Multiprocessing in Python

Smart-Home Hacks

Vendor Lock-In and the Cloud

LINUX JOURNAL

Since 1994: The original magazine of the Linux community

DEEP DIVE INTO THE CLOUD

PLUS

CLOUD HARDENING TIPS

Basic Principles for Securing Your Cloud Environment

OPEN SOURCE HISTORY

A Talk with OSI President Simon Phipps

INTRODUCING ONNX

The Open Neural Network Exchange Format

ISSUE 285 | APRIL 2018
www.linuxjournal.com

CONTENTS

APRIL 2018
ISSUE 285

CONTENTS

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Linux Journal, LLC. Subscription-related correspondence may be sent to 9597 Jones Rd, #331, Houston, TX 77065 USA. Subscription rate is $34.50/year. Subscriptions start with the next issue.

ARTICLES

AT YOUR SERVICE

SUBSCRIPTIONS: *Linux Journal* is available as a digital magazine, in PDF, EPUB and MOBI formats. Renewing your subscription, changing your email address for issue delivery, paying your invoice, viewing your account details or other subscription inquiries can be done instantly online: http://www.linuxjournal.com/subs. Email us at subs@linuxjournal.com or reach us via postal mail at *Linux Journal*, 9597 Jones Rd #331, Houston, TX 77065 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 different 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*, 9597 Jones Rd #331, Houston, TX 77065 USA. Letters may be edited for space and clarity.

SPONSORSHIP: We take digital privacy and digital responsibility seriously. We've wiped off all old advertising from *Linux Journal* and are starting with a clean slate. Ads we feature will no longer be of the spying kind you find on most sites, generally called "adtech". The one form of advertising we have brought back is sponsorship. That's where advertisers support *Linux Journal* because they like what we do and want to reach our readers in general. At their best, ads in a publication and on a site like *Linux Journal* provide useful information as well as financial support. There is symbiosis there. For further information, email: sponsorship@linuxjournal.com or call +1-281-944-5188.

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 online: http://www.linuxjournal.com/author.

NEWSLETTERS: 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.

LINUX JOURNAL

EDITOR IN CHIEF: Doc Searls, doc@linuxjournal.com

EXECUTIVE EDITOR: Jill Franklin, jill@linuxjournal.com

TECH EDITOR: Kyle Rankin, lj@greenfly.net

ASSOCIATE EDITOR: Shawn Powers, shawn@linuxjournal.com

CONTRIBUTING EDITOR: Petros Koutoupis, petros@linux.com

CONTRIBUTING EDITOR: Zack Brown, zacharyb@gmail.com

SENIOR COLUMNIST: Reuven Lerner, reuven@lerner.co.il

SENIOR COLUMNIST: Dave Taylor, taylor@linuxjournal.com

PUBLISHER: Carlie Fairchild, publisher@linuxjournal.com

ASSOCIATE PUBLISHER: Mark Irgang, mark@linuxjournal.com

DIRECTOR OF DIGITAL EXPERIENCE:
Katherine Druckman, webmistress@linuxjournal.com

GRAPHIC DESIGNER: Garrick Antikajian, garrick@linuxjournal.com

ACCOUNTANT: Candy Beauchamp, acct@linuxjournal.com

COMMUNITY ADVISORY BOARD
John Abreau, Boston Linux & UNIX Group; John Alexander, Shropshire Linux User Group;
Robert Belnap, Classic Hackers UGA Users Group; Aaron Chantrill, Bellingham Linux Users Group;
Lawrence D'Oliveiro, Waikato Linux Users Group; David Egts, Akron Linux Users Group;
Michael Fox, Peterborough Linux User Group; Braddock Gaskill, San Gabriel Valley Linux Users' Group;
Roy Lindauer, Reno Linux Users Group; Scott Murphy, Ottawa Canada Linux Users Group;
Andrew Pam, Linux Users of Victoria; Ian Sacklow, Capital District Linux Users Group;
Ron Singh, Kitchener-Waterloo Linux User Group; Jeff Smith, Kitchener-Waterloo Linux User Group;
Matt Smith, North Bay Linux Users' Group; James Snyder, Kent Linux User Group;
Paul Tansom, Portsmouth and South East Hampshire Linux User Group;
Gary Turner, Dayton Linux Users Group; Sam Williams, Rock River Linux Users Group;
Stephen Worley, Linux Users' Group at North Carolina State University;
Lukas Yoder, Linux Users Group at Georgia Tech

Linux Journal is published by, and is a registered trade name of,
Linux Journal, LLC. 4643 S. Ulster St. Ste 1120 Denver, CO 80237

SUBSCRIPTIONS
E-MAIL: subs@inuxjournal.com
URL: www.linuxjournal.com/subscribe
Mail: 9597 Jones Rd, #331, Houston, TX 77065

SPONSORSHIPS
E-MAIL: sponsorship@linuxjournal.com
Contact: Publisher Carlie Fairchild
Phone: +1-281-944-5188

How Wizards and Muggles Break Free from the Matrix

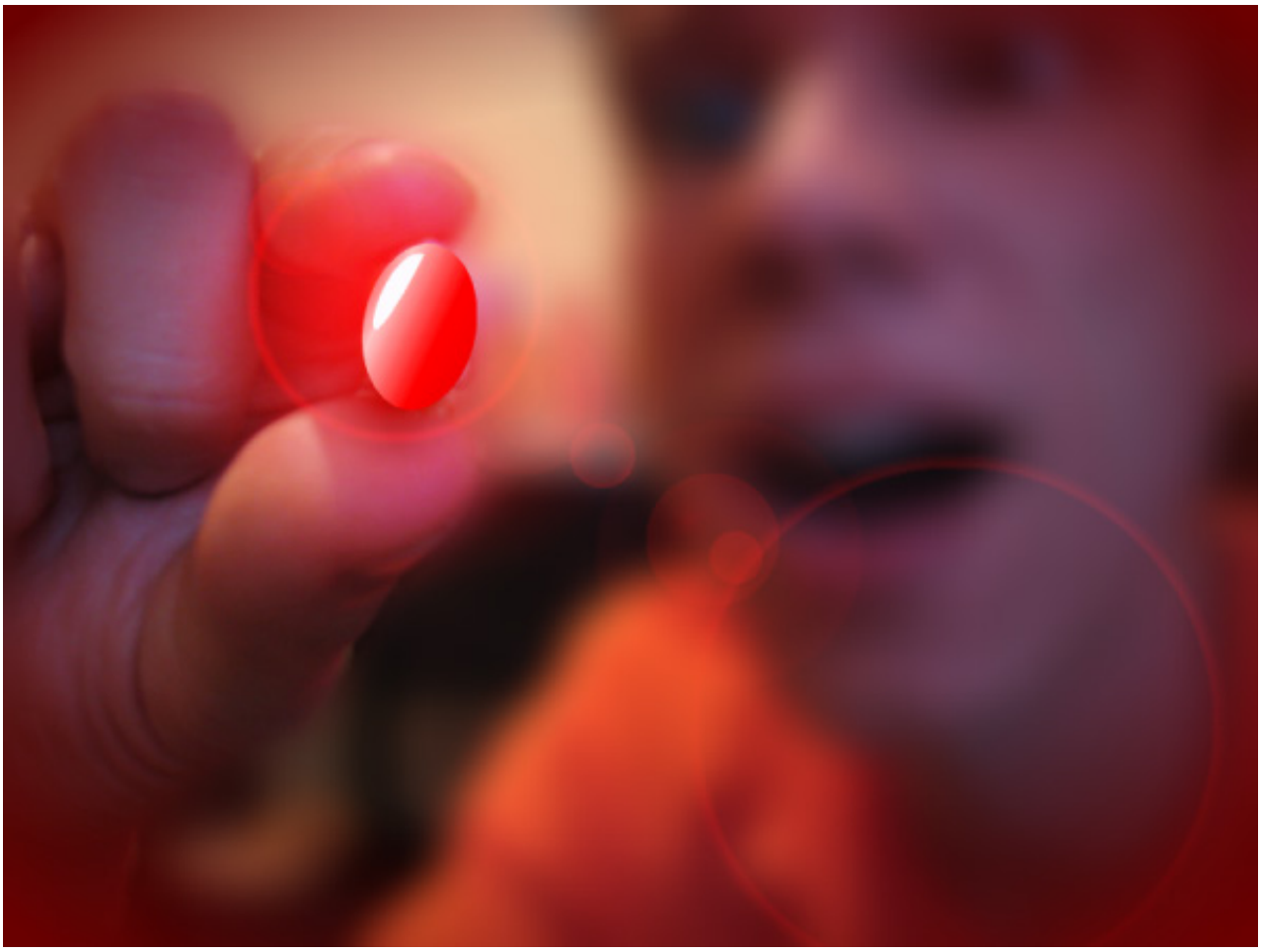
Doc Searls is a veteran journalist, author and part-time academic who spent more than two decades elsewhere on the *Linux Journal* masthead before becoming Editor in Chief when the magazine was reborn in January 2018. His two books are *The Cluetrain Manifesto*, which he co-wrote for Basic Books in 2000 and updated in 2010, and *The Intention Economy: When Customers Take Charge*, which he wrote for Harvard Business Review Press in 2012. On the academic front, Doc runs ProjectVRM, hosted at Harvard's Berkman Klein Center for Internet and Society, where he served as a fellow from 2006–2010. He was also a visiting scholar at NYU's graduate school of journalism from 2012–2014, and he has been a fellow at UC Santa Barbara's Center for Information Technology and Society since 2006, studying the internet as a form of infrastructure.

First we invented a world where everyone could be free. Then we helped build feudal castles on it, where everyone now lives. Now it's time to blow up those castles by giving everybody much better ways to use their freedom than they ever would enjoy in a castle.

By Doc Searls

I'm going to mix movie metaphors here. You'll see why.

In April 1999, a few weeks after *The Matrix* came out, the entire *Linux Journal* staff watched it in a theater not far from our headquarters at the time, in Seattle's Ballard neighborhood. While it was instantly clear to us that the movie was geek classic (hell, its hero was an ace programmer), it also was clear that the title subject—a fully convincing fake world occupied by nearly the whole human species—was an allegory (which Wikipedia calls "a metaphor whose vehicle may be a character, place or



event, representing real-world issues and occurrences").

One obvious interpretation was religious. Neo was a Christ-like savior, resurrected by a character named Trinity, who played the Holy Spirit role after Neo got killed by the Satan-like Agent Smith—all while the few humans not enslaved by machines lived in an underground city called Zion.

When the second and third installments came out in the years that followed, more bits of the story seemed borrowed from other religions: Buddhism, Gnosticism and Hinduism. Since the Wachowski brothers, who wrote and directed the films, have become the Wachowski sisters, you also can find, in retrospect, plenty of transgender takes on the series.

Then there's the philosophical stuff. Prisoners in the Matrix believe the world they inhabit is real, much as prisoners in Plato's *Allegory of the Cave* believe the shadows they see on a wall are real, because they can't tell the source of light is a fire behind them. In Plato's story, one prisoner is set free to visit the real world. In *The Matrix*, that one prisoner is Neo, his name an anagram for "The One" whose job is to rescue everybody or at least save Zion. (Spoiler: he does.)

But I didn't buy any of that, because already I saw marketing working to turn the free and open online world into a commercial habitat where—as in the fictional Matrix—human beings were reduced to batteries for giant robotic machines that harvested human attention, which they then processed and fed back to humans again.

This was the base design of the world marketing wanted to make for us in the digital age: one where each of us were "targets", "captured", "acquired", "controlled", "managed" and "locked in", so personalized "content" and "experiences" could be "delivered" to our ears and eyeballs. Marketers talked like that long before the internet showed up, but with every eyeball suddenly addressable, *personally*, the urge to jack us into marketing's Matrix became irresistible.

In fact, one reason four of us posted *The Cluetrain Manifesto* on the web that very same month was that we wanted to make clear that the internet was for everybody, not just marketing.

But, popular as *Cluetrain* was (especially with marketers), marketing got engineering—including plenty of Linux wizards—to build a Matrix for us. We live there now. Unless you have your hardware and software rigged for absolute privacy while roaming about the online world (and can you really be sure?), you're in marketing's Matrix.

The obvious parts of that Matrix are maintained by Google, Facebook, LinkedIn, Twitter, Tumblr, Pinterest, Amazon and so on. Much more of it is provisioned by names you never heard of. To see what they're up to, equip your browser with a form of tracking protection that names sources of tracking files. (Examples are

Baycloud Bouncer, Disconnect, Ghostery, Privacy Badger and RedMorph.) Then point your browser to the website of a publisher whose business side has been assimilated by the Agent Smith called "adtech"—*The Los Angeles Times*, for example. Then, check your tracking-protection tool's list of all the entities trying to spy on you.

Here are just some of the 57 suspects that Privacy Badger found for me on the *LA Times* index page:

- yieldmo.com
- trbas.com
- trbimg.com
- trbdss.com
- steelhousemedia.com
- teads.tv
- trb.com
- truste.com
- revjet.com
- rflhub.com
- rubiconproject.com
- steelhousemedia.com
- moatads.com
- ntv.io
- openx.net
- postrelease.com
- keewee.co
- krxd.net
- mathtag.net
- ml314.net
- indexwww.com
- ixiaa.com
- ensighten.com
- everesttech.net

- tronc.com
- sitescout.com
- jquery.com
- bootstrapcdn.com
- bouncexchange.com
- chartbeat.com
- cloudfront.net
- agkn.com
- adsrvr.org
- gomnit.com
- responsiveads.com
- postrelease.com

Many of those appear more than once, with different prefixes. I've also left off variants of google, doubleclick, facebook, twitter and other familiars.

Interesting: when I look a second, third or fourth time, the list is different—I suppose because third-party ad servers are busy trying to shove trackers into my browser afresh, as long as a given page is open.

When I looked up one of those trackers, "moatads", which I chose at random, most of the 1,820,000 search results were about how moatads is bad stuff. In order, this is the first page of search results:

- Remove Moatads virus (Removal Guide) - Oct 2017 update - 2 Spyware
- Moatads Malware Removal (What is moatads?) March 2018 Update ...
- What is z.moatads.com? - Webroot Community
- moatads.com
- How to remove Moatads.com fully - It-Help.info
- Uninstall Moatads virus (Uninstall Guide) - Oct 2017 updated
- Moatads Malware Removal | Mobile Security Zone
- Moatads Removal Guide | Spyware Techie
- This keeps cropping up and is a real problem. How do i get rid of it..

The fourth item says the company behind moatads, moat.com, "measures real-time Attention Analytics over 33 billion times per day". And that's just one Matrix-builder.

Clearly there is no Architect or Oracle building this Matrix, or it wouldn't suck so bad. That's to our advantage, but we're still stuck in an online world where spying is the norm rather than the exception, and personal autonomy is mostly contained within the castles of giant service providers, "social networks" and makers of highly proprietary gear.

Way back in 2013, Shoshana Zuboff called on us to "be the friction" against "the New Lords of the Ring". In later essays, she labeled the whole spying-fed advertising system both surveillance capitalism and The Big Other. If things go according to plan, her new book, *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power*, will come out soon. (Here's the Amazon link.)

People are already fighting back, whether they know it or not. PageFair's 2017 Adblock Report says at least 11% of the world's population is now blocking ads on at least 615 million devices. GlobalWebIndex says 37% of all the world's mobile users were blocking ads by January of 2016 and another 42% wanted to do so as well. Statista says the number of mobile-phone users in the world would reach 4.77 billion at some point this past year. Combine those last two numbers, and you get more than 1.7 billion people blocking ads already—a sum exceeding the population of the Western Hemisphere. All of which is why I called ad blocking the world's biggest boycott, way back in 2015. Today I'd rather think of it as a slave revolt.

But we need to be more than freed slaves. We need to be, as Shoshana says, masters of our own lives and of all the relationships we have online.

In *The Matrix*, Morpheus asks the still-captive Neo if he believes in fate. "No", says Neo, "because I don't like the idea that I'm not in control of my life."

We can't be in control of our lives as long as those lives are lived within corporate castles and we lack the tools for mastery over our virtual bodies and minds online.

It doesn't matter if Facebook, Google and the rest have no malicious intent, or if they really do want to "bring the world closer together", or "organize the world's information and make it universally accessible and useful", or "develop services that significantly improve the lives of as many people as possible". We need to be free and independent agents of our selves.

That can't happen inside the client-server systems we've had online since 1995 and earlier—systems that might as well be called slave-master. It can't happen as long as we always have to click "accept" to the terms and conditions of the online world's defaulted slave-master system. It can't happen as long as everything useful in the online world requires a login and a password. Each of those norms are walls in what Morpheus calls "a prison for your mind".

We have to think and work outside the walls in those prisons (formerly castles). And it isn't enough to free ourselves. To switch movie metaphors, *it's time for the wizards to free the muggles*. Here's a punch list of what we need to do:

- Make companies agree to our terms, rather than the other way around. As I explained in my column last month, the first company on Earth to do that is *Linux Journal*.

- Control our own self-sovereign identities and manage all the ways we are known (or not) to the administrative systems of the world. This means we will be able to...

- Sideline logins and passwords. Those are legacies of the mainframe world. Think instead of *verifiable claims*—and join one of the groups working on those. (This might help: when you're standing in line at a coffee shop, you're making a verifiable claim that you're a customer. You don't need to log in to do business there.)

- Change what all the companies we deal with know about us—such as our phone number, our last name or our email address, *in one move*.

- Pay what we want, where we want it, and how we pay for whatever we want, in our own ways.

- Call for service or support in one simple and straightforward way of our own, rather than in as many ways as there are 800 numbers to call and punch numbers into a phone before we wait on hold listening to promotions interrupted by bad music, or vice versa.

- Express loyalty in our own ways—ones that are genuine rather than coerced.

- Have a true Internet of Things, meaning one in which our things are truly ours and not just suction cups on corporate tentacles. Make our things truly ours by giving each its own cloud, which each of us will control as well.

- Own and control all our health and fitness records and how others use them.

- Help companies by generously sharing helpful facts about how we use their products and services—but in our own ways, through standard tools that work the same for every company we deal with.

- Have wallets and shopping carts that are our own—ones we can take from site to site and from app to app.

At the end of *The Matrix* trilogy, Neo succeeds at stopping the viral Agent Smith program from destroying both the machine and human worlds. But there is no vision of what all the people jacked into the Matrix would do once they were free—or if freedom was in the cards at all. In fact, all Neo does is save Zion and leave the rest of humanity living in the same old Matrix: a machine-maintained illusory existence where their only true purpose was to power the Matrix as batteries.

That previous bulleted list is a set of visions missed by both *The Matrix* and the *Harry Potter* movies. All of them give everybody far more power than even the wizards of the world—our readers and writers—now possess.

Fortunately, the internet isn't Hogwarts. Though it's a product of wizardry, everybody—wizards included—live and think inside its walls. But digital technology and the internet were designed for freedom, and not just for more enslavement on the industrial model.

So let's finish making online civilization something better than the digital feudal system we have now. ■

[Note: an ancestor of this article appeared on the *Linux Journal* website in January 2018.]

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.

LETTERS

Welcome Back, *LJ*. It Feels Like Forever!

I am very excited by the new issue! I am still digesting it, but it feels just like the old days! I had (am having?) a difficult time transitioning from print media to digital magazines. I subscribed sporadically to the digital edition after the demise of the print format in order to try to support your continued efforts, but I never found it as useful or comfortable, and I often went months without reading new issues.

I can't say enough about the depth of the articles so far—JUST RIGHT: not a textbook, but enough material to be the basis for taking informed action. This issue is literally the first digital magazine issue compelling enough to get me to read it cover to cover. I'm feeling oddly nostalgic, so I'm off to install Debian from floppy.

THANK YOU to everyone who contributed to this renaissance issue!

—Jeffrey Brown

Thanks for the kind words Jeffrey! It's been quite a wild ride, and we're so glad to be back—bigger and better than ever. We truly appreciate your support and the support of every single subscriber. We sincerely hope readers will enjoy each issue, and as always, we ask that readers continue to send any comments, requests and feedback our way.—Ed.

Regarding Doc Searls' "Help Us Cure Online Publishing of Its Addiction..."

Doc, sites loading lots of other sites' cookies are truly irritating, especially if your internet connection is not that great.

On the other hand, I LIKE targeted ads—that is, ones that more or less align to my interests. I used to receive lots of ads from Amazon.fr for fashion; well, you don't know me, but if you did, you could not imagine anybody less interested in fashion. Gradually those ads have dried up, and ads more appropriate to what I actually buy (such as electronic, mechanical, DIY items) have taken their place. Some of these I follow up and discover something of real interest. So something happened to

convince the AI to stop decorating my browsing with rubbish.

Now, I don't say that accumulating masses of personal data is the best way of generating the filters driving the ads, but if it is to be suppressed, what is going to be put in its place or is it back to the fashion ads?

A non-online alternative problem is my fixed-line telephone in France. I receive around five calls per day trying to sell me insulation, health care, double glazing and so on, and absolutely nothing I am interested in. Okay, it doesn't take too long to put the phone down (I answer "YOU RANG" in English), but not answering means I miss the one call in two weeks that is from a buddy. There is no means that the advertisers have of determining that they are wasting their time. At least the on-line irritations are getting to be somewhat interesting irritations.

So what is your proposal to substitute tracking by publishing what sort of things I might like to buy?

—Ray Foulkes, 72-year-old British electronics/computing retired techie

Doc Searls replies:

Thanks, Ray.

*In **this column** I should have made clear that there are two kinds of advertising at issue here. One is advertising and the other is adtech. I explain the differences in **An easy fix for a broken advertising system**:*

> *In the old advertising world, advertising wasn't personal. It was aimed at populations defined by the media people read, watched or listened to. Advertisers sponsored those media directly, because they wanted to reach the kinds of readers, viewers and listeners who liked particular papers, magazines, radio and TV stations, networks and programs....*
>
> *With programmatic adtech, ads follow eyeballs. Those eyeballs are tracked like*

> *animals by beacons placed in people's apps and browsers. In the online print world, readers are tagged with spyware in their browsers or apps and tracked like animals. Personal data and metadata about those readers are harvested by the spyware and munched by machines, which place ads against profiles of reader types, no matter where they show up.*
>
> *The result on the receiving end looks like old-fashioned advertising, but it's really direct response marketing (née direct mail, aka junk mail), which has always wanted to get personal, has always looked for an immediate personal response, and has always excused massive negative externalities, such as the simple fact that people hate it.*
>
> *But nearly everybody covering the industry falls for it. So does the industry itself. As I wrote in Separating Advertising's Wheat and Chaff, "Madison Avenue fell asleep, direct response marketing ate its brain, and it woke up as an alien replica of itself."*

That alien replica is the vampire I talk about. That isn't our business, and we're not interested in making it our business. If we do bring back advertising, it will be the sponsoring kind. That's what the #DoNotByte agreement is meant to welcome.

Advertising

I agree that tracking by advertisers is vile, but so is tracking by anyone. I think you miss the point that advertising (at least as I understand it) is also vile. I don't need advertising to help me make purchasing decisions. If I'm in any doubt, I'll go poll my peers for their observations. More to the point: publications that carry advertising are more beholden to those advertisers than to their subscribers. A subscriber is going to be a $35 peanut, and the management that decides what content to carry is going to care a lot more about the $35000 advertising goober than the peanut. If a publication carries advertising, it will probably lose me as a subscriber. I might buy a copy from the magazine rack to read while I'm waiting at the airport or something, but advertising seriously degrades the credibility of any publication.

—jimduba

Doc Searls replies: *Two things here.*

*First, good magazine journalism has what's called a "Chinese wall" between the editorial and publishing side of the house. (**Look it up**.) I can also tell you our Chinese wall has always been intact. I've been involved with* Linux Journal *since before it was born in 1994, and writing for it since 1996, and I cannot think of a single instance when an advertiser had any influence over anything any of us have written, or even cared about. Advertisers supported us because they liked sponsoring us and speaking to our readers. Since we were essentially a trade magazine, advertising here functioned almost as a breed of editorial copy: entries in our catalog. As writers, we might sometimes notice those ads, but never when we were writing anything. We just didn't care.*

Second, take away advertising, and most magazines disappear. Ones that persist have a hard time making enough money off subscriptions alone, but some do. Consumer Reports*, for example, has never taken ad money and seems to prosper. Right now we're not taking tracking-based advertising, but we aren't throwing out the whole advertising baby with the adtech bathwater.*

We want to make Linux Journal *follow our readers rather than vice versa and to model that approach for other magazines. That's why we want to be the first agreeing to terms of engagement our readers proffer, rather than vice versa. And we'd love to have your help in making that new hack work.*

Comment on "Security: 17 Things"

First, welcome back.

I'm going through the March 2018 issue (the re-launch issue) and reading Susan Sons' article "Security: 17 Things", and two things stuck out.

"Number 8: Use Your Credit Card": while the debate over paying interest or having money come from your checking account is not what this is about, there is the mis-information that debit liability is higher.

This used to be true. Regulation E of the Electronic Funds Transfer Act also caps liability at $50.00, if the debit card owner notifies the bank within a timely manner (usually two days). Failure to notify leaves the card holder with a larger liability.

Section 205.6 **Liability of consumer for unauthorized transfers**: "limits a consumer's liability for unauthorized electronic fund transfers, such as those arising from loss or theft of an access device, to $50; if the consumer fails to notify the depository institution in a timely fashion, the amount may be $500 or unlimited."

Going beyond this, there are other ways to pay that will protect the card for online purchases. Mobile payment (Apple Pay, Google Pay, Samsung Pay), for example, uses tokenization instead of actual card data when making purchases. For online purchases, there are tools to create one-time/one-vendor cards, such as Privacy Card or Abine's Blur.

The second issue was under heading "9. Freeze Your Credit". There are two more places that people should put security freezes in place. While the three listed in the article are the biggest three, the other two are (the fourth largest) and Chexystem.

—Chris Jenks

Susan Sons replies:

Thanks for writing, Chris.

The debit card regulations have changed somewhat, as you noticed, but because of that short two-day notification period, debit card use still bites people so much more than credit cards.

There are so many other security measures in the world worth considering that didn't quite make the cut for the article. We had to make tough choices! When the list gets too long, many people feel they can't accomplish any meaningful amount and give up. I'd never advise against looking beyond this list for more useful security measures;

it's meant as a starting place for those who are overwhelmed by the sheer amount of advice out there.

Matthew Garrett Responds to "diff -u: Detainting the Kernel"

Regarding "Detainting the Kernel" from the March 2018 issue: the kernel has the ability to "taint" itself—i.e., to set a flag that tells you something about the kernel state. Most of these flags indicate that something has happened that means the upstream kernel developers are less interested in bug reports in the kernel (because you've loaded a proprietary driver, for instance), but others are more interesting for the local admin.

If a user builds an out of tree module and loads it, the kernel will be tainted with the O flag. If that out of tree module contains proprietary code, the kernel will also be tainted with the P flag. So, loading the nvidia kernel module will cause the kernel to be tainted with both O and P.

If your kernel supports module signatures, then attempting to load an unsigned module will either fail (if the kernel is configured to enforce module signatures) or taint the kernel with the E flag. It's important to note that this isn't a security feature—if you load an unsigned module, then that module is able to modify the kernel to clear the taint flag. The only reason to taint the kernel is to inform the admin that they loaded an unsigned module. Loading the nvidia driver on this kernel would result in the E, O and P taint flags.

