

# A Look at PostgreSQL 9.5's Most Interesting Features

# LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community



**WATCH:**  
ISSUE  
OVERVIEW



MAY 2016 | ISSUE 265

<http://www.linuxjournal.com>

## SECURE TOKEN-BASED AUTHENTICATION WITH

# YubiKey 4



## Build Your Own Tiny Internet

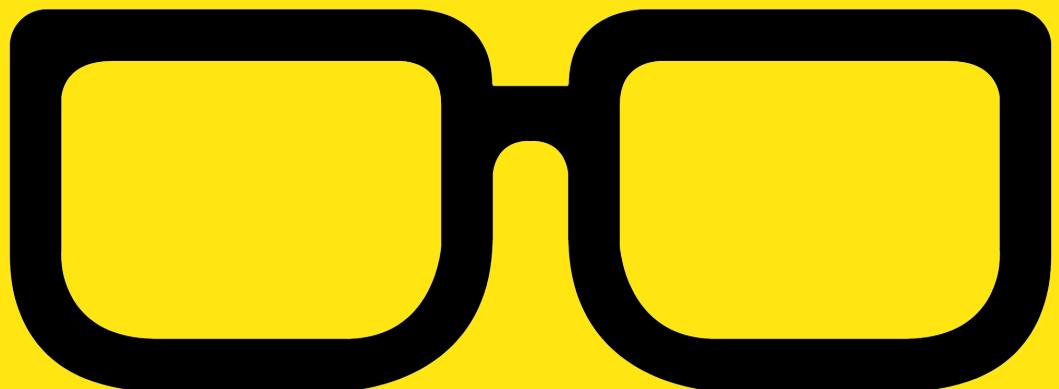
**CONFIGURE YOUR SERVER**  
to Use Your Gmail Account

—  
**TIPS FOR DEVELOPERS**  
to Prevent Compromise at the Source

## HOW TO Install Qubes and Navigate the Desktop

Practical books  
for the most technical  
people on the planet.

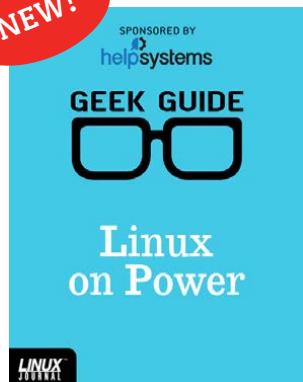
# GEEK GUIDES



Download books for free with a  
simple one-time registration.

<http://geekguide.linuxjournal.com>

**NEW!**

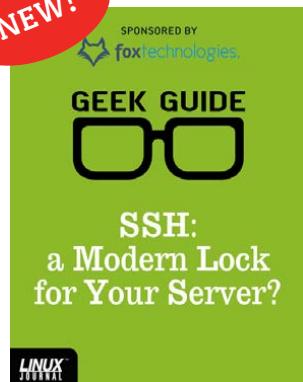


## Linux on Power

**Author:**  
Ted Schmidt

**Sponsor:**  
HelpSystems

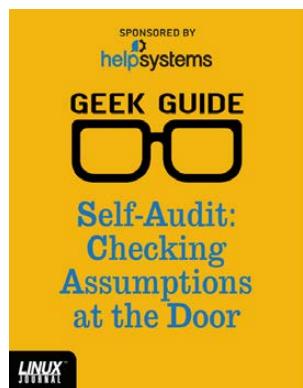
**NEW!**



## SSH: a Modern Lock for Your Server?

**Author:**  
Federico Kereki

**Sponsor:**  
Fox Technologies



## Self-Audit: Checking Assumptions at the Door

**Author:**  
Greg Bledsoe

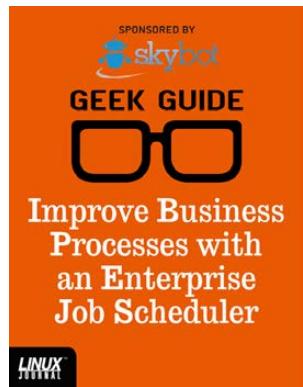
**Sponsor:**  
HelpSystems



## Agile Product Development

**Author:**  
Ted Schmidt

**Sponsor:** IBM



## Improve Business Processes with an Enterprise Job Scheduler

**Author:**  
Mike Diehl

**Sponsor:**  
Skybot



## Finding Your Way: Mapping Your Network to Improve Manageability

**Author:**  
Bill Childers

**Sponsor:**  
InterMapper



## DIY Commerce Site

**Author:**  
Reuven M. Lerner

**Sponsor:** GeoTrust



## Combating Infrastructure Sprawl

**Author:**  
Bill Childers

**Sponsor:**  
Puppet Labs

# CONTENTS

MAY 2016  
ISSUE 265



## FEATURES

### 82 Secure Token-Based Authentication with YubiKey 4

Busy Linux administrators often need to use insecure terminals, such as a co-worker's desktop, to get their jobs done.

Todd A. Jacobs provides a modern look at token-based authentication using YubiKey 4.

**Todd A. Jacobs**

### 104 The Tiny Internet Project, Part I

Use KVM to build a fully functioning "Internet" and learn all about Linux in the process.

**John S. Tonello**

#### ON THE COVER

- A Look at PostgreSQL 9.5's Most Interesting Features, p. 28
- Secure Token-Based Authentication with YubiKey 4, p. 82
- Build Your Own Tiny Internet, p. 104
- How to Install Qubes and Navigate the Desktop, p. 46
- Configure Your Server to Use Your Gmail Account, p. 54
- Tips for Developers to Prevent Compromise at the Source, p. 62

**COLUMNS**

- 28 Reuven M. Lerner's  
At the Forge**

PostgreSQL 9.5

- 38 Dave Taylor's  
Work the Shell**

The Many Paths to a Solution

- 46 Kyle Rankin's  
Hack and /**

Secure Desktops with  
Qubes: Installation

- 54 Shawn Powers'  
The Open-Source  
Classroom**

The Peculiar Case of E-mail  
in the Cloud

- 62 Susan Sons'  
Under the Sink**

Securing the Programmer,  
Part I

- 124 Doc Searls' EOF**

Privacy and the New Math

**IN EVERY ISSUE**

- 8 Current\_Issue.tar.gz**

- 10 Letters**

- 18 UPFRONT**

- 26 Editors' Choice**

- 72 New Products**

- 129 Advertisers Index**



# LINUX JOURNAL™

**Subscribe to  
Linux Journal  
Digital Edition  
for only  
\$2.45 an issue.**



**ENJOY:**

**Timely delivery**

**Off-line reading**

**Easy navigation**

**Phrase search  
and highlighting**

**Ability to save, clip  
and share articles**

**Embedded videos**

**Android & iOS apps,  
desktop and  
e-Reader versions**

**SUBSCRIBE TODAY!**

# LINUX JOURNAL

<b>Executive Editor</b>	Jill Franklin <a href="mailto:jill@linuxjournal.com">jill@linuxjournal.com</a>
<b>Senior Editor</b>	Doc Searls <a href="mailto:doc@linuxjournal.com">doc@linuxjournal.com</a>
<b>Associate Editor</b>	Shawn Powers <a href="mailto:shawn@linuxjournal.com">shawn@linuxjournal.com</a>
<b>Art Director</b>	Gerrick Antikajian <a href="mailto:gerrick@linuxjournal.com">gerrick@linuxjournal.com</a>
<b>Products Editor</b>	James Gray <a href="mailto:newproducts@linuxjournal.com">newproducts@linuxjournal.com</a>
<b>Editor Emeritus</b>	Don Marti <a href="mailto:dmarti@linuxjournal.com">dmarti@linuxjournal.com</a>
<b>Technical Editor</b>	Michael Baxter <a href="mailto:mab@cruzio.com">mab@cruzio.com</a>
<b>Senior Columnist</b>	Reuven Lerner <a href="mailto:reuven@lerner.co.il">reuven@lerner.co.il</a>
<b>Security Editor</b>	Mick Bauer <a href="mailto:mick@visi.com">mick@visi.com</a>
<b>Hack Editor</b>	Kyle Rankin <a href="mailto:lj@greenfly.net">lj@greenfly.net</a>
<b>Virtual Editor</b>	Bill Childers <a href="mailto:bill.childers@linuxjournal.com">bill.childers@linuxjournal.com</a>

## Contributing Editors

Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte  
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan • Adam Monsen

<b>President</b>	Carlie Fairchild <a href="mailto:publisher@linuxjournal.com">publisher@linuxjournal.com</a>
<b>Publisher</b>	Mark Irgang <a href="mailto:mark@linuxjournal.com">mark@linuxjournal.com</a>
<b>Associate Publisher</b>	John Grogan <a href="mailto:john@linuxjournal.com">john@linuxjournal.com</a>
<b>Director of Digital Experience</b>	Katherine Druckman <a href="mailto:webmistress@linuxjournal.com">webmistress@linuxjournal.com</a>
<b>Accountant</b>	Candy Beauchamp <a href="mailto:acct@linuxjournal.com">acct@linuxjournal.com</a>

***Linux Journal* is published by, and is a registered trade name of,  
Belltown Media, Inc.**

PO Box 980985, Houston, TX 77098 USA

## Editorial Advisory Panel

Nick Baronian  
Kalyana Krishna Chadalavada  
Brian Conner • Keir Davis  
Michael Eager • Victor Gregorio  
David A. Lane • Steve Marquez  
Dave McAllister • Thomas Quinlan  
Chris D. Stark • Patrick Swartz

## Advertising

E-MAIL: [ads@linuxjournal.com](mailto:ads@linuxjournal.com)  
URL: [www.linuxjournal.com/advertising](http://www.linuxjournal.com/advertising)  
PHONE: +1 713-344-1956 ext. 2

## Subscriptions

E-MAIL: [subs@linuxjournal.com](mailto:subs@linuxjournal.com)  
URL: [www.linuxjournal.com/subscribe](http://www.linuxjournal.com/subscribe)  
MAIL: PO Box 980985, Houston, TX 77098 USA

**LINUX** is a registered trademark of Linus Torvalds.

# Strata+Hadoop

---

## WORLD

---

P R E S E N T E D B Y

O'REILLY®

cloudera®

ExCel London | 31 May - 3 June 2016



“The Lollapalooza of big data conferences.”

— *Wired.com*

**Data science meets business.**

At Strata + Hadoop World, learn the skills and technologies you need to build successful, data-driven projects and organizations.

**[strataconf.com/uk](http://strataconf.com/uk)**

Save 20%  
Register today.  
Use code **PCLinuxJournal**

# The Power of Free

When I was a kid, I saved my money for weeks in order to buy the Aerobie (<https://en.wikipedia.org/wiki/Aerobie>). I lived in a ghetto of Detroit, and during the summer, we often would play Frisbee in the street for entertainment. I figured if I could buy the Aerobie, we'd be able to play Frisbee across multiple blocks! It would be amazing! So I hunted pop cans for weeks (in Michigan, pop can returns are 10 cents each), and finally I saved enough money to buy one. I took it home and tossed it down the street. And it kept going. I suspect it's still going, 30 years later. It flew down the street, over a block of houses and became a tiny dot in the sky. I never saw it again. Thankfully, unlike Aerobie buyers, it doesn't cost anything to start a brand-new project with Linux! We can install our operating system without worrying about license fees or saving up pop can money—and that means we aren't limited in the cool things we can do!

Reuven M. Lerner starts off the issue with an informative article on PostgreSQL 9.5. Relational databases might seem old-school, but what they do, they do very well. Reuven describes some of the cool new features of 9.5. Next up, Dave Taylor explains how to solve programming problems with the `wegrep` tool. Manipulating text via script is both incredibly fun and often very frustrating. (That frustration is part of the fun, if I'm being completely honest.) If you're a scripter who deals with text, you won't want to miss Dave's column.

Kyle Rankin continues his Qubes series this month. He



**SHAWN POWERS**

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at [shawn@linuxjournal.com](mailto:shawn@linuxjournal.com). Or, swing by the #linuxjournal IRC channel on Freenode.net.



**VIDEO:**  
Shawn Powers runs through the latest issue.

described the setup last month, and conceptually how it works, and here he covers the multi-VM model for doing actual computing. The technologies in Qubes aren't brand new, but the way of using them certainly is. Even if you're not interested in the level of security Qubes offers, you owe it to yourself to read his series if only to understand the thought process itself. It's pretty unique!

I started my column this month with a simple problem: e-mail. Now that cloud computing is the norm, it means the servers you spin up aren't in your network anymore, and so might be blocked from using your ISP's SMTP server. There are certainly ways to set up an e-mail server that you can use for proper e-mail routing, but I just need a way to send quick-and-dirty server notification messages to myself when something goes wrong. I use Postfix and Gmail to accomplish that task, and I figured I'd share my experience.

Susan Sons is back this month with a brilliant article on securing your source code, at the source! It's great to have your revision control system secured and reliable, but what if the computer you actually use to develop gets compromised? That's a bigger concern than you might realize, and Susan walks through dealing with the often overlooked problem.

We follow Susan's column with Todd A. Jacobs' tutorial on how to implement YubiKey 4 for USB token-based authentication. I have a YubiKey myself, and I have never gotten around to using it for anything more than a conversation piece. If you're looking for a way to secure your systems, but want something more than two-factor authentication, be sure to read his article. It's surprising how powerful a simple little USB device can be when used in concert with Linux.

Finally, John S. Tonello shows how to build the Internet (well, maybe not quite the Internet, but a smaller version of it for learning purposes). The free nature of Linux means you can create an elaborate and dynamic network infrastructure without worrying about licensing fees. In Part I of John's series, he begins the process of building a server farm that will in many ways function like a tiny little Internet. If you're interested in world domination, building your own Internet is a great start!

This issue is full of our regular selection tech tips, product announcements and lots of interesting information for anyone interested in Linux. The coolest part is that all our projects run on Linux, which won't cost you anything at all (unlike an Aerobie). That said, if you try to throw your Linux laptop over houses in Detroit, much like the Aerobie, you'll never get it back—at least not in the same condition as when you tossed it!■

**RETURN TO CONTENTS**

# LETTERS

---



PREVIOUS  
Current\_Issue.tar.gz

NEXT  
UpFront



## Great New Layout

I really enjoyed the new layout of the March issue. It was much more readable on my tiny netbook screen.

Maybe a font that is a bit bolder also would increase the readability, especially on low-res screens? (I know the epub edition doesn't have this limitation; however, I still prefer the PDF layout.)

—Adrien

**Shawn Powers replies:** *Thanks for the feedback! As with most changes, I'd expect a few tweaks here and there before we settle on a happy compromise between everyone's preferences.*

## Digital vs. Paper

I have been an *LJ* subscriber for many years and a Linux fan since 1994. I still remember downloading the 70+ floppy disks that comprised my very first Linux distro. I have always been an early adopter of the latest technologies, and e-readers were not an exception. These days, I have a tablet and an e-reader. Five years ago, at the time I moved from Spain to Ireland, I decided to stop buying paper books and magazines and switch to all digital, which better suited my roaming lifestyle. However, for some reason that I did not know at the time, it kept being extremely difficult to concentrate when reading certain types of content on digital formats, basically the ones that were more study-focused, such as technical articles and books, language study material and, in general, anything that

required memorization and comprehension.

Recently, I read that some research studies have found that this is a general issue that happens to (almost) everyone and that has to do with losing “context”. Our brains are still pretty much shaped for the hunter-gatherer lifestyle, which we abandoned very recently from a biological and evolutionary perspective, and writing and reading is an invention that hasn’t been with us long enough to make a mark in our brain. So we have incorporated the reading experience in one part of our brain that was used for geo-location in our hunting excursions, and that is why we use the context to remember what we read. In this particular case, context means how thick each side of the book was when we were reading a passage, how things looked on the page and on the opposite one, and so on.

In digital formats, we lose context because small things, like the frame, look the same on each page, and we do not have a physical feeling about how far we are in the book nor how much it is left to finish it.

After reading this study, I decided to stop forcing myself against my own nature and started to buy technical stuff again on paper and leave the e-reader and tablet for fiction and other non-study content. And, this is where *LJ* comes to the fore: I always have considered *LJ* one of my sources of new technical knowledge, and I use it as a way to stay connected to what’s happening out there and a first pointer to things I should take a look at and study in more detail. But since *LJ* switched to all-digital, I am reading it less and less, and there are months when I do not even browse it at all.

I think that in light of this new scientific evidence, I would like to propose that *LJ* once again (as some other readers have asked in the past) switch back to paper format.

I cannot provide the original source where I read about the scientific study I mention, but I think it was in the science section of *The Economist* sometime in 2014, which normally does summaries

of prestigious publications, such as *Science* magazine. Nevertheless, there are other sources I can provide, such as these two:

<http://www.theguardian.com/books/2014/aug/19/readers-absorb-less-kindles-paper-study-plot-ereader-digitisation> and <http://www.pri.org/stories/2014-09-18/your-paper-brain-and-your-kindle-brainarent-same-thing>.

—Juan J. Olmedilla Arregui

**Shawn Powers replies:** *It's an odd transitional period we're in, that's for certain. I also struggle with technical material on a tablet. The strange thing is, I don't seem to have any problem absorbing information from a Web browser on a computer. I don't have any explanation for why that's the case.*

*As far as moving back to a paper format, I don't think that's a financially feasible option at the publishing level. Not only is the process expensive, but advertisers have moved to digital expectations in their purchases. Thankfully, it's a fairly easy process to get a digital issue onto paper. I used to print every issue of TUX magazine, and then use a comb binder to make my own magazine. Some other folks are printing Linux Journal using on-line printing companies (for their own use, of course).*

*Your rationale is the same reason I still haven't purchased a Safari subscription and keep buying technical books. I'm curious to see what the future holds as the "digital generations" grow up. Thanks for the links as well, I appreciate it.*

## **OpenSUSE LXDE Outperforms Lubuntu, Mint/LXDE in All Key Aspects**

After using Ubuntu/Lubuntu and more recently Mint/LXDE during the past eight years, I switched to OpenSUSE/LXDE, and it has been outstandingly better and should be noted as such.

I don't mean how "pretty" it is or how easy it is to install, but as far as system performance, stability of apps (turned some from being nearly unusable to perfect) and speed.

I've got more detailed information on how it overcame some serious problems I had with Mint, which I can let you have if you're interested. I'm not a journalist, but I think anyone who is contemplating venturing into Linux or has considered the OpenSUSE option, should be encouraged to try it out. There is, as always, a downside, but the negatives are far outweighed by many positives.

—Nigel Hinton

**Shawn Powers replies:** Thanks Nigel. Some of the folks here at Linux Journal are die-hard OpenSUSE fans. For me, I usually consider ease of use as one of my most important aspects when choosing a distro. Unfortunately, that typically means what I'm most familiar with (Debian/Ubuntu, in my case). Thanks for the info on LXDE and OpenSUSE; I wouldn't have given it a try otherwise!

## Suggestion: Backdoor to Smartphone

This short video explains how biometrics make a backdoor to password-protected personal secrets: <https://youtu.be/5e2oHZccMe4>.

Many citizens are being misguided collectively by the media and get trapped in a false sense of improved security generated by the addition of biometric functions to smartphones, tablets and PCs, while many criminals presumably understand what this situation means.

The role of leading media like you is very obvious. I hope that you will not hesitate to play that role.

—Hitoshi Kokumai

**Shawn Powers replies:** I think it's important for folks to realize that biometrics can add a layer of security to existing technologies, but replacing existing models with biometrics only is definitely a risk. That said, as long as an understanding of those concerns exists, there are some viable reasons for using biometrics only. (Securing my home gaming laptop, for instance, if I'm not concerned about it being compromised.) I share your concern, however, that biometrics might seem more secure than they really are. Thanks for the reminder.

## Note from LJ Author Charles Fisher on Elliptic Curve

In my previous article on ciphers in TLS and SSH ["Cipher Security" in the September 2015 issue], there was no detailed discussion of Elliptic Curve (EC). The default settings are considered safe, but considerable suspicion of tampering exists for the standard curves. Stronger EC settings are available, and what follows is a short guide to enable them.

OpenSSL on Oracle Linux V7 supports three Elliptic Curves:

```
$ openssl ecparam -list_curves
secp384r1 : NIST/SECG curve over a 384 bit prime field
secp521r1 : NIST/SECG curve over a 521 bit prime field
prime256v1: X9.62/SECG curve over a 256 bit prime field
```

All of these curves are endorsed and supported as safe for financial transactions, government data and other sensitive content. The default curve used in TLS for EC is prime256v1.

However, both the 256 and the 384 curves come under heavy criticism from multiple sources for unexplained constants used in the formulas.

The prime256v1 curve has received particular sentiments of alarm:  
<https://github.com/nodejs/node/issues/1495>.

Daniel J. Bernstein, well-known in both cryptography and software development, dismisses both the 256 and 384 curves as tainted (<http://safecurves.cr.yp.to>):

<b>NIST P-256</b>	<b>manipulatable</b>	Coefficients generated by hashing the unexplained seed c49d3608 86e704936a6678e1 139d26b7 819f7e90.
<b>NIST P-384</b>	<b>manipulatable</b>	Coefficients generated by hashing the unexplained seed a335926a a319a27a1d00896a 6773a482 7acdac73.

Recent research on TLS acknowledges concern for the mysterious constants in the 256 and 384 curves (<https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf>): "Unfortunately, the most widely supported ECDH parameters, those

specified by NIST, are now viewed with suspicion due to NSA influence on their design, despite no known or suspected weaknesses.”

However, the secp521r1 curve appears to be more forthrightly designed, and has received praise instead of distrust (<http://blog.cr.yp.to/20140323-ecdsa.html>): “To be fair I should mention that there’s one standard NIST curve using a nice prime, namely  $2^{521}-1$ ; but the sheer size of this prime makes it much slower than NIST P-256.”

For non-HTTPS EC applications not involving a Web browser, especially where both endpoints are implemented with modern OpenSSL, prefer secp521r1. For the highest quality EC within HTTPS, prefer secp384r1, since secp521r1 support is not consistent between Chrome, Firefox and IE. The prime256v1 curve should be avoided, unless speed is preferred to quality.

The stunnel utility makes it easy to select alternate curves. Pass the `curve=secp521r1` option in the configuration file, with the exact

## LINUX JOURNAL

on your  
e-Reader

Customized  
**Kindle** and **Nook**  
editions  
available

**LEARN MORE**



name as listed in the previous "ecparam" example. Otherwise, just append the output of `openssl ecparam -name secp521r1` to your certificate file.

There are more reasons to avoid Elliptic Curve in TLS. In addition to the hints of subterfuge, there are software patent questions within the US that have come before courts of law, some by holders of significant intellectual property: <http://security.stackexchange.com/questions/3519/can-ecc-be-used-without-infringing-on-patents>.

Recent rapid-fire lawsuits have seen settlements from major corporations over patents on Elliptic Curve: [http://www.theregister.co.uk/2015/12/01/cryptopeak\\_sues\\_](http://www.theregister.co.uk/2015/12/01/cryptopeak_sues_).

If you have chosen to forego Elliptic Curve for any of these reasons, be aware that there is a small security ratings boost on the <http://ssllabs.com> scanner for using 4096-bit Diffie-Hellman primes in situations not involving EC, although this also imposes a heavy speed penalty.

### PHOTO OF THE MONTH

Remember, send your Linux-related photos to [leditor@linuxjournal.com](mailto:leditor@linuxjournal.com)!

### WRITE LJ A LETTER

We love hearing from our readers. Please send us your comments and feedback via <http://www.linuxjournal.com/contact>.

**RETURN TO CONTENTS**

# LINUX JOURNAL

## At Your Service

**SUBSCRIPTIONS:** *Linux Journal* is available in a variety of digital formats, including PDF, .epub, .mobi and an on-line digital edition, as well as apps for iOS and Android devices. Renewing your subscription, changing your e-mail address for issue delivery, paying your invoice, viewing your account details or other subscription inquiries can be done instantly on-line: <http://www.linuxjournal.com/subs>. E-mail us at [subs@linuxjournal.com](mailto:subs@linuxjournal.com) or reach us via postal mail at *Linux Journal*, PO Box 980985, Houston, TX 77098 USA. Please remember to include your complete name and address when contacting us.

**ACCESSING THE DIGITAL ARCHIVE:** Your monthly download notifications will have links to the various formats and to the digital archive. To access the digital archive at any time, log in at <http://www.linuxjournal.com/digital>.

**LETTERS TO THE EDITOR:** We welcome your letters and encourage you to submit them at <http://www.linuxjournal.com/contact> or mail them to *Linux Journal*, PO Box 980985, Houston, TX 77098 USA. Letters may be edited for space and clarity.

**WRITING FOR US:** We always are looking for contributed articles, tutorials and real-world stories for the magazine. An author's guide, a list of topics and due dates can be found on-line: <http://www.linuxjournal.com/author>.

**FREE e-NEWSLETTERS:** *Linux Journal* editors publish newsletters on both a weekly and monthly basis. Receive late-breaking news, technical tips and tricks, an inside look at upcoming issues and links to in-depth stories featured on <http://www.linuxjournal.com>. Subscribe for free today: <http://www.linuxjournal.com/enewsletters>.

**ADVERTISING:** *Linux Journal* is a great resource for readers and advertisers alike. Request a media kit, view our current editorial calendar and advertising due dates, or learn more about other advertising and marketing opportunities by visiting us on-line: <http://www.linuxjournal.com/advertising>. Contact us directly for further information: [ads@linuxjournal.com](mailto:ads@linuxjournal.com) or +1 713-344-1956 ext. 2.



Register Early  
and **SAVE!**

**All-Access Pass**  
**ONLY \$695**

Register before May 13, 2016

**Expo-Only Pass**  
**\$25**

"This is an excellent conference to attend to get an understanding from both a business and technical sense of Wearables and IOT."

—Kevin Jones, Electrical Engineer, Dept. of Defense

## Come to the Largest Wearable-IoT OEM Expo of 2016!

Go under the hood of the hottest wearable devices!

# Wearables TechCon

**July 18-20, 2016**

**San Jose Convention Center**

The Heart of Silicon Valley

A BZ Media Event

**PLUS!**  
**Internet  
of Things  
TechCon**

Learn how to design, build and develop apps  
for the wearable technology revolution  
at Wearables TechCon 2016!

**More than 60 Tech Sessions and Panels**

- Product Design
- Software Design
- Leading SDKs
- Embedded System Development
- Application Development
- Electronic Engineering for Wearables Devices

...and more tricks and techniques that will set your wearable project apart!