Unfortunately some users see the additional E taint and are unhappy, either because they didn't expect it (because it didn't happen with previous kernels) or because they have objections to the kernel wanting them to sign modules. I wrote a patch that allowed distributions to build kernels that could be configured to enforce module signatures, but wouldn't taint if an unsigned module was allowed to load.

With this patch applied and the option enabled, loading the nvidia driver would still result in the O and P flags being set, and upstream kernel developers would still be

able to tell that a proprietary driver had been loaded and ask the user to reproduce the bug without that code. However, the kernel would also be usable in cases where you want to require signed drivers.

Distributions and kernel developers don't really care if you've loaded an unsigned module, so the removal of the E taint flag doesn't change anything. But it does mean that distributions can enable support for module signatures without causing confusion for users, which makes it easier for distributions to continue doing development work on new security features that make everyone safer.

—Matthew Garrett

Bravo! What a Refreshing New Look and Approach

Incredible—one could focus on all the great articles due to the stunning sleek look of the new *LJ*. You've managed to completely reinvent yourself, the Linux news scene and the world of journalism all in one fantastic issue. This is a watershed moment—returning control to readers and defining a whole new world of vendor respect for customers. Magnificent! I am so glad that I decided to re-subscribe, it was worth the wait. And to vendors who may have doubts—I look forward to signing up with those that decide to provide products with value.

—John

We are so glad to hear you like the new issue and the new concept! Thank you for taking the time to let us know.—Ed.

SEND *LJ* A LETTER *We'd love to hear your feedback on the magazine and specific articles. Please write us here or send email to ljeditor@linuxjournal.com.*

PHOTOS *Send your Linux-related photos to ljeditor@linuxjournal.com, and we'll publish the best ones here.*

The Road Less Traveled: Certifications Can Chart a Great Career in Linux and Open Source

Taz Brown writes about the challenges of a career in IT and her goals of helping to increase diversity in the field and bring Linux to urban education.

The year is now 2018, and the world has changed tremendously in so many ways. One thing that's changed significantly is the way we learn and the way we demonstrate that knowledge. No longer is a college degree enough, particularly in the area of Information Technology (IT). Speak to two technologists about how they paved their way in the field, and you will get, oftentimes, completely different stories.

It's one of the things I like most about IT. You often can work with many different people with varying experiences, backgrounds and stories about how they came to enter the field, and one of the most common paths to IT is through certifications.

My path to IT could and would not have happened without certifications. First, my college degree was not in any tech or computer science concentration or track. I did not begin my career in IT, and therefore, gaining the knowledge I needed to enter the field began and continues with certifications. Now, this is not to say that I did not need to gain practical experience in order to be able to do the job, but had I only had practical experience and no certifications, I likely wouldn't have attracted the recruiters that I did.

I started with some CompTIA certifications like A+ and Network+, and Microsoft certs like the MCSA, focusing on Windows 7 and Windows Server. So after putting in 25–30 hours a week studying and practicing—and this was all with just a laptop, mind you—I obtained those certifications. But after getting those certifications, I wanted more—more knowledge and skills, that is. I was able to obtain a job in IT on the HelpDesk, and after a few years, and a few more certifications, I became a Systems Administrator.

So fast-forward ten years, and I am now a Sr. Linux Systems Engineer. I moved into the field of Linux about five years ago, because I saw a trend that I could not resist—a niche market. And, it has paid off, but with advancing my career came the need yet again to prove myself, and so I have been focused on the Red Hat track of certification for the last few years.

I have some Linux certifications, but the ones that have been the most important to me at this stage in my career are those from Red Hat. I currently possess the RHCSA (Red Hat Certified Systems Administrator), and for the last few months, I've been preparing to take and pass the RHCE (Red Hat Certified Engineer). My ultimate goal is to obtain the RHCA (Red Hat Certified Architect).

These exams are arguably some of the hardest certifications to achieve because they are "performance-based". You will not find "a, b, c or d" on these exams. The exams are so challenging because you have to *show* your work. You have to execute a variety of tasks, such as password recovery, configuring YUM, setting SELinux properly, configuring ACL/permissions and so on.

I purchased a copy of Workstation Pro and installed virtual servers running RHEL 7, although I could have gone with a free open-source option: VirtualBox. I set up a base image with all the necessary networking configurations, and then I cloned it to create three other servers, one being an IPA server and the other two being a server and a client with three network interface cards each. I knew that I planned to grow my environment as I continued to study and practice for the multiple Red Hat exams I'd be taking.

In my journey to Sr. Linux Systems Administrator, I've met more and more women and African Americans who either have entered the field or have obtained their RHCSAs, or


RHCSA/RHCE
Red Hat Linux Certification
Seventh Edition (Exams EX200 & EX300)
Michael Jang
Alessandro Orsaria
UNIX AND LINUX
ADMINISTRATION
STUDY GUIDE
DELL

maybe the RHCE, but often that's as far as I have seen them go.

So, I want to start a series of articles where I discuss my road to the RHCA. I plan to discuss my challenges (right now, sometimes saying on target and focused can be difficult with all the other stuff life brings, but I have set my goals, which I plan to cover in my next article), and I also hopefully will highlight others who are particularly unrepresented but who have taken the plunge to become Red Hat-certified as well.

Basically, I want anyone who reads this to know that there is no rule book for how to get there. You just need to have a plan, a razor-sharp focus and dedication, and there is nothing you can't achieve.

At the time of writing this article, I am a week out from taking my RHCE. I'll let you know what happens and more as soon as I can. Right now, I am focused on network teaming, http/https, dns, nfs, smb, smtp, ssh, ntp and database services.

I also want to see more people of color and women learn Linux and make it a career. The numbers are minimal at this point, but I know that this can change, and I hope I can somehow make a difference. For me, I believe it starts with learning and education and having the option to learn about Linux in urban schools—not just typing and the use of Windows. I think the use of Raspberry Pis and learning Python can spark creativity, ingenuity and zeal for something not thought of as having a place. There is a place; every supercomputer in the world runs on Linux.

I have started mentoring others and helping them begin their journey of learning and understanding Linux and going even further, preparing for certifications, because it is something that I know I must do if I want to see the change and increase the representation. The opportunities are endless, and the rewards are even greater. I look forward to furthering my career in Linux and open source and bringing more along with me.

—Taz Brown

Readers' Choice Awards

This month the categories are Best Editor, Best Domain Registrar and Best Laptop. Note that all the contenders listed were nominated by readers via Twitter. Be sure to check LinuxJournal.com for new polls each week and vote for your favorites!

BEST EDITOR

- **Vim: 35%**
- Emacs: 19%
- Sublime Text: 10%
- Atom: 8%
- Other: 7%
- nano: 6%
- Geany: 5%
- Gedit: 4%
- Kate: 4%
- KWrite: 1%

Award-winner Vim is an extremely powerful editor with a user interface based on Bill Joy's 40-plus-year-old vi, but with many improved-upon features, including extensive customization with key mappings and plugins. *Linux Journal* reader David Harrison points out another Vim feature "is that it's basically everywhere. It's available on every major platform."

The very features that make Vim so versatile also have been known to intimidate beginners. Perhaps that's why *Linux Journal* has featured nearly 100 articles on Vim so far. Readers generally agree though; any learning curve is worth the effort, and again this year, they award Vim the Best Editor title.

Richard Stallman's GNU Emacs comes in a respectable second place in this year's *Linux Journal* Readers' Choice Awards Best Editor category. Emacs is often referred to as "the most powerful editor ever made". Interesting pop culture trivia: Emacs was featured in the movie *Tron: Legacy*.

BEST DOMAIN REGISTRAR

Here's the breakdown of the top three (too many were nominated to list here:

- **Namecheap: 22%**
- Gandi 17%
- GoDaddy 11%

Namecheap is an independent domain registrar founded in 2000 by Richard Kirkendall. He launched it with the idea that "the average people of the internet deserved value-priced domains and stellar service." Now with more than 5 million domains under its management, Namecheap has become a leader in both the budget-domain and web-hosting spaces. No hidden fees and a general sense of honesty and transparency from Namecheap are reasons cited by *Linux Journal* readers for their devotion to the company.

Who to watch for: Gandi, in a close second place, comes in as another favorite. *Linux Journal* reader Mathieu T. commented, "I am currently moving everything to GANDI. Easy to use and clear interface, and a reactive support."

BEST LAPTOP

- **Lenovo: 32%**
- Dell: 25%
- System76: 11%
- ASUS 9%
- Other 7%
- Apple 5%
- Purism 4%
- Toshiba 2%
- Clevo 1%
- MSI 1%
- Sony 1%
- Tuxedo 1%
- Xiaomi 1%
- ZaReason 1%

The ThinkPad began life at IBM, but in 2005, it was purchased by Lenovo along with the rest of IBM's PC business. Lenovo evolved the line, and today the company is well known as a geek favorite. Lenovo's ThinkPads are quiet, fast and arguably have one of the best keyboards (fighting words!). *Linux Journal* readers say Lenovo's Linux support is excellent, leaving many to ponder why the

company doesn't ship laptops with Linux installed.

Who to watch for: System76, a small shop out of Denver, Colorado, is a Linux community favorite, coming in a very respectable third place in our poll among some heavyweight vendors. System76 is first and foremost a Linux shop, and as a result, its laptops ship with Linux pre-installed. In fact, according to Wikipedia, "The number 76 in the company name alludes to the year 1776, the year in which the American Revolution took place. The company founders hope likewise to ignite an open source revolution, ultimately leading to a situation in which consumers do not rely primarily on proprietary software."

FOSS Project Spotlight: Ravada

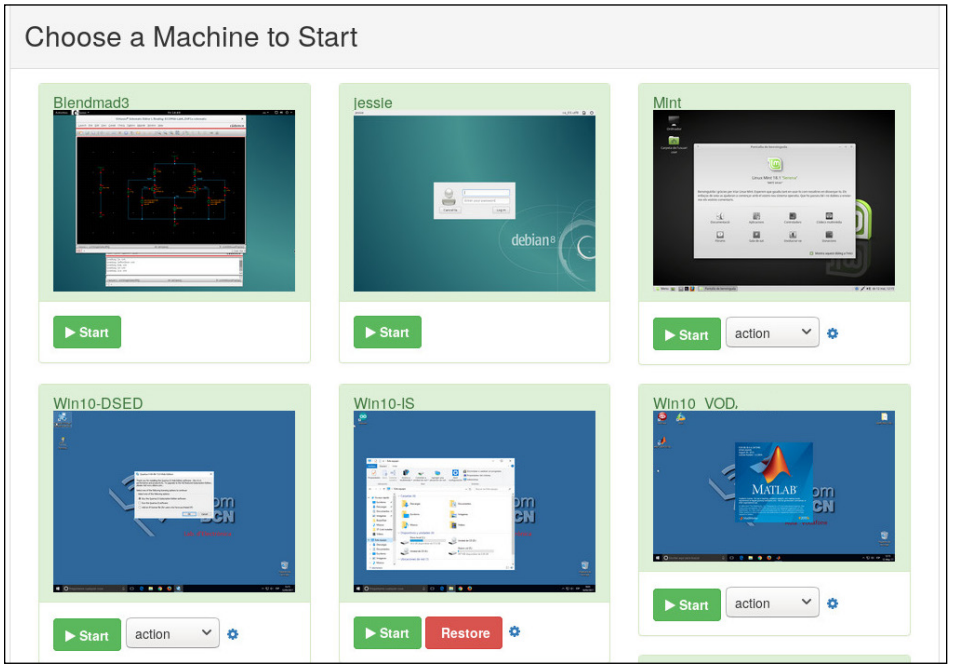
Ravada is an open-source project that allows users to connect to a virtual desktop.

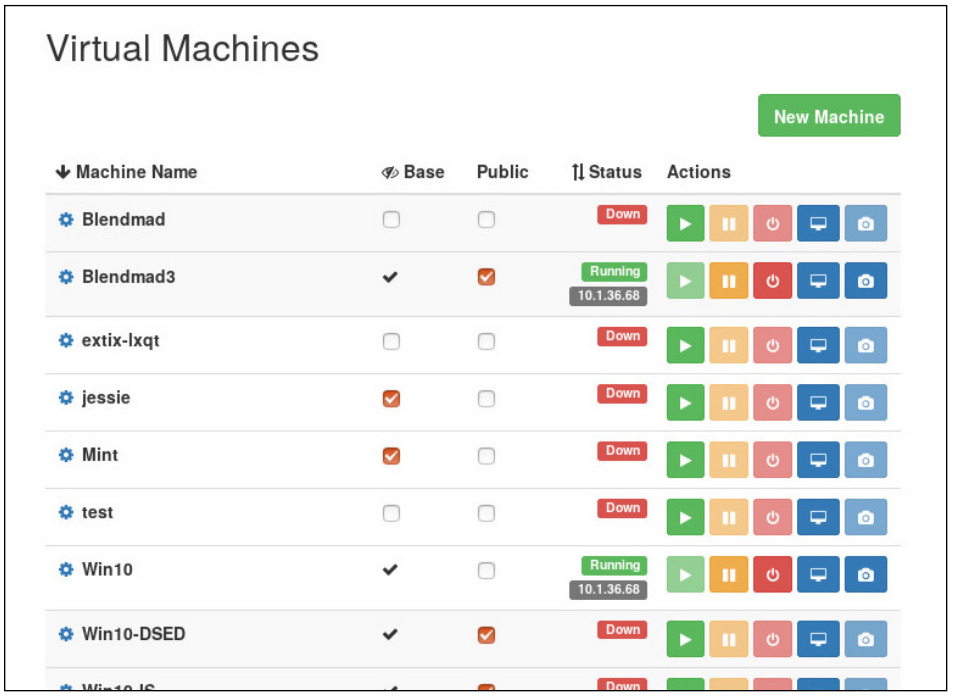
Currently, it supports KVM, but its back end has been designed and implemented in order to allow future hypervisors to be added to the framework. The client's only requirements are a web-browser and a remote viewer supporting the spice protocol.

Ravada's main features include:

- KVM back end.
- LDAP and SQL authentication.
- Kiosk mode.
- Remote access for Windows and Linux.
- Light and fast virtual machine clones for each user.
- Instant clone creation.
- USB redirection.
- Easy and customizable end-user interface (i18n, l10n).
- Administration from a web browser.

It's very easy to install and use. Following the documentation, virtual machines can be deployed in minutes. It's an early release, but it's already used in production. The project is open source, and you can download the code





from GitHub. Contributions welcome!

—Francesc Guasch

Why Do We Do It?

Why does a painter paint? Why does a carpenter build? Why does a chef cook? Why does an electronic engineer design, and why does a software programmer code? Speaking from my personal experiences, I'm going to answer those questions with this: to create something out of nothing. There is an art to conceiving an idea and, when using the right tools, bringing it to fruition.

I studied Electronic Engineering (EE) in school, learning the very basics of what makes good hardware design. I put together resistors, capacitors, transistors, operational amplifiers, microprocessors and more onto breadboards and, in turn, observed the miracle of my creations. It didn't stop there—next came the programming of such devices, writing microcode and eventually "operating systems" in their simplest of forms (using a lot of assembly language) to extend the functionality of my creations. The hardware gave these "creatures" life, but the software gave them brains. The excitement. The thrill. The adrenaline of never knowing what to expect. Was there a flaw in my design? And if so, how will I address it? Will I need an oscilloscope or a JTAG debugger? This new sense of responsibility gave me purpose. It gave me the motivation to persist and move on to bigger and greater challenges.

It's almost two decades later, and almost nothing has changed. Today, while I tend to focus more on the software and a bit less on the hardware, I am still driven by the same emotions: the need to create and the desire to watch my creation mature. This has now extended beyond the confines of single board computers or embedded environments and across a wider ecosystem and living on larger networks. I now watch my creations impact companies, institutions and individuals worldwide. In my career, there is no greater feeling than this. And although I may not get the recognition for such things, it doesn't hold me back nor does it deter me from continuing. I am not in it for the money or the fame (although, both would be nice). I am more interested in the adventure and not the destination.

Now I ask you, the reader: *Why do* **you** *do it?*

Send your answers to ljeditor@linuxjournal.com.

—Petros Koutoupis

A Good Front End for R

R is the de facto statistical package in the Open Source world. It's also quickly becoming the default data-analysis tool in many scientific disciplines.

R's core design includes a central processing engine that runs your code, with a very simple interface to the outside world. This basic interface means it's been easy to build graphical interfaces that wrap the core portion of R, so lots of options exist that you can use as a GUI.

In this article, I look at one of the available GUIs: RStudio. RStudio is a commercial program, with a free community version, available for Linux, Mac OSX and Windows, so your data analysis work should port easily regardless of environment.

For Linux, you can install the main RStudio package from the download page. From there, you can download RPM files for Red Hat-based distributions or DEB files for Debian-based distributions, then use either `rpm` or `dpkg` to do the installation.

For example, in Debian-based distributions, use the following to install RStudio:

```
sudo dpkg -i rstudio-xenial-1.1.423-amd64.deb
```

It's important to note that RStudio is only the GUI interface. This means you need to install R itself as a separate step. Install the core parts of R with:

```
sudo apt-get install r-base
```

There's also a community repository of available packages, called CRAN, that can add huge amounts of functionality to R. You'll want to install at least some of them in

order to have some common tools to use:

```
sudo apt-get install r-recommended
```

There are equivalent commands for RPM-based distributions too.

At this point, you should have a complete system to do some data analysis.

When you first start RStudio, you'll see a window that looks somewhat like Figure 1.

The main pane of the window, on the left-hand side, provides a console interface where you can interact directly with the R session that's running in the back end.

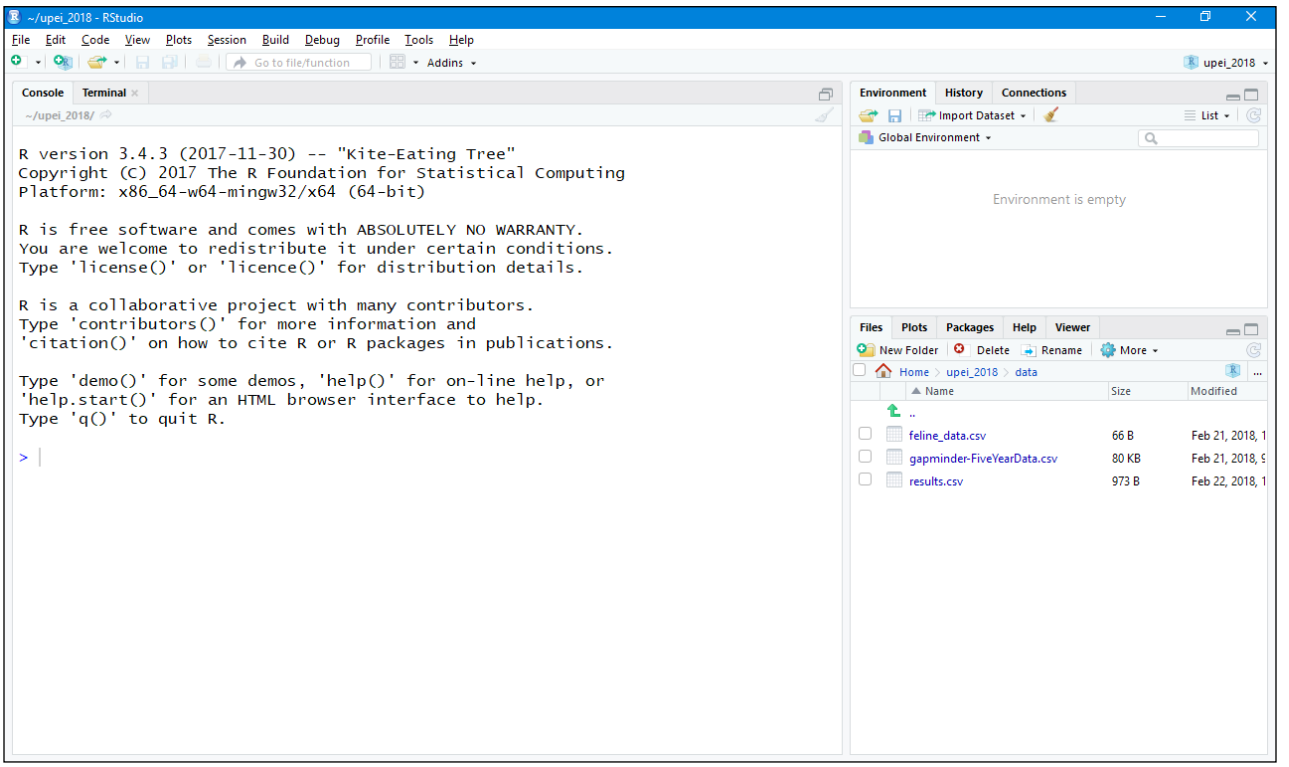


Figure 1. RStudio creates a new session, including a console interface to R, where you can start your work.

The right-hand side is divided into two sections, where each section has multiple tabs. The default tab in the top section is an environment pane. Here, you'll see all the objects that have been created and exist within the current R session.

The other two tabs provide the history of every command given and a list of any connections to external data sources.

The bottom pane has five tabs available. The default tab gives you a file listing of the current working directory. The second tab provides a plot window where any data plots you generate are displayed. The third tab provides a nicely ordered view into R's library system. It shows a list of all of the currently installed libraries, along with tools to manage updates and install new libraries. The

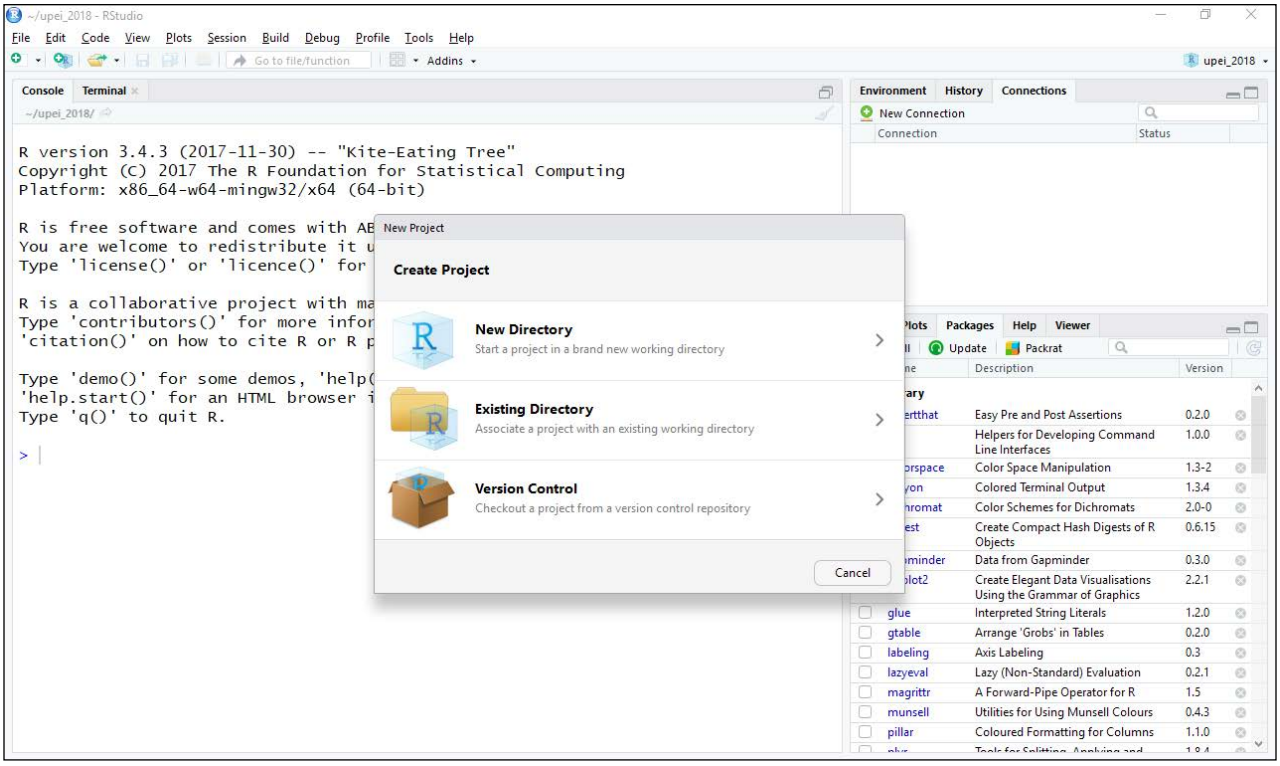


Figure 2. When you create a new project, it can be created in a new directory, an existing directory or be checked out from a code repository.

fourth tab is the help viewer. R includes a very complete and robust help system modeled on Linux man pages. The last tab is a general "viewer" pane to view other types of objects.

One part of RStudio that's a great help to people managing multiple areas of research is the ability to use projects. Clicking the menu item File→New Project pops up a window where you can select how your new project will exist on the filesystem.

As an example, let's create a new project hosted in a local directory. The file display in the bottom-right pane changes to the new directory, and you should see a new file named after the project name, with the filename ending .Rproj. This file contains the configuration for your new project. Although you can

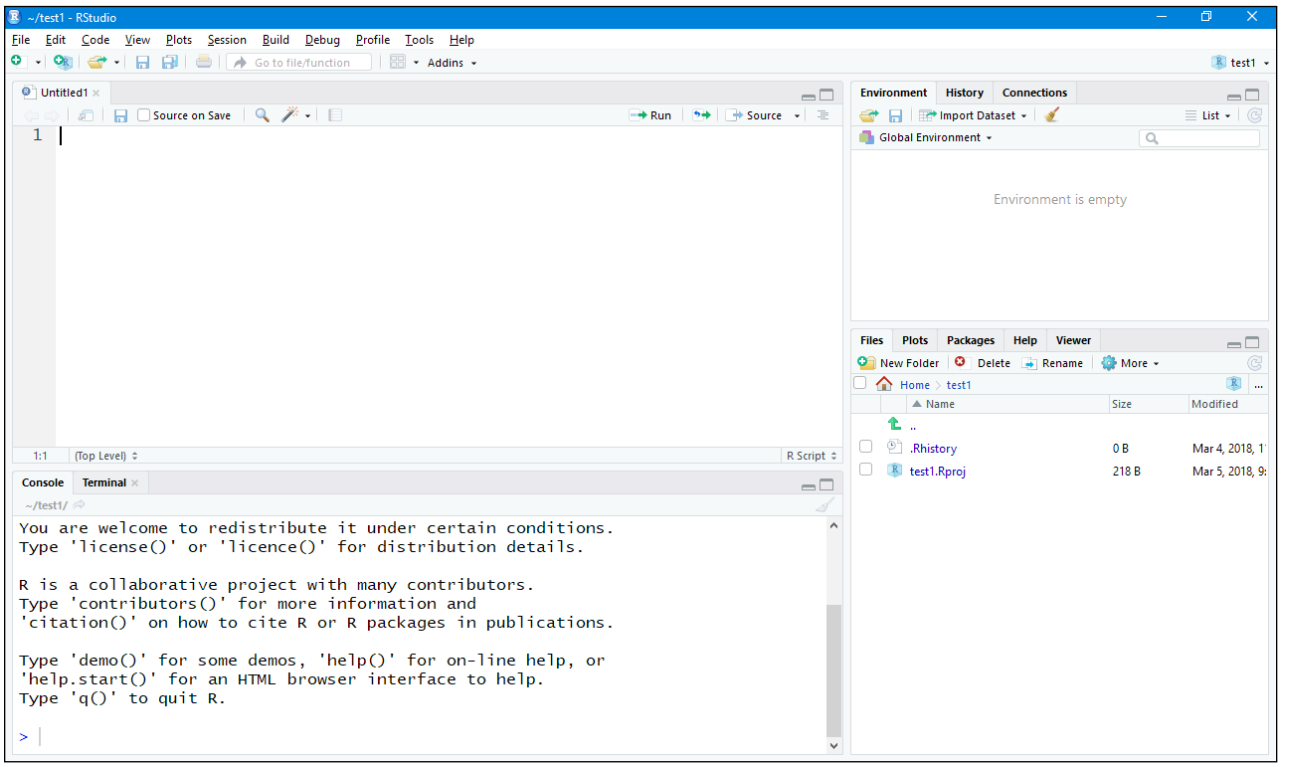


Figure 3. The script editor allows you to construct more complicated pieces of code than is possible using just the console interface.

interact with the R session directly through the console, doing so doesn't really lead to easily reproduced workflows. A better solution, especially within a project, is to open a script editor and write your code within a script file. This way you automatically have a starting point when you move beyond the development phase of your research.