[www.WearablesTechCon.com](http://www.WearablesTechCon.com)





PREVIOUS  
Letters

NEXT  
Editors' Choice



## diff -u

### What's New in Kernel Development

**Michal Hocko** has submitted some code to rework the detection of **out-of-memory conditions**. Earlier versions were “requests for comments”, but with no major objections, Michal now is asking that the code be added to **linux-next** to widen its testing base.

The current patch was just an incremental update, but Michal said that the goal of the larger project:

...is an attempt to make the OOM detection more deterministic and easier to follow because each reclaimer basically tracks its own progress which is implemented at the page allocator layer rather [than] spread out between the allocator and the reclaim.

He asked folks to come up with new testing scenarios, because the OOM killer is notoriously hard to get right. Ideally, it works properly under “standard load”, but how can there be a standard load when Linux is used throughout the world for every conceivable purpose?

**Hugh Dickins** reported a problem with the new code: it would kill his **TmpFS** instance during heavy swap, whereas the old OOM killer would let it run forever. But, he did acknowledge that “tmpfs

is and always has been a problem for OOM-killing, given that it takes up memory.”

Various other folks posted the results of their tests as well, mostly showing success, though there were some long discussions of particular technical details and a bunch of code revisions as Michal addressed people’s concerns.

**Balbir Singh** posted patches to allow updating a running Linux kernel on **PowerPC**. This kind of “live patching” is one of the holy grails of operating system development. In order to do it, you have to transfer all the running infrastructure from the existing kernel to the new one. This involves changing stored memory locations, replacing functions at the proper time and other strange things, all of which must be done individually. It’s the kind of feature that, before it was possible, people said never would work. Even now,

**Torsten Duwe** replied to Balbir’s post, saying, “Have you tested this patch? Replacing a function in the kernel? Replacing a function in a module? For local calls? For global calls? I strongly doubt so because it does not work this way.”

In fact, Torsten wasn’t saying that the whole thing didn’t work that way, only that the particular approach was problematic and needed further discussion. Meanwhile, **Petr Mladek** tested the patch and reported failure. He described the test results as, “!!! KABOOM !!!” Balbir replied, “Very good test case”, and proceeded to investigate fixes.

After some further effort, Balbir said:

The previous revision was nacked by Torsten, but compared to the alternatives at hand I think we should test this approach. Ideally we want all the complexity of live-patching in the live-patching code and not in the [user’s] patch. The other option is to accept v4 and document the limitation to patch writers of not patching functions [that have more than] 8 arguments.

He posted a new patch, which Torsten nacked again, saying, “I nacked it because I was confident it couldn’t work. Same goes for this one, sorry. My good intention was to save us all some work.” Commenting on some of the technical problems, he added, “Using

heuristics to determine whether the call was local or global makes me feel highly uncomfortable; one day it will break and nobody will remember why."

Referring to Balbir's earlier comment, **Michael Ellerman** said it was unlikely that anyone would want to live patch a function with more than eight arguments, but he added:

With the current proposals, we have no way of preventing them from doing so. Which means the first sign they'll get that it doesn't work is when they've applied the patch and their production system goes down. And not even when they insert the patch, only when the patched function is called, possibly some time later.

He also remarked, "perhaps in reality most people are only applying live patches from their distro, in which case the distro should have tested it. But I don't know for sure."

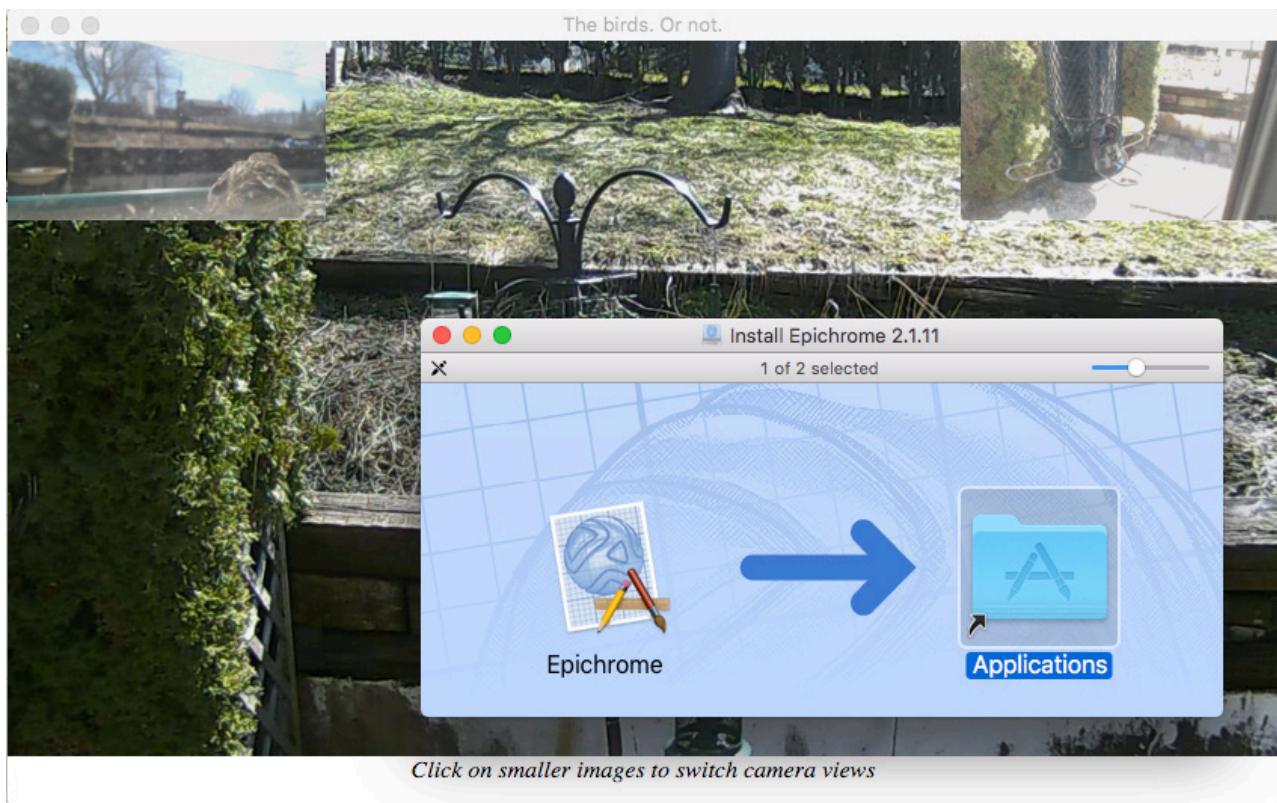
And in spite of these remaining problems, Michael added that he wanted to see these patches go into the kernel and continue testing and developing. He said, "I think we can probably come up with a fully robust solution. But not tonight, and not this week."

Throughout all of this, Balbir affirmed that there were plenty of problems to solve. He said:

I've been working with the constraints we have to get a solution that does not put the burden on the patch writer. That is why this is marked experimental as it needs a lot of testing. I think we should mark live patching on PPC as experimental to begin with.

And added, "I am keen on getting live-patching working. I think v4 with the documented limitation is fine."

And, the quest for the grail continues. Everyone involved in this discussion, in spite of their sometimes dire predictions and warnings, were all enthusiastic to get some form of this code into the kernel, even with the heavy set of caveats that currently exist. One day, live patching on PowerPC will just be normal, but for now, it remains the insane upside-down province of these lunatic visionaries.—Zack Brown



# Non-Linux FOSS: Chrome, for One

When I use OS X, I really like the Fluid app for making standalone Web applications. The problem is, Fluid isn't free unless you want the basic version. I don't mind paying for an application (and I did pay for Fluid), but it seems like something as simple as a single site browser shouldn't be something that costs money.

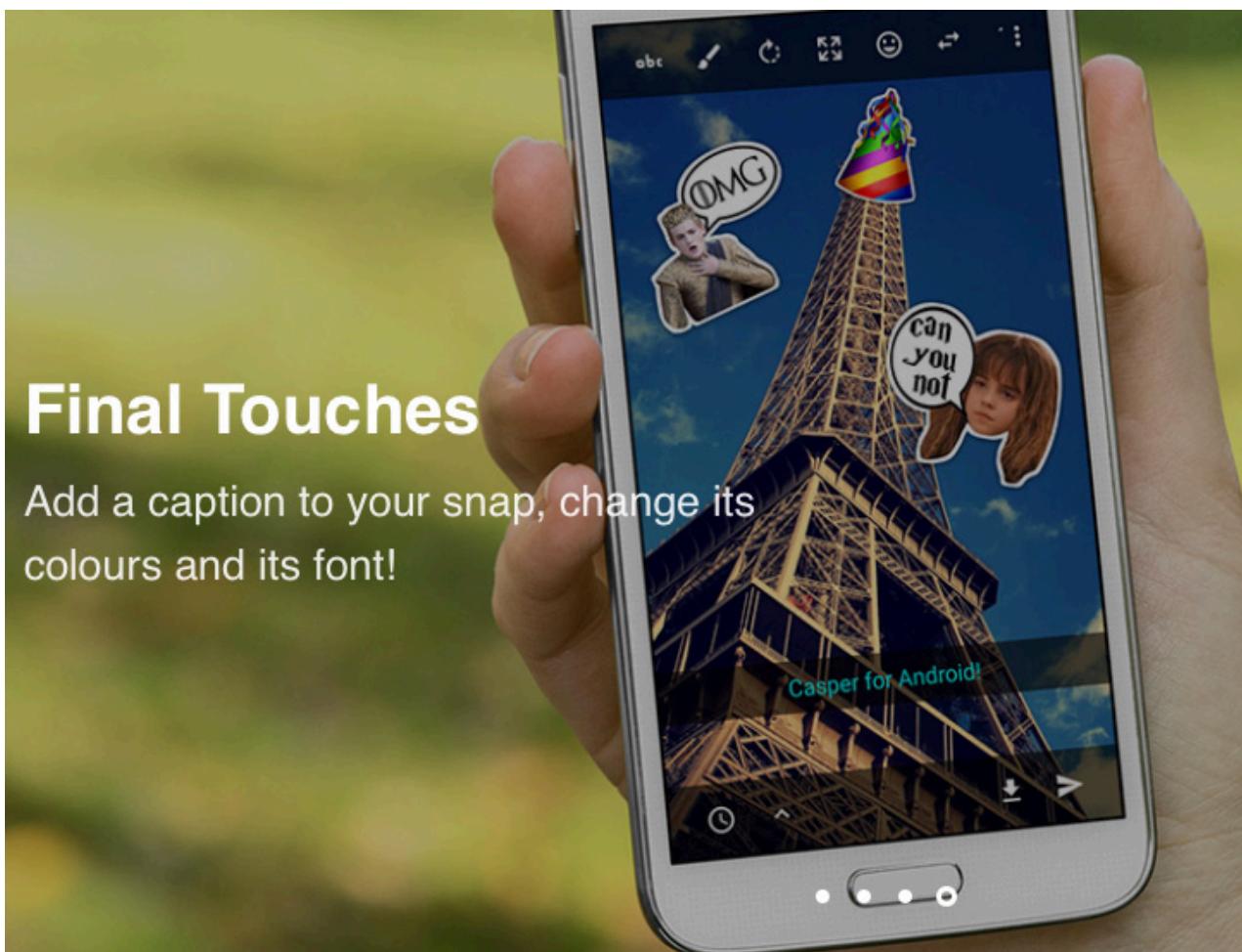
Thankfully, the folks at Epichrome feel the same way. They've created an open-source project at <https://github.com/dmarmor/epichrome> that allows you to create standalone OS X applications that use Chrome in order to provide a single site browser.

You can customize the way the app looks, give it a custom app icon and even register it as a browser on the system, so you can have it open when clicking on a specific Web site from inside your Web browser. I personally use it for my BirdCam, but it's a great way to turn any Web site into an "app" that you can launch from your dock.—Shawn Powers

# Android Candy: Snapchat, for Hoarders!

I like Snapchat, but if I'm being totally honest, it's not something I use every day. I like it because my kids send me goofy pictures and videos, and it makes me happy that they think to include me in their Snapchatty world.

The thing is, Snapchat is designed with the notion that photos and videos should be ephemeral, and once they're viewed, they



## Final Touches

Add a caption to your snap, change its colours and its font!

(Photo from <http://casper.io>)

disappear. I don't share that particular sentiment, and I tend to err the other way. Namely, I not only keep photos and videos, but I also generally back them up in three different places, geographically separated in case of failure. Also, I'm old, and I forget things.

Thankfully, there's a non-official Snapchat client made for folks like me. It's called Casper (<http://casper.io>), and it's not available in the Google Play store. Installing it is fairly simple, but because it allows you to save photos and videos, it's not allowed to be distributed in the Play store. Also, I'm sure the folks at Snapchat don't really like the app, since it goes against one of the app's core "features", but if you're okay with all those issues, I highly recommend it. The app is free, and you can use it as your Snapchat client, even if you don't intend to save the photos or videos your kids send you. But, if your kids are as interesting as mine, you'll probably want to!—Shawn Powers

## LINUX JOURNAL on your **Android** device

Download the app  
now from the  
**Google Play Store.**



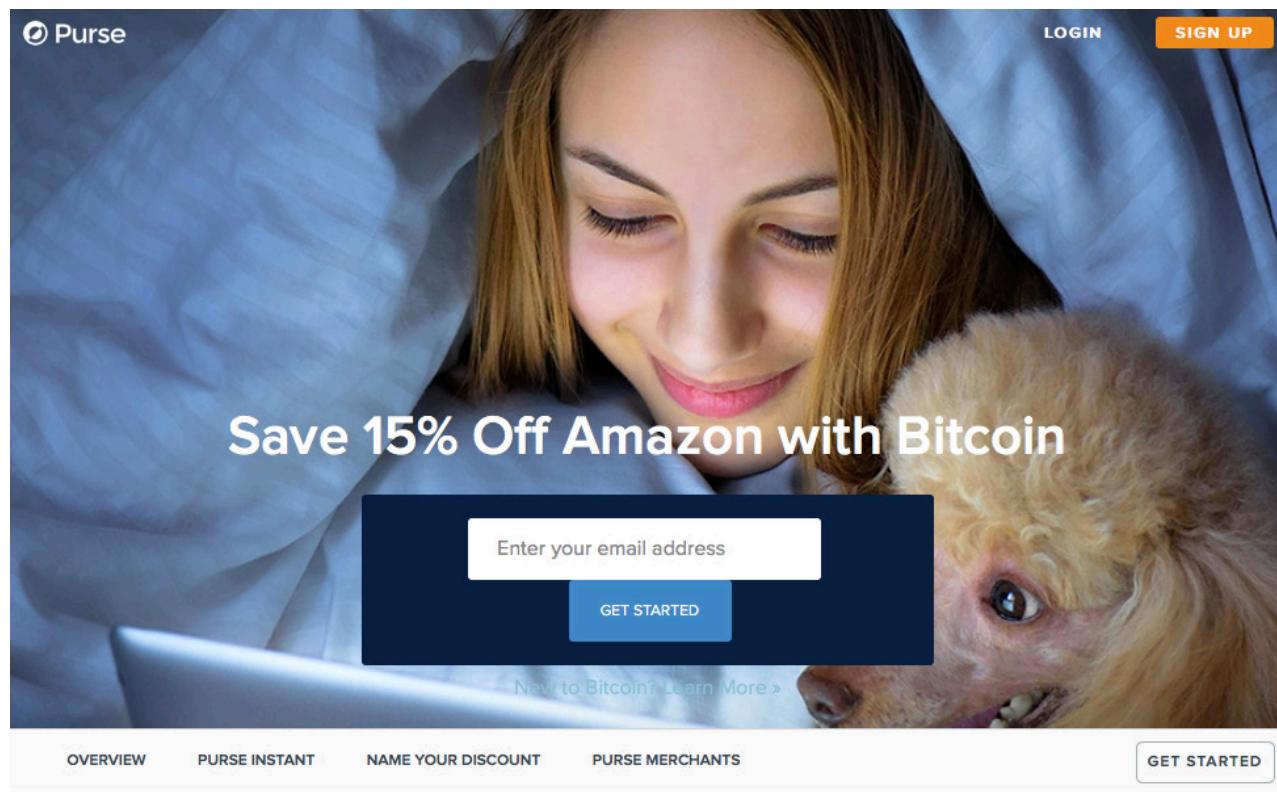
[www.linuxjournal.com/android](http://www.linuxjournal.com/android)

For more information about advertising opportunities within *Linux Journal* iPhone, iPad and Android apps, contact John Grogan at +1-713-344-1956 x2 or [ads@linuxjournal.com](mailto:ads@linuxjournal.com).

# Bitcoin on Amazon! Sort of...

I was a Bitcoin fan before it was popular. That means I had thousands of Bitcoins. It also means I sold my thousands of Bitcoins for less than \$1 each. Still, the technology fascinates me, and although cryptocurrencies have risen and fallen, I'm still a fan.

There are several places to use Bitcoin on a regular basis. One of my favorites, which I've mentioned before, is the Humble Bundle (<http://www.humblebundle.com>). I've also ordered from Overstock.com using Bitcoin. I often look for places I can spend Bitcoin and try to shop at them when possible. Like most people, however, I usually end



(Photo via <http://purse.io>)

up buying things on Amazon.

Thankfully, with the help of Purse.IO (<http://www.purse.io>), you can indirectly purchase items from Amazon using Bitcoin! I've been using Purse.IO for more than a year, but it seemed too good to be true, so I hesitated writing about it. The premise is this: some people have Amazon gift cards but would rather have Bitcoin. I have Bitcoin and want to buy stuff from Amazon. Using wish lists and an escrow system with Purse.IO, we trade. In fact, people are willing to give a discount in order to get Bitcoin for their gift cards, and so I can order items from Amazon and get a large percentage (sometimes more than 25%) off the retail price. I usually agree to around 13% off, and there are folks willing to send me things from Amazon. Once the items from my wish list arrive, Purse.IO releases the Bitcoin from escrow, and both parties are happy.

I'm sure there are people who try to scam others by ordering items with bad credit cards and so on, but since you release the escrow only when the item actually arrives, the risk is pretty low. I don't think I'd want to spend vast amounts of Bitcoin via Purse.IO, but since I don't have vast amounts of Bitcoin, it works out quite well for me. If you have some Bitcoin and wonder what to do with it, check out Purse.IO. It's pretty cool!

—Shawn Powers

## THEY SAID IT

**If you don't risk anything you risk even more.**

—Erica Jong

**Life is a great big canvas; throw all the paint on it you can.**

—Danny Kaye

**Bite off more than you can chew, then chew it. Plan more than you can do, then do it.**

—Anonymous

**A man travels the world over in search of what he needs and returns home to find it.**

—George Moore

**In terms of being late or not starting at all, then it's never too late.**

—Alison Headley

# EDITORS' CHOICE



PREVIOUS  
UpFront

NEXT

Reuven M. Lerner's  
At the Forge



## Glass Padding?

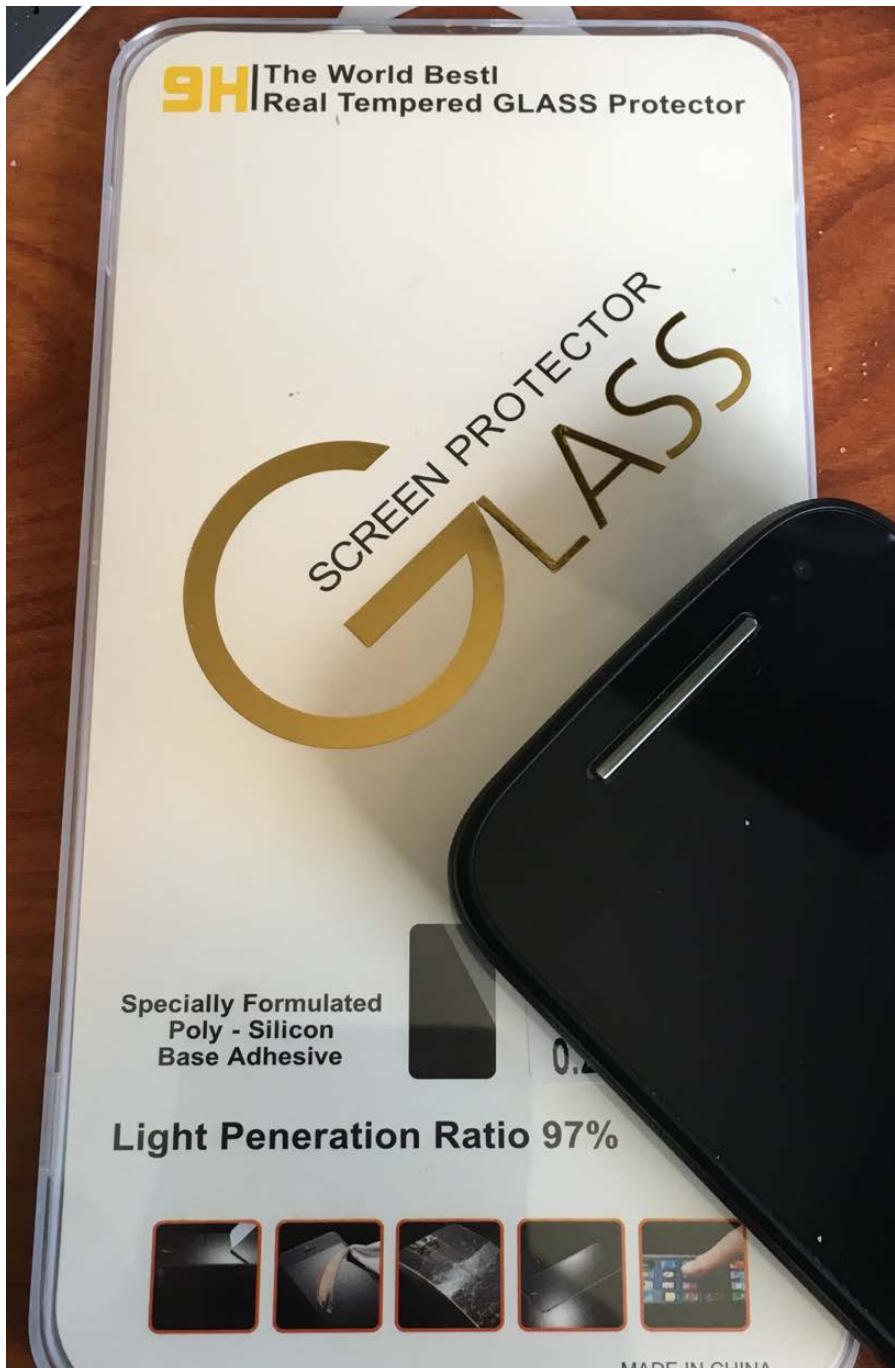
When it comes to covering my cell phone, I tend toward minimalism. I like to buy the smallest (although still powerful) phone possible, so the thought of adding a bulky case seems wrong. I also don't like screen protectors, because they generally get cloudy, and they don't feel as nice when using the screen. That said, I still usually buy them, because I've forgotten about my phone when getting out of the car and dropped it screen first on gravel far too many times.

While ordering a screen protector after damaging mine on pavement recently, I noticed that "glass" screen protectors are becoming quite popular. *Glass*. To protect the glass. Isn't that like making a helmet out of eggshells? The notion of protecting a glass screen with more glass just seems absurd. And yet, the screen "protectors" get great reviews. So I tried one.

First off, even with the thinnest tempered glass screen protector, they are noticeably thicker than plastic film-based protectors. They're also a little more challenging to apply. The truth is, however, that even with the extra thickness, a tempered glass screen protector is amazing to use! The screen is just as responsive, the surface feels just like glass (duh!), and it doesn't get cloudy with regular use. In fact, it's by far the best "covered screen" experience I've ever had on a cell phone. But still, it's a piece of glass adhered to a piece of glass. I'm not sure how that is really protection.

Except that apparently it is. I've gone through two tempered glass screen protectors in the past few months. Both times I dropped the





phone, it landed right on the glass, and both times it was on rough gravel. The first time, the screen protector completely shattered. The second time it cracked, but didn't shatter. Both times, the glass underneath wasn't harmed even a little.

I'll be honest, I don't understand the physics behind protecting a layer of glass with another layer of glass. Somehow it seems to work though. If you wrapped an egg in a second eggshell, I don't think it would endure a

fall to the ground, but the glass screen protector apparently uses physics in ways I don't comprehend. And for that reason, I'm giving "tempered glass screen protectors" this month's Editors' Choice Award. If you've been hesitant to try them, I urge you to reconsider!—Shawn Powers

**RETURN TO CONTENTS**

# PostgreSQL 9.5

Reuven describes some of the most interesting features of PostgreSQL 9.5.



**REUVEN M.  
LERNER**

◀ PREVIOUS  
Editors' Choice

NEXT  
Dave Taylor's  
Work the Shell ▶

**LONGTIME READERS OF THIS COLUMN** know that although NoSQL databases certainly have their place, I'm definitely a fan of relational (SQL) databases. And when choosing a relational database, my hands-down favorite is PostgreSQL. I've been using it for many years, starting soon after it was released, and although those early versions of PostgreSQL were packed with features, there also were significant limitations in nearly every aspect.

Today, however, PostgreSQL is recognized not just as a sophisticated database, but as a platform that can be used for storing and retrieving data, for working with data on many types of remote systems, and for manipulating and analyzing that data. Just before sitting down to write this article, I finished teaching a course in PostgreSQL to a high-tech company—and throughout the course, even people who have been working with PostgreSQL for years said, "I didn't know that it could do that too!"

PostgreSQL 9.5, which was released earlier this

Reuven M. Lerner offers training in Python, Git and PostgreSQL to companies around the world. He blogs at <http://blog.lerner.co.il>, tweets at @reuvenmlerner and curates <http://DailyTechVideo.com>. Reuven lives in Modi'in, Israel, with his wife and three children.

# Perhaps the most talked-about feature in PostgreSQL 9.5 is what's known as “UPSERT” in database circles.

year, might not have earth-shattering features, but it continues in a long tradition of carefully managed, rock-solid releases, combining new functionality with improved performance.

I've recently recommended that my clients start to upgrade to PostgreSQL 9.5. Along the way, I've been looking into some of the improvements and additions that were contained in this release. In this article, I review some of the more interesting ones and describe how they can be used. At the end of the day, a database is designed to help you extract useful information easily and quickly; most of these features fit precisely into that category.

## UPSERT

Perhaps the most talked-about feature in PostgreSQL 9.5 is what's known as “UPSERT” in database circles. The basic idea is that you should be able to `INSERT` a new record into a table, but if the new record would collide with an existing one, then you `UPDATE` the existing row instead.