When you click File→New File→R Script, a new pane opens in the top left-hand side of the window.

From here, you can write your R code with all the standard tools you'd expect in a code editor. To execute this code, you have two options. The first is simply to click the run button in the top right of this editor pane. This will run either the single line where the cursor is located or an entire block of code that

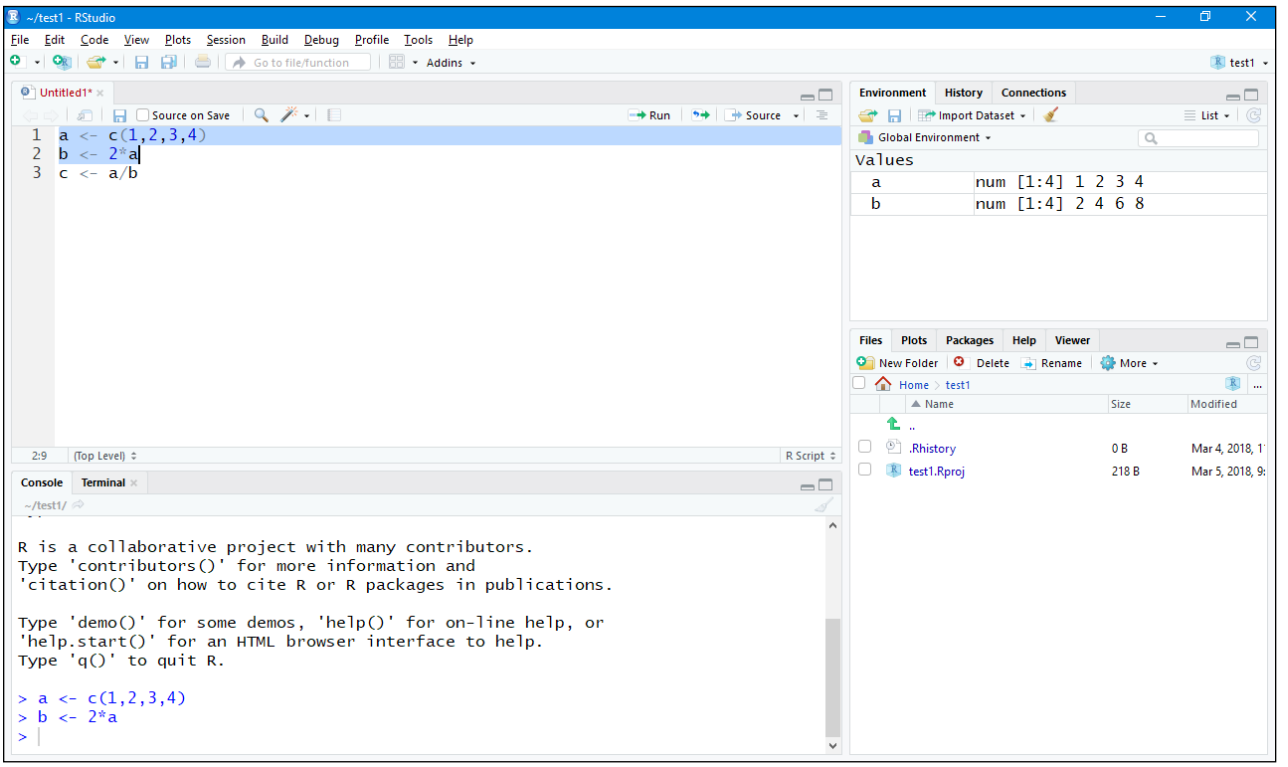


Figure 4. You can enter code in the script editor and then have them run to make code development and data analysis a bit easier on your brain.

previously had been highlighted.

If you have an entire script file that you want to run as a whole, you can click the source button in the top right of the editor pane. This lets you reproduce analysis that was done at an earlier time.

The last item to mention is data visualization in RStudio. Actually, the data visualization is handled by other libraries within R. There is a very complete, and complex, graphics ability within the core of R. For normal humans, several libraries are built on top of this. One of the most popular, and for good reason, is ggplot. If it isn't already installed on your system, you can get it with:

```
install.packages(c('ggplot2'))
```

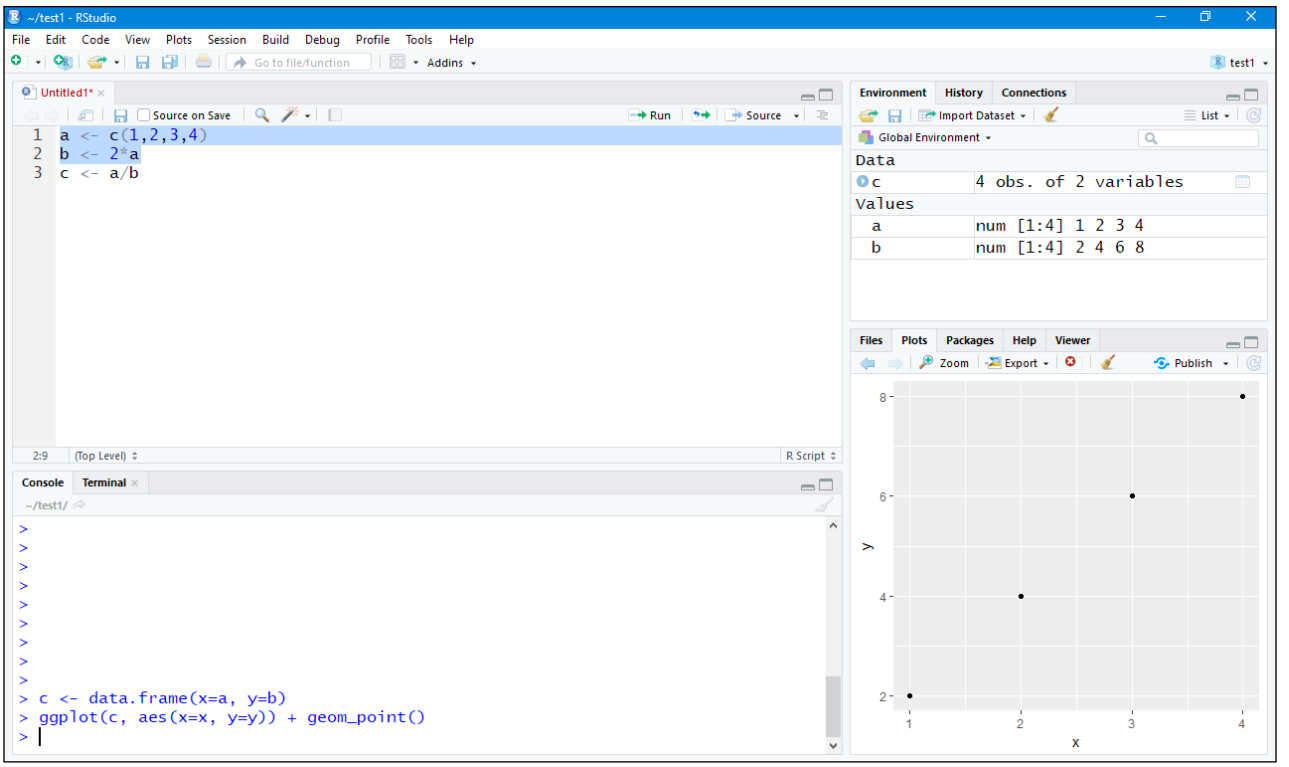


Figure 5. ggplot2 is one of the most powerful and popular graphing tools available in the R environment.

Once it's installed, you can make a simple scatter plot with this:

```
library(ggplot2)
c <- data.frame(x=a, y=b)
ggplot(c, aes(x=x, y=y)) + geom_point()
```

As you can see, ggplot takes dataframes as the data to plot, and you control the display with `aes()` function calls and `geom` function calls. In this case, I used the `geom_point()` function to get a scatter plot of points. The plot then is generated in the bottom-left pane.

There's a lot more functionality available in RStudio, including a server portion that can be run on a cluster, allowing you to develop code locally and then send it off to a server for the actual processing.

—Joey Bernard

News Briefs

Visit LinuxJournal.com for daily news briefs.

- Private Internet Access announced that it is open-sourcing its software: "We believe that the shift to open source is the right move for a privacy-focused business, and recognise that code transparency is key. We appreciate that our code may not be perfect, and we hope that the wider FOSS community will get involved, provide feedback, feature requests, bug fixes and generally help provide a greater service to the wider privacy movement." You can check out the repo on GitHub and find them at #privateinternetaccess on chat.freenode.net.

- Eric Raymond has started a new project after ranting about the state of the Uninterruptible Power Supply (UPS) market, as *The Register* reports. The Upside project is hosted on GitLab and "is currently defining requirements and developing a specification for a 'high quality UPS that can be built from off-the-shelf parts in any reasonably well-equipped makerspace or home electronics shop'." You can read Eric's original UPS rant here.

- An image of Scarlett Johansson is being used as a malware attack vector on PostgreSQL, *ZDNet* reports. According to security firm Imperva, this attack "places malware, which cryptomines Monero, on PostgreSQL DBMS servers."

- Google is banning cryptocurrency ads, according to *Ars Technica*: "Google is not only banning advertisement of cryptocurrencies themselves, but also 'initial coin offerings, cryptocurrency exchanges, cryptocurrency wallets, and cryptocurrency trading advice.'"

- Purism announced that it's collaborating with cryptography pioneer Werner Koch "to integrate hardware encryption into the company's Librem laptops and forthcoming Librem 5 phone. By manufacturing hardware with its own software and services, Purism will include cryptography by default pushing

the industry forward with unprecedented protection for end-user devices."

- Purism also recently announced that it has added "tamper-evident features" to its laptops, making it the "most secure laptop under customer control". The laptops are integrated with Trammel Hudson's Heads security firmware, giving users full control of the boot process.

- Washington state is the first to pass net neutrality protection into law. Gov. Jay Inslee signed the bill on March 5, 2018, stating "Today we make history: Washington will be the first state in the nation to preserve the open internet."

- You now can use a Raspberry Pi 3 to build your own cheap car head unit, thanks to Huan Truong's Crankshaft, a "'turnkey GNU/Linux distribution', which only needs to be downloaded and written to an SD card for the Raspberry Pi 3 tablet", as *ZDNet* reports. See Huan's Reddit announcement for more info and to report bugs.

- The BleachBit open-source system cleaner released version 2.0, with "major improvements to infrastructure, security, stability, and the framework". BleachBit does two things: "removes junk files to make more room" and "removes private information like a virtual shredder". You can download it here.

Simple Cloud Hardening

Apply a few basic hardening principles to secure your cloud environment.

By Kyle Rankin

Kyle Rankin is a Tech Editor and columnist at *Linux Journal* and the Chief Security Officer at Purism. He is the author of *Linux Hardening in Hostile Networks*, *DevOps Troubleshooting*, *The Official Ubuntu Server Book*, *Knoppix Hacks*, *Knoppix Pocket Reference*, *Linux Multimedia Hacks* and *Ubuntu Hacks*, and also a contributor to a number of other O'Reilly books. Rankin speaks frequently on security and open-source software including at BsidesLV, O'Reilly Security Conference, OSCON, SCALE, CactusCon, Linux World Expo and Penguicon. You can follow him at @kylerankin.

I've written about simple server-hardening techniques in the past. Those articles were inspired in part by the *Linux Hardening in Hostile Networks* book I was writing at the time, and the idea was to distill the many different hardening steps you might want to perform on a server into a few simple steps that everyone should do. In this article, I take the same approach only with a specific focus on hardening cloud infrastructure. I'm most familiar with AWS, so my hardening steps are geared toward that platform and use AWS terminology (such as Security Groups and VPC), but as I'm not a fan of vendor lock-in, I try to include steps that are general enough that you should be able to adapt them to other providers.

New Accounts Are (Relatively) Free; Use Them

One of the big advantages with cloud infrastructure is the ability to compartmentalize your infrastructure. If you have a bunch of servers racked in the same rack, it might be difficult, but on cloud infrastructures, you can take advantage of the technology to isolate one customer from another or to isolate one of your infrastructure types from the others. Although this doesn't

come completely for free (it adds some extra overhead when you set things up), it's worth it for the strong isolation it provides between environments.

One of the first security measures you should put in place is separating each of your environments into its own high-level account. AWS allows you to generate a number of different accounts and connect them to a central billing account. This means you can isolate your development, staging and production environments (plus any others you may create) completely into their own individual accounts that have their own networks, their own credentials and their own roles totally isolated from the others. With each environment separated into its own account, you limit the damage attackers can do if they compromise one infrastructure to just that account. You also make it easier to see how much each environment costs by itself.

In a traditional infrastructure where dev and production are together, it is much easier to create accidental dependencies between those two environments and have a mistake in one affect the other. Splitting environments into separate accounts protects them from each other, and that independence helps you identify any legitimate links that environments need to have with each other. Once you have identified those links, it's much easier to set up firewall rules or other restrictions between those accounts, just like you would if you wanted your infrastructure to talk to a third party.

Lock Down Security Groups

One advantage to cloud infrastructure is that you have a lot tighter control over firewall rules. AWS Security Groups let you define both ingress and egress firewall rules, both with the internet at large and between Security Groups. Since you can assign multiple Security Groups to a host, you have a lot of flexibility in how you define network access between hosts.

My first recommendation is to deny all ingress and egress traffic by default and add specific rules to a Security Group as you need them. This is a fundamental best practice for network security, and it applies to Security Groups as much as to traditional firewalls. This is particularly important if you use the Default security group, as it allows unrestricted internet egress traffic by default, so that should be

one of the first things to disable. Although disabling egress traffic to the internet by default can make things a bit trickier to start with, it's still a lot easier than trying to add that kind of restriction after the fact.

You can make things very complicated with Security Groups; however, my recommendation is to try to keep them simple. Give each server role (for instance web, application, database and so on) its own Security Group that applies to each server in that role. This makes it easy to know how your firewall rules are being applied and to which servers they apply. If one server in a particular role needs different network permissions from the others, it's a good sign that it probably should have its own role.

The role-based Security Group model works pretty well, but it can be inconvenient when you want a firewall rule to apply to all your hosts. For instance, if you use centralized configuration management, you probably want every host to be allowed to talk to it. For rules like this, I take advantage of the Default Security Group and make sure that every host is a member of it. I then use it (in a very limited way) as a central place to define any firewall rules I want to apply to all hosts. One rule I define in particular is to allow egress traffic to any host in the Default Security Group—that way I don't have to write duplicate ingress rules in one group and egress rules in another whenever I want hosts in one Security Group to talk to another.

Use Private Subnets

On cloud infrastructure, you are able to define hosts that have an internet-routable IP and hosts that only have internal IPs. In AWS Virtual Private Cloud (VPC), you define these hosts by setting up a second set of private subnets and spawning hosts within those subnets instead of the default public subnets.

Treat the default public subnet like a DMZ and put hosts there only if they truly need access to the internet. Put all other hosts into the private subnet. With this practice in place, even if hosts in the private subnet were compromised, they couldn't talk directly to the internet even if an attacker wanted them to, which makes it much more difficult to download rootkits or other persistence tools without setting up elaborate tunnels.

These days it seems like just about every service wants unrestricted access to web ports on some other host on the internet, but an advantage to the private subnet approach is that instead of working out egress firewall rules to specific external IPs, you can set up a web proxy service in your DMZ that has more broad internet access and then restrict the hosts in the private subnet by hostname instead of IP. This has an added benefit of giving you a nice auditing trail on the proxy host of all the external hosts your infrastructure is accessing.

Use Account Access Control Lists Minimally

AWS provides a rich set of access control list tools by way of IAM. This lets you set up very precise rules about which AWS resources an account or role can access using a very complicated syntax. While IAM provides you with some pre-defined rules to get you started, it still suffers from the problem all rich access control lists have—the complexity makes it easy to create mistakes that grant people more access than they should have.

My recommendation is to use IAM only as much as is necessary to lock down basic AWS account access (like sysadmin accounts or orchestration tools for instance), and even then, to keep the IAM rules as simple as you can. If you need to restrict access to resources further, use access control at another level to achieve it. Although it may seem like giving somewhat broad IAM permissions to an AWS account isn't as secure as drilling down and embracing the principle of least privilege, in practice, the more complicated your rules, the more likely you will make a mistake.

Conclusion

Cloud environments provide a lot of complex options for security; however, it's more important to set a good baseline of simple security practices that everyone on the team can understand. This article provides a few basic, common-sense practices that should make your cloud environments safer while not making them too complex. ■

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.

Multiprocessing in Python

Reuven M. Lerner teaches Python, data science and Git to companies around the world. His free, weekly "better developers" email list reaches thousands of developers each week; subscribe here. Reuven lives with his wife and children in Modi'in, Israel.

Python's "multiprocessing" module feels like threads, but actually launches processes.

By Reuven M. Lerner

Many people, when they start to work with Python, are excited to hear that the language supports threading. And, as I've discussed in previous articles, Python does indeed support native-level threads with an easy-to-use and convenient interface.

However, there is a downside to these threads—namely the global interpreter lock (GIL), which ensures that only one thread runs at a time. Because a thread cedes the GIL whenever it uses I/O, this means that although threads are a bad idea in CPU-bound Python programs, they're a good idea when you're dealing with I/O.

But even when you're using lots of I/O, you might prefer to take full advantage of a multicore system. And in the world of Python, that means using processes.

In my article "Launching External Processes in Python", I described how you can launch processes from within a Python program, but those examples all demonstrated that you can launch a program in an external process. Normally,

when people talk about processes, they work much like they do with threads, but are even more independent (and with more overhead, as well).

So, it's something of a dilemma: do you launch easy-to-use threads, even though they don't really run in parallel? Or, do you launch new processes, over which you have little control?

The answer is somewhere in the middle. The Python standard library comes with "multiprocessing", a module that gives the feeling of working with threads, but that actually works with processes.

So in this article, I look at the "multiprocessing" library and describe some of the basic things it can do.

Multiprocessing Basics

The "multiprocessing" module is designed to look and feel like the "threading" module, and it largely succeeds in doing so. For example, the following is a simple example of a multithreaded program:

```
#!/usr/bin/env python3

import threading
import time
import random

def hello(n):
    time.sleep(random.randint(1,3))
    print("[{0}] Hello!".format(n))

for i in range(10):
    threading.Thread(target=hello, args=(i,)).start()

print("Done!")
```

In this example, there is a function (`hello`) that prints "Hello!" along with whatever argument is passed. It then runs a `for` loop that runs `hello` ten times, each of them in an independent thread.

But wait. Before the function prints its output, it first sleeps for a few seconds. When you run this program, you then end up with output that demonstrates how the threads are running in parallel, and not necessarily in the order they are invoked:

```
$ ./thread1.py
Done!
[2] Hello!
[0] Hello!
[3] Hello!
[6] Hello!
[9] Hello!
[1] Hello!
[5] Hello!
[8] Hello!
[4] Hello!
[7] Hello!
```

If you want to be sure that "Done!" is printed after all the threads have finished running, you can use `join`. To do that, you need to grab each instance of `threading.Thread`, put it in a list, and then invoke `join` on each thread:

```
#!/usr/bin/env python3

import threading
import time
import random

def hello(n):
```

```
    time.sleep(random.randint(1,3))
    print("[{0}] Hello!".format(n))

threads = [ ]
for i in range(10):
    t = threading.Thread(target=hello, args=(i,))
    threads.append(t)
    t.start()

for one_thread in threads:
    one_thread.join()

print("Done!")
```

The only difference in this version is it puts the thread object in a list ("threads") and then iterates over that list, joining them one by one.

But wait a second—I promised that I'd talk about "multiprocessing", not threading. What gives?

Well, "multiprocessing" was designed to give the feeling of working with threads. This is so true that I basically can do some search-and-replace on the program I just presented:

- threading → multiprocessing
- Thread → Process
- threads → processes
- thread → process

The result is as follows:

```
#!/usr/bin/env python3

import multiprocessing
import time
import random

def hello(n):
    time.sleep(random.randint(1,3))
    print("[{0}] Hello!".format(n))

processes = [ ]
for i in range(10):
    t = multiprocessing.Process(target=hello, args=(i,))
    processes.append(t)
    t.start()

for one_process in processes:
    one_process.join()

print("Done!")
```

In other words, you can run a function in a new process, with full concurrency and take advantage of multiple cores, with `multiprocessing.Process`. It works very much like a thread, including the use of `join` on the `Process` objects you create. Each instance of `Process` represents a process running on the computer, which you can see using `ps`, and which you can (in theory) stop with `kill`.

What's the Difference?

What's amazing to me is that the API is almost identical, and yet two very different things are happening behind the scenes. Let me try to make the distinction clearer with another pair of examples.

Perhaps the biggest difference, at least to anyone programming with threads and

processes, is the fact that threads share global variables. By contrast, separate processes are completely separate; one process cannot affect another's variables. (In a future article, I plan to look at how to get around that.)

Here's a simple example of how a function running in a thread can modify a global variable (note that what I'm doing here is to prove a point; if you really want to modify global variables from within a thread, you should use a lock):

```
#!/usr/bin/env python3

import threading
import time
import random

mylist = [ ]

def hello(n):
    time.sleep(random.randint(1,3))
    mylist.append(threading.get_ident())   # bad in real code!
    print("[{0}] Hello!".format(n))

threads = [ ]
for i in range(10):
    t = threading.Thread(target=hello, args=(i,))
    threads.append(t)
    t.start()

for one_thread in threads:
    one_thread.join()

print("Done!")
print(len(mylist))
print(mylist)
```

The program is basically unchanged, except that it defines a new, empty list (`mylist`) at the top. The function appends its ID to that list and then returns.

Now, the way that I'm doing this isn't so wise, because Python data structures aren't thread-safe, and appending to a list from within multiple threads eventually will catch up with you. But the point here isn't to demonstrate threads, but rather to contrast them with processes.

When I run the above code, I get:

```
$ ./th-update-list.py
[0] Hello!
[2] Hello!
[6] Hello!
[3] Hello!
[1] Hello!
[4] Hello!
[5] Hello!
[7] Hello!
[8] Hello!
[9] Hello!
Done!
10
[123145344081920, 123145354592256, 123145375612928,
 ↪123145359847424, 123145349337088, 123145365102592,
 ↪123145370357760, 123145380868096, 123145386123264,
 ↪123145391378432]
```

So, you can see that the global variable `mylist` is shared by the threads, and that when one thread modifies the list, that change is visible to all the other threads.

But if you change the program to use "multiprocessing", the output looks a

bit different:

```
#!/usr/bin/env python3

import multiprocessing
import time
import random
import os

mylist = [ ]

def hello(n):
    time.sleep(random.randint(1,3))
    mylist.append(os.getpid())
    print("[{0}] Hello!".format(n))

processes = [ ]
for i in range(10):
    t = multiprocessing.Process(target=hello, args=(i,))
    processes.append(t)
    t.start()

for one_process in processes:
    one_process.join()

print("Done!")
print(len(mylist))
print(mylist)
```

Aside from the switch to multiprocessing, the biggest change in this version of the program is the use of `os.getpid` to get the current process ID.

The output from this program is as follows:

```
$ ./proc-update-list.py
[0] Hello!
[4] Hello!
[7] Hello!
[8] Hello!
[2] Hello!
[5] Hello!
[6] Hello!
[9] Hello!
[1] Hello!
[3] Hello!
Done!
0
[]
```

Everything seems great until the end when it checks the value of `mylist`. What happened to it? Didn't the program append to it?

Sort of. The thing is, there is no "it" in this program. Each time it creates a new process with "multiprocessing", each process has its own value of the global `mylist` list. Each process thus adds to its own list, which goes away when the processes are joined.

This means the call to `mylist.append` succeeds, but it succeeds in ten different processes. When the function returns from executing in its own process, there is no trace left of the list from that process. The only `mylist` variable in the main process remains empty, because no one ever appended to it.

Queues to the Rescue

In the world of threaded programs, even when you're able to append to the global `mylist` variable, you shouldn't do it. That's because Python's data structures aren't thread-safe. Indeed, only one data structure is guaranteed to be thread safe—the `Queue` class in the `multiprocessing` module.

Now, queues in the world of multithreaded programs prevent issues having to do with thread safety. But in the world of multiprocessing, queues allow you to bridge the gap among your processes, sending data back to the main process.

Queues are FIFOs (that is, "first in, first out"). Whoever wants to add data to a queue invokes the `put` method on the queue. And whoever wants to retrieve data from a queue uses the `get` command.

Now, queues in the world of multithreaded programs prevent issues having to do with thread safety. But in the world of multiprocessing, queues allow you to bridge the gap among your processes, sending data back to the main process. For example:

```
#!/usr/bin/env python3

import multiprocessing
import time
import random
import os
from multiprocessing import Queue

q = Queue()

def hello(n):
    time.sleep(random.randint(1,3))
    q.put(os.getpid())
    print("[{0}] Hello!".format(n))
```

```
processes = [ ]
for i in range(10):
    t = multiprocessing.Process(target=hello, args=(i,))
    processes.append(t)
    t.start()

for one_process in processes:
    one_process.join()

mylist = [ ]
while not q.empty():
    mylist.append(q.get())

print("Done!")
print(len(mylist))
print(mylist)
```

In this version of the program, I don't create `mylist` until late in the game. However, I create an instance of `multiprocessing.Queue` very early on. That `Queue` instance is designed to be shared across the different processes. Moreover, it can handle any type of Python data that can be stored using "pickle", which basically means any data structure.

In the `hello` function, it replaces the call to `mylist.append` with one to `q.put`, placing the current process' ID number on the queue. Each of the ten processes it creates will add its own PID to the queue.

Note that this program takes place in stages. First it launches ten processes, then they all do their work in parallel, and then it waits for them to complete (with `join`), so that it can process the results. It pulls data off the queue, puts it onto `mylist`, and then performs some calculations on the data it has retrieved.

The implementation of queues is so smooth and easy to work with, it's easy to forget that these queues are using some serious behind-the-scenes operating system magic to keep things coordinated. It's easy to think that you're working with threading, but that's just the point of multiprocessing; it might feel like threads, but each process runs separately. This gives you true concurrency within your program, something threads cannot do.