For example, let's assume you have the following table:

```
CREATE TABLE People (
    id          SERIAL PRIMARY KEY,
    ssn         TEXT NOT NULL,
    first_name  TEXT NOT NULL,
    last_name   TEXT NOT NULL,
    UNIQUE(ssn)
);
```

The above table assigns the `id` column to be a primary key, using PostgreSQL's `SERIAL` pseudo-type to create a sequence and ensure that it

auto-increments when you don't set it explicitly. But, say you also want to keep track of people's Social Security Numbers (on the assumption that everyone in the database is from the United States), which also are supposed to be unique and, thus, get a `UNIQUE` index. Then you have the person's first and last names.

If you try to `INSERT` a new person into this table, it should work fine:

```
INSERT INTO People (ssn, first_name, last_name)
VALUES ('123-456-7890', 'John', 'Smith');
```

If you try to `INSERT` the same row again, you'll get an error. Even if the user's SSN is the same, but the name has changed, this won't work. But of course, that's the whole idea of having the `UNIQUE` clause in the table definition; you want to ensure that no other row can have the same SSN.

But, there likely will be many cases when you basically want to say to the database, "If this is a new SSN, then insert the new row. But if someone with this SSN already exists, then update their record to reflect the new name."

You can do this by modifying the `INSERT` statement to include an `ON CONFLICT` clause. This clause can work in a few ways. First, it can indicate that you silently want to ignore such conflicts:

```
INSERT INTO People (ssn, first_name, last_name)
VALUES ('123-456-7890', 'Richard', 'Roe')
ON CONFLICT DO NOTHING;
```

The above query basically tells PostgreSQL, "Insert this row if you can. If not, don't worry about it."

A more common action, and the source of the "UPSERT" term, is to say "`ON CONFLICT DO UPDATE`", providing an `UPDATE`-like statement:

```
INSERT INTO People (ssn, first_name, last_name)
VALUES ('123-456-7890', 'Richard', 'Roe');
ON CONFLICT DO
    UPDATE SET first_name = 'Richard', last_name = 'Roe'
    WHERE ssn = '123-456-7890';
```

If you have assigned a default value to one of the columns in question, you even can say:

```
SET colname = DEFAULT
```

and the default value will be inserted.

If you want to set a value based on an existing one, you can do that, by using the standard `TableName.colname` syntax.

Note that UPSERT is one of those things that you should think about very carefully. Do you really want to replace an existing record with a new one? Or, do you want to enforce the uniqueness and, thus, get an error if you try to `INSERT` a new row whose unique values clash with those already in the database? That's a question that only you can answer, and it's the reason why you have to add that to your `INSERT` statement explicitly.

That said, this functionality, used appropriately, allows you to shorten your queries and make your logic clearer.

## Better Grouping

One of the first things you learn in SQL is to count things. For example:

```
SELECT COUNT(*) FROM People;
```

That'll tell you how many people you have overall. But in most cases, you want to break that apart, finding out how many people of various sorts you have. I'm going to create a new `People` table and populate it based on a recently released movie:

```
CREATE TABLE People (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    species TEXT NOT NULL,
    gender CHAR(1) NOT NULL,
    side TEXT NOT NULL
);
```

```
INSERT INTO People (name, species, gender, side)
VALUES ('Luke', 'Human', 'M', 'Good'),
       ('Leia', 'Human', 'F', 'Good'),
       ('Han', 'Human', 'M', 'Good'),
       ('Chewbacca', 'Human', 'M', 'Good'),
       ('Kylo', 'Human', 'M', 'Evil'),
       ('Phasma', 'Human', 'F', 'Evil'),
       ('Rey', 'Human', 'F', 'Good'),
       ('Finn', 'Human', 'M', 'Good'),
       ('R2D2', 'Droid', 'D', 'Good'),
       ('C3P0', 'Droid', 'D', 'Good'),
       ('BB8', 'Droid', 'D', 'Good')
;
```

You can find the gender breakdown pretty easily, by using `GROUP BY`:

```
SELECT gender, COUNT(*) FROM People GROUP BY gender;
```

But, what if you're interested in finding out the breakdown by several factors, not just gender? You could issue multiple queries. With PostgreSQL 9.5, you also can use `GROUPING SETS`. This is used with `GROUP BY` and lets you pass a list of the columns by which you want to group. Thus, you can write:

```
SELECT species, gender, side, COUNT(*)
  FROM People
GROUP BY GROUPING SETS (species, gender, side);
```

The result is a table, showing how many humans vs. droids, male vs. female vs. droid, and good vs. evil are in the movie:

species	gender	side	count
Droid			3
Human			8

		Evil	2
		Good	9
	D		3
	F		3
	M		5

If you want to get a count of all records, you can add an empty set of parentheses to your call to GROUPING SETS:

```
SELECT species, gender, side, COUNT(*)
  FROM People
GROUP BY GROUPING SETS (species, gender, side, ());
```

Now you'll get an additional row, indicating how many total rows are in the table. However, that row will have NULL values in each column, so if your other columns contain NULL values, you might have some ambiguity problems.

GROUPING SETS can take multi-layer sets, as well:

```
SELECT species, gender, side, COUNT(*)
  FROM People
GROUP BY GROUPING SETS ((species, gender),
                         (species, gender, side));
```

What if you don't want to see the numbers from individual columns, but combinations? That is, maybe you want to find how many good females there are or how many male humans. If you're running an on-line advertising system, you might want to provide your users (à la Facebook's ad manager) with the ability to break down advertising by country, gender, interests or something else. For such cases, now there is the CUBE facility, which provides all of the possible combinations for GROUPING SETS:

```
SELECT species, gender, side, COUNT(*)
  FROM People
GROUP BY CUBE (species, gender, side);
```

The result looks like this:

species	gender	side	count
Droid	D	Good	3
Droid	D		3
Droid			3
Human	F	Evil	1
Human	F	Good	2
Human	F		3
Human	M	Evil	1
Human	M	Good	4
Human	M		5
Human			8
			11

You can think of CUBE as providing all of the permutations of columns.

A similar type of analysis, known as ROLLUP, also breaks things down in multiple layers, starting with the full list, then all but the final one, then all but the final two, until you get down to an empty list. This is useful when you have a hierarchy—imagine a salary table in which each person's location, division and team are indicated. You then could get total salary (with `SUM`) or average salary (with `AVG`) across those different layers.

These new options to `GROUP BY` were added to help people using PostgreSQL for their data analysis in the ever-growing world of “big data” and data science. Some commercial databases have offered this functionality for some time, and now it has been added to PostgreSQL as well.

## Track Commit Timestamp

This is a small feature, but one that will be welcome in many quarters. For years, Ruby on Rails has added `created_at` and `modified_at` columns to every ActiveRecord model, because it's

so useful to have the timestamp at which a record was either created or modified. PostgreSQL 9.5 optionally allows you to add this to any table.

You must activate this feature in the `postgresql.conf` configuration file, and if you change its value, you need to restart PostgreSQL in order to see the effects. When activated, the line will look like this:

```
track_commit_timestamp = one
```

Now, when you create a new table and add some rows:

```
CREATE TABLE Stuff (
    id SERIAL PRIMARY KEY,
    thing TEXT NOT NULL
);
```

```
INSERT INTO Stuff (thing) values ('a');
INSERT INTO Stuff (thing) values ('b');
INSERT INTO Stuff (thing) values ('c');
```

A normal `SELECT` on that table will give the following:

```
[local]/reuven=# select * from stuff;
+----+-----+
| id | thing |
+----+-----+
| 1  | a     |
| 2  | b     |
| 3  | c     |
+----+-----+
(3 rows)
```

But, if you use the `pg_xact_commit_timestamp` function on the normally hidden `xmin` column, you can find the timestamp for

each row:

```
[local]/reuven=# SELECT pg_xact_commit_timestamp(xmin), * FROM stuff;
+-----+-----+-----+
| pg_xact_commit_timestamp | id | thing |
+-----+-----+-----+
| 2016-03-24 12:07:16.591932+02 | 1 | a      |
| 2016-03-24 12:07:16.592771+02 | 2 | b      |
| 2016-03-24 12:07:16.593563+02 | 3 | c      |
+-----+-----+-----+
```

Remember that the timestamp is held frozen for a transaction. Thus, if you `INSERT` several rows at the same time, they'll have the same timestamp:

```
INSERT INTO Stuff (thing) values ('d'), ('e'), ('f');
```

```
[local]/reuven=# SELECT pg_xact_commit_timestamp(xmin), * from stuff;
+-----+-----+-----+
| pg_xact_commit_timestamp | id | thing |
+-----+-----+-----+
| 2016-03-24 12:07:16.591932+02 | 1 | a      |
| 2016-03-24 12:07:16.592771+02 | 2 | b      |
| 2016-03-24 12:07:16.593563+02 | 3 | c      |
| 2016-03-24 12:10:15.647167+02 | 4 | d      |
| 2016-03-24 12:10:15.647167+02 | 5 | e      |
| 2016-03-24 12:10:15.647167+02 | 6 | f      |
+-----+-----+-----+
```

## And Much More

Those are the features I'll be using the most, but PostgreSQL 9.5 includes a lot more than what I have space to describe here. Foreign data wrappers, which allow PostgreSQL to talk to other databases, have gotten much smarter, including the ability to import foreign schemas. JSON operators have become more sophisticated, making PostgreSQL into (ironically) one of the fastest and most fully featured NoSQL

databases. New BRIN indexes are a good compromise between speed, accuracy and size. And of course, there are numerous performance improvements as well.

And although it's still too early to talk about it seriously, there already has been discussion of whether the next version will be called 9.6 or 10.0—in part because it looks like the next version will include some truly killer features. About a year from now, we'll be able to explore those and see just how well they make this amazing database even more amazing.

### Conclusion

As I have learned to expect, the latest version of PostgreSQL offers a host of new features and enhancements that make it not just a rock-solid database on which to run your operations, but also one that offers a great deal of flexibility to do so. If you're already a die-hard PostgreSQL user, you'll probably enjoy these new features and should plan to upgrade soon. And if not, well, then maybe you'll take a look at it! ■

### RESOURCES

PostgreSQL's home page is at <http://postgresql.org>. A wiki page describing and documenting all of the changes in 9.5 is available at [https://wiki.postgresql.org/wiki/What's\\_new\\_in\\_PostgreSQL\\_9.5](https://wiki.postgresql.org/wiki/What's_new_in_PostgreSQL_9.5).

Send comments or feedback via  
<http://www.linuxjournal.com/contact>  
or to [ljeditor@linuxjournal.com](mailto:ljeditor@linuxjournal.com).

[RETURN TO CONTENTS](#)

## The Many Paths to a Solution

Dave explores the many ways to solve programming problems in Linux with `wegrep`.



DAVE TAYLOR

---

Dave Taylor has been hacking shell scripts since the dawn of the computer era. Well, not really, but still, 30 years is a long time! He's the author of the popular *Wicked Cool Shell Scripts* and *Teach Yourself Unix in 24 Hours* (new edition just released!). He can be found on Twitter as @DaveTaylor and at his tech site: <http://www.AskDaveTaylor.com>.

---

PREVIOUS

Reuven M. Lerner's  
At the Forge

NEXT

Kyle Rankin's  
Hack and /



**A PROJECT I'M INVOLVED WITH** has made me think about how there are always many solution paths for any given problem in the Linux universe. For this other project, I wanted to cobble together a version of grep that let me specify proper regular expressions without having to worry about the `-E` flag and get a context for the matches too.

These are both popular expansions to grep, of course: the former demonstrated by both grep `-E` and the egrep shortcut, while the latter task is done with grep `-C` and, on some UNIX and Linux systems, `wgrep`.

But, there are a lot of different ways to create that particular functionality that don't involve relying on a modern version of grep; older versions might have the `-E` flag, but don't include support for contextualization.

So in this article, I thought it would be interesting to look at different ways to produce what I shall call

wegrep, a version of grep that includes both the `-C` contextual window and the `-E` regular expression pattern support.

### Wrapper, Maybe We Just Need a Wrapper

If you have the modern GNU grep, which you can ascertain by simply trying to use the `-C` flag, this all becomes easy:

```
$ grep -C  
grep: option requires an argument -- C
```

There's a pretty gnarly usage statement after this, but if your version can understand the `-C` or its wordy sibling `-context`, you're in luck.

Enter a "wrapper", a simple script that changes the default behavior of a program. At its simplest, it actually can be a system alias, so this:

```
alias ls="/bin/ls -F"
```

is a sort of wrapper, ensuring that whenever I run the `ls` command, the `-F` flag is specified.

For this smarter version of grep, I simply could tell the user what flags to use or set specific flags with `GREP_OPTIONS`, an environment variable, but let's build out `wegrep`, as discussed.

For usage, it's going to be as simple as possible: command, pattern, source file. Like this:

```
wegrep '^Alice' wonderland.txt
```

This would search the file `wonderland.txt` for the regex "Alice", rooted to the beginning of a line.

Easily done:

```
grep=/usr/bin/grep  
if [ $# -ne 2 ] ; then  
    echo "Usage: wegrep [pattern] filename" ; exit 1  
fi  
$grep -C2 -n -E "$1" "$2"
```

I even added some error checking to ensure that the user specified the right number of parameters, with a simple error message to hide some of the complexity of the real grep command.

For a test file, I'm going to use the first four paragraphs of Lewis Carroll's immortal *Alice in Wonderland*, as downloaded from Project Gutenberg (<http://www.gutenberg.org>).

Here's the result of my first invocation:

```
$ sh wegrep '^Alice' wonderland.txt
11-Down the Rabbit-Hole
12-
13:Alice was beginning to get very tired of sitting by her
14-sister on the bank, and of having nothing to do: once
15-or twice she had peeped into the book her sister was
--
--
26-
27-There was nothing so very remarkable in that; nor did
28:Alice think it so very much out of the way to hear the
29-Rabbit say to itself, 'Oh dear! Oh dear! I shall be
30-late!' (when she thought it over afterwards, it
```

You can see that grep does a good job with this task, showing me two lines of context above and below each match, and denoting which line contains the match itself by having the : separate the line number from the content.

But what if your version of grep doesn't have support for the -C flag? What if you actually need to identify which lines match the pattern, then roll your own context display?

### Building Your Own Context

Since grep is still available, and all but the most ancient of grep implementations support the -E flag to allow the user to specify a regular expression, the task can be broken into two parts: identify which lines match, then figure out a way to list lines (n-2)...n...(n+2), as shown in the above output.

The first task can be done surprisingly easily because grep has a handy -n flag that appends line numbers. With that, getting a list of which lines match the specified pattern is straightforward.

But, let's see what's output first:

```
$ grep -n -E '^Alice' wonderland.txt
13:Alice was beginning to get very tired of sitting by her
28:Alice think it so very much out of the way to hear the
```

Now it's a job for Superman! I mean, um, cut:

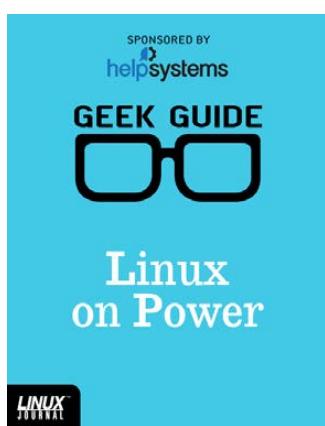
```
grep -n -E "$pattern" "$file" | \
cut -d: -f1
13
28
```

---

# Linux Journal eBook Series GEEK GUIDES

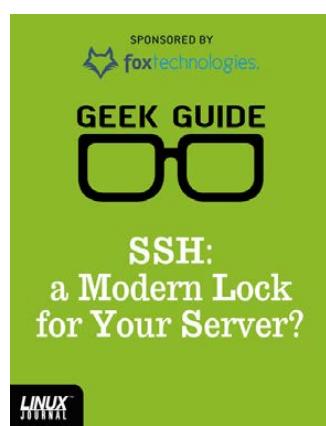


Practical books for the most technical people on the planet.



### Linux on Power

**Author:**  
Ted Schmidt  
**Sponsor:**  
HelpSystems



### SSH: a Modern Lock for Your Server?

**Author:**  
Federico Kereki  
**Sponsor:**  
Fox Technologies

Go to <http://geekguide.linuxjournal.com>

Let's switch to the other task of showing a range of lines centered on the specified line. You could do this with a tortured pairing of head and tail, but sed is a much better tool for the job this time.

In fact, sed makes it easy. Want to grab lines 12, 13 and 14? This'll do the trick:

```
sed '12,14p' wonderland.txt
```

Well, not quite. The problem is that the default behavior of sed is to echo every line it sees in addition to whatever the user specifies, so you'll end up with every line from wonderland.txt and additionally have lines 12–14 appear a second time as the statement is matched and executed (the p suffix means "print").

That's why if you're going to do anything with sed, it's critical to know its -n flag, which surpasses its desire to output every line it reads. Now here's a working command:

```
$ sed -n '12,14p' wonderland.txt
```

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once

Can you see how to chain these together? It all can be done in a simple for loop (particularly if you ignore error checking for now). But again, there's another small step required: the line count n prior and n subsequent to the matching line n need to be calculated. That's easy math:

```
before=$(( $match - $context ))
after=$(( $match + $context ))
```

Here context specifies whether you want 1, 2, 3 or more lines of context above and below the matching line.

Let's give this a whirl:

```
#!/bin/sh
```

## WORK THE SHELL

```
# wegrep - grep with context and regular expressions
grep=/usr/bin/grep
sed=/usr/bin/sed
if [ $# -ne 2 ] ; then
    echo "Usage: wegrep [pattern] filename" ; exit 1
fi
for match in $($grep -n -E "$1" "$2" | cut -d: -f1)
do
    before=$(( $match - $context ))
    after=$(( $match + $context ))
    $sed -n '${before}, ${after}p' "$2"
done
exit 0
```

Except it turns out that there are two critical bugs in the above code, as is immediately apparent when you run your first test:

```
$ sh wegrep '^Alice' wonderland.txt

wegrep: line 14: 13:Alice - : syntax error in expression
→(error token is ":Alice - ")
```

Can you see the first bug? Line 14 is the calculation for the variable `before`.

So what's wrong? You need to initialize `context` with a value, so the mathematical expression is essentially:

15 +

Which is correctly flagged as an error. Easily fixed.

The second bug is more subtle, however, but here's the clue when you run the script with `context` defined as 1 near the top of the script:

```
$ sh wegrep '^Alice' wonderland.txt
sed: 1: "${before}, ${after}p": unexpected EOF (pending }'s)
sed: 1: "${before}, ${after}p": unexpected EOF (pending }'s)
```

## Now can you see the error? It's a subtle and common problem in shell scripts: I'm using the wrong quotation marks.

That's definitely odd. It's `sed` that's complaining, but what's wrong with the line that invokes `sed`?

Let's have another look at that line:

```
$sed -n '${before},${after}p' "$2"
```

Now can you see the error? It's a subtle and common problem in shell scripts: I'm using the wrong quotation marks. Remember, in a shell script, single quotation marks prevent the interpretation of variables. Switch it to double quotation marks, and everything now works great:

```
$ sh wegrep '^Alice' wonderland.txt
```

```
Alice was beginning to get very tired of sitting by her  
sister on the bank, and of having nothing to do: once  
There was nothing so very remarkable in that; nor did  
Alice think it so very much out of the way to hear the  
Rabbit say to itself, 'Oh dear! Oh dear! I shall be
```

Now another problem rears its head: how do you differentiate between blocks that have matched? Easy, add `-----` before and after each match by adding a few echo statements to the for loop:

```
for match in $($grep -n -E "$1" "$2" | cut -d: -f1)  
do  
    before=$(( $match - $context ))  
    after=$(( $match + $context ))  
    echo "-----"
```

```
sed -n "${before},${after}p" "$2"
echo "----"
done
```

This works, but it's a bit clunky as output goes, although it pretty closely matches what modern grep does with the `-c` flag:

```
$ sh wegrep '^Alice' wonderland.txt
```

```
-----
```

```
Alice was beginning to get very tired of sitting by her
sister on the bank, and of having nothing to do: once
```

```
-----
```

```
-----
```

```
There was nothing so very remarkable in that; nor did
Alice think it so very much out of the way to hear the
Rabbit say to itself, 'Oh dear! Oh dear! I shall be
```

```
-----
```

As a purist, I'd much rather have one dashed line between output blocks, one before the first match and one after the last, with no doubling of lines.

That's not hard to do, and there's a second task of adding back line numbers and ideally denoting which line has the match to the regular expression. But I'm out of room, so those tasks will have to wait until next month.■

Send comments or feedback via  
<http://www.linuxjournal.com/contact>  
or to [ljeditor@linuxjournal.com](mailto:ljeditor@linuxjournal.com).

**RETURN TO CONTENTS**

# Secure Desktops with Qubes: Installation

A secure desktop starts with a secure installation.



KYLE RANKIN

---

Kyle Rankin is a Sr. Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *The Official Ubuntu Server Book*, *Knoppix Hacks* and *Ubuntu Hacks*. He is currently the president of the North Bay Linux Users' Group.

---

PREVIOUS

Dave Taylor's  
Work the Shell

NEXT

Shawn Power's  
The Open-Source  
Classroom



**THIS IS THE SECOND IN A MULTIPART SERIES ON THE QUBES OPERATING SYSTEM.** In my first article, I gave an overall introduction to Qubes and how it differs from most other desktop Linux distributions, namely in the way it focuses on compartmentalizing applications within different VMs to limit what attackers have access to in the event they compromise a VM. This allows you to use one VM for regular Web browsing, another for banking and a different one for storing your GPG keys and password manager. In this article, I follow up with a basic guide on how to

download and install Qubes, along with a general overview of the desktop and the various default VM types.

## Download and Verify the Qubes ISO

You can download the latest version of Qubes from <https://www.qubes-os.org/downloads>, and on that page, you will find links to download the ISO image for the installer as well as more detailed instructions on how to create a bootable USB disk with the Qubes ISO (currently the latest 3.1 ISO is larger than will fit on a standard DVD, so you will need to stick with a USB-based install for that version).

In addition to the ISO, you also should download the signature file and signing key files via their links on the same download page. The signature file is a GPG signature using the Qubes team's GPG signing key. This way, you can verify not only that the ISO wasn't damaged in transit, but also that someone in between you and the Qubes site didn't substitute a different ISO. Of course, an attacker that could replace the ISO also could replace the signing key, so it's important to download the signing key from different computers on different networks (ideally some not directly associated with you) and use a tool like sha256sum to compare the hashes of all the downloaded files. If all the hashes match, you can be reasonably sure you have the correct signing key, given how difficult it would be for an attacker to Man-in-the-Middle multiple computers and networks.

Once you have verified the signing key, you can import it into your GPG keyring with:

```
$ gpg --import qubes-master-signing-key.asc
```

Then you can use gpg to verify the ISO against the signature:

```
$ gpg -v --verify Qubes-R3.1-x86_64.iso.asc Qubes-R3.1-x86_64.iso
gpg: armor header: Version: GnuPG v1
gpg: Signature made Tue 08 Mar 2016 07:40:56 PM PST using RSA
      ↗key ID 03FA5082
gpg: using classic trust model
```

```
gpg: Good signature from "Qubes OS Release 3 Signing Key"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:           There is no indication that the signature belongs
gpg:           ↪to the owner.
Primary key fingerprint: C522 61BE 0A82 3221 D94C A1D1 CB11
                           ↪CA1D 03FA 5082
gpg: binary signature, digest algorithm SHA256
```

What you are looking for in the output is the line that says “Good signature” to prove the signature matches. If you see the warning like in the above output, that’s to be expected unless when you added the Qubes signing key to your keyring, you took the additional step to edit it and mark it as trusted.

## Install Qubes

The Qubes installation process is either pretty straightforward and simple or very difficult depending on your hardware. Due to a combination of the virtualization and other hardware support Qubes needs, it may not necessarily run on hardware that previously ran Linux. Qubes provides a hardware compatibility list on its site so you can get a sense of what hardware may work, and the Qubes site is starting to create a list of certified hardware with the Purism Librem 13 laptop as the first laptop officially certified to run Qubes.

Like most installers, you get the opportunity to partition your disk, and you either can accept the defaults or take a manual approach. Note that Qubes defaults to encrypting your disk, so you will need to have a separate /boot partition at the very least. Once the installer completes, you will be presented with a configuration wizard where you can choose a few more advanced options, such as whether to enable the sys-usb USB VM. This VM gets all of your USB PCI devices and acts as protection for the rest of the desktop from malicious USB devices. It’s still an experimental option with some advantages and disadvantages that I will cover in a future column. It’s off by default, so if you are unsure, just leave it unchecked during the install—you always can create it later.

The install also gives you the option of installing either KDE, XFCE or both. If you choose both, you can select which desktop environment you want to

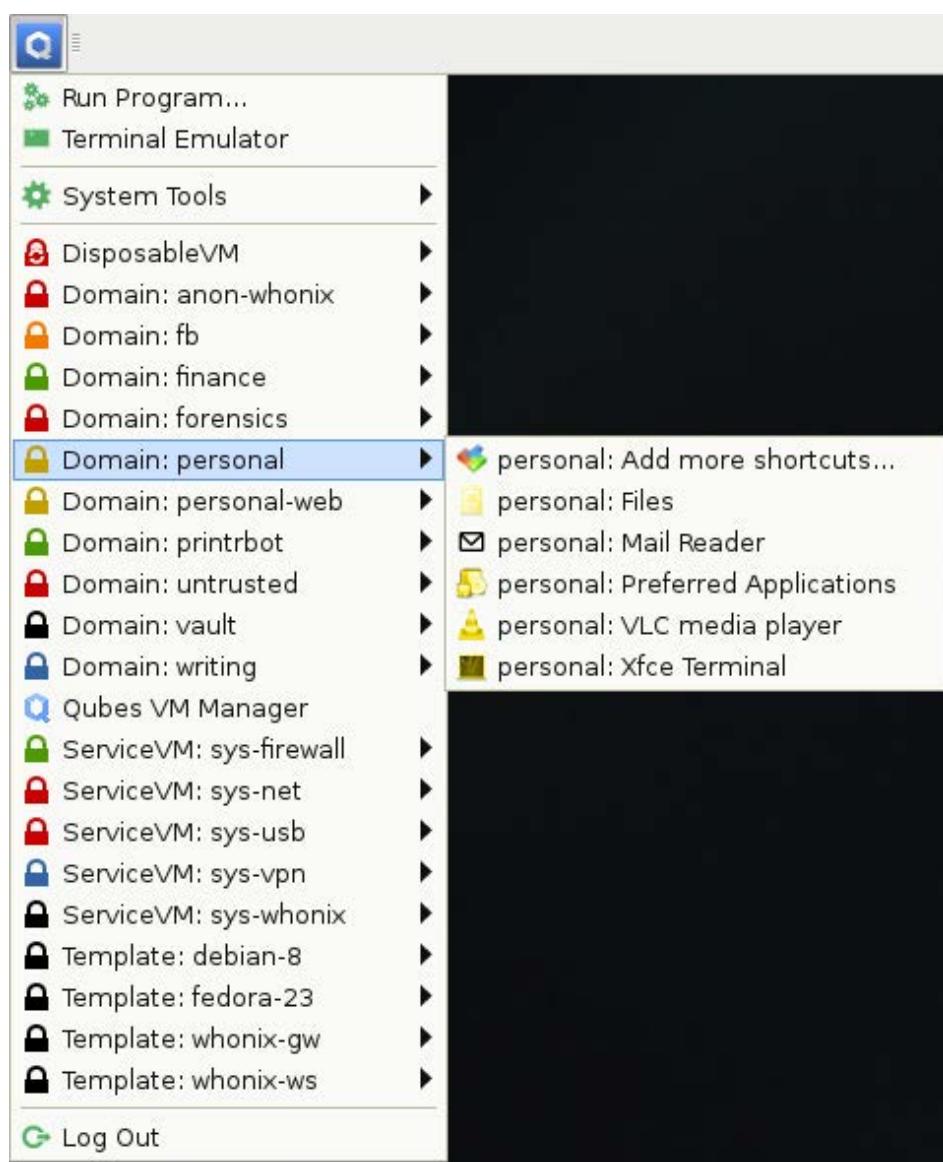
use at login as with any other Linux distribution. Given how cheap disk space is these days, I'd suggest just installing both so you have options.

## The Qubes Desktop

Whether you choose KDE or XFCE as your desktop environment, the general way that Qubes approaches desktop applications is the same, so instead of focusing on a particular desktop environment, I'm going to try to keep my descriptions relatively generic so that they apply to either KDE or XFCE.

The first thing you may notice is that instead of organizing applications into categories, the Qubes application menu is a list of different classes of VMs. Under

each of these VMs is a default set of applications, but note that it isn't a complete list of available applications—that would make the menu too unwieldy. Instead, you choose which applications you want to make available for each VM by selecting Add more shortcuts from that VM's submenu (Figure 1). This brings up a window that allows you to move



**Figure 1. An Example Qubes Desktop Menu**

application shortcuts over to the menu. Note that Qubes detects only applications that provide a .desktop link (the same way they automatically would show up in other desktop environments).

Qubes categorizes VMs in the desktop menu into groups based on their VM type. It's important to understand the purpose of each category, as it will help you make more secure decisions about what to do (and what not to do) in each type of VM. Here are the main categories:

- **Disposable VM:** these also are referred to as dispVMs and are designed for one-time use. When you launch a Disposable VM, it creates a brand-new VM instance based on a template and launches an application (usually a Web browser if launched from the menu, but it could be any application available within the VM's template). When you close that application, all data for that VM is erased. You can open multiple Disposable VMs at a time, and each is run within its own container. Disposable VMs are useful for opening risky e-mail attachments, browsing risky Web pages or any other activity where the chance of being compromised is high. If attackers do happen to compromise your Disposable VM, they had better act fast, because the entire environment will disappear once you close the window.
- **Domain VM:** these also often are referred to as appVMs, and they're the VMs where most applications are run and where users spend most of their time. When you want to segregate activities, you do so by creating different appVMs and assigning them different trust levels based on a range of colors from red (untrusted) to orange, yellow, green, blue, purple, grey and black (ultimately trusted). For instance, you may have a red untrusted appVM you use for general-purpose Web browsing, another yellow appVM for most trusted Web browsing that requires a login and still another even more trusted green appVM that you use just for banking. If you do both personal activities and work from the same laptop, appVMs provide a great way to keep work and personal files and activities completely separate.
- **Service VM:** service VMs are split into subcategories of netVMs and proxyVMs, and they're VMs that typically run in the background

and provide your appVMs with services (usually network access). For instance, the sys-net netVM is assigned all of your network PCI devices and is the untrusted VM that provides the rest with external network access. The sys-firewall proxyVM connects to sys-net, and other appVMs use it for network access. Because sys-firewall acts like a proxy, it allows you to create custom firewall rules for each appVM that connects to it, so you can, for example, create a banking VM that can access only port 443 on your bank's Web site. The sys-whonix proxyVM provides you with an integrated Tor router, so any appVMs you connect to it automatically route their traffic over Tor. You can configure which Service VM your appVM uses for its network (or if it has network access at all) through the Qubes VM Manager.

- **Template VM:** Qubes includes a couple different Linux distribution templates from which you can base the rest of the VMs. Other VMs get their root filesystem template from a Template VM, and once you shut the appVM off, any changes you may have made to that root filesystem are erased (only changes in /rw, /usr/local and /home persist). When you want to install or update an application, you turn on the corresponding Template VM, perform the installation or update, then turn it back off. Then the next time you reboot an appVM based on that template, it will get the new application. A compromise of a Template VM would mean a compromise of any appVMs based on that template, so generally speaking, you leave Template VMs off and turn them on only temporarily to update or install new software. You even can change which Template VM an appVM is based on after it is set up. Because only your personal settings persist anyway, think of it like installing a new Linux distribution but keeping your /home directory from the previous install.

## Installing Applications

Installing applications in Qubes is a bit different from a regular desktop Linux distribution because of the use of Template VMs. Let's say that you want to install GIMP on your personal appVM. Although you could install the software directly inside the appVM with yum, dnf or apt-get, depending on the distribution the appVM uses, that application would

last only until you turn the personal appVM off (and it wouldn't show up in the desktop menu). To make applications persist, you just identify the Template VM the appVM is based from, and then in the desktop menu, you can select the debian-8: Packages or fedora-23: Software option from the menu to start the VM and launch a GUI application to install new software. Alternatively, you also can just open a terminal application from the corresponding template VM and use yum, dnf or apt-get to install the software.

Once you install an application, if it provides a .desktop shortcut and installs it in the standard places, Qubes automatically will pick it up and add it to the list of available applications for your appVM. That doesn't automatically make it visible in the menu though. To add it to the list of visible applications, you have to select the Add More Shortcuts option from within that appVMs menu and drag it over to the list of visible applications. Otherwise, you always can just open a terminal within the appVM and launch it that way.

## The Qubes VM Manager

The Qubes VM Manager provides a nice graphical interface for managing all of the VMs inside Qubes. The primary window shows a list of all running VMs, including the CPU, RAM and disk usage, the color you've assigned them, and the template from which they are based. There is a list

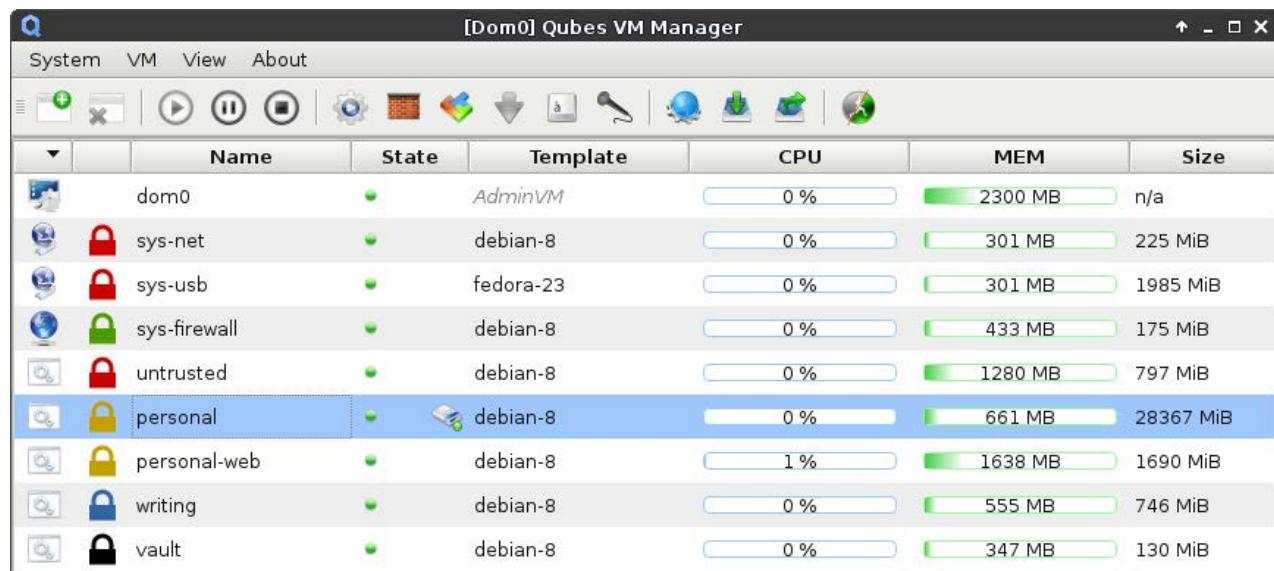


Figure 2. The Qubes VM Manager with Some Running VMs

of buttons along the top that let you perform various operations against a VM you've selected, including creating a new VM or removing an existing one, powering a VM on or off, changing its settings and toggling the list to show only running VMs or all of your VMs.

You can potentially tweak a lot of different settings with a VM, but the VM manager makes creating new VMs or changing normal settings relatively simple and organized. Some of the main settings you may want to tweak include the color to assign the VM, how much RAM or disk it can have at maximum, what template it uses and to which netVM it is connected. In addition, you can set up custom firewall rules for your VM, assign PCI devices to it and configure your application shortcut menu.

The VM manager is one of the nice points that makes it easier to navigate around what otherwise would be a pretty complicated system of command-line commands and configuration files. That, combined with some of the other Qubes tools like its copy-and-paste method (Ctrl-Shift-c to move from an appVM's clipboard to the global clipboard, highlight the appVM to paste into, then Ctrl-Shift-v to move it to that appVM's clipboard) and its command-line and GUI file manager tools that let you copy files between appVMs, all make an environment that's much easier to use than you might expect given the complexity.

Although this article will get you started with Qubes, where things really get interesting is in how you organize appVMs and divide tasks between them. In my next article, I will go over some of my own approaches to compartmentalization both on my personal and work laptops, and I'll highlight some additional Qubes security features that make it hard for me to go back to a regular old Linux desktop. ■

Send comments or feedback via  
<http://www.linuxjournal.com/contact>  
or to [leditor@linuxjournal.com](mailto:leditor@linuxjournal.com).

**RETURN TO CONTENTS**

# The Peculiar Case of E-mail in the Cloud

Configure your server to use your Gmail account.



**SHAWN POWERS**

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at [shawn@linuxjournal.com](mailto:shawn@linuxjournal.com). Or, swing by the #linuxjournal IRC channel on Freenode.net.

PREVIOUS  
Kyle Rankin's  
Hack and /

NEXT  
Susan Sons'  
Under the Sink

**MOST OF THE TIME WHEN I START A PROJECT**, or spin up a virtual server, it's done in my own basement "server farm". Not too many years ago, if I wanted those services to be public, I'd simply port-forward from my static IP into my personal machines. Or, perhaps I'd set up a name-based virtual host as a reverse proxy if I needed to expose a Web app. The nice thing about hosting projects locally is that when I need to send e-mail messages (usually error notifications), I simply can send them through my ISP's SMTP server. Granted, that's gotten a little more complex through the years, as ISPs are starting to lock down their mail servers and relay mail only for valid domains, but that just means I have to register my

static IP properly. It works nicely.

The problem is, the issues I've had with my office Internet connection during the past year really have forced me to reconsider how I host public-facing services. I've been forced, by necessity, to spin up cloud instances and host my numerous projects remotely. I'm actually rather thankful for the need, because although it's not free to host projects remotely, it's fairly inexpensive and much more convenient. I still have my Raspberry Pi colocated in Austria for free (thanks again, Kyle Rankin, for pointing me to that awesome service!). Unfortunately, the Raspberry Pi isn't powerful enough for many of the crazy things I try on-line. It struggles, for instance, to host MySQL. So my main "project" server is a Google Compute instance that I end up paying about \$15/month to keep active. That's not cheap, but I actually think I might be able to turn off one of my ESXi machines at home now, and I suspect it uses more than that in electricity.

### **The Problem**

The problem with Internet-hosted servers is that the lack of a usable SMTP relay makes e-mail very difficult. Yes, it's possible to install Postfix as a full-blown e-mail server, but I have no desire to worry about securing my own e-mail server from attacks attempting to use me as a SPAM relay. And although installing a non-relaying e-mail server certainly is possible, I've found that unless you configure SPF records, MX records and get particularly lucky, e-mail sent from a cloud instance often never arrives at the destination. This is especially true if you spin up servers on the fly.

The truth of the matter is, the only reason I want e-mail in the first place is so I can get notifications from my servers when something goes wrong. I don't ever need to reply to the e-mails. I don't really care where the e-mails come from (address-wise). I just want to have confidence that my notifications will get to me!

### **The Solution**

If you install Postfix on your server, it's possible to use a Gmail account to send all e-mail on your system. There are a few downsides to this method, but the configuration is simple, and Google's e-mail servers are very reliable. Plus, because you're not acting as an e-mail server yourself, you

don't have to worry about having your e-mail rejected by recipients. It's legitimately coming from gmail.com.

The first unfortunate consequence is that for its simplest implementation, you need to enable "less secure apps" to log in to your Gmail account. I actually set up a separate gmail.com account for my server, and then I don't worry about the less secure setting. Thankfully, if this is a concern, it's possible to use two-factor authentication (more on that later).

Second, if you use Gmail as your e-mail relay, every e-mail will be rewritten to come "from" the gmail.com address. For me, this is a non-issue, because I just want my servers to e-mail me reliably when things go wrong. So although this won't be an issue for many servers, it's certainly not a feasible way to provide multi-user e-mail on your server. You can send from multiple users on your Linux system, but every e-mail that is sent will have its headers rewritten so they come from the same address! Thankfully, only the "from" address itself is rewritten, so messages come from an address formed like "User 1 <user@gmail.com>" and "User 2 <user@gmail.com>". So even though the underlying addresses are different, you still can tell from which user the e-mail messages are coming. This is useful for me, as it helps determine which app is sending me failure information!

## The Procedure

First, you need to install Postfix along with the tools needed for enabling SASL connections. Your procedure will vary based on distribution, but for Ubuntu/Debian folks, it will go something like this:

```
sudo apt-get install postfix mailutils libsasl2-2  
➥ca-certificates libsasl2-modules
```

When Postfix installs, it will ask what type of system you're configuring. Multiple options will work, since you're editing the main file afterward, but I recommend you choose "Internet Site" and answer the questions accordingly. (Again, don't worry too much about what answers you put in the setup dialog, most of it will get overridden by your modifications anyway.)

Next, edit the main.cf file:

```
sudo nano /etc/postfix/main.cf
```

Then, change or add the following stanza of information somewhere in the file. Pay close attention, because there will be a few lines that look similar, but are subtly different. You'll probably have to add all the lines below:

```
relayhost = [smtp.gmail.com]:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/postfix/cacert.pem
smtp_use_tls = yes
```

Now you need to create that cacert.pem file. You could just reference the original file directly, but I like to have all the required files in one folder—that makes it easier to replicate when spinning up new servers:

```
sudo cat /etc/ssl/certs/Thawte_Premium_Server_CA.pem >
➥/etc/postfix/cacert.pem
```

In order to send mail, you need to have your authentication information on the server. Create the file from scratch:

```
sudo nano /etc/postfix/sasl/sasl_passwd
```

Enter your Gmail account information. It feels wrong to type a user name and password into a file, but you're going to lock the file's permissions pretty tight in the next step. Use this format in the file:

```
[smtp.gmail.com]:587 USER@gmail.com:PASSWORD
```

The **USER** and **PASSWORD** obviously need to be substituted with your account credentials. You also can use a Google Hosted Domain account,

just use the full e-mail address instead of @gmail.com. Then secure the file and create a hash database so Postfix can read it properly:

```
sudo chmod 400 /etc/postfix/sasl/sasl_passwd  
sudo postmap /etc/postfix/sasl/sasl_passwd
```

Finally, reload Postfix and test outgoing e-mail:

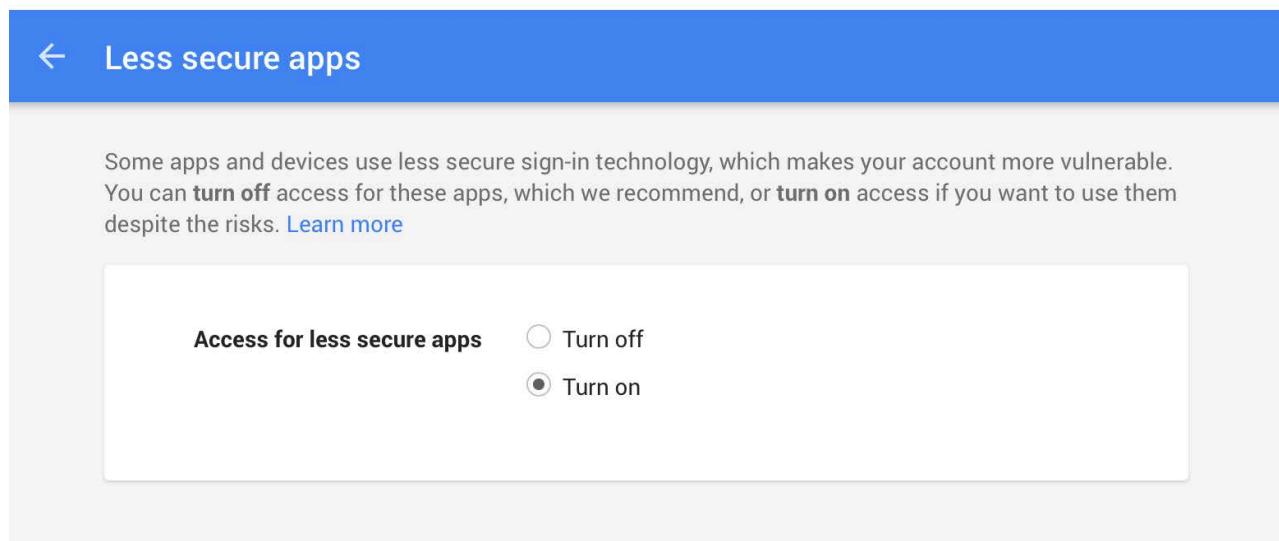
```
sudo service postfix reload  
echo "It Worked" | mail -s "Email Test" anotheruser@example.com
```

## Troubleshooting

It's very possible the e-mail will fail. If you get an error like this in your log files:

```
SASL authentication failed; server smtp.gmail.com[1.2.3.4]  
→said: 534-5.7.14 Please log in via your web browser  
→and then try again.
```

the most probable reason is that secure login setting I mentioned earlier. Like I mentioned previously, I don't have a problem with doing this on an account I created specifically for relaying e-mail. If you're



**Figure 1. It seems odd to “turn on” insecurity.**

using your actual e-mail address though, I don't really recommend it.

If you're still interested in softening the security to fix this problem, log in to your Gmail account and head to: <https://www.google.com/settings/security/lesssecureapps> (Figure 1).

You should be able to switch this to "turn on", which turns on the ability for less secure apps to log in. (That sounds a bit counterintuitive, turning "on" a more insecure method, but if you read closely, that's what you want to do.)

Once you make that change, try sending an e-mail again, and it should go through. The original e-mail actually probably will go through too, since Postfix keeps trying to send failed messages.

### If You Prefer More Security

I've had mixed results using a Google App Password and two-factor authentication for Postfix e-mail relay setups. I'll leave this as an exercise for those folks who don't want to allow less secure authentication or who already use two-factor authentication on their accounts. (This might be the only option for Google Hosted Domain users whose administrators have not enabled the "less secure app" feature.)

The first step is to turn on two-factor authentication on your account. Otherwise, you won't be able to generate App Passwords. Head over to <http://accounts.google.com/SmsAuthConfig> in order to enable and configure two-factor authentication.

Then, create an App Password for your server setup here:

<https://security.google.com/settings/security/app passwords>.

Once you have the App Password, copy it into your Postfix authentication file in place of the password entered earlier. You'll need to re-create the password map too using the `postmap` command. Then restart Postfix, and try sending an e-mail. If it doesn't work, check your log files and start troubleshooting there. Like I said, I've had mixed results.

### Not Just Gmail!

If you are struggling with your Gmail account, or just prefer not to rely on Google for relaying your information, the good news is

this procedure works for any SMTP server. In fact, the configuration might be far simpler for other SMTP servers. If you have an e-mail account from your ISP, you likely can use that account by tweaking the settings above to match your ISP's account information. I think I have a charter.net e-mail address that I've never used for anything. I suspect many folks have similar addresses.

E-mail might be a dying form of communication, but for things like server notifications, it's hard to beat. The problem is, there are so many security concerns over relaying e-mail, it can be frustratingly difficult to configure one of the oldest messaging protocols!

Usually when I'm setting up a new server, I quickly install Postfix and configure it like this using a Chef or Puppet method for quick and reliable configuration. If you have a simpler or different method for enabling e-mail on cloud servers, drop me an e-mail at [shawn@linuxjournal.com](mailto:shawn@linuxjournal.com). I love hearing other solutions, and I'll share any really great solutions with the rest of the class in a future issue! ■

Send comments or feedback via  
<http://www.linuxjournal.com/contact>  
or to [ljeditor@linuxjournal.com](mailto:ljeditor@linuxjournal.com).

**RETURN TO CONTENTS**

O'REILLY®

# OSCON

OPEN SOURCE CONVENTION

May 16-19, 2016  
Austin, TX

The original (also the biggest, baddest & broadest) open source gathering comes to Austin.



Save 20%  
Register today.

Use code **PCLinuxJournal**

# Securing the Programmer, Part I



SUSAN SONS

In a previous column, “Chain of Custody”, I wrote about some of the ways a piece of software could be compromised between the developer and the consumer. Here, I describe how developers can improve their practices in administering their development workstations to prevent compromise at the source.

---

PREVIOUS

Shawn Power's  
The Open-Source  
Classroom



NEXT  
New Products



Susan Sons serves as a Senior Systems Analyst at Indiana University’s Center for Applied Cybersecurity Research (<http://cacr.iu.edu>), where she divides her time between helping NSF-funded science and infrastructure projects improve their security, helping secure a DHS-funded static analysis project, and various attempts to save the world from poor information security practices in general. Susan also volunteers as Director of the Internet Civil Engineering Institute (<http://icei.org>), a nonprofit dedicated to supporting and securing the common software infrastructure on which we all depend. In her free time, she raises an amazing mini-hacker, writes, codes, researches, practices martial arts, lifts heavy things and volunteers as a search-and-rescue and disaster relief worker.

**I HAVE A FAVORITE SAYING:** “If you are a systems administrator, you have the keys to the kingdom. If you are an open-source programmer, you don’t know which or how many kingdoms you have the keys to.” We send our programs out into the world to be run by anyone for any purpose. Think about that: by anyone, for any purpose. Your code might

be running in a nuclear reactor right now, or on a missile system or on a medical device, and *no one told you*. This is not conjecture; this is everyday reality. Case in point: the US Army installed gpsd on all armor (tanks, armored personnel carriers and up-armored Humvees) *without telling its developers* (<http://esr.ibiblio.org/?p=3818>).

This two-part series focuses on the needs of infrastructure software developers—that is, developers of anything that runs as root, has a security function, keeps the Internet as a whole working or is life-critical. Of course, one never knows where one’s software will be run or under what circumstances, so feel free to follow this advice even if all you maintain is a toddler login manager.

Part I covers basic security concepts and hygiene: how to think about security needs and how to keep your development system in good shape to reduce the risk of major computing security mishaps. Part II, in a future issue of *Linux Journal*, will take things a step further and discuss what it takes to improve the security culture and practices of the projects you develop for.

This guide isn’t going to teach you everything about security. It will give you an idea of what to do, but in many cases, you’ll need to rely on man pages and other documentation to get the “how”. I did that both for brevity and to ensure that this article covers various Linux distributions equally and without becoming out of date in a matter of weeks.

I chose the controls here carefully. It is the set of controls that is consistently available across Linux distributions, realistic for developers to maintain even if they are developing open-source software as a side project and can’t put many hours into it. It’s maintainable without extensive training and has highest impact for the security of the software being developed. All of those things are judgment calls, and I welcome debate about them. The goal of this guide is not “ultimate security” or the fabled “uncrackable system”. It is to raise the bar for security hygiene among open-source infrastructure software developers significantly from where it is right now.

I’d love to find that, in a year from now, we’re all much more secure and can iterate on our standards again. In my perfect world, I write this article every spring, we all up our game a notch, and the following spring, we are prepared to make the jobs of ransomware developers,

spammers, oppressive governments, corporate spies and so on even harder than before.

## Concepts

I know programmers—being one myself—and programmers don't just want to be told that something works, we want to know why and how it works. Thus, before I introduce a checklist for the open-source developer, I'm beginning with some underlying security concepts.

**CIA and Software Development:** No, not *that* CIA. Confidentiality, Integrity and Availability: these are the three goals of security. In the Open Source world, we usually are most concerned with integrity: is this the software the developer I trust made, and can I be sure it hasn't been tampered with? Availability typically comes second: can I get a copy of this software, and its documentation, when I need it? Confidentiality usually comes in last, as it applies only to a few parts of our practice: private keys and other credentials, vulnerabilities we still are working to patch and some sensitive intra-project communications. Even so, it is rare that those things need remain confidential indefinitely.

**Risk:** “*A ship is safe in harbor, but that's not what ships are for.*”  
—William G.T. Shedd

I can make my laptop perfectly secure. I can do this by removing its battery, filling its ports with epoxy, tying it to some bricks, and then dropping it to the bottom of the Marianas Trench. There, in the deepest ocean chasm, it will be awfully hard to get to, and anyone who tries will find a hunk of pulverized metal and plastic that couldn't take the pressure or the salt water.

Of course, at that point, what good does the laptop do me or anyone else?

Computing involves risk. It always has involved risk, but never more so than now, when we are constantly connected and running systems so complex as to be virtually un-auditable. It is rare for me to work on a machine so small in scope that I could audit its every line of code in a lifetime, let alone before the next kernel patch comes out. When I do see such a machine, it is invariably a single-purpose, life-critical component maintained at great expense.

Still, the world doesn't seem to be ending (yet). This is because we do have the power to mitigate risk to manageable levels:

- We can render a risk irrelevant (if I don't store credit-card numbers, I'm

not at risk of a credit-card database breach).

- We can transfer a risk to someone else (if I insure my laptop, the insurance company pays if it is stolen, not me).
- We can lower the likelihood of a risk (if I never transmit or store passwords in the clear, it is less likely that they will be compromised).
- We can lower the impact of a risk (if I use two-factor authentication, compromising a password alone does nothing).