Conclusion

Threading is easy to work with, but threads don't truly execute in parallel. Multiprocessing is a module that provides an API that's almost identical to that of threads. This doesn't paper over all of the differences, but it goes a long way toward making sure things aren't out of control. ■

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.

Smart-Home Lightning Hacks

Home automation should make life simpler, not more complex!

By Shawn Powers

Shawn Powers is Associate Editor here at *Linux Journal*, and has been around Linux since the beginning. He has a passion for open source, and he loves to teach. He also drinks too much coffee, which often shows in his writing.

Kyle Rankin occasionally uses the "lightning hacks" format for his Hack and / *LJ* column when he has a bunch of neat topics to cover that wouldn't be enough for a complete article on their own. Thinking along those lines for this article, I figured it would be great to cover various home-automation stuff I do. Not only is it fun to share ideas, but if I make a list of all the cool things I'm currently doing, it will make it easier to compare the functionality of open-source options I'd like to explore next. If you haven't been dipping your toes into the world of home automation, maybe some of these hacks will change your mind.

My Setup

Most home-automation ideas can be implemented in multiple ways. In fact, I'm counting on that as I look into less-proprietary methods in the near future. But right now, I'm using a Samsung SmartThings hub. Yes, it is proprietary, but Samsung really has opened up the API and allowed developers to create device drivers and apps to customize the platform. I think SmartThings is the most feature-complete solution for home automation

right now, but it does have a few frustrations. The most annoying is that it requires a constant connection to the internet in order to function. Most folks are frustrated with the inherent privacy concerns of home automation taking place in the cloud, and that's a big problem. For me, the more frustrating aspect is the effect shoddy internet service has on a home. If the internet goes down, so does 90% of my house! I have a few workarounds, but I know that a solid (not fast) internet connection is vital if your solution is cloud-based like SmartThings.

Anyway, my setup consists of the following:

- Samsung SmartThings Hub v2.
- Amazon Echo devices all over the house.
- Google Home devices all over the house.
- Sonos speakers in most rooms.
- Various lights, switches, sensors and so on.

Having both Amazon Echo and Google Home isn't something I encourage; it's just that I have a habit of trying new technology, and they are both so amazing, I haven't yet chosen one over the other. Thankfully, they're pretty happy to function together.

Hack 1: the Mailman Detector

In my home, my office is all the way at the back of the house. In most homes, that's not a big deal, but in my case, my office is purposefully separate from the main living area so that when I'm recording videos, the house sounds don't interrupt. During the day, I'm usually home working alone, so I never know if a package has been delivered. The mailman could kick the front door down, and I'd never hear it in my office. My solution was to install a doorbell with a label about my office being all the way in the back (Figure 1), but sadly, most delivery folks just drop and run. So I decided to automate.

Figure 1. No one rings this. Maybe they feel bad bothering me?

The main device for detecting motion on the porch is, not surprisingly, a motion detector. I'm using an older Samsung motion detector, but if you're considering buying something for a hack like this, look at the reviews online. Samsung's latest upgrade gets many poor reviews. I'm using the Samsung model because it came with my hub, but I've had incredibly good luck with the Aeotec Multisensor (Figure 2).

Sensing motion is only the first step, of course, because I need to be notified when someone is on my front porch. My first solution was to have SmartThings trigger a text-to-speech message on the Sonos speaker in my office. The problem with that is that in the middle of recording videos, my speaker would blare "there's someone on your front porch, master", which was funny, but also frustrating during recording. After a half day of that, I switched to a push notification from SmartThings. I'd see it on my phone, but it wouldn't interrupt what I was doing.

Figure 2. This sensor looks a bit like a camera, so be prepared for glares.

I'm still working on a few issues with my mailman trap. After I adjusted the sensitivity (an app option), the system worked well all summer. Once the leaves started to fall, however, the slightest breeze would trigger a notification as the fallen leaves rustled past. No degree of sensitivity adjustment would find the balance between not notifying of leaves and still triggering when the mailman walked up. Thankfully, I was able to move the sensor to the overhang on my porch. It points down and never sees leaves, but it easily notices people on the porch. The key points: place your sensors well and don't go overboard on notifications. The loud ones are cool, but they become annoying quickly.

Hack 2: the Shower Vent

I'm a bit finicky about my shower experience. I really dislike being cold, so when I get into the shower, I usually don't turn on the exhaust vent, because it makes a

very unpleasant draft. Because I'm usually naked while showering, that draft is really annoying. Yet, if I don't turn on the exhaust fan, the bathroom turns into a swamp. Thankfully, that same Multisensor I mentioned before has a humidity sensor as well. Since it functions on battery power, I can place the sensor in the bathroom without worrying about electrocuting anyone.

The hack requires that your vent is on a smart-home controlled switch, but apart from that, it's a simple event trigger. When the humidity level reaches a certain point, the exhaust fan kicks on. Then it stays on until the level drops back down to normal. I'll be honest, the "turning back off" thing is something I hadn't considered when setting up the automation. Now, the bathroom controls its own humidity quite nicely.

As a bonus, the multisensor has a motion detector, so I pointed it so that it sees the door, and now at night when I stumble in to pee, the light comes on automatically. The only downside of the sensor is that it looks a bit like a spy camera, so having it in the bathroom creeps some people out. Thankfully, it's less creepy when it points at the door, and it doesn't need to be inside the shower in order to detect humidity. Still, you might need to explain the system to guests so they don't think you're a pervert.

Hack 3: Digital Dog Squad

I live in a low-crime area of northern Michigan, but when we're out of town, I still worry. I currently have security cameras up, but before that, I sort of booby-trapped our house when we were gone for an extended time. I thought it was silly, but then a desperate text message made me realize how powerful automation can be. I've had several iterations of this hack.

My first faux-security setup was to have our house in an "away" mode, and then tie the door sensors to an event that would play an MP3 file over all the Sonos speakers in the house. I got a vicious-sounding recording of dogs barking, and then whenever a door opened (while in away mode), all the speakers would turn their volume up and play the sound.

This was incredibly fun.

Unfortunately, it wasn't really a deterrent; it was just a great way to scare the crap out of people as they came into the house. I realized that although it was great at "catching" someone, it did very little to convince them they were in danger. Anyone casing the house would realize there were no dogs. The best we could wish for was criminals leaving over concern the neighbors would hear the fake dogs. But still, it was fun to put the house in away mode right before my teenage daughters opened the door.

Version 2 of the hack was far more effective. I used the same triggering mechanism, but rather than dogs barking, I had the system play a text-to-speech message at medium volume from the kitchen. It was just loud enough to hear clearly, but obviously not intended to startle anyone. The message was something like: "Unauthorized motion detected. Authorities contacted. Please stand by for video surveillance upload."

It seems a bit cheesy, but I know it's effective, because my brother in law stopped by our house one evening unannounced. He frantically texted me and called my wife trying to assure us that no one was breaking into the house and to please call the police so they didn't arrest him. It was hilarious.

Is this sort of hack a great security system? Absolutely not. But it certainly is a fun way to make your smart house freak people out. So don't rely on digital barking dogs to scare away a thief, but by all means, scare your teenage kids when when come home past curfew.

Hack 4: the Door Is Always Locked

First off, let me warn you about a really stupid mistake. Don't tie the ability to unlock your door to an Amazon Echo or Google Home. I realized my mistake one day when I saw our Echo through the window as I approached our front door. I literally yelled at my house, "Echo, turn off front door lock", and sure enough, my door unlocked. The strange wording is because Samsung and Amazon try hard to avoid situations like that, and I had to do some crazy hacking in order to get the ability to unlock the door. I now see why it's such a dumb idea to have a verbal command to unlock your door.

Figure 3. These locks are awesome. They're easy to install too.

And, I feel absurdly stupid. Nevertheless, my next lock automation is worth setting up. But please, learn from my foolishness.

SmartThings has a cool automation that allows you to keep your door locked all the time. This is a really great feature, because even when you're home, it's a good idea to keep your doors locked. Unfortunately, I remember to lock the door only when leaving and at night when going to bed. Usually. So now, using a combination of the Kwikset 910 deadbolt (Figure 3) and a Samsung door sensor (Figure 4), our doors lock themselves after being shut for one minute. It doesn't matter what time of day or who is home; after one minute, the deadbolt locks. The system is smart enough *not* to lock the deadbolt if the door is open, so you won't get a deadbolt sticking out of the door while you haul in groceries.

Figure 4. These actually detect opening, temperature and vibration—so cool!

This automation is one of the most useful of all the automations in my house. I never worry whether we locked the doors. I never worry that someone might sneak into the house while I'm back in my office. The door is always locked. And since the Kwikset 910 has a keypad for unlocking, there's no worry that my family will be locked out. Plus, with SmartThings, there's a great automation called Lock Manager, which allows me to give friends and family temporary codes from anywhere. I also can create "burner" codes that will work only a set number of times.

Having a deadbolt that is connected to the network and that potentially could be hacked does concern me a bit; I won't lie. But having the convenience of automatically locking the door and being able to re-key remotely is worth it for me.

Hack 5: a Family of Big Brothers

I've mentioned Life360 before. It's always nice to see where your family members are, and although that might seem creepy, we can all track each other (Figure 5). It's not just parents spying on their kids, it's family members being able to track each other. We're not creeped out by it, but it isn't for everyone. If you're okay with the idea of your every movement being tracked, it allows for some great home automation.

Samsung has a "presence sensor", which is a device that the SmartThings hub senses as it approaches. If you're concerned about your movements being tracked, the keychain-like fobs might be a better solution for you. Still, Life360 functions as a presence sensor, plus more.

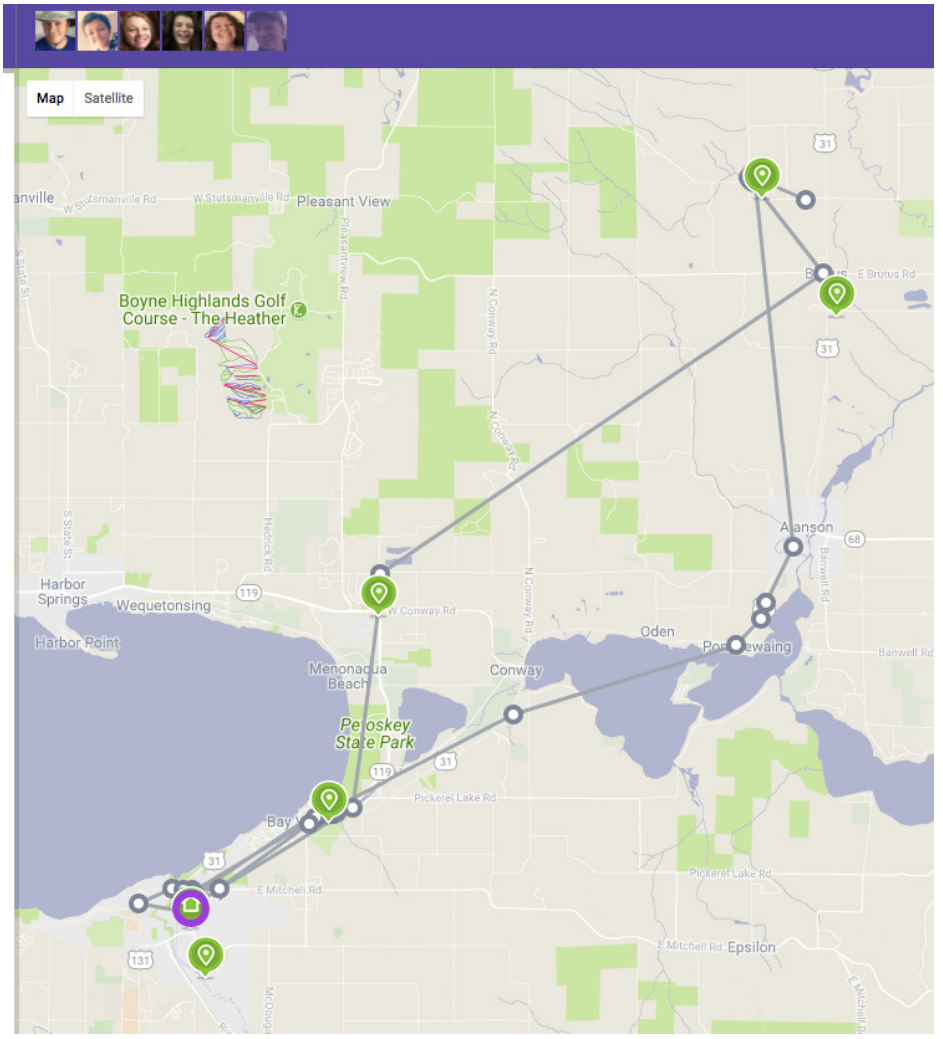


Figure 5. Just a day of errands. Creepy? A little. Useful? For sure.

I have our system configured so that when everyone is "gone" (Life360 tells the hub when people come and go) to adjust the thermostat, turn off lights and stop the mailman notifications. It's a great way to save energy, and it requires nothing more than everyone taking their phones with them. My teenage daughters never have a problem remembering their phones, so it works well.

Another cool bonus is that there is direct integration with Amazon Echo and Life360 now. We can be sitting in the living room watching TV, and say, "Echo, ask Life360 where is Lydia", and Alexa will tell us Lydia's location. I pay for the "pro" version of Life360, so I've configured multiple named locations all over town. If Lydia is at Starbucks, for instance, Alexa will tell us that she's at Starbucks and how long it will take her to get home. If she's not at a pre-defined location, it still tells us her location (street names) and how long it will take her to get home. It's really useful, and the privacy concerns don't bother us. Again, that level of familial transparency isn't for everyone, but it means fewer calls and texts to driving teenagers. For me, that alone is worth it.

My Hacks, Your Hacks

I purposefully didn't explain the specific instructions for setting up these hacks, because the process will be different for every smart-home implementation. For me, the "ah-ha" moments are coming up with interesting ways to make my smart house work for me. If you have trouble implementing any of these hacks and no amount of googling helps, drop me an email via the LJ contact form (or email ljeditor@linuxjournal.com), and I'll try to clarify what I did.

All that said, I'd *love* to hear unique and useful ways you're using the smart-home devices in your house. The implementation is often the easy part; it's the creative thinking that is the magic. Please share your smart-home ideas with me via email, ljeditor@linuxjournal.com, and I'll be sure to share the really cool ones with everyone, giving you full credit. Have fun, and hack on! ■

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.

What's New in Kernel Development

Zack Brown is a tech journalist at *Linux Journal* and *Linux Magazine*, and is a former author of the "Kernel Traffic" weekly newsletter and the "Learn Plover" stenographic typing tutorials. He first installed Slackware Linux in 1993 on his 386 with 8 megs of RAM and had his mind permanently blown by the Open Source community. He is the inventor of the *Crumble* pure strategy board game, which you can make yourself with a few pieces of cardboard. He also enjoys writing fiction, attempting animation, reforming Labanotation, designing and sewing his own clothes, learning French and spending time with friends'n'family.

Linus' Posting Habits

Linus Torvalds sometimes is criticized for bombastically cursing out kernel developers. He does do this, but it's not his default behavior, and I think the real nature of when and how he posts to the mailing list is interesting. For example, he stayed out of the whole discussion of how to replace the **BitKeeper** revision control system for a long time, letting various projects guess frustratingly at his desires, before he finally took a break from Linux development to design and implement **git**.

In other cases, he's allowed developers to lambaste each other savagely for days or longer over key elements of the kernel or the behaviors of new hardware peripherals, only to enter the debate later on, generally to propose a simpler solution that neither camp had thought of.

Sometimes he'll enter a discussion for apparently no other reason than that a particular bug or piece of code interests him, and he works with whoever posted a given patch to get the kinks out or track down underlying problems.

In general, Linus tends to stay out of most discussions, coming in primarily only on controversial issues after he's developed a

position of his own.

But yes, sometimes he comes down pretty hard on developers, generally saying they already should know better than to do a particular thing, or when he feels the developer is hawking a particular corporate goal that goes against the spirit of open-source development—although I doubt he'd put it that way himself. He doesn't seem to like making overtly evangelical statements and tends to stick to the "if it works, I like it" attitude that he took when adopting BitKeeper.

Occasionally, Linus gets a technical issue wrong—he'll claim something about the kernel as being true that isn't. An example of that happened recently, when **Amir Goldstein** posted a patch to alter a string hashing function. It was a small patch that preserved some dropped bits in the 64-bit case and left the 32-bit case virtually unchanged. Amir asked for Linus' advice because he couldn't find a maintainer for the file, and Linus had been one of the people most recently to change the code.

Linus didn't agree that Amir's code left the 32-bit case unchanged. And he singled out a call to the **__hash_32()** function as being particularly time-consuming. He rejected Amir's patch, saying, "the patch as-is doesn't seem to buy anything, and only adds cost."

But when Amir pointed out that his patch hadn't actually added the call to __hash_32(), that the call had been there already, Linus took another look and replied, "Oh, duh. My bad. It was indeed there already. Let me go back and look at the history of this thing."

He later replied to his own post, saying:

> After having looked more at it, I take back all my complaints about the patch, you were right and I was mis-reading things or just being stupid.
>
> I also don't worry too much about the possible performance impact of this on 64-bit, since most architectures that actually care about performance end up not using this very

> much (the dcache code is the most performance-critical, but the word-at-a-time case uses its own hashing anyway).
>
> So this ends up being mostly used for filesystems that do their own degraded hashing (usually because they want a case-insensitive comparison function).
>
> A _tiny_ worry remains, in that not everybody uses DCACHE_WORD_ACCESS, and then this potentially makes things more expensive on 64-bit architectures with slow or lacking multipliers even for the normal case.
>
> That said, realistically the only such architecture I can think of is PA-RISC. Nobody really cares about performance on that, it's more of a 'look ma, I've got warts^W an odd machine' platform.
>
> So I think your patch is fine, and all my initial worries were just misplaced from not looking at this properly.
>
> Sorry.

To me the interesting details about this kind of post are that Linus never seems to drag out an admission of error. He won't cling to a mistaken idea out of embarrassment; he also avoids making excuses for why he got the thing wrong or dismissing the opposing side as unimportant. He also seems to put in some real work in developing a proper understanding of the situation after seeing that he made a mistake and moves directly into dealing with whatever the new technical details might be.

I personally like all that. And for me, it adds context to the times when he gets angry at developers. It seems to happen mostly—or only—when he feels that developers should have seen and acknowledged their own error by a given point.

Intel Design Flaw Fallout

For weeks, the world's been talking about severe **Intel** design flaws affecting many

CPUs and forcing operating systems to look for sometimes costly workarounds.

Linux patches for these issues are in a state of ongoing development. Security is always the first priority, at the expense of any other feature. Next would probably be the general speed of a running system for the average user. After that, the developers might begin piecing together any features that had been pulled as part of the initial security fix.

But while this effort goes on, the kernel developers seem fairly angry at Intel, especially when they feel that Intel is not doing enough to fix the problems in future processors.

In response to one set of patches, for example, **Linus Torvalds** burst out with, "All of this is pure garbage. Is Intel really planning on making this shit architectural? Has anybody talked to them and told them they are f*cking insane?" He went on, "the IBRS garbage implies that Intel is _not_ planning on doing the right thing for the indirect branch speculation. Honestly, that's completely unacceptable." And then he said:

> The whole IBRS_ALL feature to me very clearly says "Intel is not serious about this, we'll have an ugly hack that will be so expensive that we don't want to enable it by default, because that would look bad in benchmarks". So instead they try to push the garbage down to us. And they are doing it entirely wrong.

He went on, even more disturbingly, to say:

> The patches do things like add the garbage MSR writes to the kernel entry/exit points. That's insane. That says "we're trying to protect the kernel". We already have retpoline there, with less overhead.
>
> So somebody isn't telling the truth here. Somebody is pushing complete garbage for unclear reasons. Sorry for having to point that out....As it is, the patches are COMPLETE AND UTTER GARBAGE....WHAT THE F*CK IS GOING ON?

At one point, **David Woodhouse** offered a helpful technical summary of the whole situation for those of us on the edge of our seats:

> This is all about Spectre variant 2, where the CPU can be tricked into mispredicting the target of an indirect branch. And I'm specifically looking at what we can do on *current* hardware, where we're limited to the hacks they can manage to add in the microcode.
>
> The new microcode from Intel and AMD adds three new features.
>
> One new feature (IBPB) is a complete barrier for branch prediction. After frobbing this, no branch targets learned earlier are going to be used. It's kind of expensive (order of magnitude ~4000 cycles).
>
> The second (STIBP) protects a hyperthread sibling from following branch predictions which were learned on another sibling. You *might* want this when running unrelated processes in userspace, for example. Or different VM guests running on HT siblings.
>
> The third feature (IBRS) is more complicated. It's designed to be set when you enter a more privileged execution mode (i.e. the kernel). It prevents branch targets learned in a less-privileged execution mode, BEFORE IT WAS MOST RECENTLY SET, from taking effect. But it's not just a "set-and-forget" feature, it also has barrier-like semantics and needs to be set on *each* entry into the kernel (from userspace or a VM guest). It's *also* expensive. And a vile hack, but for a while it was the only option we had.
>
> Even with IBRS, the CPU cannot tell the difference between different userspace processes, and between different VM guests. So in addition to IBRS to protect the kernel, we need the full IBPB barrier on context switch and vmexit. And maybe STIBP while they're running.
>
> Then along came Paul with the cunning plan of "oh, indirect branches can be exploited? Screw it, let's not have any of *those* then", which is retpoline. And it's a *lot* faster than frobbing IBRS on every entry into the kernel. It's a massive performance win.

So now we *mostly* don't need IBRS. We build with retpoline, use IBPB on context switches/vmexit (which is in the first part of this patch series before IBRS is added), and we're safe. We even refactored the patch series to put retpoline first.

But wait, why did I say "mostly"? Well, not everyone has a retpoline compiler yet…but OK, screw them; they need to update.

Then there's Skylake, and that generation of CPU cores. For complicated reasons they actually end up being vulnerable not just on indirect branches, but also on a "ret" in some circumstances (such as 16+ CALLs in a deep chain).

The IBRS solution, ugly though it is, did address that. Retpoline doesn't. There are patches being floated to detect and prevent deep stacks, and deal with some of the other special cases that bite on SKL, but those are icky too. And in fact IBRS performance isn't anywhere near as bad on this generation of CPUs as it is on earlier CPUs *anyway*, which makes it not quite so insane to *contemplate* using it as Intel proposed.

That's why my initial idea, as implemented in this RFC patchset, was to stick with IBRS on Skylake, and use retpoline everywhere else. I'll give you 'garbage patches', but they weren't being 'just mindlessly sent around'. If we're going to drop IBRS support and accept the caveats, then let's do it as a conscious decision having seen what it would look like, not just drop it quietly because poor Davey is too scared that Linus might shout at him again.

I have seen *hand-wavy* analyses of the Skylake thing that mean I'm not actually lying awake at night fretting about it, but nothing concrete that really says it's OK.

If you view retpoline as a performance optimisation, which is how it first arrived, then it's rather unconventional to say "well, it only opens a *little* bit of a security hole but it does go nice and fast so let's do it".

But fine, I'm content with ditching the use of IBRS to protect the kernel, and I'm not even surprised. There's a *reason* we put it last in the series, as both the most contentious

> and most dispensable part. I'd be *happier* with a coherent analysis showing Skylake is still OK, but hey-ho, screw Skylake.
>
> The early part of the series adds the new feature bits and detects when it can turn KPTI off on non-Meltdown-vulnerable Intel CPUs, and also supports the IBPB barrier that we need to make retpoline complete. That much I think we definitely *do* want. There have been a bunch of us working on this behind the scenes; one of us will probably post that bit in the next day or so.
>
> I think we also want to expose IBRS to VM guests, even if we don't use it ourselves. Because Windows guests (and RHEL guests; yay!) do use it.
>
> If we can be done with the shouty part, I'd actually quite like to have a sensible discussion about when, if ever, we do IBPB on context switch (ptraceability and dumpable have both been suggested) and when, if ever, we set STIPB in userspace.

Most of the discussion on the mailing list focused on the technical issues surrounding finding actual solutions. But Linus was not alone in finding the situation unacceptable. A variety of developers, including David, were horribly offended, not by the design flaw itself, but by the way they perceived Intel to be handling the subsequent situation—the poor technical fixes, the lack of communication between Intel developers and the kernel community, and as Linus pointed out, the potential choice by Intel not to fix some of the problems at all.

Speeding Up the Un-Speed-Up-able

Sometimes kernel developers can work in parallel for years without realizing it. It's one of the inefficiencies of a distributed system that tends to work out as a benefit when you have enough contributors to be able to spare the extra labor—it's sort of a "with enough eyeballs, all bugs are shallow" kind of thing.

This time **Kirill A. Shutemov** wanted to scratch a particular itch in the **memory encryption code**. Memory encryption occurs at such a heavily used place in the kernel that a difference of just a couple opcodes can make a noticeable speed

difference to a running system. Anything short of introducing security holes would be worth considering, in order to shave off those few **Assembly** instructions.

But Kirill had a problem—his code made things worse. He wanted to convert the `__PHYSICAL_MASK` variable into what he called a "patchable constant"—a value that had the simplicity of a constant at runtime, but that could be set dynamically by configuration options or at the bootloader command line before booting the system.

So far, he had come up with a patch that did this—and that would be applicable to other variables that might be faster to implement as constants. Unfortunately, as implemented, although it achieved the feature he wanted, it caused **GCC** to produce code that was less efficient than what it had produced before. Instead of speeding things up, this resulted in a 0.2% slowdown on his test system.

He posted his patch, along with a request for ideas, to the linux-kernel mailing list.

Linus Torvalds replied with an analysis of the opcodes produced by GCC, showing why they were slower and why Kirill's general approach always would suffer from the slowdown issue. In particular, Linus pointed out that Kirill moved constant values into registers, which never would be optimal, and his code also included a **movabsq** instruction, which he felt was rarely needed.

Linus said he actually really wanted this feature in spite of the complex overhead required to accomplish it; in fact, he preferred an even more complex approach, along the lines of something **H. Peter Anvin** had attempted about 18 months earlier. Of that approach, he said:

> It's rather more complex, but it actually gives a much bigger win. The code itself will be much better, and smaller. The *infrastructure* for the code gets pretty hairy, though.

Peter replied that he'd actually been working on this very project for some time from a secret, undisclosed location at the bottom of a well in a town with no name. In fact, his current approach was yet more ambitious (and complex) than what he'd

tried 18 months earlier. On the other hand, as he put it, he was now about 85% done with the whole thing, and the code "mostly needs cleanup, testing, and breaking into reasonable chunks." He added:

> The main reason I haven't submitted it yet is that I got a bit overly ambitious and wanted to implement a whole bunch of more complex subcases, such as 64-bit shifts on a 32-bit kernel. The win in that case is actually quite huge, but it requires data-dependent code patching and not just immediate patching, which requires augmentation of the alternatives framework.

So it looks like Kirill's itch is about to be scratched…by Peter. An element of the kernel so deep and tangled that it seemed intended never to be touched is being tugged apart beyond its apparent natural limits. And, all of this is achieved by the mere addition of a big honking pile of insanely complex code that even seasoned developers balked at touching. ■

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.

Tackling L33t-Speak

How to script a l33t-speak translator.

By Dave Taylor

Dave Taylor has been hacking shell scripts on Unix and Linux systems for a really long time. He's the author of *Learning Unix for Mac OS X* and *Wicked Cool Shell Scripts*. He can be found on Twitter as @DaveTaylor and you can reach him through his tech Q&A site Ask Dave Taylor.

My daughter and I were bantering with each other via text message this morning as we often do, and I dropped into a sort of mock "leet speak". She wasn't impressed, but it got me thinking about formulaic substitutions in language and how they represent interesting programming challenges.

If you're not familiar with "leet speak" it's a variation on English that some youthful hackers like to use—something that obscures words sufficiently to leave everyone else confused but that still allows reasonably coherent communication. Take the word "elite", drop the leading "e" and change the spelling to "leet". Now replace the vowels with digits that look kind of, sort of the same: l33t.

There's a sort of sophomoric joy in speaking—or writing—l33t. I suppose it's similar to pig latin, the rhyming slang of East Londoners or the reverse-sentence structure of Australian shopkeepers. The intent's the same: it's us versus them and a way to share with those in the know without everyone else understanding what you're saying.

At their heart, however, many of these things are just substitution ciphers. For example, "apples and pears" replaces

"stairs", and "baked bean" replaces "queen", in Cockney rhyming slang.

It turns out that l33t speak is even more formalized, and there's actually a Wikipedia page that outlines most of its rules and structure. I'm just going to start with word variations and letter substitutions here.

The Rules of L33t Speak

Okay, I got ahead of myself. There aren't "rules", because at its base, leet speak is a casual slang, so l33t and 733T are both valid variations of "elite". Still, there are a lot of typical substitutions, like dropping an initial vowel, replacing vowels with numerical digits or symbols (think "@" for "a"), replacing a trailing "s" with a "z", "cks" with "x" (so "sucks" becomes "sux"), and the suffixed "ed" becomes either 'd or just the letter "d".

All of this very much lends itself to a shell script, right? So let's test some mad skillz!

For simplicity, let's parse command-line arguments for the l33t.sh script and use some level of randomness to ensure that it's not too normalized. How do you do that in a shell script? With the variable `$RANDOM`. In modern shells, each time you reference that variable, you'll get a different value somewhere in the range of `1..MAXINT`. Want to "flip a coin"? Use `$(($RANDOM % 2))`, which will return a zero or 1 in reasonably random order.

So the fast and easy way to go through these substitutions is to use `sed`—that old mainstay of Linux and UNIX before it, the stream editor. Mostly I'm using `sed` here, because it's really easy to use substitute/pattern/newpattern/—kind of like this:

```
word="$(echo $word | sed "s/ed$/d/")"
```

This will replace the sequence "ed" with just a "d", but only when it's the last two letters of the word. You wouldn't want to change education to ducation, after all.

Here are a few more that can help:

```
word="$(echo $word | sed "s/s$/z/")"
word="$(echo $word | sed "s/cks/x/g;s/cke/x/g")"
word="$(echo $word | sed "s/a/@/g;s/e/3/g;s/o/0/g")"
word="$(echo $word | sed "s/^@/a/")"
word="$(echo $word |  tr "[[:lower:]]" "[[:upper:]]")"
```

In order, a trailing "s" becomes a trailing "z"; "cks" anywhere in a word becomes an "x", as does "cke"; all instances of "a" are translated into "@"; all instances of "e" change to "3"; and all instances of "o" become "0". Finally, the script cleans up any words that might start with an "a". Finally, all lowercase letters are converted to uppercase, because, well, it looks cool.

How does it work? Here's how this first script translates the sentence "I am a master hacker with great skills":

```
I AM A M@ST3R H@XR WITH GR3@T SKILLZ
```

More Nuances

It turns out that I missed some nuances of Leet and didn't realize that most often the letter "a" is actually turned into a "4", not an "@", although as with just about everything about the jargon, it's somewhat random.

In fact, every single letter of the alphabet can be randomly tweaked and changed, sometimes from a single letter to a sequence of two or three symbols. For example, another variation on "a" is "/-\" (for what are hopefully visually obvious reasons).

Continuing in that vein, "B" can become "|3", "C" can become "[", "I" can become "1", and one of my favorites, "M" can change into "[]V[]". That's a lot of work, but since one of the goals is to have a language no one else understands, I get it.

There are additional substitutions: a word can have its trailing "S" replaced by a "Z", a trailing "ED" can become "'D" or just "D", and another interesting one is

that words containing “and”, “anned” or “ant” can have that sequence replaced by an ampersand (&).

Let’s add all these L337 filters and see how the script is shaping up.

But First, Some Randomness

Since many of these transformations are going to have a random element, let’s go ahead and produce a random number between 1–10 to figure out whether to do one or another action. That’s easily done with the `$RANDOM` variable:

```
doit=$(( $RANDOM % 10 ))       # random virtual coin flip
```

Now let’s say that there’s a 50% chance that a -ed suffix is going to change to “’D” and a 50% chance that it’s just going to become “D”, which is coded like this:

```
if [ $doit -ge 5 ] ;  then
  word="$(echo $word | sed "s/ed$/d/")"
else
  word="$(echo $word | sed "s/ed$/'d/")"
fi
```

Let’s add the additional transformations, but not do them every time. Let’s give them a 70–90% chance of occurring, based on the transform itself. Here are a few examples:

```
if [ $doit -ge 3 ] ;  then
  word="$(echo $word | sed "s/cks/x/g;s/cke/x/g")"
fi

if [ $doit -ge 4 ] ;  then
  word="$(echo $word | sed "s/and/\&/g;s/anned/\&/g;
     s/ant/\&/g")"
fi
```

And so, here's the second translation, a bit more sophisticated:

```
$ l33t.sh "banned? whatever. elite hacker, not scriptie."
B&? WH4T3V3R. 3LIT3 H4XR, N0T SCRIPTI3.
```

Note that it hasn't realized that "elite" should become L337 or L33T, but since it *is* supposed to be rather random, let's just leave this script as is. Kk? Kewl.

If you want to expand it, an interesting programming problem is to break each word down into individual letters, then randomly change lowercase to uppercase or vice versa, so you get those great ransom-note-style WeiRD LeTtEr pHrASes. ■

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.

DEEP DIVE

INTO THE CLOUD

Everything You Need to Know about the Cloud and Cloud Computing, Part I

An in-depth breakdown of the technologies involved in making up the cloud and a survey of cloud-service providers.

By Petros Koutoupis

The cloud has become synonymous with all things data storage. It additionally equates to the many web-centric services accessing that same back-end data storage. But the term also has evolved to mean so much more.

Cloud computing provides more simplified access to server, storage, database and application resources, with users provisioning and using the minimum set of requirements they see fit to host their application needs. In the past decade alone, the paradigm shift toward a wider and more accessible network has forced both hardware vendors and service providers to rethink their strategies and cater to a new model of storing information and serving application resources. As time continues to pass, more individuals and businesses are connecting themselves to this greater world of computing.

What Is the Cloud?

Far too often, the idea of the "cloud" is confused with the general internet. Although

it's true that various components making up the cloud can be accessible via the internet, they are not one and the same. In its most general terms, cloud computing enables companies, service providers and individuals to provision the appropriate amount of computing resources dynamically (compute nodes, block or object storage and so on) for their needs. These application services are accessed over a network—and not necessarily a public network. Three distinct types of cloud deployments exist: public, private and a hybrid of both.

The public cloud differentiates itself from the private cloud in that the private cloud typically is deployed in the data center and under the proprietary network using its cloud computing technologies—that is, it is developed for and maintained by the organization it serves. Resources for a private cloud deployment are acquired via normal hardware purchasing means and through traditional hardware sales channels. This is not the case for the public cloud. Resources for the public cloud are provisioned dynamically to its user as requested and may be offered under a pay-per-usage model or for free.

Some of the world's leading public cloud offering platforms include:

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform
- IBM Cloud (formerly SoftLayer)

As the name implies, the hybrid model allows for seamless access and transitioning between both public and private deployments, all managed under a single framework.

For those who prefer either to host their workload internally or partially on the public cloud—sometimes motivated by security, data sovereignty or compliance—private and hybrid cloud offerings continue to provide the same amount of service but all

within your control.

Using cloud services enables you to achieve the following:

- Enhance business agility with easy access and simplified deployment of operating systems/applications and management of allocated resources.

- Reduce capital expenses and optimize budgets by reducing the need to worry about the hardware and overall data center/networking infrastructure.

- Transform how you deliver services and manage your resources that flexibly scale to meet your constantly evolving requirements.

- Improve operations and efficiency without worrying about hardware, cooling or administration costs—you simply pay for what you need and use (this applies to the public option).

Adopting a cloud model enables developers access to the latest and greatest tools and services needed to build and deploy applications on an instantly available infrastructure. Faster development cycles mean that companies are under pressure to compete at a faster velocity or face being disrupted in the market. This new speed of business coupled with the mindset of "on-demand" and "everything-as-a-service" extends beyond application developers into finance, human resources and even sales.

The Top Players in the Industry

For this Deep Dive series, I focus more on the public cloud service providers and their offerings with example deployments using Amazon's AWS. Why am I focusing on the public cloud? Well, for most readers, it may appear to be the more attractive option. Why is that? Unlike private cloud and on-premises data-center deployments, public cloud consumption is designed to relieve maintainers of the burden to invest and continuously update their computing hardware and networking infrastructures. The reality is this: hardware gets old and it gets old relatively quickly.

Public clouds are designed to scale, theoretically, without limit. As you need to provision more resources, the service providers are well equipped to meet those requirements. The point here is that you never will consume all of the capacity of a public cloud.

The idea is to reduce (and potentially remove) capital expenditure (capex) significantly and focus more on the now-reduced operational expenditure (opex). This model allows for a company to reduce its IT staffing and computing hardware/software costs. Instead of investing heavily upfront, companies are instead subscribing to the infrastructure and services they need. Remember, it's a pay-as-you-go model.

Note: service providers are not limited to the ones mentioned here. This listing consists of the providers with the largest market share.

Amazon

First launched in 2006, Amazon Web Services (AWS) offers a suite of on-demand cloud computing, storage and networking resources, and data storage. Today, AWS additionally provides an extensive list of services that focus on compute, storage, networking, database, analytics, developer tools and many more. The most well known of these services are the Amazon Elastic Compute Cloud (EC2) and the Amazon Simple Storage Service (S3). (I cover those offerings and more in Part II of this series.)

Amazon stands at the forefront of the public cloud with the largest market share of users—by a substantial amount at that. Being one of the very first to invest in the cloud, Amazon has been able to adapt, redefine and even set the trends of the industry.

Microsoft

Released to the general public in 2010, Microsoft Azure redefined computing to existing Microsoft ecosystems and beyond. Yes, this includes Linux. Much like AWS, Azure provides customers with a large offering of attractive cloud products that cover compute, storage, databases and more. Microsoft has invested big in the cloud, and it definitely shows. It's no surprise that Microsoft currently is the second-largest service provider, quickly approaching Amazon's market share.

Google

Google's entry into this market was an evolving one. Embracing the cloud and cloud-style computing early on, Google essentially would make available internally implemented frameworks to the wider public as it saw fit. It wasn't until around 2010 with the introduction of Google Cloud Storage that the foundations of the Google Cloud Platform were laid for the general public and future cloud offerings. Today, Google remains a very strong competitor.

IBM

IBM announced its cloud ambitions in 2012. The vision (and name) has evolved since that announcement, but the end result is the same: compute, storage and so on, but this time, with an emphasis on security, hybrid deployments and a connection to an Artificial Intelligence (AI) back end via the infamous Watson framework.

Architectural Overview

The whole notion of cloud computing has evolved since its conception. The technology organically grew to meet new consumer needs and workloads with even newer features and functionality (I focus on this evolution in Part II of this series). It would be extremely unfair and unrealistic of me to say that the following description covers all details of your standard cloud-focused data centers, so instead, I'm making it somewhat general.

Regions and Zones

In the case of Amazon, AWS places its data centers across 33 availability zones within 12 regions worldwide. A region is a separate geographic location. Each availability zone has one or more data centers. Each data center is equipped with 50,000–80,000 servers, and each data center is configured with redundant power for stability, networking and connectivity. Microsoft Azure runs across 36 global regions, each region consisting of a single data center serving that particular region. IBM Cloud operates from at least 33 single data-center regions worldwide, each consisting of thousands (if not tens of thousands) of servers.

As a user, you are able to place a virtual instance or resource in multiple locations

and across multiple zones. If those resources need to be replicated across multiple regions, you need to ensure that you adjust the resource to do exactly that.

This approach allows both users and corporations to leverage the cloud provider's high availability. For instance, if I am company *foo*, providing service *bar* and, in turn, rely on AWS to expose all the components I need to host and run service *bar*, it would be in my best interest to have those resources replicated to another zone or even a completely different region. If, for instance, region A experiences an outage (such as an internal failure, earthquake, hurricane or similar), I still would continue to provide service *bar* from region B (a standard failover procedure).

Every server or virtual instance in the public cloud is accessible via a public IP address. Using features like Amazon's Virtual Private Cloud (VPC), users are able to provision isolated virtual networks logically to host their AWS resources. And although a VPC is restricted to a particular region, there are methods by which one private network in one region can peer into the private network of another region.

Storage

When it comes to storage in the cloud, things begin to get interesting. At first glance and without any explanation, the illustration shown in Figure 1 may appear somewhat complicated, but I still ask that you take a moment to study it.

At the bottom, there are storage servers consisting of multiple physical storage devices or hard drives. There will be multiple storage servers deployed within a single location. Those physical volumes are then pooled together either within a single location or spanning across multiple locations to create a logical volume.

When a user requests a data volume for either a virtual machine instance or even to access it across a network, that volume of the specified size will be carved out from the larger pool of drives. In most cases, users can resize data volumes easily and as needed.

In the cloud, storage is presented to users in a few ways. First is the traditional block

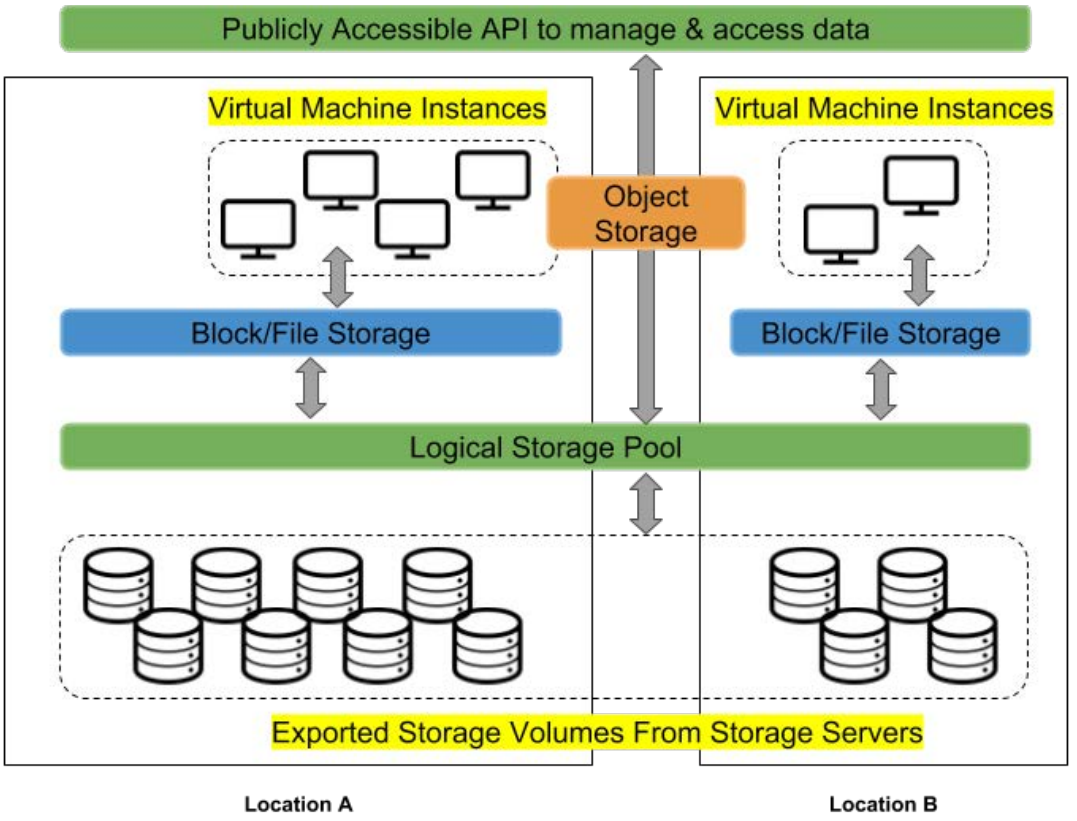


Figure 1. A High-Level Overview of a Typical Storage Architecture in the Cloud

device or filesystem, following age-old access methods, which can be mapped to any virtual machine running any operating system of your choice. These are referred to as Elastic Block Storage (EBS) and Elastic File System (EFS) on AWS. The other and more recent method is known as object storage and is accessible over a network via a REpresentational State Transfer (REST) API over HTTP.

Object storage differs from block/filesystem storage by managing data as objects. Each object will include the data itself, alongside metadata and a globally unique identifier. A user or application will access this object by requesting it by its unique identifier. There is no concept of a file hierarchy here, and no directories or subdirectories either. The technology has become quite standard for the more modern web-focused style of computing. It is very common to see it used to store photos, videos, music and more. Mentioned earlier, the AWS object solution is referred to as S3.

Virtualization

The key to the cloud's success is centered around the concept of Infrastructure-as-a-Service (IaaS) and its capabilities to serve virtual machines via a hypervisor's API. A hypervisor allows you to host multiple operating systems (that is, virtual machines) on the same physical hardware. Most modern hypervisors are designed to simulate multiple CPU architectures, which include Intel (x86 and x86-64), ARM, PowerPC and MIPS. In the case of the cloud, through a web-based front end, you are in complete control of all your allocated computing resources and also are able to obtain and boot a new server instance within minutes.

Think about it for a second. You can commission a single server or thousands of server instances simultaneously and within minutes—not in hours or days, but *minutes*. That's pretty impressive, right? All of this is controlled with the web service Application Program Interface (API). For those less familiar, an API is what glues services, applications and entire systems together. Typically, an API acts as a public persona for a company or a product by exposing business capabilities and services. An API geared for the cloud can be invoked from a browser, mobile application or any other internet-enabled endpoint.

With each deployed server instance, again, you are in full control. Translation: you have root access to each one (with console output) and are able to interact with them however you need. Through that same web service API, you are able to start or stop whichever instance is necessary. Cloud service providers allow users to select (virtual) hardware configurations—that is, memory, CPU and storage with drive partition sizes. Users also can choose to install from a list of multiple operating systems (including Linux distributions and Microsoft Windows Server) and software packages. And finally, it's worth noting, the more resources selected, the more expensive the server instance becomes.

Containers

Containers are about as close to bare metal that you can get when running virtual machines. Hosting virtual machines imposes very little to no overhead. This feature limits, accounts for and isolates CPU, memory, disk I/O and network usage of one

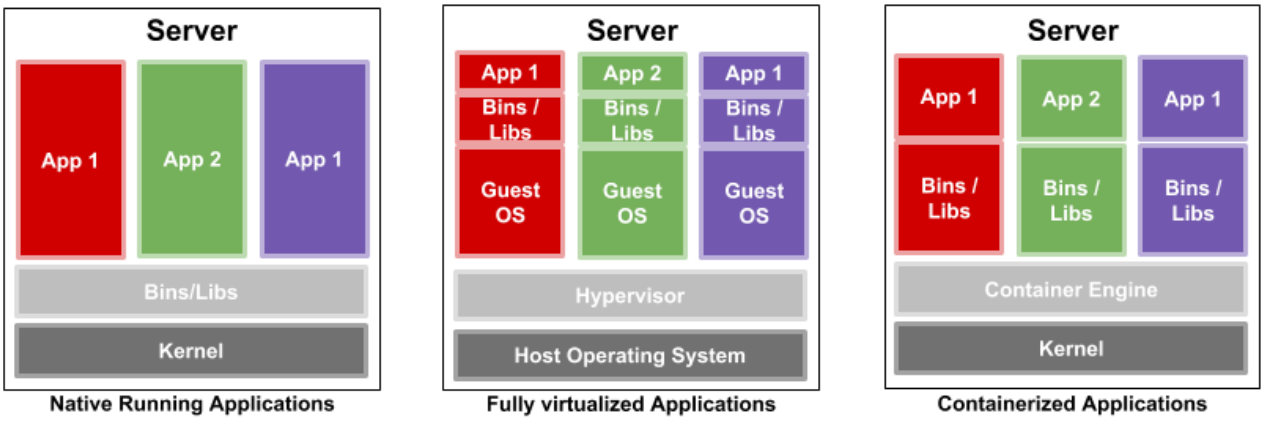


Figure 2. Illustrating the Difference between Running Applications on Bare-Metal Systems, Hypervisors and Containers

or more processes. Essentially, containers decouple software applications from the operating system, giving users a clean and minimal operating environment while running everything else in one or more isolated "containers".

This isolation prevents processes running within a given container from monitoring or affecting processes running in another container. Also, these containerized services do not influence or disturb the host machine. The idea of being able to consolidate many services scattered across multiple physical servers into one is one of the many reasons data centers have chosen to adopt the technology. This method of isolation adds to the security of the technology by limiting the damage caused by a security breach or violation. An intruder who successfully exploits a security hole on one of the applications running in that container is restricted to the set of actions possible within that container.

In the context of the cloud, containers simplify application deployment immensely by not only isolating the application from an entire operating system (virtualized or not) but also by being able to deploy with a bare minimum amount of requirements in both software and hardware, further reducing the headache of maintaining both.

Serverless Computing

Cloud native computing or serverless computing are more recent terms describing

the more modern trend of deploying and managing applications. The idea is pretty straightforward. Each application or process is packaged into its own container, which, in turn, is orchestrated dynamically (that is, scheduled and managed) across a cluster of nodes. This approach moves applications away from physical hardware and operating system dependency and into their own self-contained and sandboxed environment that can run anywhere within the data center. The cloud native approach is about separating the various components of application delivery.

This may sound identical to running any other container in the cloud, but what makes cloud native computing so unique is that you don't need to worry about managing that container (meaning less overhead). This technology is hidden from the developer. Simply upload your code, and when a specified trigger is enabled, an API gateway (maintained by the service provider) deploys your code in the way it is intended to process that trigger.

The first thing that comes to mind here is Amazon's AWS Lambda. Again, under this model, there's no need to provision or manage physical or virtual servers. Assuming it's in a stable or production state, simply upload your code and you are done. Your code is just deployed within an isolated containerized environment. In the case with Lambda, Amazon has provided a framework for developers to upload their event-driven application code (written in Node.js, Python, Java or C#) and respond to events like website clicks within milliseconds. All libraries and dependencies to run the bulk of your code are provided for within the container.

As for the types of events on which to trigger your application, Amazon has made it so you can trigger on website visits or clicks, a REST HTTP request to their API gateway, a sensor reading on your Internet-of-Things (IoT) device, or even an upload of a photograph to an S3 bucket.

Now, how do all these components come together? It begins with users accessing your service through a website or via an application on their mobile devices. The web server and the various components on which it relies may be hosted from locally managed containers or even virtual machines. If a particular function is required

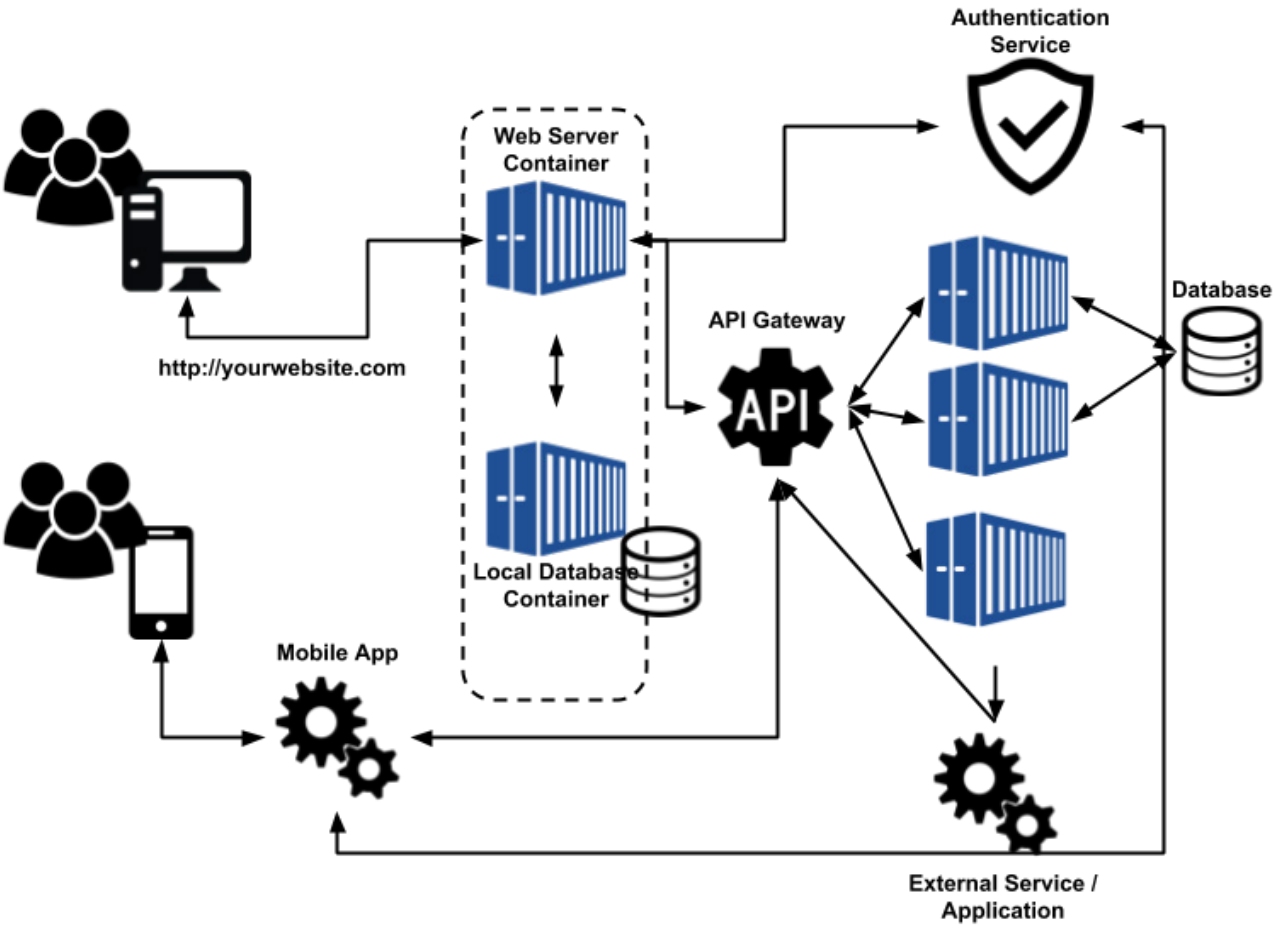


Figure 3. A Typical Model for Cloud Native Computing

by either the web server or the mobile application, it will reach out to a third-party authentication service, such as AWS Identity and Access Management (IAM) services, to gain the proper credentials for accessing the serverless functions hosted beyond the API gateway. When triggered, those functions will perform the necessary actions and return with whatever the web server or mobile application requested. (I cover this technology more in-depth in Part II.)