## Controls

A control is something that mitigates risk in information security.

**Full-Disk Encryption:** “Full-disk encryption”, often abbreviated FDE, is something of a misnomer. It really means full *partition* encryption in most cases. Many distributions, such as Red Hat and Ubuntu, offer full-disk encryption as a check-box option at install time. Some others, such as Slackware and Gentoo, require some manual intervention in preparing the disks, but there is reasonably good documentation available.

Now that you know how easy it is, let’s talk about why you do it: encrypting all of your storage (including swap!) protects you against theft of your powered-down or hibernated computer. If you are not using FDE, and your machine is lost or stolen, not only is your personal information compromised, but so are your code-signing keys, the SSH keys you use to check in code and access servers, and any information you may have on not-yet-patched vulnerabilities. An attacker could release patches that look as if they came from you, and most likely no one would be the wiser.

Off-line attacks on the passphrases of private keys can and do happen. Full-disk encryption makes it unlikely that the keys can be recovered and almost certain that you will have had time to revoke them in the meantime. It also gives your team plenty of time to patch and publish vulnerabilities before a thief can make use of any information that may have been on your machine.

**Operating System:** Quite simply: if you need Windows, run it in a VM or on another machine. *Do not* dual-boot on your development machine. Windows is generally prone to collecting malware, and in a scenario

## How you handle your private encryption keys is paramount to their usefulness.

where the machine boots directly to Windows (as opposed to running it in a virtual machine), Windows may have the opportunity to overwrite your motherboard's firmware with malicious software that would then impact your Linux system.

This isn't to say that Windows cannot be reasonably secured, but it's a large, hard-to-manage attack surface, especially for those of us who specialize in Linux and may keep a Windows system around only for something like gaming purposes.

**Password Management:** Pick a decent password manager, and use it. Recycling passwords is not okay. Using weak passwords is not okay. No one can remember a large assortment of strong passwords. Save your brain by memorizing only what you have to.

Many Linux users ask me if a password manager is really safe. What if my password is held in RAM? What if my laptop is stolen? Full-disk encryption will protect your machine's contents, including your password management database, and most password management software has a layer of encryption of its own. I'm not talking about perfect security (and you may want to keep your 2–5 most valuable passwords in your head). I'm talking about making decent security manageable. A weak password out in the wild is far more open to attack than a password management database behind a password on an encrypted hard drive on your laptop.

I could write an article entirely about the pros and cons of various password managers, but the short version is this: *any* password manager that doesn't upload all your credentials to the cloud is better than not using a password manager.

**Key Management:** How you handle your private encryption keys is paramount to their usefulness. If attackers get a copy of one of your keys—especially if two-factor authentication is not in play—they easily can brute-force its password off-line and use it to impersonate you. The NSA's or China's or EvilCorp's latest back door could go out under

your signature. So, do the following:

- Never store private keys on a system that does not have full-disk encryption.
- Avoid creating passwordless private keys unless you understand the implications of doing so, and have another protection in place to prevent their abuse, such as encrypting the key with a key stored on a separate hardware token.
- Keep a log of which keys you have in use where, so that if a key is compromised, you can ensure that it is revoked and replaced with a new key.
- Use a different key on each of your machine. If one of your keys is compromised, once it is used, you know from which key it was which of your machines is in question. If all machines use the same key, you likely will have no idea where the compromise came from. Having multiple keys also is helpful when you are having someone else revoke a key for you. If I find that my laptop's key may be compromised, but I'm away at a conference, I can make a couple phone calls to get that key revoked in the two or three places where it could cause the most damage. Then, when I get home, I can log in via my desktop (which has its own key) to place a new public key for my laptop in the appropriate places.
- For GPG keys and other systems that allow it, create a revocation certificate for each key and have a friend store those certificates in case of emergency. This way, even if you have a major compromise or lose access to your private key, your friend can mark that key as compromised and no longer valid. This does not give your friend the ability to impersonate you, only to revoke your keys.

**Backups:** Your backups should be protected as well as you protect your primary systems, otherwise they are as much a liability as they are an asset. Backups always should be encrypted, and it is especially important that if you back up to a cloud service, you have configured your backup system such that the relevant encryption keys reside with you locally and

never are shared with the service storing the encrypted data. Be sure to keep a backup of your keys or passwords for your backup data store somewhere safe and separate from that data store, such as in a fireproof safe, if they are not memorized.

**Multifactor Authentication:** Quite simply, multifactor authentication is your friend. Use it wherever it is available. In most cases, this is either a password or an SSH key combined with a second authentication factor, such as a hardware token, a soft token on your phone that provides a time-sensitive shortcode or the service's ability to send you an SMS or other out-of-band confirmation.

Passwords are fairly easy to compromise. SSH keys are less so, but at this point, two-factor (or more) authentication has become so easy and accessible, there is no excuse not to use it.

GitHub, Google and many of the other services we use regularly offer two-factor authentication to help protect our accounts, and several different options exist for employing it on our own infrastructure as well.

**Configuration Management:** If you have a complex enough system that the overhead is worth it, consider a configuration management system, such as Ansible or Puppet. For most of us, with respect to our individual development machines, this will not be the case. A much simpler and lighter solution is to install and enable etckeeper, which will keep a revision history of your system configuration in a git or hg repository, automatically updating it on package manager events. You can trigger updates manually when editing configuration files yourself.

Although etckeeper doesn't give you the centralized management features of a true configuration management system, when combined with good backups, it provides the feature most important to the security of a single-machine setup: auditability of configuration. That audit trail can be invaluable when things go wrong.

**Updates:** Keeping up with security updates should be a no-brainer, but many developers simply become lazy or avoid updates because they are afraid they'll have to resolve some new conflict or failure as a result.

However, doing critical development work on a machine that is a week or more behind on security patches is simply taking a massive risk on behalf of every user of your software for the sake of some convenience. You have a package manager; use it.

**Firewall:** If your machine is a desktop that will live behind a hardware firewall, you may not need to run a firewall on the machine itself. However, if you do not have a hardware firewall protecting your machine, or if it is mobile (a laptop you travel with, or a desktop if you ever bring it to hackfests or LAN parties), it needs to be firewalled.

Almost every Linux firewall is a wrapper around iptables, which is good, because iptables is fast, powerful and reliable. Which wrapper you use doesn't matter terribly much. I have been using ipkungfu for a while now to avoid having to hand-compose iptables rules on my laptop, but other options are just as good.

**SSH:** You may want to ssh between your development machines for a variety of reasons: to transfer files, to check just one setting or to use a build environment home from a laptop abroad. There are many good guides to running SSH servers, but here are some general tips for your first-pass check over sshd configuration:

- Never run sshd on your development machine unless you really need to and don't have it start on boot. A development machine is a machine you are usually sitting at, so running sshd when not in use is a needless risk, and it's easy to start it when it is needed.
- Never allow ssh in as root. If you need root access, you can ssh in as a regular user and then use su or sudo to gain root privileges. This helps to ensure that a single compromised key or password will not give an attacker root privileges.
- Never allow password-only SSH. Either require key-based authorization or a password plus a second factor, such as a one-time password from a soft token application.
- Check logs or run a monitoring script to stay informed about attempts to brute-force your sshd.

**Isolating Users and Services:** One simple way to increase your system's security is to isolate various services from one another using system accounts. Some distributions do this by default most of the time,

# OSS Developer Security Checklist

- Full Disk Encryption on workstation(s) and backups (including swap).
- No Windows dual-boot (VMs are okay).
- Using a password manager.
- Never recycle passwords/passphrases.
- Use reasonably strong passwords/ passphrases for the context.
- No keys stored on unencrypted media.
- No keys stored passwordless (see article for exceptions).
- Private encryption keys never entrusted to anyone else (double-check cloud backup systems).
- Revocation certificates for all keys stored with a trusted friend.
- 2FA is used wherever practical.
- 2FA tokens never stored on same device used to log in or store primary credential.
- Configuration management (at least ex-post-facto change recording) employed on all systems.
- root user or a user with unlimited no-password sudo privileges is not used for day-to-day computing, coding and so on.
- Local services run for convenience/ testing, such as a Web server, gitolite and so on, run as their own user account rather than as root and have access only to files in their own directory, which is clearly documented.
- No remotely accessible services are autorun on boot; remote services run only when needed.
- No FTP dæmon is running on workstation(s).
- Password auth for SSH is disabled. Only key-based or multifactor auth allowed.
- SSH as root is disabled.
- Workstation screen is locked every time I walk away, even for a minute.
- Application firewall (probably iptables) configured and running on workstation(s).
- All packages checked for updates multiple times/week.
- Adobe Flash not installed (or disabled in all browsers).
- Lightning and FireWire ports disabled in BIOS/UEFI firmware settings.

but you should check that your machine is doing it wherever practical:

- Do not use root, or a user that can exercise root privileges without a password (for example, via passwordless sudo) for everyday tasks. Always use the least system access necessary for any given task.

- Do not run every service on your machine as root. In general, never run anything as root unless you must. Developers frequently run local instances of a number of services for testing and development purposes. These should be isolated to their own runs-as users to help contain any exposure that one service causes. Apache may run as “apache” or “www-data”; a git server may run as “git” or “gitolite”; a mail service may run as “mail”. Whatever the names are, it is important that services are separated.
- Never allow two users to share the same system account. System accounts are free; make more of them as needed.

### What Not to Use on a Development Machine:

- If you must run Adobe Flash (try not to run it at all), enable it in only one browser, and do everything other than the one thing you tolerate Flash for in a different browser. Better yet: run Flash only within a VM dedicated to that purpose, if you can. Flash is closed-source, riddled with holes and simply can’t be secured.
- Do not run an FTP service. FTP is incredibly insecure, as is FTPS. This is not to be confused with SFTP, which serves a similar function but operates over the more trustworthy SSH protocol.
- Disable all FireWire and Lightning ports on your machine, in the BIOS or UEFI firmware if possible, otherwise by physically disconnecting them or filling them with epoxy. FireWire and Lightning use direct memory access, meaning that an attacker connecting to one of these ports while you are at a conference and looking away for a moment (even with your computer’s screen locked) can dump the contents of your RAM (or change it) as long as your machine is powered up.■

Send comments or feedback via  
<http://www.linuxjournal.com/contact>  
or to [ljeditor@linuxjournal.com](mailto:ljeditor@linuxjournal.com).

**RETURN TO CONTENTS**

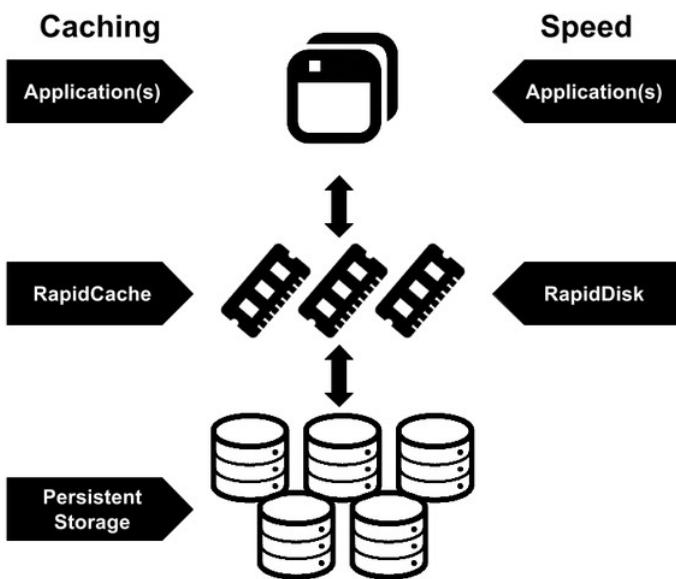
# NEW PRODUCTS



PREVIOUS  
Susan Sons'  
Under the Sink



NEXT  
Feature:  
YubiKey 4



## Petros Koutoupis' RapidDisk

RapidDisk is an open-source and enhanced Linux RAM drive solution led by BDFL Petros Koutoupis that allows users to create, resize and remove RAM drives dynamically or map those same RAM drives as a cache to slower data volumes. The latest version 4.0 release adds a series of complementary improvements, such as kernel module optimizations, code cleanup/redesign and bug fixes. RapidDisk consists of a collection of kernel modules, an administration utility, high-availability scripts and a RESTful API for third-party integration. By design, RapidDisk volumes are thinly provisioned and will allocate memory only upon usage.

<http://rapiddisk.org>



## The Qt Company's Qt Start-Up

The Qt Company is proud to offer a new version of the Qt for Application Development package called Qt Start-Up, the company's C++-based framework of libraries and tools that enables the development of powerful, interactive and cross-platform applications and devices. Now used by around one million developers worldwide, the Qt Company seeks to expand its user base by targeting smaller enterprises. The new Qt Start-Up, available to companies with an annual revenue of less than \$100,000, enables small and start-up companies to harness the full power of the Qt application and UI development framework in their products. The Qt Start-Up package is just as powerful as the regular Qt for Application Development package, but it's available at a much lower price. The Qt Company's rationale for Qt Start-Up is to enable a new generation of innovators and help them build successful businesses assisted by the powerful application development and UI creation tools within the Qt framework.

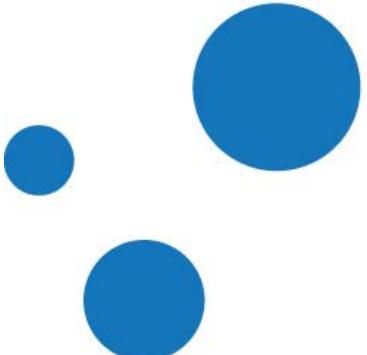
<http://qt.io>



## Linaro's ARM-Based Developer Cloud

As the adoption of ARM-based servers accelerates and IoT applications rapidly evolve, software developers are demanding access to requisite hardware and software-reference platforms. In response, Linaro released Linaro Developer Cloud, a new cloud-based native ARMv8 development environment, which can be used to design, develop, port and test server, cloud and IoT applications without substantial upfront hardware investment. The Developer Cloud is the combination of ARM-based silicon vendors' server hardware platforms, emerging cloud technologies and many Linaro member-driven projects, including server-class boot architecture, kernel and virtualization. The Developer Cloud is based on OpenStack, leveraging both Debian and CentOS, as the underlying cloud OS infrastructure. It will use ARM-based server platforms from Linaro members AMD, Cavium, Huawei and Qualcomm Technologies, Inc., and will expand with demand and as new server platforms come to market.

<http://linaro.org>

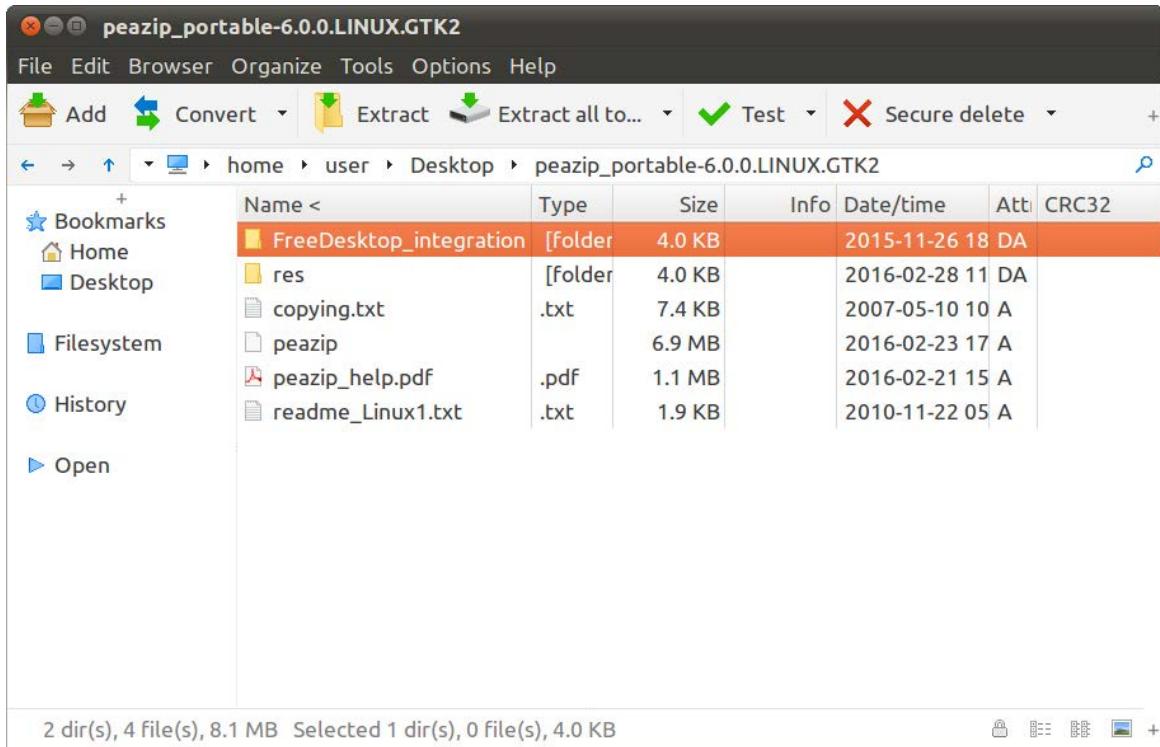


# VARNISH SOFTWARE

## Varnish Software's Varnish Massive Storage Engine

The headlining feature of the new Varnish Massive Storage Engine (MSE) 2.0 from Varnish Software is cache persistence. This new capability in MSE, an exclusive module of Varnish Plus Web optimization suite, allows Web sites to retain data across restarts and reboots and ensures that, in the case of a system crash, cache content will not be lost. Furthermore, users can repair and maintain their sites as quickly as possible. Varnish MSE was designed specifically for the high-performance needs of video distribution, content delivery networks (CDNs) and large-cache use cases that require enabling of the Varnish caching layer to handle multi-terabyte data sets. For big CDNs, an empty cache is a no-go, argues Varnish, and reintroducing large cache objects, of the size for which MSE is required, is time-consuming and expensive. With Varnish MSE 2.0, all objects remain in cache with minimal performance reduction.

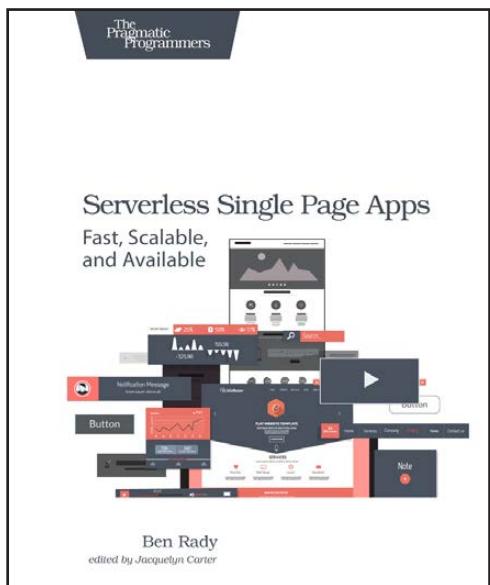
<http://varnish-software.com>



## PeaZip

Free of charge for any use and free of any kind of advertising bundle, PeaZip is an open-source (LGPL) file archiver, a free alternative to software like WinRar and WinZip, for Linux and Windows. Version 6.0.0 is a new major release of PeaZip that adds new features to the existing functionality. Innovations include ability to open/extract in the new RAR5 format, full support for Unicode filenames, quick search and new themes to customize PeaZip's look and feel. PeaZip now supports more than 180 archive types (including 7Z, RAR, TAR, ZIP, ZIPX and so on) and provides useful file management features, such as encryption, file split, verify hash and secure delete. PeaZip adds that, unlike most other classic file archivers like WinZip and WinRar, its archiver is natively portable and cross-platform/multiplatform software that is available for computers or tablets running 32-bit and 64-bit Windows (9x, NT/2K/XP, Vista/7/8/10, ReactOS, Wine), Linux or BSD x86 and x86-64.

<http://peazip.org>



## Ben Rady's *Serverless Single Page Apps* (The Pragmatic Programmers)

You don't need to manage your own servers to build powerful Web applications. Need proof? Pick up tech author Ben Rady's new book *Serverless Single Page Apps: Fast, Scalable, and Available*, a guide to creating single-page apps that run entirely on Web services, scale to millions of users and cost amazingly little. Readers of Rady's book will skip over building an application server, avoid messing around with middle-tier infrastructure and get right to the Web app their customers want. Using a Web browser, a prepared workspace and an editor, readers learn the fundamental technologies behind modern single-page apps and use Web standards to create lean Web applications that can take advantage of the newest technologies. They'll also deploy the application quickly using Amazon S3 and utilize Amazon Cognito to connect with providers like Google and Facebook to manage user identities. Other topics include DynamoDB for reading and writing user data directly from the browser and Amazon Lambda for creation of scalable custom microservices. *Serverless Single Page Apps* is for those who either have never built a Web application before and seasoned Web developers looking for an alternative to complex server-side Web frameworks.

<http://pragprog.com>

# ServersCheck's Thermal Imaging Camera Sensor



Monitoring data centers with sensors over conventional temperature probes has huge advantages, says facilities monitoring specialist ServersCheck. The company's new technology is what it bills

as "the world's first thermal imaging camera sensor that works with SNMP and Modbus". While a traditional temperature sensor reports the nearby temperature, ServersCheck's patent-pending thermal imaging sensor performs a thermal scan of what it sees within its 50° field-of-view camera. Every two seconds, it checks the temperature at 4,800 points, and this thermal image array is then converted into SNMP and Modbus data for easy integration with monitoring platforms. The network monitoring or building management platform receives transmission from the wired or cellular base unit—that is, the SensorGateway, which connects to the sensors via RJ45. This new technology opens up myriad new monitoring opportunities, says ServersCheck, particularly relating to data centers and distributed infrastructure sites. The new sensors can monitor individual changes or events within the environment that don't necessarily affect the overall conditions and often go undetected. Furthermore, monitoring the contacts and switches at substations with massive current loads on a 24x7 basis streamlines the maintenance process and safeguards against massive failures.

<http://serverscheck.com>

2016

WOMEN IN TECHNOLOGY

# SUMMIT

Executive women, entrepreneurs, and technology  
thought leaders from around the world will converge  
in Silicon Valley... **Join Us!**

**June 5th - 7th**

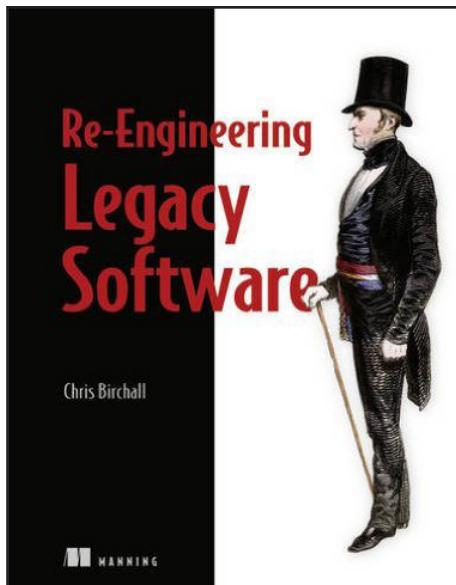
DoubleTree by Hilton  
San Jose, California

[witi.com/summit](http://witi.com/summit)



**Special Offer:** Use promo code **WOMEN** by  
May 15th and **Save \$100** Off Registration!

# Chris Birchall's *Re-Engineering Legacy Software* (Manning Publications)



Chances are high that you didn't write the application you're currently working on. Most developers inherit projects built on an existing codebase that reflects design patterns, usage assumptions, infrastructure and tooling from another time and another team (and the docs are complete rubbish). To help you breathe new life into your legacy project, pick up Chris Birchall's new book *Re-Engineering Legacy Software*. Birchall's book is an experience-driven guide to revitalizing inherited projects, covering refactoring, quality metrics, toolchain and workflow, continuous integration, infrastructure automation and organizational culture. On the purely technical side, readers will learn techniques for introducing dependency injection for code modularity, quantitatively measuring quality and automating infrastructure. On the strategic side, readers will develop practical processes for deciding whether to rewrite or refactor, team organization and even convincing management that quality matters. Core topics include deciphering and modularizing awkward code structures, effectively integrating and automating tests, replacing an outdated build system and infrastructure automation using tools like Vagrant and Ansible.

<http://manning.com>

Please send information about releases of Linux-related products to [newproducts@linuxjournal.com](mailto:newproducts@linuxjournal.com) or New Products c/o Linux Journal, PO Box 980985, Houston, TX 77098. Submissions are edited for length and content.

[RETURN TO CONTENTS](#)

THE  
**PERL  
CONFERENCE**



**PERL 6**

*June 19-24*

[PerlConference.Org](http://PerlConference.Org)

# Secure Token-Based Authentication with YubiKey 4

YubiKeys replace other one-time password and two-factor authentication systems with a USB token. This article provides in-depth coverage of common YubiKey configurations and defines a Vagrant virtual machine to speed up configuration testing.

TODD A. JACOBS



PREVIOUS  
New Products

NEXT

Feature: The Tiny Internet Project, Part I



**O**ne-time password systems like S/Key and OTPW (see my article “Configuring One-Time Password Authentication with OTPW” in the January 2013 issue of *Linux Journal*) are designed to protect your login credentials when you need to connect from an untrusted host, such as a co-worker’s desktop, or a public terminal at an Internet café or over an unencrypted protocol like telnet. Such systems rely on a challenge-response password list that you must carry around, and periodically update and print from secure terminals. This can be a huge hassle.

In contrast, two-factor authentication systems (with or without one-time passwords) require a second authentication step in order to prove your identity. This “second authentication factor” can be provided through token-based systems, such as Yubico’s YubiKeys. YubiKeys are commercial (but inexpensive) security devices that provide support for a variety of one-time password and two-factor authentication methods in several portable and platform-independent USB form-factors.

Unlike tokens that require you to enter a PIN or transcribe a result, the YubiKey 4 is designed for ease of use. Simply plug it in to a USB port, and it will be seen as a USB keyboard that can interact directly with your authentication dialogs.



Figure 1. YubiKey 4 from Yubico



Figure 2. Inserting a YubiKey 4 into a USB Port

# YubiKey 4 Authentication Protocols and Cryptographic Features

## **YUBIKEY 4 AUTHENTICATION PROTOCOLS:**

In addition to the Yubico OTP protocol pre-configured in Slot 1, the YubiKey 4 supports several other one-time password protocols. At the time of this writing, the YubiKey 4 natively supports a number of additional protocols, such as OATH-HOTP (HMAC-Based One-Time Passwords), FIDO U2F (Universal 2nd Factor) and HMAC-SHA1 Challenge-Response. The YubiKey 4 also provides the OATH-TOTP (Time-Based One-Time Passwords) authentication when used with the GPLv3 Yubico Authenticator desktop application.