Security

It's generally assumed that private cloud and on-premises solutions are more secure than the public cloud options, but recent studies have shown this to not be the case. Public cloud service providers spend more time and resources consulting with

security experts and updating their software frameworks to limit security breaches. And although this may be true, the real security challenge in leveraging public cloud services is using it in a secure manner. For enterprise organizations, this means applying best practices for secure cloud adoption that minimizes lock-in where possible and maximizes availability and uptime.

I already alluded to identity and authentication methods, and when it comes to publicly exposed services, this feature becomes increasingly important. These authentication services enable users to manage access to the cloud services from their providers. For instance, AWS has this thing called IAM. Using IAM, you will be able to create and manage AWS users/groups and with permissions that allow/deny their access to various AWS resources or specific AWS service APIs. With larger (and even smaller) deployments, this permission framework simplifies global access to the various components making up your cloud. It's usually a free service offered by the major providers.

Summary

It's fair to say that this article covers a lot: a bit about what makes up the cloud, the different methods of deployment (public, private and hybrid) and the types of services exposed by the top cloud-service providers, followed by a general overview of how the cloud is designed, how it functions and how that functionality enables the features we enjoy using today.

In Part II of this series, I further explore the cloud using real examples in AWS. So, if you haven't done so already, be sure to register an account over at AWS. Charges to that account may apply. ■

Petros Koutoupis, *LJ* Contributing Editor, is currently a senior platform architect at IBM for its Cloud Object Storage division (formerly Cleversafe). He is also the creator and maintainer of the Rapid Disk Project. Petros has worked in the data storage industry for well over a decade and has helped pioneer the many technologies unleashed in the wild today.

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.

Everything You Need to Know about the Cloud and Cloud Computing, Part II: Using the Cloud

How to get started with AWS, install Apache, create an EFS volume and much more.

By Petros Koutoupis

The cloud is here to stay, regardless of how you access data day to day. Whether you are uploading and sharing new photos with friends in your social-media account or updating documents and spreadsheets alongside your peers in your office or school, chances are you're connecting to the cloud in some form or another.

In the first part of this series, I explored what makes up the cloud and how it functions when all of its separate moving pieces come together. In this article, building from Part I's foundations, I cover using the cloud through some actual examples.

Getting Started with AWS

For the purposes of this article, I'm focusing on a few of the top offerings provided by Amazon Web Services (AWS). Please know that I hold no affiliation to or with Amazon, nor am I stating that Amazon offerings exceed

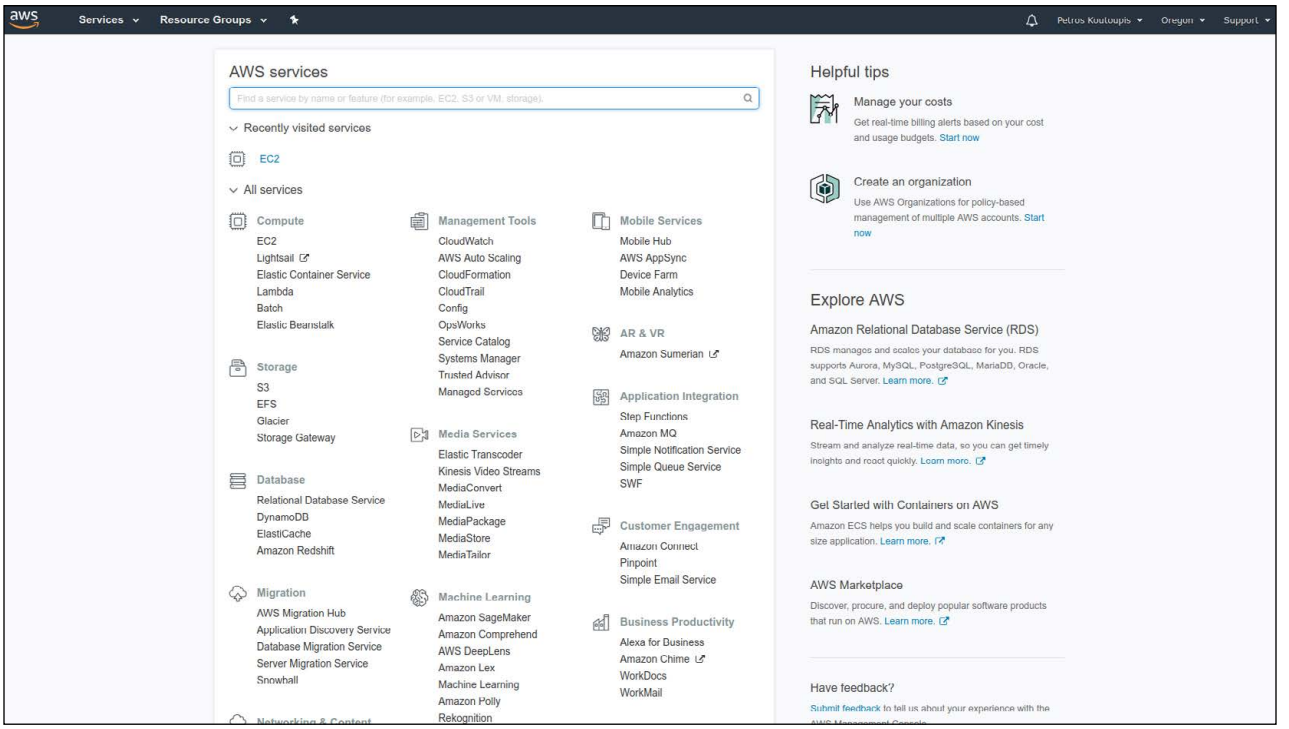


Figure 1. The AWS Main Dashboard of Services and Resources

those of its competitors.

If you haven't already, be sure to register an account. But before you do, understand that charges *may* apply. Amazon, may provide a *free* tier of offerings for a limited time, typically a year, to newly registered users. In most cases, the limitations to these offerings are far less than ideal for modern use cases. It is a pay-as-you go model, and you'll be charged only as long as the instance or service continues to be active.

As soon as you are registered and logged in from within your web browser, you'll be greeted by a fairly straightforward dashboard.

Compute

At first, companies leveraging cloud compute applied a straight copy-and-paste of their very own data centers for deploying standard web/application/database servers.

The model was the same. There is nothing wrong with that approach. The transition for most converting from on-premises to the cloud would have been somewhat seamless—at least from the perspective of the user accessing those resources. The only real difference being that it was just in a different data center and without the headache of maintaining the infrastructure supporting it.

In the world of AWS, virtual compute servers are managed under the Elastic Cloud Computing (EC2) stack, from whole virtual instances to containers and more. Let's begin an example EC2 experiment by navigating to the EC2 dashboard.

After you select "Launch instance", you'll be presented with a screen where you can select from a preconfigured image of various Linux and Windows installations. For this example, let's choose Ubuntu Server 16.04.

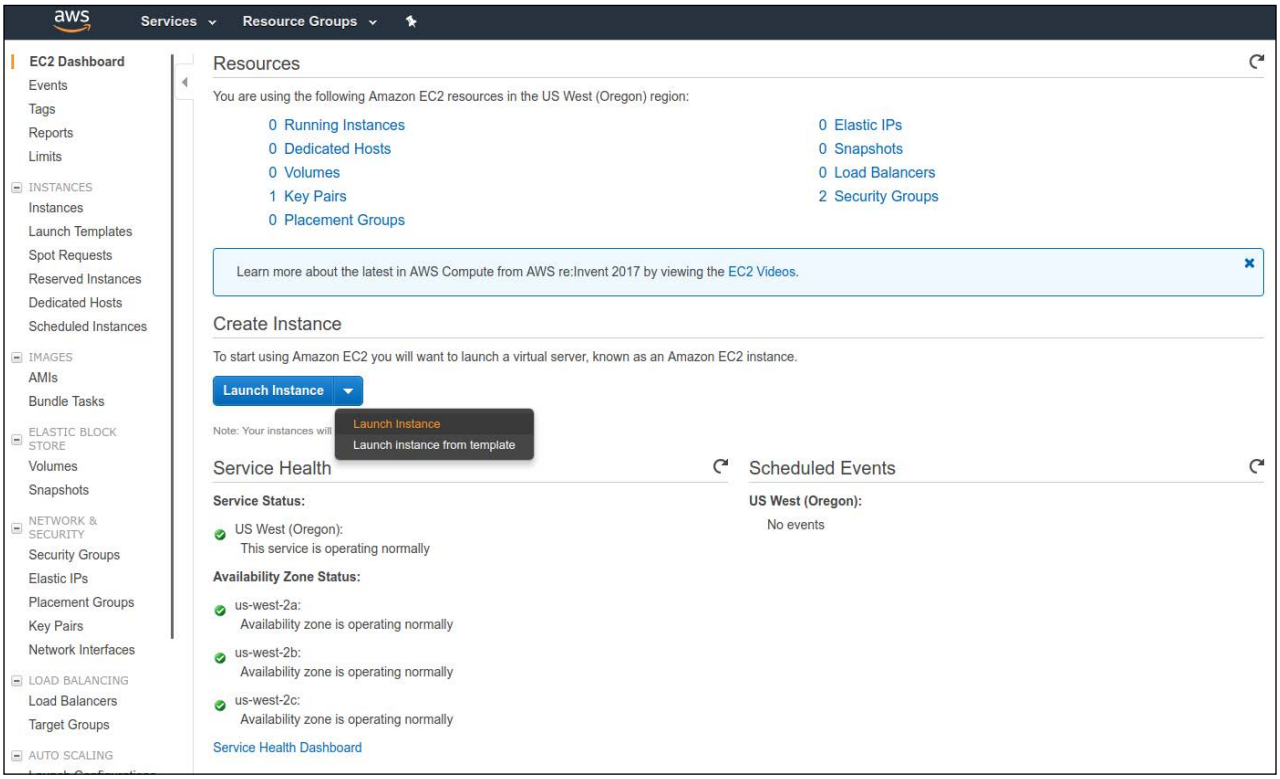


Figure 2. The Elastic Cloud Computing Dashboard

The following screen provides the option of defining the number of virtual cores or processors required, the total amount of memory and the network speed. Remember, the more cores and memory defined in this instance, the more costly.

From this point, you can decide whether to customize this particular instance further with additional local storage (using the Elastic Block Storage or EBS framework), security policies, deployment region (data center) and so on. For this example, let's select "Review and Launch".

Confirm that all your settings are correct, and also note the instance's Security Group name (you'll revisit this shortly). Press Launch.

This next step is very important. You'll be prompted to either associate an existing public/private key pair to access this virtual machine or create a new one. When created, be sure to download and store a copy of the generated PEM file somewhere safe. You will need this later.

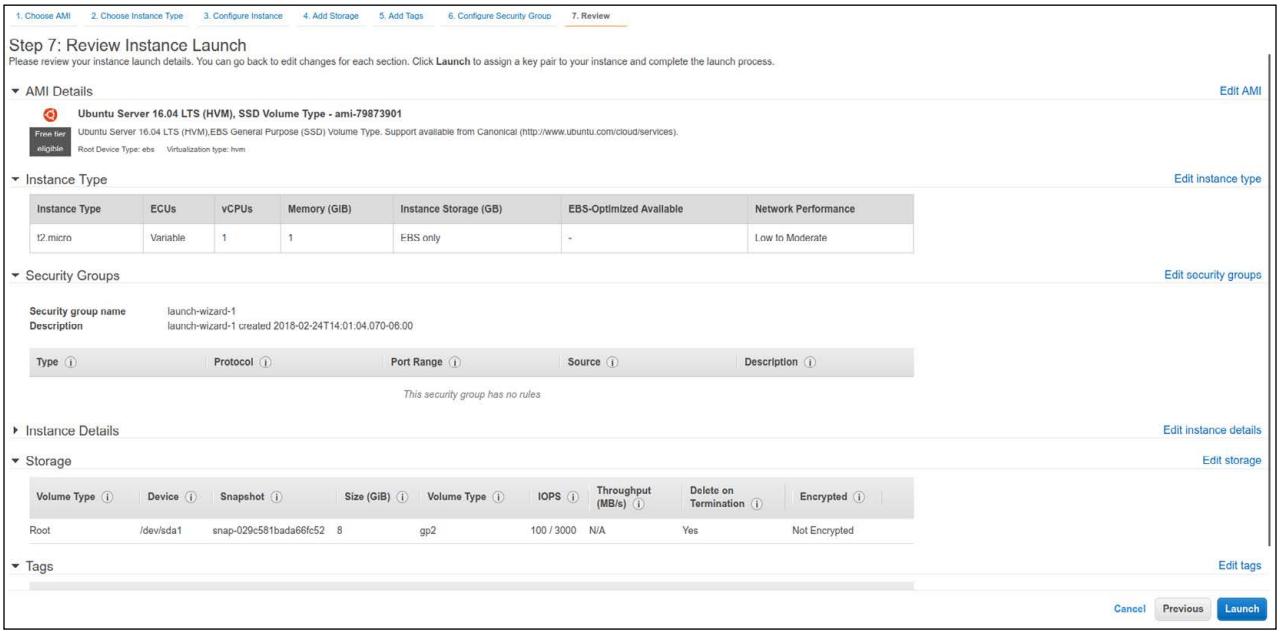


Figure 3. The EC2 Instance Review Screen

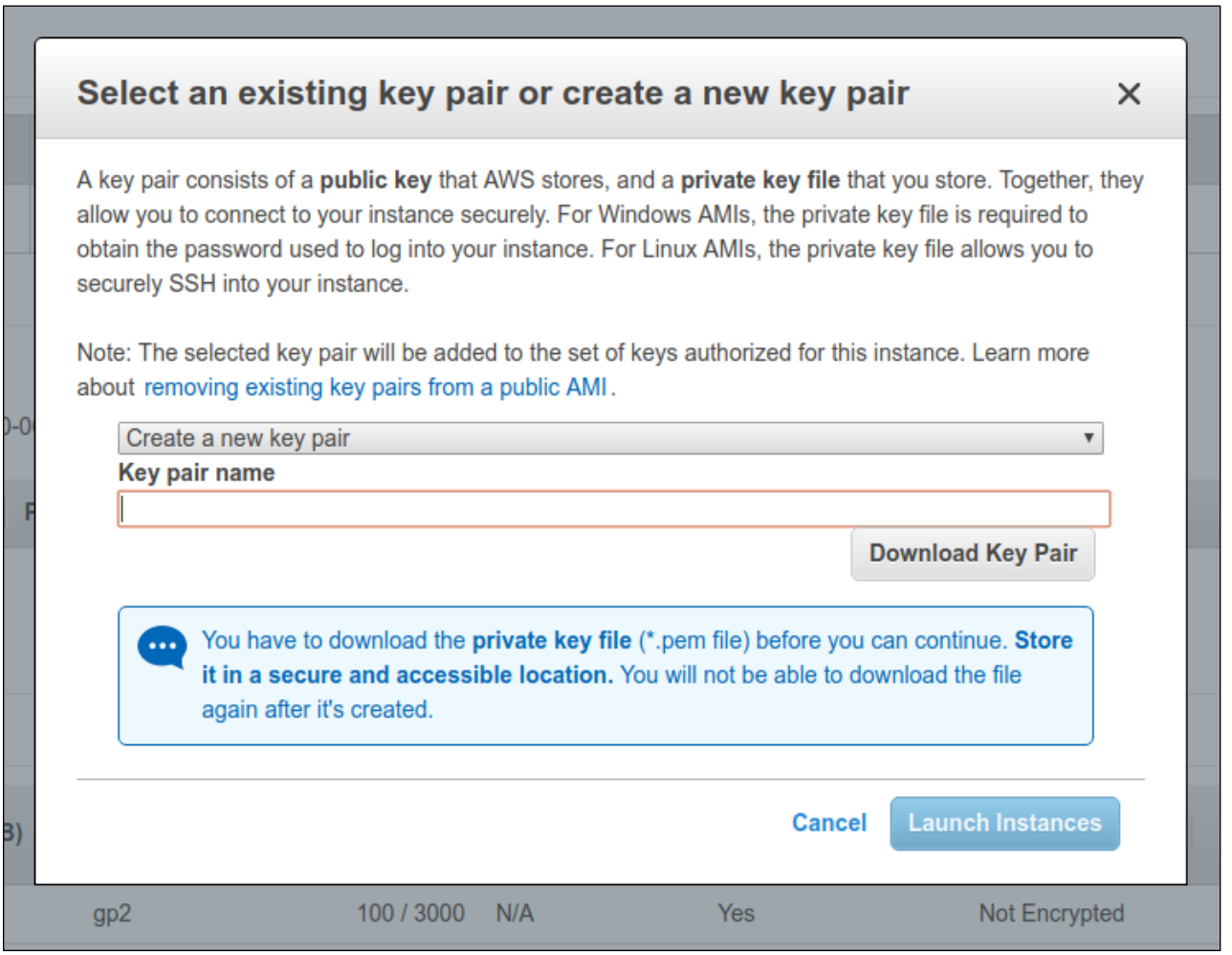


Figure 4. The EC2 Public/Private Key Pair Creation Window

Launch your instance and go to the Instances Dashboard. Your virtual machine is now up and running, and you should be able to log in to it using that PEM file.

Now, locate that PEM file on your local machine and change its permissions to only read-access for the owner:

```
$ chmod 400 Linux-Journal.pem
```

Using SSH and referencing that same PEM file, log in to that instance by its public IP

address and as the user ubuntu:

```
$ ssh -i Linux-Journal.pem ubuntu@35.165.122.94
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1049-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

To run a command as administrator (user "root"), use
"sudo <command>". See "man sudo_root" for details.

ubuntu@ip-172-31-21-167:~$
```

Voilà! You did it. You deployed a virtual machine on AWS and are now connected to it. From this point on, you can treat it like any other server. For instance, now you can install Apache:

```
$ sudo apt install apache2
```

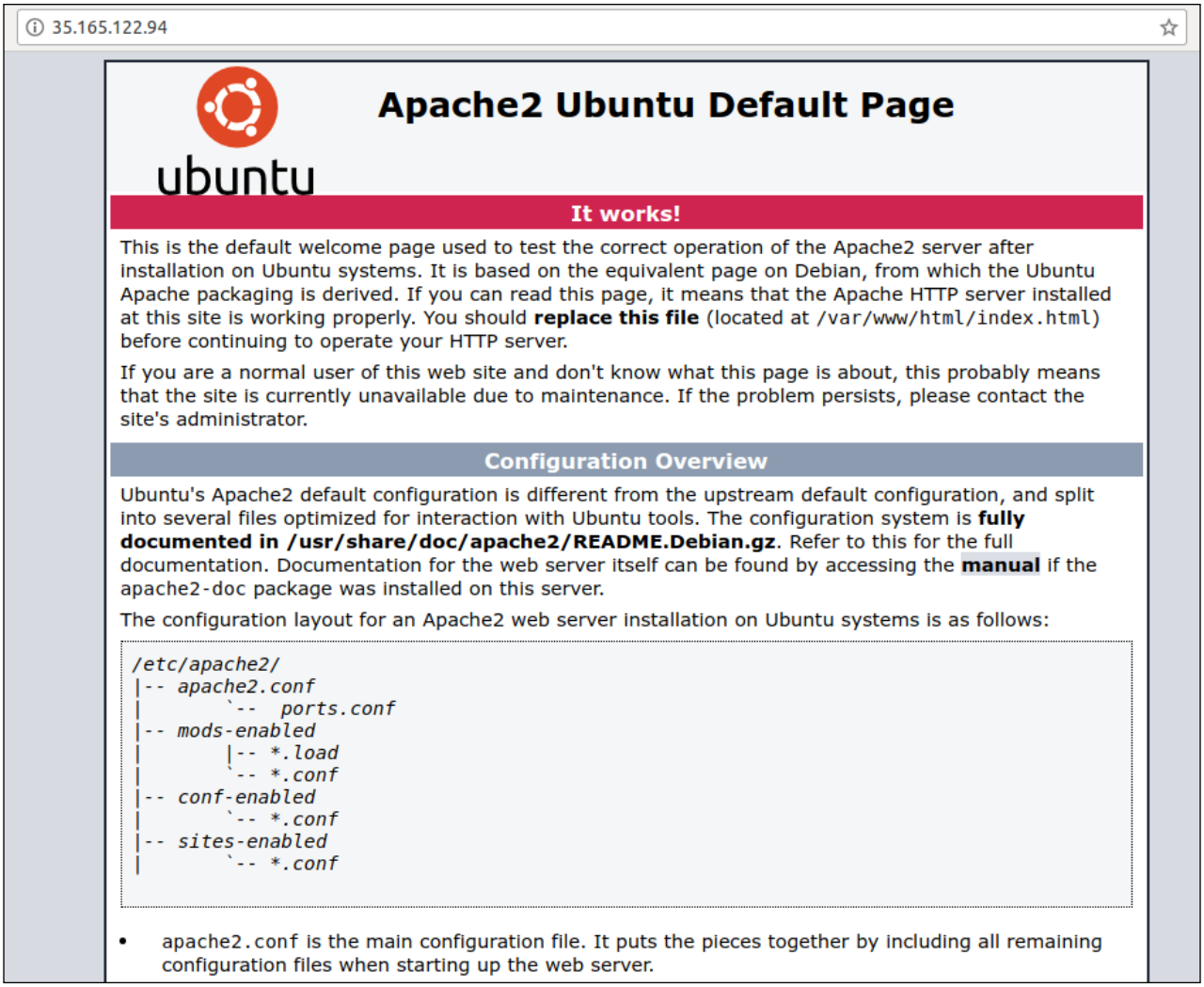


Figure 5. Accessing Port 80 on the Virtual Machine Instance

Now that Apache is installed and running, you need to modify that same Security Group used by this instance to allow access to port 80 over the same public IP address. To do that, navigate to the instance dashboard menu option “Security Groups” located under the Network and Security category. Find your Security Group in the list, select it, and at the bottom of the selection table, click the Inbound tab. Next, let’s add a rule to allow incoming access on port 80 (HTTP). After all is done, open a web browser, and paste in the machine’s public IP address.

Now, what if you don't need a full-blown operating system and care about running only one or two applications instead? Taking from the previous Apache example, maybe you just need a web server. This is where Elastic Container Service (ECS) comes into the picture. ECS builds on top of EC2 in that you are required to spin up an EC2 instance to host the uploaded Docker instances from your Amazon-hosted Docker image registry. Docker is the most popular of container technologies. It is a userspace and lightweight virtualization platform that utilizes Linux Control Groups (cgroups) and namespaces to manage resource isolation.

As users continue to entrench themselves in the cloud, they'll start to rely on the provider's load balancers, DNS managers and more. The model discussed earlier evolves to accommodate those requirements and simplify its management. There even may be a desire to plug in to the provider's machine learning or analytics platforms and more.

Cloud Native Applications

The traditional data center starts to disappear when going serverless. It's a very elastic model—one where services spin up and down, always in response to the demand. Much less time is spent logged in to a server, and instead, DevOps engineers spend most of their time writing API code to connect all the dots in their products. The fact that Linux runs everything underneath is sort of lost in the equation. There is less of a need to know or even care.

As one would expect, container technologies have helped accelerate cloud adoption. Think about it. You have these persistent containerized application images that within seconds are spun up or down as needed and balanced across multiple nodes or data-center locations to achieve the best in quality of service (QoS). But, what if you don't want to be bothered by all the container stuff and instead care only about your application along with its functions? This is where Amazon's Lambda helps. As I mentioned in Part I, with Lambda, you don't need to be concerned with the container. Just upload your event-driven application code (written in Node.js, Python, Java or C#) and respond to events, such as website clicks, within milliseconds. Lambda scales automatically to support the exact needs of your application.

As for the types of events (labeled an event source) on which to trigger your application, or code handlers, Amazon has made it so you can trigger on website visits or clicks, a REST HTTP request to its API gateway, a sensor reading on your Internet-of-Things (IoT) device, or even an upload of a photograph to an S3 bucket. This API gateway forms the bridge that connects all parts of AWS Lambda. For example, a developer can write a handler to trigger on HTTPS request events.

Let's say you need to enable a level of granularity to your code. Lambda accommodates this by allowing developers to write modular handlers. For instance, you can write one handler to trigger for each API method, and each handler can be invoked, updated and altered independent of the others.

Lambda allows developers to combine all required dependencies (that is, libraries, native binaries or even external web services) to your function into a single package, giving a handler the freedom to reach out to any of those dependencies as it needs them.

Now, how does this compare to an Amazon AWS EC2 instance? Well, the short answer is that it's a lot more simplified, and by simplified, I mean there is zero to no overhead on configuring or maintaining your operating environment. If you need more out of your environment that requires access to a full-blown operating system or container, you'll spin up an EC2 virtual instance. If you need only to host a function or special-purpose application, that is where Lambda becomes the better choice. With Lambda, there isn't much to customize—and sometimes, less is good.

Storage

If you recall from Part I, in the cloud, multiple local volumes are pooled together across one or more sites and into a larger set of storage pools. When volumes are requested in block, filesystem or object format, they are carved out of those larger pools. Let's look at some of these AWS offerings.

Elastic File System

The Amazon Elastic File System (EFS) provides users with a simplified, highly available and very scalable file storage for use with EC2 instances in the cloud. Just like with

anything else offered by Amazon, the storage capacity of an EFS volume is elastic in that it can grow and shrink dynamically to meet your application's needs. When mounted to your virtual machine, an EFS volume provides the traditional filesystem interfaces and filesystem access semantics.

To create a new EFS volume, select EFS from the main AWS dashboard and then click on the "Create file system" button. You'll be directed to a screen where you need to configure your new filesystem. To simplify things, let's select the same Security Group used by the EC2 instance from the previous example.

Next, give your filesystem a name and verify its options.

Once confirmed and created, you'll see a summary that looks similar to Figure 8. *Note that the filesystem will not be ready to use from a particular location until its "Life cycle state" reads that it's available.*

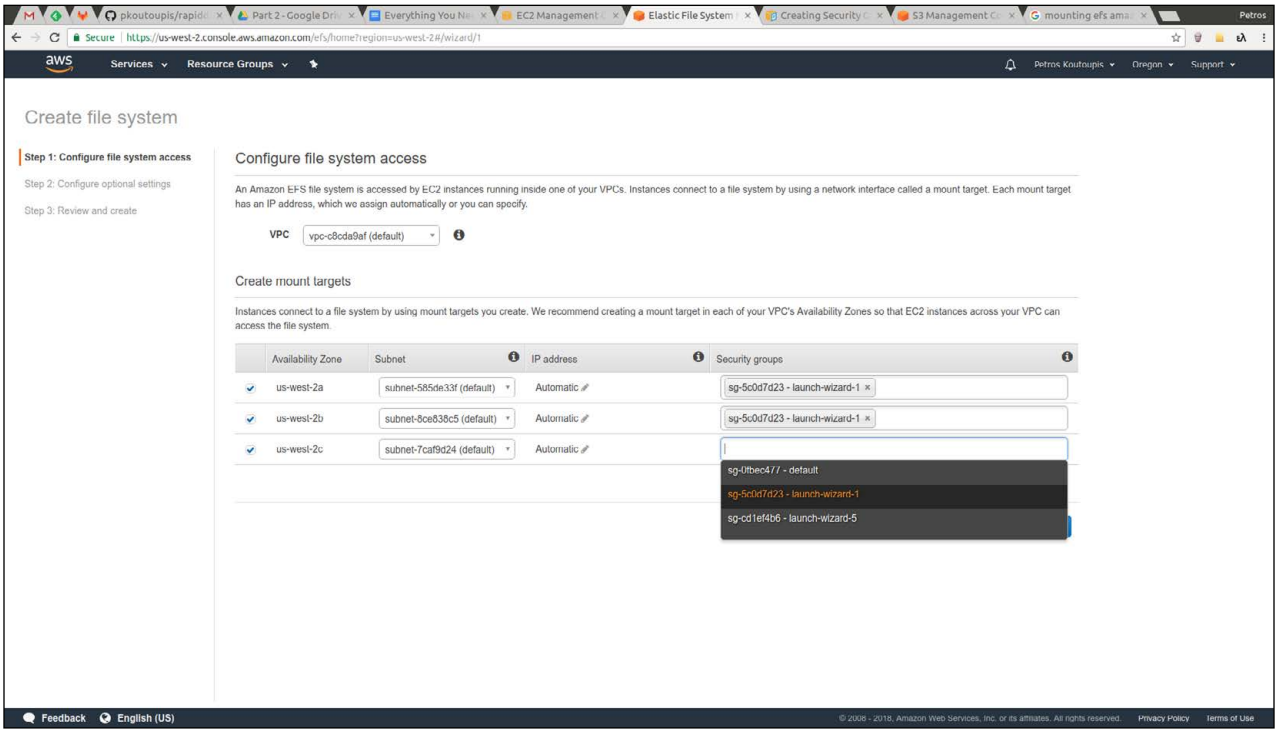


Figure 6. Creating a New Filesystem and Assigning It to a Security Group

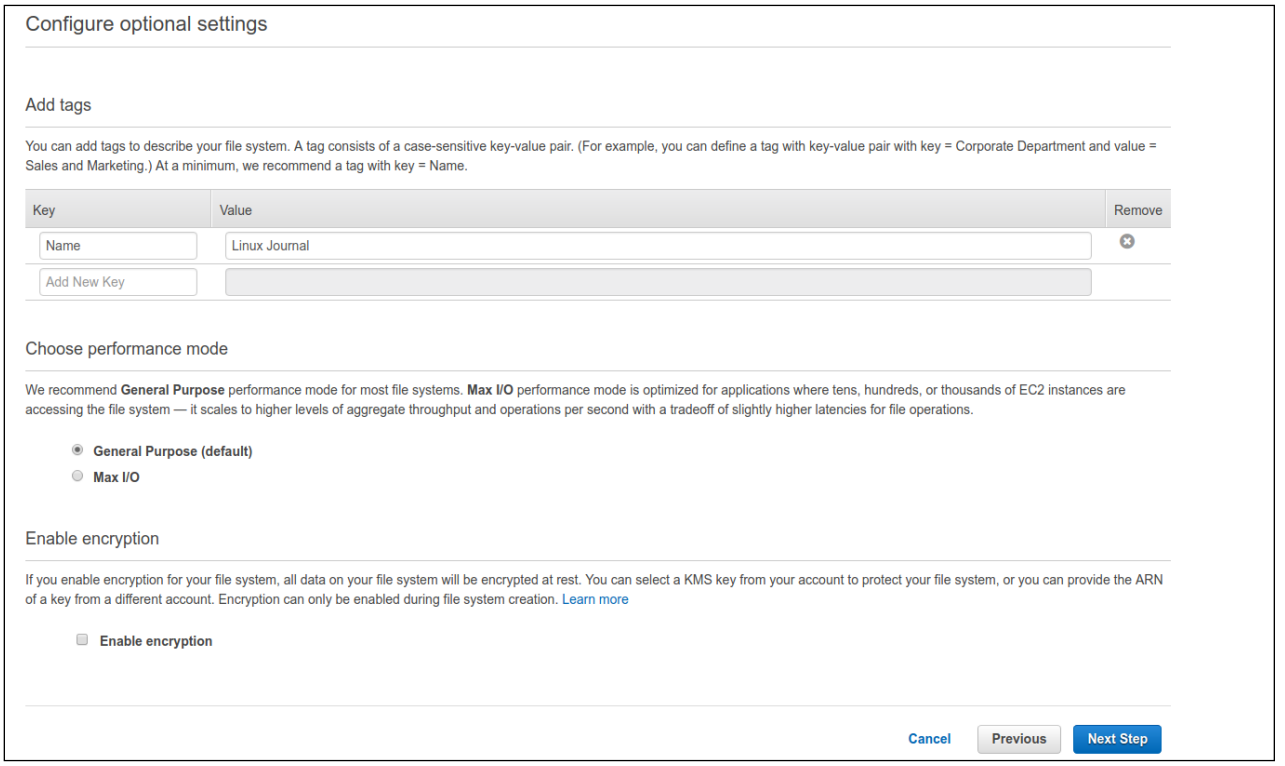


Figure 7. Setting a Name and Configuring Options

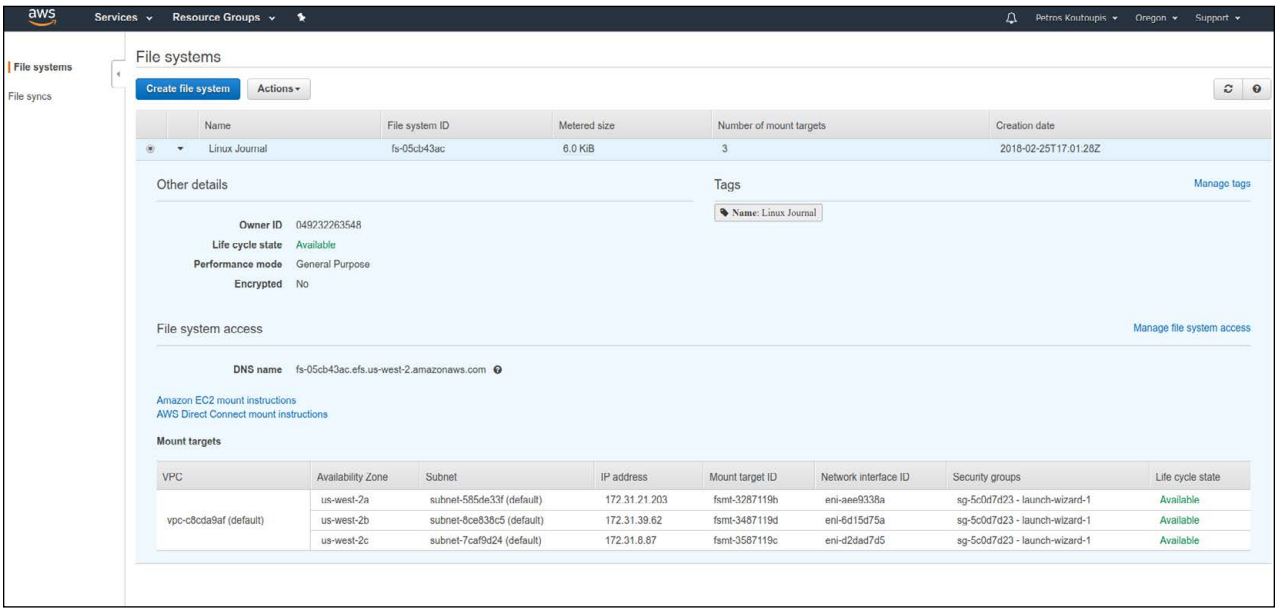


Figure 8. Filesystem Summary

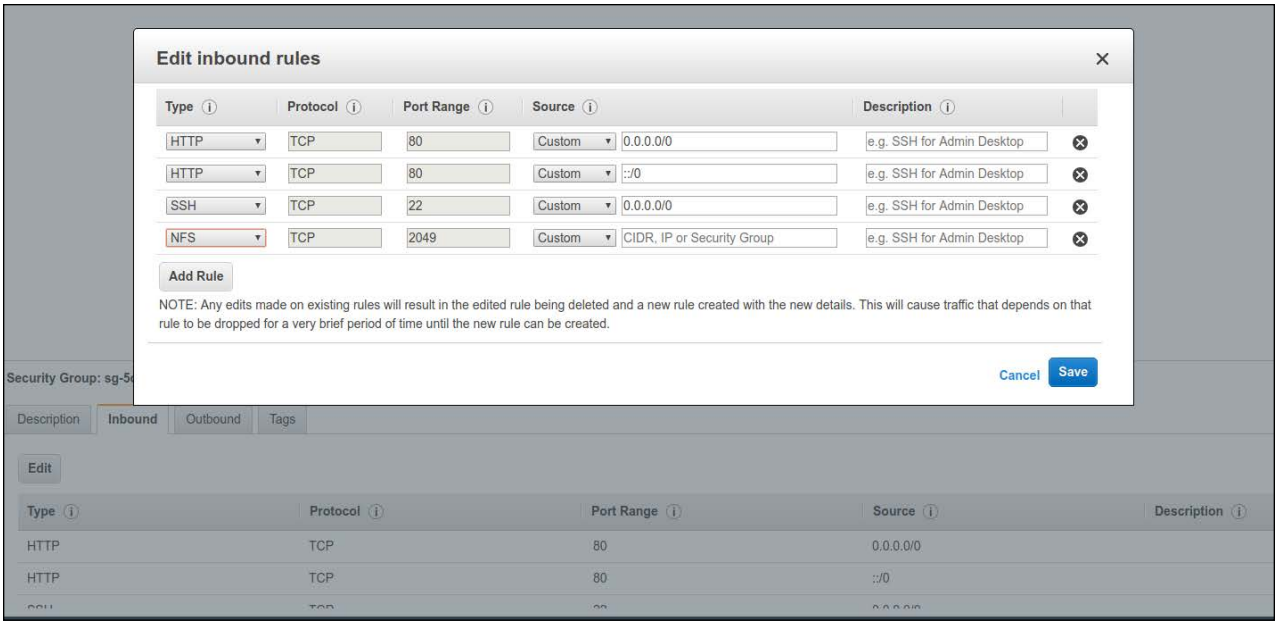


Figure 9. Adding NFS to Your Security Group

Using the same EC2 instance, install the NFS packages from the distribution's package repository:

```
$ sudo apt-get install nfs-common
```

Before proceeding, you need to add NFS to your Security Group. This applies to both inbound and outbound traffic.

From the console of your virtual machine instance, mount the NFS filesystem:

```
$ sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=
↪1048576,hard,timeo=600,retrans=2 fs-05cb43ac.efs.us
↪-west-2.amazonaws.com:/ /mnt
```

Verify that the volume is mounted (note: in the last line below under Filesystem, the name is truncated so the output would all fit; that entry

should read `fs-05cb43ac.efs.us-west-2.amazonaws.com`):

```
$ df
Filesystem       1K-blocks    Used        Available Use% Mounted on
udev                499312       0           499312   0% /dev
tmpfs               101456    3036            98420   3% /run
/dev/xvda1         8065444 1266640          6782420  16% /
tmpfs               507268       0           507268   0% /dev/shm
tmpfs                 5120       0             5120   0% /run/lock
tmpfs               507268       0           507268   0% /sys/fs/cgroup
tmpfs               101456       0           101456   0% /run/user/1000
fs-05...  9007199254739968       0 9007199254739968   0% /home/ubuntu/efs
```

And there you have it! An EFS volume is now connected to your EC2 instance, and you're able to read from and write to it like any other filesystem.

S3

The Simple Storage Service (S3) supplies applications a front end to store and retrieve millions if not billions of data content from buckets at massive scale. Traditional filesystems aren't capable of cataloging such a large listing of data and, in turn, serve that data within a reasonable amount of time. Object storage solves that. It isn't a filesystem but instead a high-level cataloging system. When you `PUT` a file into your bucket, you do so with a tag ID. Then when you `GET` that file from the same bucket, you request it by the same tag ID.

At a high level, you don't see how the data is stored or managed, and technically, you shouldn't care. Each object storage solution has its own methods by which they save object content, and sometimes it's as simple as saving each individual object as a file under a nested directory structure and on top of a traditional filesystem but then not making this visible to the end user or application. The application will access this data or bucket by using a REST API and communicating to the bucket's endpoint over HTTP.

Amazon's S3 API has become somewhat standard, and other object storage solutions

maintain compatibility with this API alongside their very own. The motivation hopefully is to migrate those same AWS S3 users over to that other object storage platform.

Glacier

Amazon's Glacier offers its users an extremely secure, durable and low-cost (as low as $0.004 per Gigabyte *per* month) alternative to data archiving and long-term backup hosted on their cloud. It is a way for most companies to ditch their age-old and very limited local file servers and tape drives. How often has your company found itself struggling either to reduce the consumption or add to the capacity of their local archive system? It happens a lot more often than you would think. Glacier alleviates all of that headache and concern.

Private Clouds and OpenStack

When I stepped into the data storage industry more than 15 years ago, it was a very different time with very different customers. This was the era of Storage Area Networks (SAN) and Network Attached Storage (NAS). The cloud did not exist. Our customers were predominantly mid-to-large enterprise users. These companies were single vendor shops. If you were an HP shop, you bought HP. EMC, you bought EMC. NetApp, you bought NetApp, and so on. From the customer's point of view, there was a level of comfort in knowing that you needed to interact only with a single vendor for purchasing, management and support.

About a decade ago, this mentality started to change. Exciting new technologies cropped up: virtualization, flash and software-defined everything. These same technologies eventually would enable the then future cloud. Customers wanted all of those neat features. However, the problem was that the large vendors didn't offer them—at least not in the beginning. As a result, customers began to stray away from the single-vendor approach. What that meant was multiple vendors and multiple management interfaces. Now there were too many moving parts, each needing a different level of expertise in the data center.

There was a light at the end of the tunnel, however: enter OpenStack. OpenStack glued all of those moving components together (that is, storage, compute and

networking). The project is an Apache-licensed open-source framework designed to build and manage both public and private clouds. Its interrelated components control hardware pools of processing, storage and networking resources, all managed through a web-based dashboard, a set of command-line utilities or through an API. Even though OpenStack exports and publishes its own unique API, the project does strive to maintain compatibility with competing APIs, which include Amazon's EC2 and S3.

OpenStack's primary goal was to create a single and universal framework to deploy and manage various technologies in the data center dynamically. Originally started in 2010, an effort jointly launched by Rackspace Hosting and NASA, the project has since grown exponentially and attracted a wide number of supporters and users. The secret to OpenStack's success is convergence. By providing a single and standardized framework, it brought order back to an almost unmanageable ecosystem. Its recent popularity should come as no surprise.

Modern company-backed Linux distributions—which include Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES) and Ubuntu Server—have gone to great lengths not only to simplify the technology but also to support it. And although I don't spend much more time discussing the technology here, as it deserves its own dedicated write-up, let me state that if you desire to run your very own private cloud deployment, OpenStack is definitely the way to do it.

Summary

In closing, why use the cloud? If the following list doesn't convince you, I don't know what will:

- Flexibility.
- Speed to develop and deploy.
- Fault tolerance (and an efficient disaster recovery plan).
- Access to the latest and greatest software.

- Reduced operational costs.
- Increased cross-team collaboration.
- Remote access (work from anywhere).
- Security.
- And more...

The availability of a public cloud platform that offers best-of-breed performance, an incredibly wide and ever-growing selection of services, and global coverage is a powerful and, it could be said, necessary, addition to your IT strategy. With larger organizations, you may have to plan on building your own internal cloud for certain types of workloads—and the availability of OpenStack grants the best of both worlds in terms of scalability, ownership and utilization of existing data-center assets.

There is a lot I didn't cover here due to space limitations—for instance, AWS and other cloud providers add very advanced security options to limit access to various resources and do so by creating user and group policies. You also are able to expand on the rules of the security groups used by your EC2 instances and further restrict certain ports to specific IP addresses or address ranges. If and when you do decide to leverage the cloud for your computing needs, be sure to invest enough time to ensure that those settings are configured properly. ■

Petros Koutoupis, *LJ* Contributing Editor, is currently a senior platform architect at IBM for its Cloud Object Storage division (formerly Cleversafe). He is also the creator and maintainer of the Rapid Disk Project. Petros has worked in the data storage industry for well over a decade and has helped pioneer the many technologies unleashed in the wild today.

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.

The Agony and the Ecstasy of Cloud Billing

Cloud billing is inherently complex; it's not just you.

By Corey Quinn

Back in the mists of antiquity when I started reading *Linux Journal*, figuring out what an infrastructure was going to cost was (although still obnoxious in some ways) straightforward. You'd sign leases with colocation providers, buy hardware that you'd depreciate on a schedule and strike a deal in blood with a bandwidth provider, and you were more or less set until something significant happened to your scale.

In today's brave new cloud world, all of that goes out the window. The public cloud providers give with one hand ("Have a full copy of any environment you want, paid by the hour!"), while taking with the other ("A single Linux instance will cost you $X per hour, $Y per GB transferred per month, and $Z for the attached storage; we simplify this pricing into what we like to call 'We Make It Up As We Go Along'").

In my day job, I'm a consultant who focuses purely on analyzing and reducing the Amazon Web Services (AWS) bill. As a result, I've seen a lot of environments doing different things: cloud-native shops spinning things up without governance, large enterprises transitioning into the public cloud with legacy applications that don't exactly support that model without some serious tweaking, and cloud migration projects that somehow lost their way severely enough that they were declared acceptable as they were, and the "multi-cloud" label was slapped on to them. Throughout all of this, some themes definitely have emerged that I find that people

don't intuitively grasp at first. To wit:

- It's relatively straightforward to do the basic arithmetic to figure out what a current data center would cost to put into the cloud as is—generally it's a lot! If you do a 1:1 mapping of your existing data center into the cloudy equivalents, it invariably will cost more; that's a given. The real cost savings arise when you start to take advantage of cloud capabilities—your web server farm doesn't need to have 50 instances at all times. If that's your burst load, maybe you can scale that in when traffic is low to five instances or so? Only once you fall into a pattern (and your applications support it!) of paying only for what you need when you need it do the cost savings of cloud become apparent.

- One of the most misunderstood aspects of Cloud Economics is the proper calculation of Total Cost of Ownership, or TCO. If you want to do a break-even analysis on whether it makes sense to build out a storage system instead of using S3, you've got to include a lot more than just a pile of disks. You've got to factor in disaster recovery equipment and location, software to handle replication of data, staff to run the data center/replace drives, the bandwidth to get to the storage from where it's needed, the capacity planning for future growth—and the opportunity cost of building that out instead of focusing on product features.

- It's easy to get lost in the byzantine world of cloud billing dimensions and lose sight of the fact that you've got staffing expenses. I've yet to see a company with more than five employees wherein the cloud expense wasn't dwarfed by payroll. Unlike the toy projects some of us do as labors of love, engineering time costs a lot of money. Retraining existing staff to embrace a cloud future takes time, and not everyone takes to this new paradigm quickly.

- Accounting is going to have to weigh in on this, and if you're not prepared for that conversation, it's likely to be unpleasant—you're going from an old world where you could plan your computing expenses a few years out and be pretty close to accurate. Cloud replaces that with a host of variables to account for, including variable costs depending upon load, amortization of Reserved Instances, provider

price cuts and a complete lack of transparency with regard to where the money is actually going (Dev or Prod? Which product? Which team spun that up? An engineer left the company six months ago, but 500TB of that person's data is still sitting there and so on).

The worst part is that all of this isn't apparent to newcomers to cloud billing, so when you trip over these edge cases, it's natural to feel as if the problem is somehow your fault. I do this for a living, and I was stymied trying to figure out what data transfer was likely to cost in AWS. I started drawing out how it's billed to customers, and ultimately came up with the "AWS Data Transfer Costs" diagram shown in Figure 1.

If you can memorize those figures, you're better at this than I am by a landslide! It isn't straightforward, it's not simple, and it's certainly not your fault if you don't somehow intrinsically know these things.

That said, help is at hand. AWS billing is getting much more understandable, with the

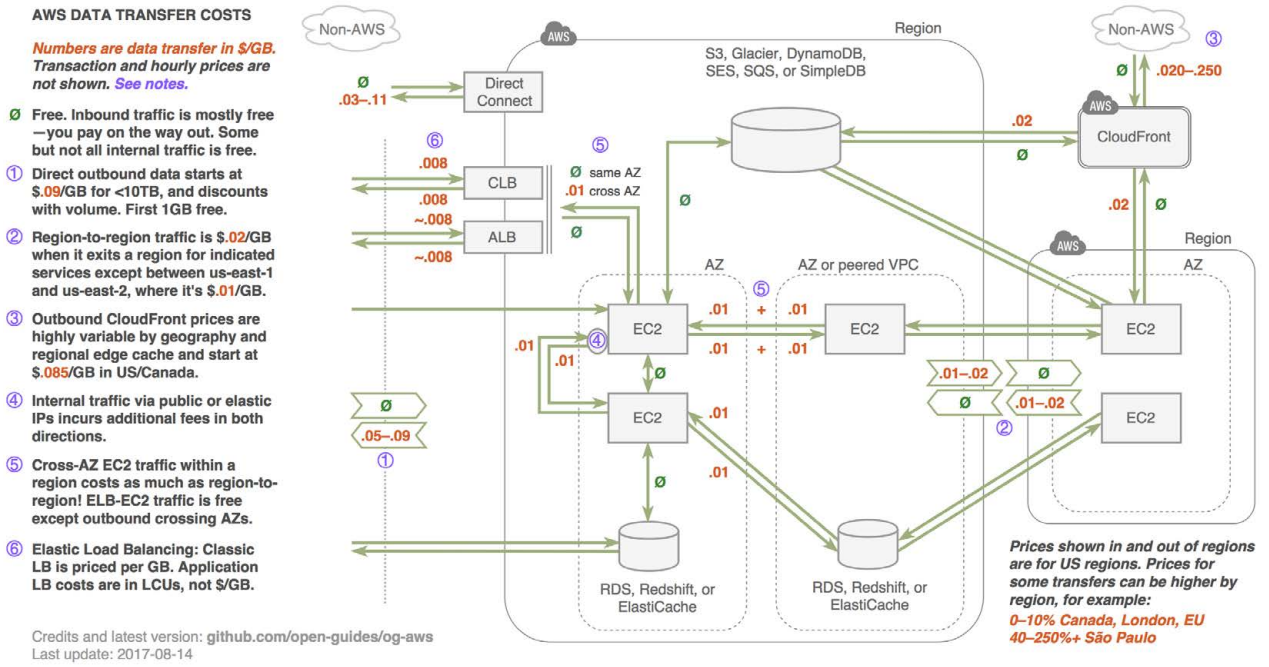


Figure 1. A convoluted mapping of how AWS data transfer is priced out.

advent of such things as free Reserved Instance recommendations, the release of the Cost Explorer API and the rise of serverless technologies. For their part, Google's GCP and Microsoft's Azure learned from the early billing stumbles of AWS, and as a result, both have much more understandable cost structures. Additionally, there are a host of cost visibility Platform-as-a-Service offerings out there; they all do more or less the same things as one another, but they're great for ad-hoc queries around your bill. If you'd rather build something you can control yourself, you can shove your billing information from all providers into an SQL database and run something like QuickSight or Tableau on top of it to aide visualization, as many shops do today.

In return for this ridiculous pile of complexity, you get something rather special—the ability to spin up resources on-demand, for as little time as you need them, and pay only for the things that you use. It's incredible as a learning resource alone—imagine how much simpler it would have been in the late 1990s to receive a working Linux VM instead of having to struggle with Slackware's installation for the better part of a week. The cloud takes away, but it also gives. ■

Corey Quinn is currently a Cloud Economist at the Quinn Advisory Group, and an advisor to ReactiveOps. Corey has a history as an engineering director, public speaker and cloud architect. He specializes in helping companies address horrifying AWS bills, and curates LastWeekinAWS.com, a weekly newsletter summarizing the latest in AWS news, blogs and tips, sprinkled with snark.

Vendor Lock-in: Now in the Cloud!

Vendor lock-in has moved from corporate infrastructure into the cloud, only this time many are all too happy to embrace it.

By Kyle Rankin

I started my professional career as a general-purpose "computer guy" for a small office. I wore many hats in this job including desktop help desk, server administrator, webmaster, network administrator, security engineer, and in general, if it plugged in, I was probably responsible for it. This was back in the era where, although Linux was making some inroads in the web server market, Microsoft absolutely dominated both the desktop and the corporate server markets. It was expected back then that offices of any size from ten to a thousand people would not only be running Windows on the desktop, but also that the back office would be running Windows servers.

Those Microsoft services weren't necessarily better than the alternatives, but they won out anyway because of vendor lock-in. The domination of the desktop market meant that Microsoft could develop proprietary protocols like SMB (file sharing) and MAPI (email client syncing), and add them to Windows and Microsoft Office. Once SMB was baked in to Windows, it became easy for the boss of a small office to turn his or her desktop into the office file server without adding any extra software. As the company grew, that desktop was replaced by a standalone Windows server, and when you found out that your ISP (which you were using for corporate email up to this point) didn't support the shared calendar feature you saw in Outlook, you found out that Exchange and its MAPI protocol did.

How Ease Leads to Lock-in

People pick single-vendor solutions often not because they are better, but because they are easier. There's a built-in assumption that if you buy two products from the same vendor, they will work together better than if one of the products were from a different vendor. Yet when everything is from the same vendor, that vendor can develop proprietary protocols that have special features just for their products that competitors either can't use or have to pay a steep licensing fee to use. Beyond Microsoft, this can be found in routing protocols that work only on Cisco equipment or flavors of SQL that work only on Oracle databases. In all these cases, there is a built-in incentive to enhance their proprietary protocols and ignore bugs, performance issues or compatibility problems with the open standards they support.

It's this embracing of proprietary protocols and standards that leads to vendor lock-

in. When two different vendors interoperate, they generally pick open standards and public protocols. If you don't like one of the vendors, you can replace it with a different vendor that supports the same standards and protocols. With vendor lock-in, you end up using features that exist only in proprietary protocols, and it becomes much harder to replace them with another vendor. No matter how good the competitor's solution is, the time and effort to switch away from proprietary protocols becomes too great.

When Linux entered the server scene, its stability and free (as in cost) nature started nibbling away at the UNIX web, application and database servers in data centers. With tools like Samba, it also started quietly replacing unstable Windows file servers. Linux's strong support of open standards highlighted some of the problems with proprietary protocols, and it's largely because of it (and UNIX before it) that the internet we use today relies on open protocols like TCP/IP, DNS and HTTP. There was a point when that wasn't a sure thing.

The Cloud Is Just Another Vendor

I bring all of this up because we are repeating the same exact mistakes from the past, only this time in the cloud. I chalk it up to a combination of younger engineers who didn't live through the last wave of closed standards, Linux geeks who are still so focused on ancient enemies like Microsoft that they never noticed the giants who took their place, and senior engineers who know better but are happy to trade vendor lock-in for ease, because after decades in the industry, they are starting to get tired. When you consider just how much of the modern internet runs on one of only a few cloud providers (and let's be honest, most of it runs on only one of them), this complacence is creating a monoculture that puts far too much control in one organization's hands.

Monocultures can work great up until the point that there's a problem. Farming monocultures, such as cotton in the South and potatoes in Ireland, were prosperous up until there was a boll weevil and a blight. With the cloud, you have your entire business on one infrastructure, and there's this assumption that the infrastructure as a whole never would have a problem—at least until it does. At that point, you find

out not only that the cloud provider had a fault, but that its fault was actually *your fault* for putting all of your eggs in one basket. Maybe the basket was a single server, maybe it was a single availability zone, maybe it was S3 in a single region. The point is, no matter how big the outage, it's your fault, because if you back up one step, you will always find that when all your eggs are in a single basket, fault tolerance is on you—at least until that basket is the cloud provider itself.