## **ADDITIONAL CRYPTOGRAPHIC FEATURES:**

The YubiKey 4 also is configurable as an OpenPGP Smartcard with support for strong RSA keys up to 4,096 bits and several ECC (Elliptic Curve Cryptography) key types. These keys support the usual OpenPGP encryption and signing functions used by Pretty Good Privacy (PGP) and GNU Privacy Guard (GPG).

As a bonus feature, the YubiKey 4 supports a special authentication key that can be used by GNU Privacy Guard's gpg-agent for SSH public key authentication. Although outside the scope of this article, a YubiKey configured for SSH authentication meets all the requirements to be considered a true two-factor authentication method.

Since secret key material can't be extracted from a YubiKey even when plugged in to a compromised host, the smartcard features of the YubiKey make it ideal for carrying around your OpenPGP secret keys for creating digital signatures or decrypting messages on multiple hosts, or for using SSH public key authentication from an untrusted host without fear of giving up your secret key.

In some modes, a user may be required to activate a touch-sensitive spot on the token to authenticate when challenged, and in others, all that's required is the device's physical presence. In either case, this simplified approach to token-based authentication improves the user experience tremendously.

### YubiKey Configuration for Yubico OTP

**Using a Factory-Default YubiKey:** The YubiKey 4 is a rugged USB token that comes pre-configured for cloud-based authentication using 128-bit AES keys. In this mode, YubiKeys generate 44-character one-time passwords on demand. Although the YubiKey 4 is ready to use the Yubico OTP protocol via the YubiCloud service out of the box, an additional configuration slot is available to enable other protocols and local authentication options.

### Warning: Why YubiKey Slot 1 Is Special

Each YubiKey has two configuration slots. The first slot is factory-configured to support the Yubico OTP protocol, and this protocol validates against the Internet-accessible YubiCloud. Don't overwrite this slot!

Overwriting a slot isn't a reversible action. You can't reload the slot with previous data. You'll be required to create a *new* key with a special "vv" prefix (rather than the Yubico-standard "cccccc" prefix) to assign to Slot 1, and then upload that new key to the YubiCloud. This is only desirable in corporate settings where additional information (such as organizational unit) needs to be encoded into the prefix, or when compliance requirements require the use of an on-premises hardware security module (HSM), such as the YubiHSM.

Although you *can* overwrite Slot 1 with a custom configuration, clobbering the factory-default configuration creates unnecessary complications. Therefore, most users and administrators should limit custom configuration settings to Slot 2. However, if you *do* overwrite the Slot 1 configuration, refer to the extensive Yubico documentation for information on generating a new AES key and uploading it to the YubiCloud or an on-premises authentication server.

In other words, if you simply want to use the factory-default configuration for token-based validation against the YubiCloud, the YubiKey token requires no further configuration. The key itself is ready to use the moment you open the package and plug it in to a USB port!

Of course, even though the YubiKey arrives pre-configured, Linux services still need to know how to validate a key and how to associate an identity with that key. To do that, configure your Linux devices to support the Yubico OTP protocol through OpenSSH and PAM as described below.

**Configure SSH for Challenge-Response Authentication:** To use Yubico OTP or any of the YubiKey's other challenge-response protocols for SSH authentication, you must first configure your OpenSSH daemon to support it. On Ubuntu 15.04, which is the reference system for this article, the following settings must be present in the /etc/ssh/sshd\_config file:

```
ChallengeResponseAuthentication yes  
UsePAM yes
```

Reload the SSH server to enable the new configuration options to take effect. Note that the Vagrantfile provided in the next section will handle the configuration changes and reloading automatically for you.

**Supporting Yubico OTP in SSH with PAM:** The standard Linux mechanism for authentication is PAM, which stands for Pluggable Authentication Module. Yubico provides a BSD-licensed PAM module, which is available on GitHub and in the repositories of many recent distributions.

Once you have installed Yubico PAM, include the module in any PAM-enabled service that you want to protect with a YubiKey. Although this most often is the SSH service, you also can use it for console logins, X11 logins or other PAM-protected services. Because it's possible to lock yourself out of a machine through improper PAM configuration, and because not all X11 display managers support challenge-response authentication, this article remains focused on the common use case of protecting the SSH service.

If you have Vagrant and VirtualBox installed, along with direct access to the Internet, you can use the Vagrantfile shown in Listing 1 to create a test instance. The Vagrantfile also will set up Yubico PAM for you while provisioning the virtual machine (VM). This can save you a great deal of time troubleshooting when setting up a YubiKey test environment.

**Listing 1. Vagrantfile to Create a Test Instance**

```
Vagrant.configure(2) do |config|
  config.vm.box = 'ubuntu/vivid64'

  if defined? VagrantVbguest
    config.vbguest.auto_update = false
  end

  config.vm.provider :virtualbox do |v|
    v.gui = true
    v.customize ['modifyvm', :id, '--usb', 'on']
    v.customize [
      'usbfilter', 'add', '0',
      '--target', :id,
      '--name', 'yubikey',
      '--vendorid', '1050',
      '--productid', '0407',
    ]
  end

  config.vm.provision :shell, inline: <<- 'SHELL'
    authfile='/etc/yubikey_mappings'

    packages=(
      libpam-{yubico,otpw}
      otpw-bin
      yubikey-personalization
    )
    export DEBIAN_FRONTEND=noninteractive
    apt-get install -y "${packages[@]}"

    if [[ ! -f "$authfile" ]]; then
      touch "$authfile"
      chmod 640 "$authfile"
    fi

    file='/etc/pam.d/sshd'
    comment="# Enable Yubicloud authentication."
    config='auth      sufficient pam_yubico.so'
    config+="config id=16 authfile=$authfile"
    if ! fgrep -q pam_yubico.so "$file"; then
      sed -i "2a\\$comment\\n$config\\n" "$file"
    fi

    comment="# Require OTPW one-time passwords."
    config="# auth      requisite pam_otpw.so\\n"
    config+="# session    optional pam_otpw.so"
    if ! egrep -q 'req.*pam_otpw' "$file"; then
      sed -i \
        "/pam_yubico/a\\n$comment\\n$config" \
        "$file"
    fi
  end
end
```

```

fi

comment='# Require YubiCloud authentication.'
config='# auth      requisite    pam_yubico.so'
config="$config id=16 authfile=$authfile"
if ! egrep -q 'req.*pam_yubico.so' "$file"; then
    sed -i \
        "/pam_yubico/a\\\$comment\\n$config" \
        "$file"
fi

dir='/etc/yubico'
if [[ ! -d "$dir" ]]; then
    mkdir -pm 1777 "$dir"
fi

file='/etc/pam.d/common-auth'
comment='# Enable HMAC-SHA1 to succeed without '
comment+=" a second factor, even with\n"
comment+="# encrypted home directories."
config='# auth      sufficient   '
config+='pam_yubico.so '
config+='mode=challenge-response '
config+="#chalresp_path=/etc/yubico"
if ! fgrep -q pam_yubico.so "$file"; then
    sed -i \
        -e "# pam-auth-update(8)/a\\\$comment" \
        -e "# pam-auth-update(8)/a\\$config" \
        "$file"
fi

file='/etc/ssh/sshd_config'
keyword='ChallengeResponseAuthentication'
if ! egrep -q "^$keyword yes" "$file"; then
    sed -ri "s/($keyword) no/\1 yes/" "$file"
    service ssh reload
fi

dir='/etc/udev/rules.d/'
rules=($dir/*)
if [[ ${#rules} -ne 2 ]]; then
    url="https://raw.githubusercontent.com"
    url+="${url}/Yubico/yubikey-personalization"
    url+="${url}/master"
    url+="${url}/[69-70]-yubikey.rules"
    cd "$dir"
    curl -sLO "$url"
    udevadm trigger
fi
SHELL
end

```

Bring up the VM with the following command:

```
$ vagrant up --provider virtualbox
```

If you have a real server to work with or prefer to configure an existing box rather than a VM, ensure that the `pam_yubico.so` module is included at the very top of your PAM configuration file for the SSH service. On the Ubuntu reference platform for this article, the PAM configuration file is `/etc/pam.d/sshd`, but it may be different on your distribution.

As an example, the top of your file should look similar to the following:

```
# PAM configuration for the Secure Shell service

# Enable YubiCloud authentication.
auth sufficient pam_yubico.so id=16 authfile=/etc/yubikey_mappings

# Standard Un*x authentication.
@include common-auth
```

This particular PAM configuration prompts the user for YubiKey authentication before falling back to a standard password prompt. See the sidebar on precedence for more information about the order in which authentications happens.

**Map Users to YubiKeys:** With the OpenSSH and PAM configuration now done, there is one more step to complete before YubiKey authentication will work: mapping system users to authorized tokens.

First, it's important to understand *why* this step is necessary, especially when using the YubiCloud. The challenge-response configuration you've implemented here pairs a token holding a secure element (the YubiKey) with an AES key stored in the YubiCloud. Without a way to authorize tokens, a user could use *any* valid token to log in, even if the token wasn't issued to that user. In addition, unless you map tokens to specific accounts, you can't identify the holder of a token nor limit the accounts a given token can be used to access.

To address these concerns, you will use a file defined by your

## Precedence of Authentication Steps

OpenSSH and PAM both contain authentication rules and can support very complex configurations when necessary. However, when configured on the reference platform as described in this article, the precedence of authentication mechanisms (unless overridden by the SSH client) will be as follows:

1. Public key authentication.
2. YubiKey authentication.
3. UNIX password authentication.

This is generally the appropriate order of precedence, but if necessary, it can be adjusted through one or more of the following mechanisms:

1. The precedence defined by `PreferredAuthentications` in the SSH client.
2. The authentication methods allowed by your SSH server configuration.
3. Multiple authentications as required by the `AuthenticationMethods` server option.
4. The PAM stack defined for your SSH service.

PAM configuration to map tokens to users. This restricts who may use YubiKeys to authenticate to the system, as well as tying a given YubiKey to a specific user or system account.

This mapping is done using a 12-character modified hexadecimal (ModHex) prefix unique to each AES key. The prefix for factory-configured YubiKeys starts with “cccccc”, although custom or

reconfigured YubiKeys always start with “vv”. The following script will enable you to identify the necessary prefix for your YubiKey easily:

```
#!/usr/bin/env bash

read -p 'Enter Yubico OTP: '
echo "${REPLY:0:12}"
```

1. Place the script into your path and make it executable.
2. Insert your YubiKey into an available USB port.
3. Run the script.
4. When prompted, *briefly* press the round sensor for 0.3 to 1.5 seconds to issue a one-time password using the factory-default setting in Slot 1. Please note that longer presses of 2.5–5.0 or 8–15 seconds trigger other behaviors; see the YubiKey Manual for details.
5. Copy the prefix; this will be associated with a user name in the mapping file defined in your PAM configuration.

When you run the script and trigger your YubiKey, you should see output similar to the following:

```
$ bash yubikey-id.sh
Enter Yubico OTP: cccccrlivbljtkhdinncllelldrktdefttuuuecfjdbh
ModHex Prefix: cccccrlivbl
```

Using this example, you would map “ccccccrlivbl” to the user assigned that particular YubiKey device. You do that in the authfile defined in your PAM configuration file.

If you use the Vagrantfile from Listing 1, you can configure the mapping file from the host machine with a single command and a tap of

your YubiKey:

```
vagrant ssh -- \
"echo 'vagrant:$(yubikey-id.sh)' | \
sudo tee -a /etc/yubikey_mappings"
```

Otherwise, log in to your test system and edit the mapping file in the form of <username>:<prefix>. To assign more than one YubiKey per user, separate each prefix from the next one with a colon.

**Testing Your YubiKey:** Now you're ready to test authentication with Yubico OTP! If possible, remain logged in as root on the box that you're testing while you verify the configuration in another window. If you have made a mistake in one of the configuration files, you may not be able to reconnect.

If you are using the Vagrant test instance, use the following command to perform a valid test of Yubico OTP:

```
vagrant ssh -- \
-S none \
-o PreferredAuthentications=keyboard-interactive
```

This will force the SSH client to disable connection sharing so that it won't reuse an existing connection where you already may have authenticated with an SSH key or password. In addition, it will tell the client to try challenge-response before presenting any SSH keys.

If you are not using Vagrant, use the `-S none` and `-o PreferredAuthentications=keyboard-interactive` arguments anyway to ensure you are performing a valid test. However, feel free to modify other aspects of SSH client behavior to fit your specific environment.

If your OpenSSH and PAM settings are correct, you will see a prompt similar to the one shown in Figure 3.

```
$ vagrant ssh -- -S none -o PreferredAuthentications=keyboard-interactive
YubiKey for `vagrant': 
```

Figure 3. OpenSSH Dialog Prompting User for YubiKey

To continue, press the sensor on your YubiKey for 0.3 to 1.5 seconds to submit your one-time password. Assuming that your mapping file is correct, and that the PAM module is able to connect to the Internet and reach the YubiCloud service, you will be logged in.

If you're also prompted for your system password but haven't configured two-factor authentication, as described later in this article, it's likely that your mapping file or ability to connect to the YubiCloud service is the problem. Double-check the mapping file and your Internet connectivity. If problems persist, refer to the Yubico PAM documentation for instructions on how to turn on debug-level logging.

## Adding a Second Authentication Factor

Two-factor authentication is sometimes defined as something you know (such as a PIN) and something you have, such as an OTP calculator or token. As you've implemented the system thus far, you've really implemented only one-time passwords in a way that is similar to S/Key or OTPW, but much easier for the end user. No password lists to carry around, no password prefixes to memorize—just connect, press the sensor, and bingo!

You can implement true two-factor authentication by making a minor change to your SSH server's PAM configuration. For example, you can require the use of a password (something you know) *in addition to* a YubiKey (something you have).

To do this, change the control value of Yubico PAM from "sufficient" to "requisite". In your original one-time password configuration, the module was *sufficient* by itself to grant access when Yubico OTP succeeded, but would fall back to the next authentication option if it failed. With the change to *requisite*, Yubico OTP is now required, and the PAM authentication stack will return immediately from a failure. The first four lines of /etc/pam.d/sshd now contain:

```
# PAM configuration for the Secure Shell service

# Require token-based Yubico OTP authentication.
auth      requisite pam_yubico.so id=16 authfile=/etc/yubikey_mappings
```

```
$ vagrant ssh -- -S none -o PreferredAuthentications=keyboard-interactive  
YubiKey for `vagrant':  
Password: 
```

Figure 4. Two-Factor Authentication with YubiKey and Password

Your login dialog now should look like Figure 4, even when your YubiKey successfully validates against the YubiCloud.

A compromised machine can't extract elements from the YubiKey for later use or replay a one-time password using the Yubico OTP protocol once the password has been used. Even assuming that you are logged in on a compromised workstation, two-factor authentication is surprisingly robust even when exposing your password to the local host. By combining your password with the one-time passwords provided by the YubiKey token, and given the necessity of having physical possession of the token in order to issue one-time passwords, your overall risk is actually reduced.

However, if you are exposing a password that you use on multiple machines, and not all of those machines are protected by two-factor authentication, this would be an unacceptable risk. You can do better!

It's possible to modify the PAM stack further to replace the request for your Linux system password with a request for a different one-time password system, such as OTPW. For example, on the example reference platform, you might require the OTPW module *in addition* to Yubico OTP. Assuming that OTPW has been installed and configured, your PAM module in /etc/pam.d/sshd would look like this:

```
# PAM configuration for the Secure Shell service  
  
# Require token-based Yubico OTP authentication.  
auth      requisite  pam_yubico.so id=16 authfile=/etc/yubikey_mappings  
  
# Require OTPW one-time passwords.  
auth      requisite  pam_otpw.so  
session   optional   pam_otpw.so
```

## OpenSSH Authentication Methods

It's worth noting that recent versions of OpenSSH v6.2 and later support multifactor authentication through the `AuthenticationMethods` server configuration option. However, in most cases, PAM provides a more customizable experience with a wider range of supported authentication options.

At the time of this writing, using `AuthenticationMethods` instead of PAM for multifactor authentication on Linux systems appears to be of limited value. However, this feature *does* allow OpenSSH to require the user to authenticate using more than one public key at a time. This is something PAM cannot do, and the feature may grow more robust in the future.

Of course, to ensure that the system requires both Yubico OTPW and your second-factor authenticator, you also must disable standard password authentication via PAM. On the reference platform, edit `/etc/pam.d/sshd` and prefix `@include common-auth` with a comment character.

Once you've initialized OTPW with:

```
$ vagrant ssh -- otpw-gen > password-list.txt
```

you're ready to test out two-factor authentication. With public key authentication disabled, you should be prompted for an OTPW password after successfully authenticating your YubiKey.

## Local Validation

In some cases, it's undesirable to validate to the YubiCloud or other networked resource. If you can physically plug the YubiKey token in to your server or workstation, HMAC-SHA1 is a good alternative. In this mode, YubiKey's challenge-response feature is very similar to S/Key, OPIE or OTPW one-time password systems, but requires only the physical presence of the token.

**Installing the YubiKey Command-Line Personalization Tool:** The YubiKey configuration tools `ykpersonalize` and `ykpacmcfg` are available from the `yubikey-personalization` package on Ubuntu 15.04. If you’re using the `Vagrantfile` provided here, this package already has been installed and configured for you. Otherwise, go ahead and install it now.

Next, install the udev rules to give non-root users access to the YubiKey

## Debugging YubiKey Configuration Tools

Use a recent version of the command-line YubiKey Personalization Tools. At the time of this writing, the current stable version is v1.17.2. Also, ensure you’re logged in on a local TTY rather than over the network with telnet or SSH.

### **OUTDATED TOOLS:**

If you see an error like “Yubikey core error: no yubikey present” when trying to configure the token, try the following command to see if your YubiKey is being detected as a USB device:

```
$ lsusb | fgrep Yubico  
Bus 001 Device 003: ID 1050:0407 Yubico.com
```

If your device is detected, but the problem persists, the personalization tools may not be current. Use version 1.16.2 or later of the tools.

### **PERMISSION PROBLEMS:**

On the other hand, “USB error: Access denied (insufficient permissions)” is telling you that:

- You don’t have the proper udev rules installed.
- You are attempting to use the tools via SSH, rather than from a console.

device. The yubikey\_udev\_rules.sh script below downloads the latest rules for you and places them into the correct location on an Ubuntu machine (adjust the download directory on other distributions if necessary):

```
#!/usr/bin/env bash

# Purpose:
#     Download the latest udev rules for the YubiKey
#     Personalization Tools.

url="https://raw.githubusercontent.com"
url+="${url}/Yubico/yubikey-personalization"
url+="${url}/master"
url+="${url}/[69-70]-yubikey.rules"

cd /etc/udev/rules.d/
sudo curl -sLO "$url"
```

Finally, run `sudo udevadm trigger` to force udev to use the new rules you just installed. Otherwise, the rules may not take effect until the next reboot.

Make sure you download the latest udev rules from the GitHub repository and connect directly to a console or through your virtual machine's graphical user interface.

**Configuring HMAC-SHA1 Challenge-Response:** Once the udev rules are installed, you're ready to configure the key from a local console. If using the VM, log in through the graphical user interface using the user name "vagrant" and the password "vagrant".

First, use the YubiKey personalization tools to enable HMAC-SHA1 challenge-response from Slot 2:

```
# Configure Slot 2 for HMAC-SHA1.
yes | ykpersonalize -2 \
    -ochal-resp \
    -ochal-hmac \
    -ohmac-lt64 \
    -oserial-api-visible
```

Next, initialize the file that holds the challenge-response pairs using your new Slot 2 configuration:

```
ykpamcfg -2
```

Finally, configure PAM to accept the HMAC-SHA1 credentials during console logins by placing challenge-response authentication *before* the primary block in /etc/pam.d/common-auth. Use the following excerpt as guidance:

```
# /etc/pam.d/common-auth - authentication settings common to all
# services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.). The default is to use
# the traditional Unix authentication mechanisms.
#
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules. See
# pam-auth-update(8) for details.

# Enable HMAC-SHA1 to succeed without a second factor.
auth      sufficient    pam_yubico.so mode=challenge-response

# here are the per-package modules (the "Primary" block)
auth      [success=1 default=ignore]      pam_unix.so nullok_secure
```

If you're using the VM, just uncomment module in common-auth. However, leave the directory argument commented out for now. At this time, if you specify a global directory for HMAC-SHA1, the PAM module will not find the challenge-response file in the user's home directory. (I'll discuss use of the global directory in the next section.)

The changes you made in common-auth won't affect non-local connections such as SSH *unless* you're connecting to a local VM. This is typically because the host machine and the VM share access to the YubiKey. Remember that HMAC-SHA1 is for authenticating console access only!

Now that PAM has been updated, attempt to log in again from a console. You should be authenticated automatically immediately after typing your user name at the login prompt. However, when you modify PAM, login may not pick up the configuration changes immediately. If this happens, just press return a few times on your console until the screen clears and the login prompt starts again at the top of the screen.

**HMAC-SHA1 with Encrypted Home Directories:** If you use encrypted home directories, such as Ubuntu's eCryptfs, challenge-response files must be moved out of the user's home directory and into a central location, such as /etc/yubico. Otherwise, the challenge-response files can't be read or updated by PAM during the authentication process when the directory is unmounted.

On such systems, the module authors recommend a publicly writable central location with the sticky bit set. This is similar to the /tmp directory and allows users to update their challenge-response files after each log in:

```
sudo mkdir --mode=1777 /etc/yubico
```

Next, add this central directory to your PAM configuration using the chalresp\_path option. If using the VM, just uncomment the provided argument:

```
# /etc/pam.d/common-auth - authentication settings common to all
# services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.). The default is to use the
# traditional Unix authentication mechanisms.
```

```
#  
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.  
# To take advantage of this, it is recommended that you configure any  
# local modules either before or after the default block, and use  
# pam-auth-update to manage selection of other modules. See  
# pam-auth-update(8) for details.  
  
# Enable HMAC-SHA1 to succeed without a second factor, even with  
# encrypted home directories.  
auth sufficient pam_yubico.so mode=challenge-response  
chalresp_path=/etc/yubico  
  
# here are the per-package modules (the "Primary" block)  
auth [success=1 default=ignore] pam_unix.so nullok_secure
```

All users must then move their challenge-response file into this central directory and rename the file's prefix from "challenge" to their login name. For example:

```
mv ~/.yubico/challenge-* \  
"/etc/yubico/$LOGNAME-$(ykinfo -qs)"
```

**Adding a Second Factor to HMAC-SHA1:** As with Yubico OTP, it's easy to add a second factor to HMAC-SHA1 challenge-response authentication. Simply substitute `auth sufficient pam_yubico.so` with `auth required pam_yubico.so` in the common-auth file, and PAM will mandate the presence of the user's YubiKey in addition to any other required authentications. On the reference system, this will be the system password by default, but other configurations are possible.

In general, use *required* here instead of *requisite*. Using "required" ensures that the rest of the PAM stack (such as the system password prompt) is evaluated before returning a response even though authentication can't succeed if the YubiKey isn't present or valid. Not returning immediately prevents a potential attacker from seeing "Login incorrect" immediately after typing in a user name. However, this represents a trade-off for users. With "required", if users forget to insert their YubiKeys *before* attempting to login, they may think they've simply

mistyped their password and become frustrated. The trade-off between user experience and security in this particular case is yours to make.

### Summary

I have described two types of challenge-response systems using the YubiKey 4 USB tokens: Yubico OTP with YubiCloud validation and HMAC-SHA1 for local validation. I've also explored different ways to configure YubiKey authentication using PAM, including support for two-factor authentication.

The YubiKey 4 is a worthwhile replacement for other one-time password and two-factor authentication systems due to its flexibility, rugged design and convenient form factor.

In addition, the device's support for OpenPGP smartcard standards and other code-signing features makes it a good choice for administrators, developers and DevOps engineers who need secure

---

**LINUX JOURNAL**  
now available  
for iPad and  
iPhone at the  
App Store.



[www.linuxjournal.com/ios](http://www.linuxjournal.com/ios)



For more information about advertising opportunities within *Linux Journal* iPhone, iPad and Android apps, contact John Grogan at +1-713-344-1956 x2 or [ads@linuxjournal.com](mailto:ads@linuxjournal.com).

access to their secret keys when on the go.

This article provides a solid foundation for implementing the device's basic security features. Hopefully, it has also whet your appetite for exploring additional uses for the YubiKey 4. ■

---

**Todd A. Jacobs** is a veteran IT consultant who currently specializes in automation and security. In what passes for his free time, Todd is also a top contributor on Stack Overflow and Project Management Stack Exchange. In between writing, coding and (of course) drinking way too much coffee, he practices the ancient arts of marriage and fatherhood to the very best of his ability.

## Resources

### Essential Resources for This Article:

Yubico PAM Module: <https://developers.yubico.com/yubico-pam>

YubiKey Personalization Tools: <https://github.com/Yubico/yubikey-personalization>

### Additional YubiKey-Related Resources:

Yubico Home Page: <https://www.yubico.com>

Yubico Downloads: <https://www.yubico.com/support/downloads>

Yubico Documentation: <https://www.yubico.com/support/documentation>

YubiCloud Connector Libraries:

[https://developers.yubico.com/OTP/Libraries>List\\_of\\_libraries.html](https://developers.yubico.com/OTP/Libraries>List_of_libraries.html)

Supported YubiCloud Alternatives for Yubico OTP:

- YubiKey Key Storage Module (YK-KSM): <https://github.com/Yubico/yubikey-ksm>
- YubiKey OTP Validation Server (YK-VAL): <https://github.com/Yubico/yubikey-val>
- Python YubiHSM (PyHSM): <https://github.com/Yubico/python-pyhsm>

Yubico Authenticator for OATH (desktop app): <https://github.com/Yubico/yubioath-desktop>

Send comments or feedback via  
<http://www.linuxjournal.com/contact>  
or to [leditor@linuxjournal.com](mailto:leditor@linuxjournal.com).

**RETURN TO CONTENTS**

# Attend



# SPTechCon

## The SharePoint Technology Conference

### June 27-30, 2016

The Sheraton Boston



Register Early and SAVE!



**"This is the most informative conference I have been to in years. The technical discussions gave me a much better understanding of direction, advantages and challenges we face with this massive platform."**

—Jamie Tyndall, Manager, Application Development, Business Information Group

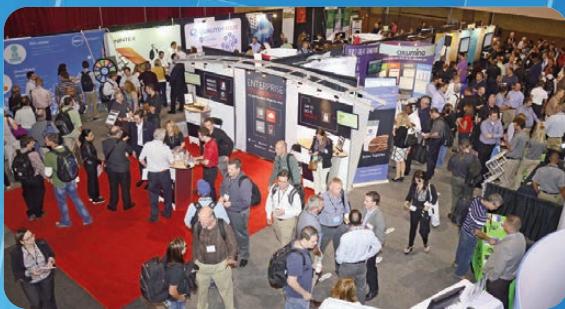
# Learn what's new in SharePoint and Office 365!



**"This was a great conference that addresses all levels, roles and abilities. Great variety of classes, great presenters, and I learned many practical things that I can take back and start implementing next week."**

—Kathy Mincey, Collaboration Specialist, FHI 360

Whether you want to learn about what's coming in SharePoint 2016, are still making the most out of SharePoint 2013 or even 2010, or getting started with Office 365, you will find the SharePoint and Office 365 training you need at SPTechCon.



**"As a newcomer to SharePoint, SPTechCon was an excellent way to begin learning more of its vast capabilities. This conference is a great way to hear about technical features and success stories of the product. Great vendors too. I will be following up with several of them about their products."**

—Jeffrey Wahl, IT Services Manager, Carbonite, Inc.

[www.spotechcon.com](http://www.spotechcon.com)

A BZ Media Event

SPTechCon™ is a trademark of BZ Media LLC. SharePoint® is a registered trademark of Microsoft.

# THE TINY INTERNET PROJECT, Part I

Learn the basics of Linux  
by building your own  
private multi-server “Internet”.

**JOHN S. TONELLO**



PREVIOUS  
Feature:  
YubiKey 4

NEXT  
Doc Searls' EOF



**A**s readers of this magazine well know, Linux drives many of the technologies we use every day, from smart TVs to Web servers. Linux is everywhere—except most homes and classrooms.

That's a problem if we want to help breed the next generation of engineers and computer scientists. In fact, if teenagers (or any other group of curious individuals) want to learn about Linux, they often must rely on a geeky friend or parent willing to show them the way.

This three-part series seeks to change that by offering a way for anyone to learn about Linux by building what is essentially a tiny, self-contained Internet. Using old equipment and free software, you'll build a private network (with your own domain name), build Web sites, set up an e-mail server, install and use a database, and set up a Linux distro mirror.

## **IF YOU LIKE TO LEARN BY DOING, BUT YOU'RE INTIMIDATED BY THE THICK LINUX TEXTS YOU FIND AT THE BOOKSTORE, THIS TINY INTERNET PROJECT IS FOR YOU.**

If you like to learn by doing, but you're intimidated by the thick Linux texts you find at the bookstore, this Tiny Internet Project is for you. If you're a teacher interested in bringing Linux to the classroom, this is a great way to do it.

At the core of the project is a Proxmox KVM environment. KVM, or kernel-based virtual machine, is an open-source alternative to often costly VM technology like VMware and Hyper-V. You'll use Proxmox to host several Ubuntu 14.04 servers (or other Linux flavors), connect them over a private network and learn a lot about Linux along the way.

The Tiny Internet Project assumes you have some basic computer skills (Windows, Mac or Linux), that you have a couple computers lying around and that you have some time to tinker. The project can be done in whole or in part, depending on your interests and needs. It's particularly designed for educators who want to introduce school-aged kids to Linux.

## FEATURE: The Tiny Internet Project, Part I

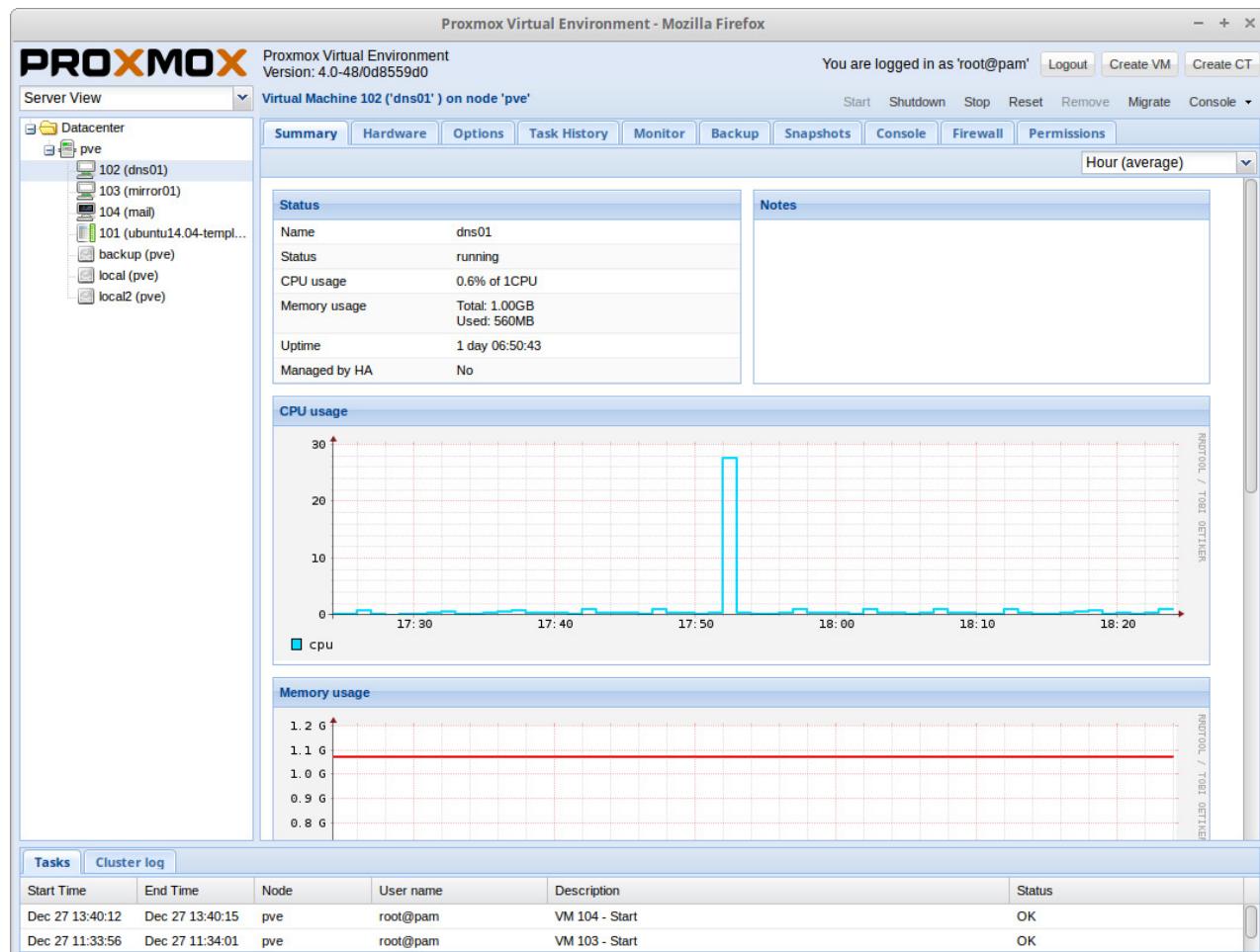


Figure 1. Proxmox

## WHAT YOU'LL BE BUILDING

You'll be using open-source software for everything in this project, so everything you need will be free to download and use. You'll also take advantage of virtualization technology, which will enable you to deploy a bunch of virtual machines. In all, you'll deploy:

- The Proxmox server to host all your virtual machines.
- Two DNS servers, a primary and a secondary.
- An e-mail server.
- One or more Web servers.

- An Ubuntu 14.04 repository mirror.

Strictly speaking, the mirror is optional. As long as you have an Internet connection, you'll be able to download new software and run Linux updates on all the servers you deploy. But the goal is to create a self-contained tiny Internet that will work without a permanent connection to the public Internet. Setting up a local mirror will enable you to do just that.

Optionally, you can build two or more Proxmox hosts and set up a cluster (I'll cover that when I describe building the Proxmox server). Obviously, you'll need one physical computer for each Proxmox host you want to add to your tiny Internet.

## PROJECT REQUIREMENTS

When it comes to hardware for this project, the goal is not to have you buy anything new, but to use stuff you already own—maybe your recently retired desktop, an old wireless router or an old laptop or netbook. If you don't have any hardware like this lying around, ask friends and family, and then consider Craigslist or eBay.

The bare minimum hardware you'll need includes:

- One 64-bit PC that supports virtualization.
- One PC that can attach to a network and run a Web browser (Windows, Mac or Linux).
- One network switch or router.

Nice to have:

- Another 64-bit PC that supports virtualization so you can build a cluster.
- Some sort of network-attached storage (NAS).
- Another old PC that can run Linux to act as a proxy server.

### HARDWARE

**The Main Server:** The key requirement for this project is a primary computer with a processor that can handle virtualization. Many, many computers made since 2010 or so have this capability, including the tower I used for my very first tiny Internet. It has:

- One Intel i3 processor (four cores).
- 8GB of memory (possible with less, but not much less).
- Two 2TB SATA drives (one drive is enough).
- Two 10/100/1000 Ethernet ports (it had one built in to the motherboard, and I added a PCI card).

To see if the computer you have in mind can become a Proxmox server, there are several ways to test it to see if it supports virtualization. There are tools for Windows and Linux, which are listed in the Resources section at the end of this article.

For those already using a Linux desktop or server, you can use existing commands to see if virtualization is supported. Open a terminal and run this simple command to do a quick check:

```
$ cat /proc/cpuinfo | grep vmx
```

It should return something that looks like the following (repeated several times for each core you have):

```
[flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
  ↪pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
  ↪sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc
  ↪arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
  ↪aperfmpf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl
  ↪vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1
  ↪sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c
  ↪rdrandlahf_lm ida arat epb xsaveopt pln pts dtherm
  ↪tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms]
```

If the flags include `vmx` (possibly highlighted red in the output), you're probably good to go. You also might check your system's BIOS. Often, virtualization is possible, but it's disabled by default. Look for it in your BIOS, enable it and reboot.

If you have additional hard drives that you'd like to use with this project, you can install them in your main server (the Proxmox host machine). Technically speaking, you need only a single drive, but having more than one can give you nice options for backing up the things you build. It's also good practice to learn how to mount multiple drives!

**The Administration PC:** You'll need some sort of second computer to act as your main administrative machine. This does not need to support virtualization. It only needs to be able to run a Web browser, maybe tinyproxy, and have an Ethernet port and Wi-Fi—or two Ethernet ports. If the machine doesn't have built-in Wi-Fi, you can get a USB dongle to do the job. The goal here is to have a machine with two network connections: one to your Tiny Internet and one to the network you use to access the Internet, such as your home or school network.

The administration PC can be your current desktop or laptop, and it can be Windows, Mac or Linux. If you're planning to have a main Proxmox server with two Ethernet ports, your administration PC needs to have only two network connections if you want to connect to your private tiny Internet and the public Internet simultaneously. One scenario also uses this PC as an http proxy server, which again needs access to both public and private networks.

I had a couple old laptops, and I successfully used the following for my administration PC:

- An IBM ThinkPad T60p (with built-in Wi-Fi and 10/100/1000 wired Ethernet).
- A Dell Mini 9 (with built-in Wi-Fi and 10/100 wired Ethernet).
- A Dell Mini 10 (with built-in Wi-Fi and 10/100 wired Ethernet).
- A first-generation Intel-based MacBook (with built-in Wi-Fi and -10/100/1000 wired Ethernet).
- A Dell GX620 (with a Wi-Fi card and built-in 10/100 wired Ethernet).

Any old tower PC or desktop will work too—nothing fancy needed!

Ideally, your administration PC will be running a flavor of Linux with a desktop environment like GNOME, KDE or Xfce. However, it's not necessary. The goal is to ease you into Linux, not to toss you into the pool cruelly.

Optional: if it's not possible to run dedicated Linux computers in your classroom (or lab), but you want to get a taste for it, you always can boot a Windows or Intel-based Mac using a USB stick. I'll go into more detail about this later, but you can learn more about making a bootable USB with Linux on the Ubuntu Web site. Information is available in the Resources section at the end of this article.

**Other PCs for Your Tiny Internet:** If you're running this project out of your den, you won't need any more computers. If you're building this in a classroom, the student machines can be much like the administration PC, though they each need just one network interface. If you're hard-wiring everyone to your tiny Internet, obviously each PC will need an Ethernet port. If you're going wireless, built-in Wi-Fi or inexpensive USB Wi-Fi dongles work great. Ideally, all the PCs in your network will be running a flavor of Linux, such as Ubuntu, Xubuntu, Linux Mint, Fedora, SUSE, CentOS, Kali—or another.

**Network Gear:** Again, you can use any switch or router you already have to create your private tiny Internet network. A single network switch or router is all you'll need to connect everything. If you already have a home network router, that can double as your tiny Internet switch, but I recommend a second to create a truly standalone system.

If you're setting up in a classroom, I strongly recommend using a wireless router so you easily can add dozens and dozens of separate student computers to your tiny Internet without running a bunch of Ethernet cables all over the place. Sure, that'll look cool, but it's not very practical.

Your tiny Internet switch (or router) requires just two ports: one connected to your main Proxmox server and one connected to your administration PC.

For my first tiny Internet, I used an old Netgear MR314 Wireless Router, which features the following:

- Four 10/100 LAN ports.
- One 10/100 WAN port.
- 802.11b wireless.

## FEATURE: The Tiny Internet Project, Part I

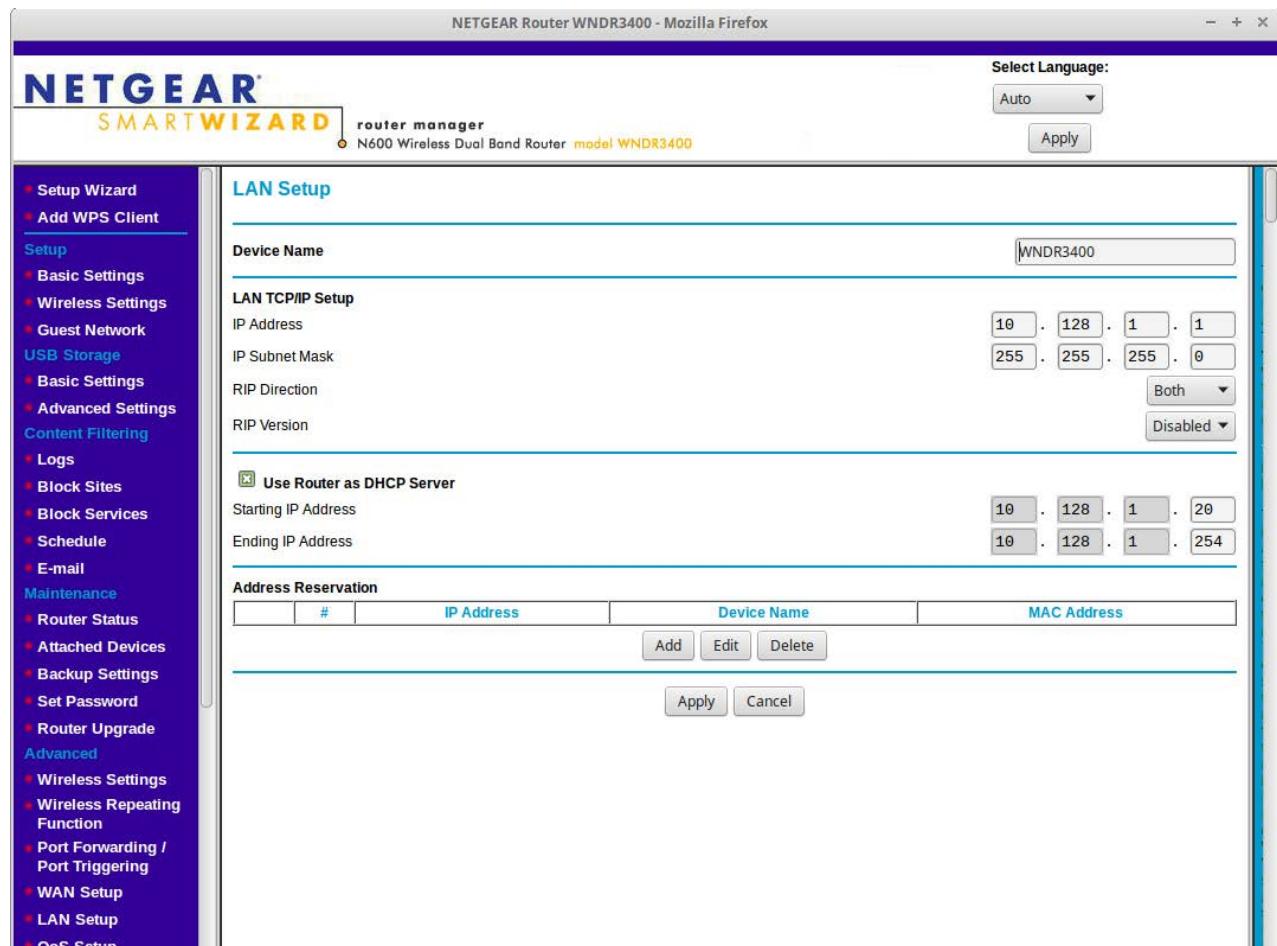


Figure 2. Netgear

Granted, this old box supports only WEP encryption, but it worked fine. Remember, your tiny Internet is self-contained, with no direct connection to the outside world. Yes, you'll want some security in place, but your main security risk is from members of your own tiny Internet, not the world.

I also tested the following networking devices with success:

- A TP-Link TL-SG108 eight-port 100/1000 switch (bought new for \$25).
- A Netgear N600 four-port 10/100 Wireless Dual-Band Router WNDR3400.

The Netgear N600 became my final choice because it has WPA2 security, wireless n capability and a USB 3.0 port for adding a USB drive (for making a poor man's NAS).

**Other Hardware:** In addition to the PCs and network gear, you'll need a few USB thumbdrives. You'll burn .iso images to these and set them up so you can boot from them. In particular, you'll create these three:

- Proxmox 4.x boot disk.
- Ubuntu 14.04 server boot disk.
- Xubuntu 14.04 Trusty Tahr (or any other Linux-flavor desktop you want).

If you don't have access to USB thumbdrives, you always can use DVDs for the purpose, but that's not nearly as easy, flexible or cheap. Still, if that's all you have, make sure you have four or five blank disks available, a decent DVD burner and disk-burning software.

Wireless capability is fairly ubiquitous in modern PCs and laptops, but your older machines may not have it. Fortunately, there are dozens of very inexpensive USB Wi-Fi dongles available (many for \$8 or so). If you're thinking of getting one (or a dozen), make sure the device works under Linux. Better still: buy devices that work with Linux, Windows or Mac.

## SOFTWARE

All the software you'll use for the Tiny Internet Project is free and open-source. Most of it's Linux software, of course, but I've also listed a few tools for Windows and Mac users, particularly the software you'll need to create bootable USB drives from an .iso file.

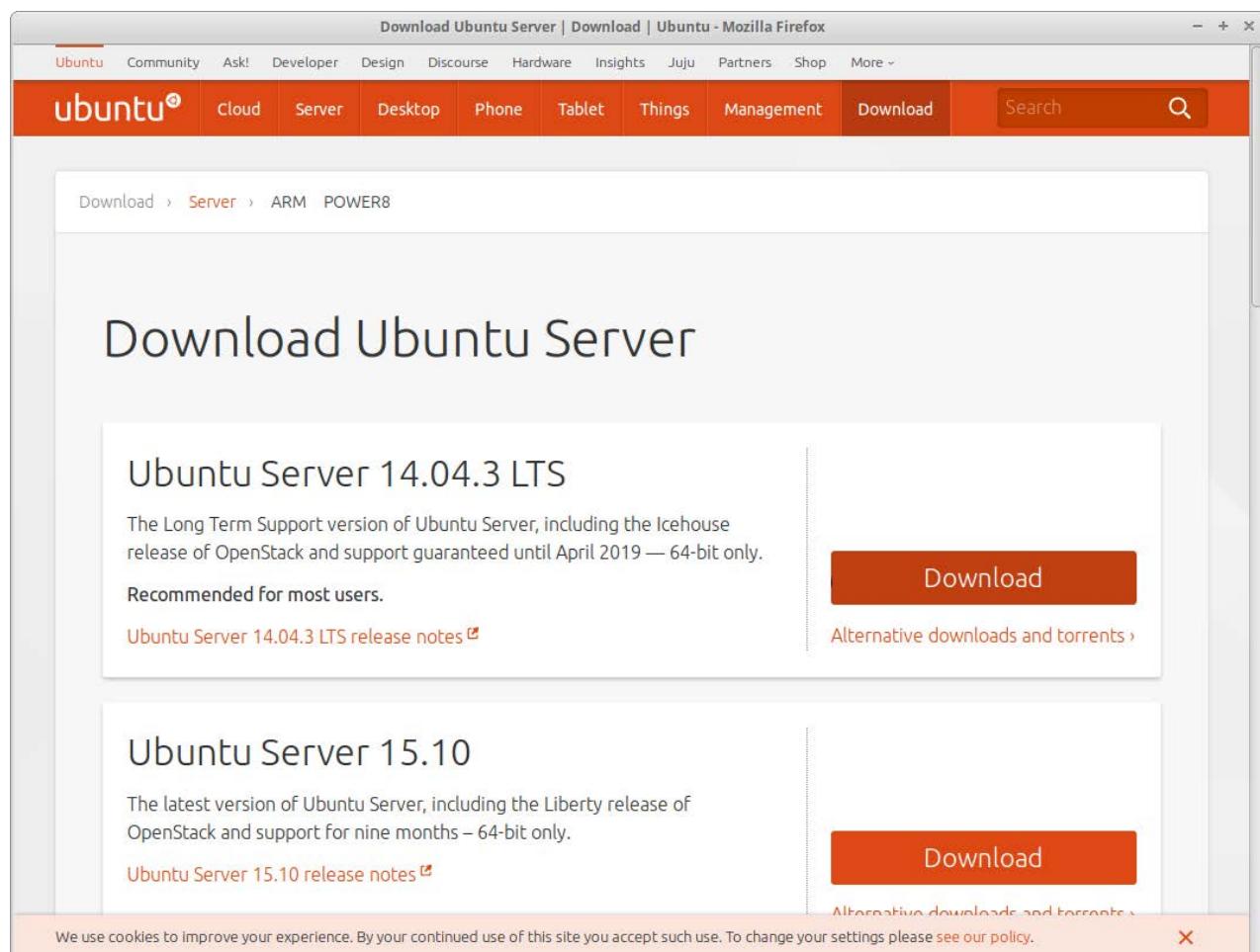
You'll notice too that I'm using Ubuntu 14.04 as the base for my virtual machines. If you would rather use, say, Fedora or SUSE, that's up to you. For brevity, I stick to Ubuntu when it comes time to talk about installation procedures.

**Proxmox 4.x:** Proxmox is an open-source KVM, or kernel-based virtual machine host. You can use many different flavors of Linux to create a KVM, but Proxmox is a good option for your tiny Internet because it comes complete. It's based on Debian, which is similar to the Ubuntu 14.04 you'll be installing, and it features an excellent browser-based management tool. It's also nice that you can install a system in minutes using the Proxmox .iso, which you'll turn into a bootable USB disk.

It's important to note that Proxmox is free to use, but offers several paid levels of support. If you want to purchase those services, that's up to you. You won't need to purchase anything for this project though.

**Ubuntu 14.04 LTS:** The long-term release of Ubuntu 14.04 (also known as Trusty Tahr) is solid, stable, flexible and makes a great foundation for all your virtual machines. Let's download and install the 64-bit version, which you'll use to build your virtual machines and VM templates. The operating system also is available in a 32-bit version, which means you can install the same operating system on all your tiny Internet computers and servers—even if some of your equipment is older. When I set up my Dell Mini 9 as a proxy server, for example, I used the 32-bit Ubuntu 14.04 for seamless integration.

You'll make a bootable USB drive from the latest Ubuntu 14.04 .iso; if you're going the DVD route, you'll create a bootable disk.



**Figure 3. Ubuntu Server**



**Figure 4. Webmin**

**Webmin 1.7x:** Webmin is a browser-based tool that makes administering (and understanding) a Linux server a lot easier. Hard-core command-line junkies will scoff at the GUI, but those new to Linux will appreciate Webmin's power and flexibility for managing everything from Apache Web services and Postfix-related mail services to updates and system health. You'll install Webmin on your base Ubuntu 14.04 VM template and use it on every server thereafter.

**apt-mirror:** The goal of the Tiny Internet Project is to build a standalone Internet, and in order to do that, you need to make all the Linux software you want (and might dream of using) available on your private network. To do that, you'll install apt-mirror on one of your Ubuntu VMs.

To replicate the Ubuntu Trusty Tahr Linux distribution, you'll need much more disk space than all your other virtual servers combined. I've done several tests using standard apt-mirror settings (without -src, or source versions), and I found that the main, security and i386 repositories total less than 100GB. When you build the mirror VM, you'll make a 200GB disk, which should give you enough space for future additions and natural growth of the repository.



**Figure 5. apt-mirror**

Once your local mirror is built, Ubuntu updates and upgrade will be very fast. However, initially downloading the mirror will take hours—even over fast Internet connections. Once it's done though, future updates go quickly and you'll have everything you need to build out and experiment with a variety of VM servers.

**Bind9:** In order to make your tiny Internet as real as possible, you'll set up a domain name server (DNS) that will allow you to give your private network a working domain. Bind9 is the latest version of bind, which allows you to set up forward and reverse zones. That means if you want to run a domain called `linuxrocks.com`, you can, and create subdomains like `dns.linuxrocks.com` or `mirror01.linuxrocks.com`. This also will make setting up your private e-mail system a lot easier and much more familiar.

**Postfix and Dovecot:** A big part of the public Internet is e-mail, and your tiny Internet would come up short if it didn't provide this important service. You'll use Postfix for mail handling and Dovecot for POP-ing or IMAP-ing the mail to e-mail clients like Thunderbird. Not only will users of your private tiny Internet be able to have their own e-mail addresses, but they'll also be able to exchange e-mail freely with one another.

These mail services are solid and reliable, and they're supported by a wide array of e-mail clients, including Thunderbird. You'll experiment with securing the mail server too, so you can learn more about mail security.

**LAMP Stack:** The combination of Linux, Apache (Web), MySQL (database) and PHP form the foundation for millions of servers around the world and across the Internet. The combination enables a wide array of Web sites (and content-management systems), database-driven Web applications and much more. You'll deploy a "base" LAMP stack on one of your Ubuntu 14.04 VMs and then make a template of it. That way, you'll be able to deploy as many different Web servers as you want.

The base LAMP VM will include:

- Apache2.
- MySQL.
- PHP 5.x.
- phpMyAdmin.

The last item, phpMyAdmin, is a popular browser-based tool for managing MySQL databases. It's robust and flexible, and perfect for learning more about databases.

**Tinyproxy:** You have a couple options when it comes to connecting your private tiny Internet to the public Internet. One is to have multiple network cards in your main Proxmox host. The other is to have a secondary computer with two network connections to serve as a proxy server. There are advantages to each, so you can decide later which way you want to proceed.

If you take the proxy path—using a separate computer to relay all your http, https and ftp requests—you'll install tinyproxy. It's very lightweight, doesn't require caching (which can take up massive amounts of disk space), and it's fast. You'll have to make some modifications to apt to enable Ubuntu updates via the proxy, but once it's set up, it works well.

**DHCP:** If you've played with a home network—a router provided by your Internet provider, for example—you're probably familiar with

## PART OF THE FUN OF RUNNING YOUR OWN TINY INTERNET IS HAVING COMPLETE CONTROL OVER ALL THE PIECES—THE SERVERS, THE CLIENT PCS AND THE NETWORK.

how DHCP works. A DHCP server hands out IP addresses to all the devices that attach to the network, whether they're computers, tablets, smartphones or thermostats.

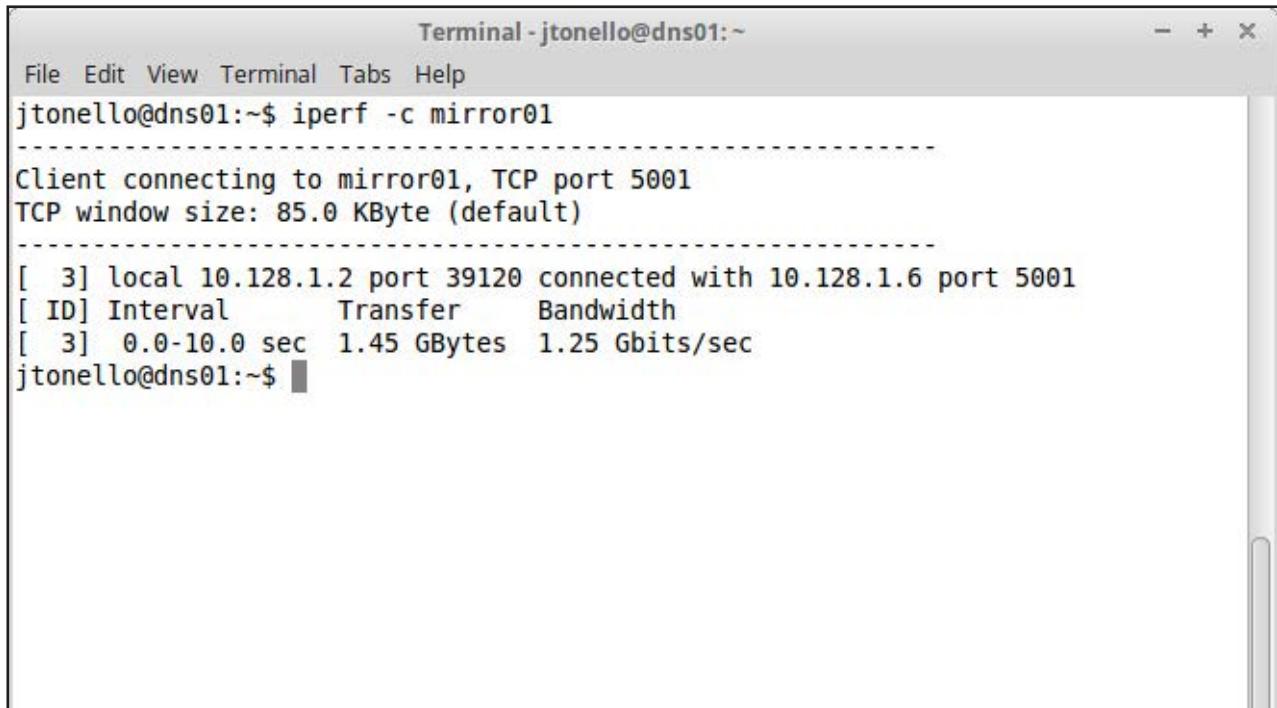
If you're deploying a large tiny Internet—say, in a classroom—having DHCP will make things easy. Yes, you can assign static IP addresses to all the machines on your private network (and some servers, indeed, must have static addresses), but for casual users, DHCP works great.

If you're using a router (such as the Netgear models I mentioned earlier), it probably has its own DHCP server built in. That means the device itself hands out IP addresses to every wired and wireless connection it makes, so all your computers, tablets and smartphones have their own unique addresses. Plug in the router, connect it to your tiny Internet, and you're done.

**isc-dhcp-server:** If instead you're planning to use a simple network switch (one that doesn't hand out addresses at all), you'll need to deploy a DHCP server. Here, you'll use `isc-dhcp-server` for the job. It's lightweight and easy to use. Even if you plan to use a router with built-in DHCP, you may want to deploy a separate `isc-dhcp-server` as a back-up (I'll talk about how to do that later in this series).

Part of the fun of running your own tiny Internet is having complete control over all the pieces—the servers, the client PCs and the network. By installing a simple tool like `iperf`, you'll be able to test the speeds at which your components communicate. If you've ever used on-line tools like Speedtest.net to test your home or workplace Internet download and upload speeds, you'll be familiar with what `iperf` does.

You'll use `iperf` to test the speeds between devices across your private network and to test speeds between servers living together on the Proxmox host. This is where it really gets interesting. Even if you're using



A screenshot of a terminal window titled "Terminal - jtonello@dns01:~". The window shows the output of an iperf command. The output indicates a client connecting to a server at 10.128.1.6 port 5001 via TCP. The TCP window size is set to 85.0 KByte (default). The test shows a single connection (ID 3) with a local port of 39120 connected to 10.128.1.2. The transfer time is 0.0-10.0 sec, and the bandwidth achieved is 1.45 GBytes (1.25 Gbits/sec).

```
Terminal - jtonello@dns01:~  
File Edit View Terminal Tabs Help  
jtonello@dns01:~$ iperf -c mirror01  
-----  
Client connecting to mirror01, TCP port 5001  
TCP window size: 85.0 KByte (default)  
-----  
[ 3] local 10.128.1.2 port 39120 connected with 10.128.1.6 port 5001  
[ ID] Interval Transfer Bandwidth  
[ 3] 0.0-10.0 sec 1.45 GBytes 1.25 Gbits/sec  
jtonello@dns01:~$ █
```

**Figure 6.** iperf

a 10Mb Ethernet switch to connect your VM host machine with your laptop, for example, you'll be able to measure gigabit speeds between your various VMs. That's because they're connected by a virtual network on a single server, limited only by the speed of the server's bus! That'll make your tiny Internet a speedy and realistic place to explore Linux.

## COMING NEXT

In the next installments, you'll take the information I've covered here and build your Proxmox host, Ubuntu mirror and e-mail and domain name servers. After that, you'll deploy a LAMP stack and build some Websites, databases and even a WordPress site.■

---

**John Tonello** is the Director of IT for NYSERNet Inc., New York state's regional optical networking company. He's been a Linux user and enthusiast since building his first Slackware system from diskette 20 years ago. Since then, he's developed Web and IT solutions for major universities, Fortune 500 companies and small start-ups. A former Cornell University IT trainer and writer, John served six years as the mayor of an Upstate New York city, where he championed the use of technology to help solve problems facing municipalities.



# NEW ORLEANS

DRUPALCON 2016

ERNEST N. MORIAL CONVENTION CENTER  
MAY 9 - 13, 2016



## Join us in the Big Easy.

With Drupal 8 newly released and thousands of community members in attendance, DrupalCon New Orleans promises to be an event to remember.

See you in New Orleans this May.  
Laissez les Bon Temps Rouler!

[neworleans2016.drupal.org](http://neworleans2016.drupal.org)

### Resources

Test your computer to see if it supports virtualization:

- Windows Users: <http://www.technorms.com/8208/check-if-processor-supports-virtualization>
- Linux Users: <http://virt-tools.org/learning/check-hardware-virt>

Create Bootable USB Sticks:

- <http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-windows>
- <http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-mac-osx>
- <http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-ubuntu>

Software Resources:

- Proxmox: <http://www.proxmox.com>
- Ubuntu: <http://www.ubuntu.com>
- Webmin: <http://www.webmin.com>
- Postfix: <http://www.postfix.org>
- Dovecot: <http://www.dovecot.org>
- Apache: <http://apache.org>
- MySQL: <http://www.mysql.com>
- PHP: <http://www.php.net>
- PhpMyAdmin: <https://www.phpmyadmin.net>
- Tinyproxy: <https://banu.com/tinyproxy>
- WordPress: <https://wordpress.com>

Other Useful Resources:

- GNOME: <https://www.gnome.org>
- KDE: <https://www.kde.org>
- Xfce: <http://www.xfce.org>

Send comments or feedback via  
<http://www.linuxjournal.com/contact>  
or to [leditor@linuxjournal.com](mailto:leditor@linuxjournal.com).

**RETURN TO CONTENTS**



Where every interaction matters.

# break down your innovation barriers

## **power your business to its full potential**

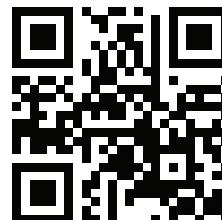
When you're presented with new opportunities, you want to focus on turning them into successes, not whether your IT solution can support them.

Peer 1 Hosting powers your business with our wholly owned FastFiber Network™, global footprint, and offers professionally managed public and private cloud solutions that are secure, scalable, and customized for your business.

Unsurpassed performance and reliability help build your business foundation to be rock-solid, ready for high growth, and deliver the fast user experience your customers expect.

**Want more on cloud?**

**Call: 844.855.6655 | [go.peer1.com/linux](http://go.peer1.com/linux) | View Cloud Webinar:**



## FREE DOWNLOADS

### The Forrester Wave™: Digital Experience Platforms, Q4 2015

The demand to be at every touchpoint in the customer lifecycle is no longer an option—it's a requirement. To manage and deliver experiences consistently across all touchpoints, organizations are looking to digital experience platforms as the foundation of their digital presence.

Get Forrester's evaluation of the best vendors, including:

- The ten providers that matter most.
- How each vendor stacks up to Forrester's criteria.
- Six needs a digital experience platform architecture must meet.

> <http://geekguide.linuxjournal.com/content/forrester-wave-digital-experience-platforms-q4-2015>

---

### ACQUIA™ The Ultimate Guide to Drupal 8 by Acquia

With 200+ new features and improvements, Drupal 8 is the most advanced version of Drupal yet. Drupal 8 simplifies the development process, enabling you to do more, in less time, with proven technologies that make it easier to be a first time Drupal user. Read this eBook, written by Angie Byron (you may know her as "webchick"), to get up to speed on the new changes in Drupal 8. Drupal 8's improvements include:

- API-driven content approach.
- Rest-first native web services.
- Seamless integration with existing technologies.
- Multilingual features and capabilities.
- Responsive by nature and mobile-first.

> <http://geekguide.linuxjournal.com/content/ultimate-guide-drupal-8>

---

### ACQUIA™ How to Choose a Great CMS by Acquia

Web Content Management Systems serve as the foundation of your digital experience strategy. Yet many organizations struggle with legacy proprietary products that can't keep pace with the new realities of digital marketing. To determine if you are in need of a new CMS, use our guide, which includes:

- An evaluation to see if your current CMS supports your digital business strategy.
- The top considerations when selecting a new CMS.
- A requirements checklist for your next CMS.
- Ten questions to ask CMS vendors.

> <http://geekguide.linuxjournal.com/content/how-choose-great-cms>

# FREE DOWNLOADS

## Fast/Flexible Linux OS Recovery

How long does it take to restore a system, whether virtual or physical, back to the exact state it was prior to a failure? Re-installing the operating system, re-applying patches, re-updating security settings takes too damn long! If this is your DR Strategy, we hope you've documented every change that's been made, on every system?!

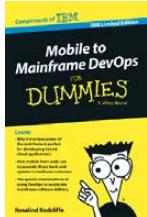
Most companies incorporate backup procedures for critical data, which can be restored quickly if a loss occurs. However, that works only if you have an OS to restore onto and the OS supports the backup.

In this live one-hour webinar, learn how to enhance your existing backup strategies for complete disaster recovery preparedness using Storix System Backup Administrator (SBAdmin), a highly flexible full-system recovery solution for UNIX and Linux systems.

Webinar: April 26, 2016 at 1:00 PM Eastern

> <http://www.linuxjournal.com/storix-recovery>

---



## Mobile to Mainframe DevOps for Dummies

In today's era of digital disruption empowered by cloud, mobile, and analytics, it's imperative for enterprise organizations to drive faster innovation while ensuring the stability of core business systems. While innovative systems of engagement demand speed, agility and experimentation, existing systems of record require similar attributes with additional and uncompromising requirements for governance and predictability. In this new book by Rosalind Radcliffe, IBM Distinguished Engineer, you will learn about:

- Responding to the challenges of variable speed IT.
- Why the mainframe is a unique and ideal platform for developing hybrid cloud applications.
- How mobile front ends can rejuvenate back-end systems to reach new customers.
- And, special considerations for using a DevOps approach to accelerate mainframe software delivery.

> <http://devops.linuxjournal.com/devops/mobile-mainframe-devops-dummies>

---

### BRAND-NEW EDITION!

## DevOps For Dummies - New Edition with SAFe®

In this NEW 2nd edition, learn why DevOps is essential for any business aspiring to be lean, agile, and capable of responding rapidly to changing customers and marketplace.

Download the E-book to learn about:

- The business need and value of DevOps.
- DevOps capabilities and adoption paths.
- How cloud accelerates DevOps.
- The Ten DevOps myths.
- And more.

> <http://devops.linuxjournal.com/devops/devops-dummies-new-edition-safe>

# Privacy and the New Math

Our last refuge is with our first principles.

DOC SEARLS and T.ROB WYATT



**DOC SEARLS**

---

Doc Searls is Senior Editor of *Linux Journal*. He is also a fellow with the Berkman Center for Internet and Society at Harvard University and the Center for Information Technology and Society at UC Santa Barbara.

---

## PREVIOUS

◀ Feature: The Tiny Internet Project, Part I

---

*Among the countless essays and posts I've read on the fight over crypto that's been going on between Apple and the FBI ([https://en.wikipedia.org/wiki/FBI-Apple\\_encryption\\_dispute](https://en.wikipedia.org/wiki/FBI-Apple_encryption_dispute)), one by the title above by T.Rob Wyatt (<https://medium.com/@tdotrob>) in Medium stood out so well that I asked if he'd like to help me adapt it into an EOF here in Linux Journal. He said yes, and here it is.—Doc*

---

T.Rob Wyatt is a security specialist with 30 years of industry experience. His company is IoPT Consulting (<https://iopconsulting.com>).

**I**n the Apple vs. FBI case, the real disputes are between math and architecture, and between open and closed. Linux can play an important role in settling those disputes, because it is on the right side of both.

Apple's case is for crypto, an application of math. The FBI's case is for a way through the crypto. The term for that architectural hole is a "back door". Since

the key to that door would be the FBI's alone, with no way for others to tell how or when they'll use it (unless the FBI shares it), the FBI's side is the closed one.

To unpack this, let's look at the case.

At a PR level, the FBI would like an outcome it says is consistent with our moral outrage over the mass murders in San Bernardino by terrorists who were killed by police and left behind an Apple iPhone 5c that the FBI wants Apple's help opening. The phone's owner was a guy whose rights we wouldn't care much about even if he were still alive, so there is little public interest served in keeping the information private.

The FBI would also like to solve what it calls the "Going Dark Issue" (<https://www.fbi.gov/about-us/otd/going-dark-issue>). Specifically, the growing use of encryption on the Internet is "eroding law enforcement's ability to quickly obtain valuable information that may be used to identify and save victims, reveal evidence to convict perpetrators, or exonerate the innocent."

Getting into the dead perp's iPhone and solving the "going dark problem" both require ways for the FBI to "see the light", we might say, through the crypto. Or, literally, for math to work one way for everybody using crypto and another way for the FBI.

While the FBI contends that the country's safety depends on "law enforcement's lawful intercept and evidence collection needs", in fact, it also depends on the math we call crypto to keep commerce and infrastructure up and running. To serve both these needs, math has to work differently for the FBI than for everyone else. But math works the same for everyone, and taking action inconsistent with this principle leads predictably to bad outcomes.

In order both to break security for the government's benefit and continue to use it for infrastructure and commerce, the government must keep the tools and methods that enable such breakage secret at all costs. But if you have a secret that breaks digital security, you don't use digital security to secure it. You use vaults, guns and worse. Once you have such a capability, keeping it secret requires tipping the balance of power away from individuals and toward the government.

The ability of individuals to keep an expressed thought secret is one of the checks and balances that nudges the power differential toward

homeostasis somewhere below Citizens 0, Government 100. Breaking crypto in commercial products eliminates the ability of citizens to keep their expressed thoughts secret and in doing so eliminates an important constraint on government power escalation. Because math works the same for everyone, eliminating one individual's security from government intrusion eliminates it for everybody.

The phone is the most intimate personal data repository in widespread use on the planet. If checks and balances fail to protect the phone, the power differential from a practical standpoint is already at Citizens 0, Government 100. So this isn't about breaking a phone. It's about breaking a system. Once it's broken, it stays broken.

Tangential to this is the argument that cracking this one phone doesn't compromise all the others. That too is provably false, and quite easily so.

See, a security model includes not just the crypto but all of the trust anchors and controls in the system. The high profile breaches in the news are almost never due to breaking the crypto, but rather from breaking one or more trust anchors in the security model.

Resilience against brute-force attack is a critical control in the iPhone's security model. This is because the cryptography is impenetrable, but human-chosen passwords are surprisingly easy to crack. According to Apple's iOS Security guide (dated September 2015, [https://www.apple.com/business/docs/iOS\\_Security\\_Guide.pdf](https://www.apple.com/business/docs/iOS_Security_Guide.pdf)):

Only 10 attempts to authenticate and retrieve an escrow record are allowed. After several failed attempts, the record is locked and the user must call Apple Support to be granted more attempts. After the 10th failed attempt, the HSM cluster destroys the escrow record and the keychain is lost forever. This provides protection against a brute-force attempt to retrieve the record, at the expense of sacrificing the keychain data in response.

Designing the phone to wipe the data after some number of failed attempts compensates for the human tendency to pick really bad passwords. Defeating that control—which is what the government wants—breaks the security model.

For this to be okay requires that math works differently for this one

phone than for all others—or for the math to work differently for government than for everyone else. But, since math works the same for everyone, the government must keep the hack secret at all costs, including escalation to a Citizen 0, Government 100 power differential if necessary. It would, after all, be “in the interest of national security” to do so, and that always supersedes the interest of any one individual.

The FBI’s case also requires that we fully trust it not to mess up. Yet it appears it already has in the San Bernardino case, says Apple (<http://www.apple.com/customer-letter/answers>):

One of the strongest suggestions we offered was that they pair the phone to a previously joined network, which would allow them to back up the phone and get the data they are now asking for. Unfortunately, we learned that while the attacker’s iPhone was in FBI custody the Apple ID password associated with the phone was changed. Changing this password meant the phone could no longer access iCloud services.

Reports *Computerworld*, “the FBI didn’t directly contest that”, but also wants additional information not backed up in iCloud (<http://www.computerworld.com/article/3036183/apple-ios/fbi-rebuts-criticism-that-it-reset-terrorists-icloud-password-after-attack.html>). Specifically:

Authorities want Apple to create a modified version of iOS that disables an auto-erase feature—triggered after 10 incorrect passcode entries—and removes the forced delays between passcode guesses. The FBI would then conduct a brute-force passcode crack from a personal computer at high speeds to uncover the passcode—which unlocks the device—and to examine all the data there.

Apple calls this a back door. The FBI insists it is not. In Hollywood, a back door gives an attacker direct login to a system, but in real life, the term refers to an intentional weakness in the security model. Or, in the words of the Jargon File, “a hole in the security of a system deliberately left in place by designers or maintainers” (<http://www.catb.org/jargon/html/B/back-door.html>). Removing the auto-wipe triggered by too many

failed password attempts is a hole in the security model big enough to drive a simulated truck through. Point goes to Apple on this one.

Corporations are much more susceptible to government coercion than a distributed Open Source community, such as the ecosystem that has grown up around Linux. And Apple itself may not be entirely clean and consistent on the matter of safeguarding individual privacy. Stewart A. Baker (<https://www.lawfareblog.com/contributors/sbaker>), a former official with the Department of Homeland Security and the National Security Agency, wrote a blog post in February titled “Deposing Tim Cook” (<https://www.lawfareblog.com/deposing-tim-cook>), in which he suggests that Apple may make compromises for the Chinese government that it won’t for the US one.

And let’s not forget that the US government is not of one mind on this. In court, we have the Communications Assistance for Law Enforcement Act (CALEA, [https://en.wikipedia.org/wiki/Communications\\_Assistance\\_for\\_Law\\_Enforcement\\_Act](https://en.wikipedia.org/wiki/Communications_Assistance_for_Law_Enforcement_Act)) written in 1994 vs. the All Writs Act ([https://en.wikipedia.org/wiki/All\\_Writs\\_Act](https://en.wikipedia.org/wiki/All_Writs_Act)) written in 1789. Administratively, we have the FBI vs. other government agencies with overlapping jurisdictions. Richard A. Clarke ([https://en.wikipedia.org/wiki/Richard\\_A.\\_Clarke](https://en.wikipedia.org/wiki/Richard_A._Clarke)), who held a number of high-level security positions under Ronald Reagan, Bill Clinton and both Bushes, said this in an interview with NPR (<http://www.npr.org/2016/03/14/470347719/encryption-and-privacy-are-larger-issues-than-fighting-terrorism-clarke-says>):

I think the Justice Department and the FBI are on their own here. You know, the secretary of defense has said how important encryption is when asked about this case. The National Security Agency director and three past National Security Agency directors, a former CIA director, a former Homeland Security secretary have all said that they’re much more sympathetic with Apple in this case. You really have to understand that the FBI director is exaggerating the need for this and is trying to build it up as an emotional case, organizing the families of the victims and all of that. And it’s Jim Comey and the attorney general is letting him get away with it.

Whichever way this case is decided, it is clear that the US and many

other governments around the world would eliminate the right and ability of their citizens to keep a secret. The government's ability to coerce corporations casts doubt on the integrity of the code those corporations produce. The "Open with a capital O" in Open Source is itself a security control that resists attacks by preserving the integrity of the code. If someone compromised open-source code, it would be possible to find it and back out the changes.

In recent years, governments have made an enemy of personal privacy, regarding it as a vulnerability within the state and a potential refuge for terrorism. That's why many vendors of secure hardware and software have fled their home countries and relocated to privacy-friendly jurisdictions.

When people stop worrying so much about the merits of a specific case and consider that the FBI (and, if it succeeds, the whole government) wants to destroy our underlying security models, geography won't matter because math works the same everywhere on the planet. At that point, the resilience of the security model and the supporting code will be the most important consideration. We will have a migration toward privacy-friendly, open-source technology, and Linux is the leading expression of that.

If the US government succeeds in its bid to break Apple's security model, its next step is to prohibit Apple from fixing the vulnerability. After that comes mandated back doors and a general prohibition on unbreakable information systems. Those sanctions would be relatively

## ADVERTISER INDEX

ADVERTISER	URL	PAGE #
DrupalCon New Orleans	<a href="http://neworleans2016.drupal.org">http://neworleans2016.drupal.org</a>	119
Drupalize.me	<a href="http://drupalize.me">http://drupalize.me</a>	131
O'Reilly OSCON	<a href="http://www.oreilly.com/pub/cpc/5732">http://www.oreilly.com/pub/cpc/5732</a>	61
O'Reilly Strata	<a href="http://conferences.oreilly.com/strata">http://conferences.oreilly.com/strata</a>	7
Peer 1 Hosting	<a href="http://go.peer1.com/linux">http://go.peer1.com/linux</a>	121
Perl6	<a href="http://PerlConference.Org">http://PerlConference.Org</a>	81
SPTechCon	<a href="http://www.sptechcon.com/">http://www.sptechcon.com/</a>	103
WeatablesDevCon	<a href="http://www.wearablesdevcon.com">http://www.wearablesdevcon.com</a>	17
WITI Women in Technology Summit	<a href="http://www.witi.com/conferences/2016/summit/">http://www.witi.com/conferences/2016/summit/</a>	79

**Thank you as always for supporting our advertisers by buying their products!**

### ATTENTION ADVERTISERS

The *Linux Journal* brand's following has grown to a monthly readership nearly one million strong. Encompassing the magazine, Web site, newsletters and much more, *Linux Journal* offers the ideal content environment to help you reach your marketing objectives. For more information, please visit <http://www.linuxjournal.com/advertising>

easy to enforce on domestic corporations but much more difficult against a worldwide development community. The good news is that this is the easiest call to action ever: Just keep doing what you do. Participation in the Linux community is the most important security control in the whole open-source model.

It's interesting to think about how much and how easily we suspend our disbelief when it comes to security. Consider, for example, the entire *Star Wars* franchise. When R2D2 needs to do some research, the physical data ports are all compatible. So are the protocols at all communication layers. Access is unlogged. All the systems involved provide sensitive confidential details to anonymous queries, and neither the queries nor command and control traffic are alarmed. Even my worst consulting clients are ten times better at security than The Empire.

As unbelievable as the security was in *Star Wars*, George Lucas stopped well short of asking us to believe that math works differently for the Empire than it does for the rebels. The US government asks that of us and more. Even if we are inclined to accept the government's proposition, it's one thing to put up with that level of surrealism for an hour or so in a theater. It's something else entirely when the future of privacy is at stake.■

Send comments or feedback via  
<http://www.linuxjournal.com/contact>  
or to [ljeditor@linuxjournal.com](mailto:ljeditor@linuxjournal.com).

**RETURN TO CONTENTS**



## Instant Access to Premium Online Drupal Training

- ✓ Instant access to hundreds of hours of Drupal training with new videos added every week!
- ✓ Learn from industry experts with real world experience building high profile sites
- ✓ Learn on the go wherever you are with apps for iOS, Android & Roku
- ✓ We also offer group accounts. Give your whole team access at a discounted rate!

Learn about our latest video releases and offers first by following us on Facebook and Twitter (@drupalizeme)!

Go to <http://drupalize.me> and get Drupalized today!