At this point, most people find solace in two things. First, if their cloud vendor were to have a major systemic problem, it wouldn't just be down—a huge swath of the internet would be down. Their cloud vendor is too big to fail, you see, so why plan for an event that probably will never happen? Second, they already are all-in with their cloud provider and are using too many proprietary services. It would be too difficult, too time-consuming and too expensive to develop an architecture that could run on multiple cloud platforms. It also would mean throwing away some of the really easy-to-use new features their platform just announced this year that may not be as great as competitors but that integrate so easily into their existing services.

The Internet Is Just Someone Else's Cloud

The bigger risk—the reason I'm writing this article—isn't your company's services, which although I'm sure are fantastic, let's be honest, we probably could live without for a few hours or even days of downtime. The bigger risk is what happens to open standards on the internet and our ability to use the internet freely if almost all of it is run by proprietary services on a handful of cloud vendors. We Linux users are so proud of how much Linux has dominated cloud infrastructure servers, yet many engineers who truly use cloud infrastructure to its fullest never touch Linux or anything related to it. They simply interact with layers of abstraction on top of a proprietary API—if you dig down far enough, you'll find some Linux software, but that hardly matters to many people at that point.

In fact, plenty of people see only having to interact with their cloud vendor's API as a feature, and if the current trends continue in this direction, cloud infrastructure will turn into the ultimate proprietary OS where it honestly doesn't matter whether there are any open standards or software under the hood, as that's all hidden from you

anyway. Individual containers or server instances take the place of processes, cloud services take the place of applications, and you as the engineer interact with it all using either their GUI, or a special language defined and controlled by your vendor.

If we get to that point, there's nothing stopping vendors from replacing any open standards behind the scenes with their own proprietary ones perfectly suited for their platforms. The question is whether those standards will start to supplant the open standards that exist outside their networks. Second- and third-place competitors will try to add interoperability features to get customers to move over, but will have a harder and harder time as these new protocols become more closed. With less competition, the winning vendor will have less motivation to innovate and more motivation to increase lock-in. Of course, you won't see those changes anyway, so there's nothing to worry about. After all, it would be too expensive for you to switch by then anyway.

I'm not writing this to say you shouldn't use the cloud. Cloud infrastructure has helped Linux and some open standards grow in a number of ways up to this point, and it has spawned a great many innovations from the speed at which we can create server infrastructure, its overall resiliency and countless open-source tools that help manage the overall ephemeral nature of its services. What I'm trying to do is remind you that vendor lock-in is actually a bad and dangerous thing. It's important to remember the lessons we learned in past eras of proprietary standards battles between vendors. Those battles threatened Linux, open standards and ultimately your freedom to control your own software. If it happened in the past, it can happen again. So think twice before you go with a single cloud vendor for everything just because it's easy. ■

Kyle Rankin is a Tech Editor and columnist at *Linux Journal* and the Chief Security Officer at Purism. He is the author of *Linux Hardening in Hostile Networks*, *DevOps Troubleshooting*, *The Official Ubuntu Server Book*, *Knoppix Hacks*, *Knoppix Pocket Reference*, *Linux Multimedia Hacks* and *Ubuntu Hacks*, and also a contributor to a number of other O'Reilly books. Rankin speaks frequently on security and open-source software including at BsidesLV, O'Reilly Security Conference, OSCON, SCALE, CactusCon, Linux World Expo and Penguicon. You can follow him at @kylerankin.

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.

OSI's Simon Phipps on Open Source's Past and Future

With an eye on the future, the Open Source Initiative's president sits down and talks with *Linux Journal* about the organization's 20-year history.

By Christine Hall

It would be difficult for anyone who follows Linux and open source to have missed the 20th birthday of open source in early February. This was a dual celebration, actually, noting the passing of 20 years since the term "open source" was first coined and since the formation of the Open Source Initiative (OSI), the organization that decides whether software licenses qualify to wear that label.

The party came six months or so after Facebook was successfully convinced by the likes of the Apache Foundation; WordPress's developer, Automatic; the Free Software Foundation (FSF); and OSI to change the licensing of its popular React project away from the BSD + Patents license, a license that had flown under the radar for a while.

The brouhaha began when Apache developers noticed a term in the license forbidding the suing of Facebook over any patent issues, which was troublesome because it gave special consideration to a single entity, Facebook, which pretty much disqualified it from being an open-source license.

Although the incident worked out well—after some grumblings Facebook relented and changed the license to MIT—the Open Source Initiative fell under some criticism

Simon Phipps delivers the keynote at Kopano Conference 2017 in Arnhem, the Netherlands.

for having approved the BSD + Patents license, with some people suggesting that maybe it was time for OSI to be rolled over into an organization such as the Linux Foundation.

The problem was that OSI had never approved the BSD + Patents.

"BSD was approved as a license, and Facebook decided that they would add the software producer equivalent of a signing statement to it", OSI's president, Simon Phipps, recently explained to *Linux Journal*. He continued:

> They decided they would unilaterally add a patent grant with a defensive clause in it. They found they were able to do that for a while simply because the community accepted it. Over time it became apparent to people that it was actually not an acceptable patent grant, that it unduly favored Facebook and that if it was allowed to grow to scale, it would definitely create an environment where Facebook was unfairly advantaged.

He added that the Facebook incident was actually beneficial for OSI and ended up being a validation of the open-source approval process:

> I think the consequence of that encounter is that more people are now convinced that the whole licensing arrangement that open-source software is under needs to be approved at OSI.
>
> I think prior to that, people felt it was okay for there just to be a license and then for there to be arbitrary additional terms applied. I think that the consensus of the community has moved on from that. I think it would be brave for a future software producer to decide that they can add arbitrary terms unless those arbitrary terms are minimally changing the rights and benefits of the community.

As for the notion that OSI should be folded into a larger organization such as the Linux Foundation?

"When I first joined OSI, which was back in 2009 I think, I shared that view", Phipps said. He continued:

> I felt that OSI had done its job and could be put into an existing organization. I came to believe that wasn't the case, because the core role that OSI plays is actually a specialist role. It's one that needs to be defined and protected. Each of the organizations I could think of where OSI could be hosted would almost certainly not be able to give the role the time and attention it was due. There was a risk there would be a capture of that role by an actor who could not be trusted to conduct it responsibly.
>
> That risk of the license approval role being captured is what persuaded me that I needed to join the OSI board and that I needed to help it to revamp and become a member organization, so that it could protect the license approval role in perpetuity. That's why over the last five to six years, OSI has dramatically changed.

This is Phipps' second go at being president at OSI. He originally served in the position from 2012 until 2015, when he stepped down in preparation for the end of

his term on the organization's board. He returned to the position last year after his replacement, Allison Randal, suddenly stepped down to focus on her pursuit of a PhD.

His return was pretty much universally seen in a positive light. During his first three-year stint, the organization moved toward a membership-based governance structure and started an affiliate membership program for nonprofit charitable organizations, industry associations and academic institutions. This eventually led to an individual membership program and the inclusion of corporate sponsors.

Although OSI is one of the best known open-source organizations, its grassroots approach has helped keep it on the lean side, especially when compared to organizations like the behemoth Linux or Mozilla Foundations. Phipps, for example, collects no salary for performing his presidential duties. Compare that with the Linux Foundation's executive director, Jim Zemlin, whose salary in 2010 was reportedly north of $300,000.

"We're a very small organization actually", Phipps said. He added:

> We have a board of directors of 11 people and we have one paid employee. That means the amount of work we're likely do behind the scenes has historically been quite small, but as time is going forward, we're gradually expanding our reach. We're doing that through working groups and we're doing that through bringing together affiliates for particular projects.

While the public perception might be that OSI's role is merely the approval of open-source licenses, Phipps sees a larger picture. According to him, the point of all the work OSI does, including the approval process, is to pave the way to make the road smoother for open-source developers:

> The role that OSI plays is to crystallize consensus. Rather than being an adjudicator that makes decisions ex cathedra, we're an organization that provides a venue for people to discuss licensing. We then identify consensus as it arises and then memorialize that consensus. We're more speaker-of-the-house than king.

> That provides an extremely sound way for people to reduce the burden on developers of having to evaluate licensing. As open source becomes more and more the core of the way businesses develop software, it's more and more valuable to have that crystallization of consensus process taking out the uncertainty for people who are needing to work between different entities. Without that, you need to constantly be seeking legal advice, you need to constantly be having discussions about whether a license meets the criteria for being open source or not, and the higher uncertainty results in fewer contributions and less collaboration.

One of OSI's duties, and one it has in common with organizations such as the Free Software Foundation (FSF), is that of enforcer of compliance issues with open-source licenses. Like the FSF, OSI prefers to take a carrot rather than stick approach. And because it's the organization that approves open-source licenses, it's in a unique position to nip issues in the bud. Those issues can run the gamut from unnecessary licenses to freeware masquerading as open source. According to Phipps:

> We don't do that in private. We do that fairly publicly and we don't normally need to do that. Normally a member of the license review mailing list, who are all simply members of the community, will go back to people and say "we don't think that's distinctive", "we don't think that's unique enough", "why didn't you use license so and so", or they'll say, "we really don't think your intent behind this license is actually open source." Typically OSI doesn't have to go and say those things to people.
>
> The places where we do get involved in speaking to people directly is where they describe things as open source when they haven't bothered to go through that process and that's the point at which we'll communicate with people privately.

The problem of freeware—proprietary software that's offered without cost—being marketed under the open-source banner is particularly troublesome. In those cases, OSI definitely will reach out and contact the offending companies, as Phipps says, "We do that quite often, and we have a good track record of helping people understand why it's to their business disadvantage to behave in that way."

One of the reasons why OSI is able to get commercial software developers to heed its advice might be because the organization has never taken an anti-business stance. Founding member Michael Tiemann, now VP of open-source affairs at Red Hat, once said that one of the reasons the initiative chose the term "open source" was to "dump the moralizing and confrontational attitude that had been associated with 'free software' in the past and sell the idea strictly on the same pragmatic, business-case grounds that had motivated Netscape."

These days, the organization has ties with many major software vendors and receives most of its financial support from corporate sponsors. However, it has taken steps to ensure that corporate sponsors don't dictate OSI policy. According to Phipps:

> If you want to join a trade association, that's what the Linux Foundation is there for. You can go pay your membership fees and buy a vote there, but OSI is a 501(c)(3). That's means it's a charity that's serving the public's interest and the public benefit.
>
> It would be wrong for us to allow OSI to be captured by corporate interests. When we conceived the sponsorship scheme, we made sure that there was no risk that would happen. Our corporate sponsors do not get any governance role in the organization. They don't get a vote over what's happening, and we've been very slow to accept new corporate sponsors because we wanted to make sure that no one sponsor could have an undue influence if they decided that they no longer liked us or decided to stop paying the sponsorship fees.

This pragmatic approach, which also puts "permissive" licenses like Apache and MIT on equal footing with "copyleft" licenses like the GPL, has traditionally not been met with universal approval from FOSS advocates. The FSF's Richard Stallman has been critical of the organization, although noting that his organization and OSI are essentially on the same page. Years ago, OSI co-founder and creator of The Open Source Definition, Bruce Perens, decried the "schism" between the Free Software and Open Source communities—a schism that Phipps seeks to narrow:

> As I've been giving keynotes about the first 20 years and the next ten years of open

> source, I've wanted to make very clear to people that open source is a progression of the pre-existing idea of free software, that there is no conflict between the idea of free software and the way it can be adopted for commercial or for more structured use under the term open source.
>
> One of the things that I'm very happy about over the last five to six years is the good relations we've been able to have with the Free Software Foundation Europe. We've been able to collaborate with them over amicus briefs in important lawsuits. We are collaborating with them over significant issues, including privacy and including software patents, and I hope in the future that we'll be able to continue cooperating and collaborating. I think that's an important thing to point out, that I want the pre-existing world of free software to have its due credit.

Software patents represent one of several areas into which OSI has been expanding. Patents have long been a thorny issue for open source, because they have the potential to affect not only people who develop software, but also companies who merely run open-source software on their machines. They also can be like a snake in the grass; any software application can be infringing on an unknown patent. According to Phipps:

> We have a new project that is just getting started, revisiting the role of patents and standards. We have helped bring together a post-graduate curriculum on open source for educating graduates on how to develop open-source software and how to understand it.
>
> We also host other organizations that need a fiduciary host so that they don't have to do their own bookkeeping and legal filings. For a couple years, we hosted the Open Hatch Project, which has now wound up, and we host other activities. For example, we host the mailing lists for the California Association of Voting Officials, who are trying to promote open-source software in voting machines in North America.

Like everyone else in tech these days, OSI is also grappling with diversity issues. Phipps said the organization is seeking to deal with that issue by starting at the

membership level:

> At the moment I feel that I would very much like to see a more diverse membership. I'd like to see us more diverse geographically. I'd like to see us more diverse in terms of the ethnicity and gender of the people who are involved. I would like to see us more diverse in terms of the businesses from which people are employed.
>
> I'd like to see all those improve and so, over the next few years (assuming that I remain president because I have to be re-elected every year by the board) that will also be one of the focuses that I have.

And to wrap things up, here's how he plans to go about that:

> This year is the anniversary year, and we've been able to arrange for OSI to be present at a conference pretty much every month, in some cases two or three per month, and the vast majority of those events are global. For example, FOSSASIA is coming up, and we're backing that. We are sponsoring a hostel where we'll be having 50 software developers who are able to attend FOSSASIA because of the sponsorship. Our goal here is to raise our profile and to recruit membership by going and engaging with local communities globally. I think that's going to be a very important way that we do it. ■

Christine Hall has been a journalist since 1971. In 2001, she began writing a weekly consumer computer column, and she started covering Linux and FOSS in 2002 after making the switch to GNU/Linux. When not writing about tech, she can be found watching Netflix or anything else she can find that's not housecleaning. Follow her on Twitter: @BrideOfLinux.

Create Dynamic Wallpaper with a Bash Script

Harness the power of bash and learn how to scrape websites for exciting new images every morning.

By Patrick Wheelan

So, you want a cool dynamic desktop wallpaper without dodgy programs and a million viruses? The good news is, this is Linux, and anything is possible. I started this project because I was bored of my standard OS desktop wallpaper, and I have slowly created a plethora of scripts to pull images from several sites and set them as my desktop background. It's a nice little addition to my day—being greeted by a different cat picture or a panorama of a country I didn't know existed. The great news is that it's easy to do, so let's get started.

Why Bash?

BAsh (The Bourne Again shell) is standard across almost all *NIX systems and provides a wide range of operations "out of the box", which would take time and copious lines of code to achieve in a conventional coding or even scripting language. Additionally, there's no need to re-invent the wheel. It's much easier to use somebody else's program to download webpages for example, than to deal with low-level system sockets in C.

How's It Going to Work?

The concept is simple. Choose a site with images you like and "scrape" the page for those images. Then once you have a direct link, you download them and set them as

the desktop wallpaper using the display manager. Easy right?

A Simple Example: xkcd

To start off, let's venture to every programmer's second-favorite page after Stack Overflow: xkcd. Loading the page, you should be greeted by the daily comic strip and some other data.

Now, what if you want to see this comic without venturing to the xkcd site? You need a script to do it for you. First, you need to know how the webpage looks to the computer, so download it and take a look. To do this, use `wget`, an easy-to-use, commonly installed, non-interactive, network downloader. So, on the command line, call `wget`, and give it the link to the page:

```
user@LJ $: wget https://www.xkcd.com/


--2018-01-27 21:01:39--  https://www.xkcd.com/
Resolving www.xkcd.com... 151.101.0.67, 151.101.192.67,
 ↪151.101.64.67, ...
Connecting to www.xkcd.com|151.101.0.67|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2606 (2.5K) [text/html]
Saving to: 'index.html'

index.html                                          100%
[=====================================================>]
2.54K  --.-KB/s    in 0s

2018-01-27 21:01:39 (23.1 MB/s) - 'index.html' saved [6237]
```

As you can see in the output, the page has been saved to index.html in your current directory. Using your favourite editor, open it and take a look (I'm using nano for this example):

```
user@LJ $: nano index.html
```

Now you might realize, despite this being a rather bare page, there's a lot of code in that file. Instead of going through it all, let's use `grep`, which is perfect for this task. Its sole function is to print lines matching your search. Grep uses the syntax:

```
user@LJ $: grep [search] [file]
```

Looking at the daily comic, its current title is "Night Sky". Searching for "night" with `grep` yields the following results:

```
user@LJ $: grep "night" index.html

<img src="//imgs.xkcd.com/comics/night_sky.png"
 ↪title="There's a mountain lion nearby, but it
 ↪didn't notice you because it's
 ↪reading Facebook." alt="Night Sky"
 ↪srcset="//imgs.xkcd.com/comics/night_sky_2x.png 2x"/>
Image URL (for hotlinking/embedding):
 ↪https://imgs.xkcd.com/comics/night_sky.png
```

The `grep` search has returned two image links in the file, each related to "night". Looking at those two lines, one is the image in the page, and the other is for hotlinking and is already a usable link. You'll be obtaining the first link, however, as it is more representative of other pages that don't provide an easy link, and it serves as a good introduction to the use of `grep` and `cut`.

To get the first link out of the page, you first need to identify it in the file programmatically. Let's try `grep` again, but this time instead of using a string you already know ("night"), let's approach as if you know nothing about the page. Although the link will be different, the HTML should remain the same; therefore, `<img src=` always should appear before the link you want:

```
user@LJ $: grep "img src=" index.html

<span><a href="/"><img src="/s/0b7742.png" alt="xkcd.com logo"
 ↪height="83" width="185"/></a></span>
<img src="//imgs.xkcd.com/comics/night_sky.png"
 ↪title="There's a mountain lion nearby, but it
 ↪didn't notice you because it's reading Facebook."
 ↪alt="Night Sky" srcset="//imgs.xkcd.com/comics/
↪night_sky_2x.png 2x"/>
<img src="//imgs.xkcd.com/s/a899e84.jpg" width="520"
 ↪height="100" alt="Selected Comics" usemap="#comicmap"/>
```

It looks like there are three images on the page. Comparing these results from the first `grep`, you'll see that <img src="//imgs.xkcd.com/comics/night_sky.png" has been returned again. This is the image you want, but how do you separate it from the other two? The easiest way is to pass it through another `grep`. The other two links contain "/s/"; whereas the link we want contains "/comics/". So, you need to `grep` the output of the last command for "/comics/". To pass along the output of the last command, use the pipe character (|):

```
user@LJ $: grep "img src=" index.html | grep "/comics/"

<img src="//imgs.xkcd.com/comics/night_sky.png"
 ↪title="There's a mountain lion nearby, but it
 ↪didn't notice you because it's reading Facebook."
 ↪alt="Night Sky" srcset="//imgs.xkcd.com/comics/
↪night_sky_2x.png 2x"/>
```

And, there's the line! Now you just need to separate the image link from the rest of it with the `cut` command. `cut` uses the syntax:

```
user@LJ $: cut [-d  delimeter] [-f field] [-c characters]
```

To cut the link from the rest of the line, you'll want to cut next to the quotation mark and select the field before the next quotation mark. In other words, you want the text between the quotes, or the link, which is done like this:

```
user@LJ $: grep "img src=" index.html | grep "/comics/" |
 ↪cut -d\" -f2

//imgs.xkcd.com/comics/night_sky.png
```

And, you've got the link. But wait! What about those pesky forward slashes at the beginning? You can cut those out too:

```
user@LJ $: grep "img src=" index.html | grep "/comics/" |
 ↪cut -d\" -f 2 | cut -c 3-

imgs.xkcd.com/comics/night_sky.png
```

Now you've just cut the first three characters from the line, and you're left with a link straight to the image. Using `wget` again, you can download the image:

```
user@LJ $: wget imgs.xkcd.com/comics/night_sky.png


--2018-01-27 21:42:33--  http://imgs.xkcd.com/comics/night_sky.png
Resolving imgs.xkcd.com... 151.101.16.67, 2a04:4e42:4::67
Connecting to imgs.xkcd.com|151.101.16.67|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 54636 (53K) [image/png]
Saving to: 'night_sky.png'

night_sky.png                                      100%
[============================================================>]
53.36K  --.-KB/s    in 0.04s
```

```
2018-01-27 21:42:33 (1.24 MB/s) - 'night_sky.png'
 ↪saved [54636/54636]
```

Now you have the image in your directory, but its name will change when the comic's name changes. To fix that, tell `wget` to save it with a specific name:

```
user@LJ $: wget "$(grep "img src=" index.html | grep "/comics/"
 ↪| cut -d\" -f2 | cut -c 3-)" -O wallpaper
--2018-01-27 21:45:08--  http://imgs.xkcd.com/comics/night_sky.png
Resolving imgs.xkcd.com... 151.101.16.67, 2a04:4e42:4::67
Connecting to imgs.xkcd.com|151.101.16.67|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 54636 (53K) [image/png]
Saving to: 'wallpaper'

wallpaper                                          100%
[=================================================================>]
53.36K  --.-KB/s    in 0.04s

2018-01-27 21:45:08 (1.41 MB/s) - 'wallpaper' saved [54636/54636]
```

The `-O` option means that the downloaded image now has been saved as "wallpaper". Now that you know the name of the image, you can set it as a wallpaper. This varies depending upon which display manager you're using. The most popular are listed below, assuming the image is located at /home/user/wallpaper.

GNOME:

```
gsettings set org.gnome.desktop.background picture-uri
 ↪"File:///home/user/wallpaper"
gsettings set org.gnome.desktop.background picture-options
 ↪scaled
```

Cinnamon:

```
gsettings set org.cinnamon.desktop.background picture-uri
 ↪"file:///home/user/wallpaper"
gsettings set org.cinnamon.desktop.background picture-options
 ↪scaled
```

Xfce:

```
xfconf-query --channel xfce4-desktop --property
 ↪/backdrop/screen0/monitor0/image-path --set
 ↪/home/user/wallpaper
```

You can set your wallpaper now, but you need different images to mix in. Looking at the webpage, there's a "random" button that takes you to a random comic. Searching with `grep` for "random" returns the following:

```
user@LJ $: grep random index.html

<li><a href="//c.xkcd.com/random/comic/">Random</a></li>
<li><a href="//c.xkcd.com/random/comic/">Random</a></li>
```

This is the link to a random comic, and downloading it with `wget` and reading the result, it looks like the initial comic page. Success!

Now that you've got all the components, let's put them together into a script, replacing www.xkcd.com with the new c.xkcd.com/random/comic/:

```
#!/bin/bash

wget c.xkcd.com/random/comic/

wget "$(grep "img src=" index.html | grep /comics/ | cut -d\"
```

```
 ↪-f 2 | cut -c 3-)" -O wallpaper

gsettings set org.gnome.desktop.background picture-uri
 ↪"File:///home/user/wallpaper"
gsettings set org.gnome.desktop.background picture-options
 ↪scaled
```

All of this should be familiar except the first line, which designates this as a bash script, and the second `wget` command. To capture the output of commands into a variable, you use `$()`. In this case, you're capturing the grepping and cutting process—capturing the final link and then downloading it with `wget`. When the script is run, the commands inside the bracket are all run producing the image link before `wget` is called to download it.

There you have it—a simple example of a dynamic wallpaper that you can run anytime you want.

If you want the script to run automatically, you can add a cron job to have cron run it for you. So, edit your cron tab with:

```
user@LJ $: crontab -e
```

My script is called "xkcd", and my crontab entry looks like this:

```
@reboot /bin/bash /home/user/xkcd
```

This will run the script (located at /home/user/xkcd) using bash, every restart.

Reddit

The script above shows how to search for images in HTML code and download them. But, you can apply this to any website of your choice—although the HTML code will be different, the underlying concepts remain the same. With that in mind, let's tackle downloading images from Reddit. Why Reddit? Reddit is possibly the largest

blog on the internet and the third-most-popular site in the US. It aggregates content from many different communities together onto one site. It does this through use of "subreddits", communities that join together to form Reddit. For the purposes of this article, let's focus on subreddits (or "subs" for short) that primarily deal with images. However, any subreddit, as long as it allows images, can be used in this script.

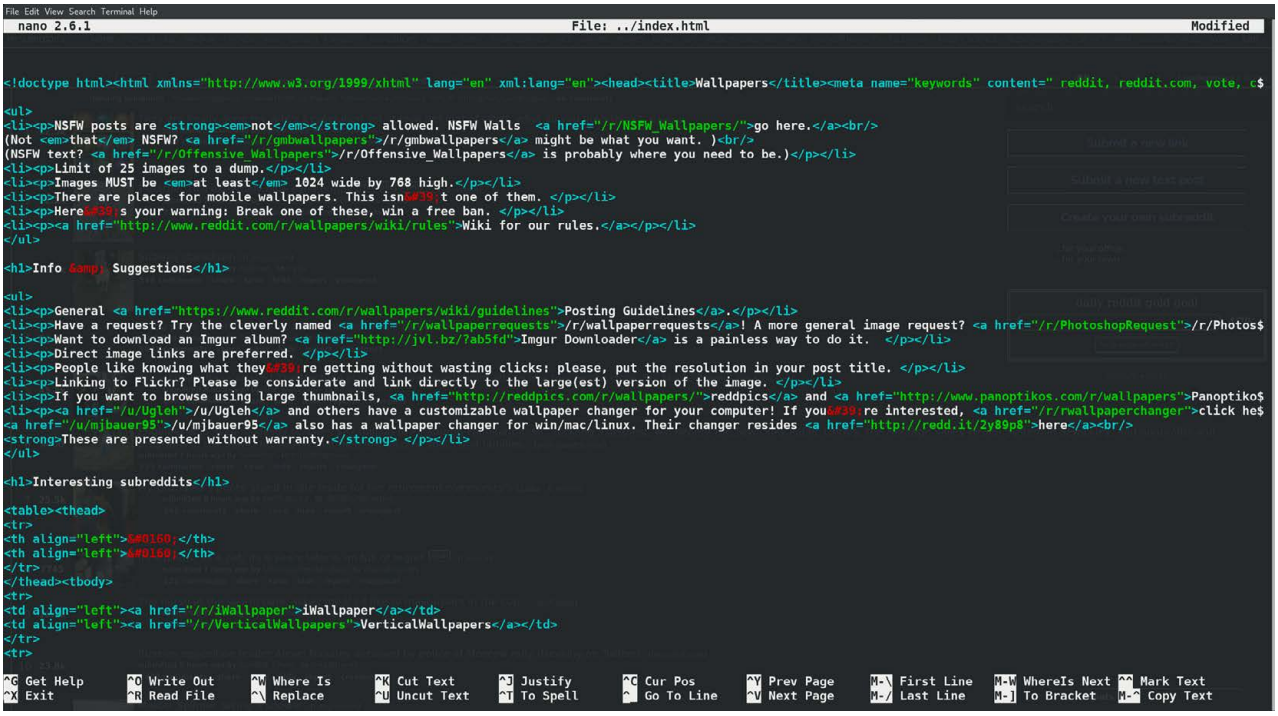


Figure 1. Scraping the Web Made Simple—Analysing Web Pages in a Terminal

Diving In

Just like the xkcd script, you need to download the web page from a subreddit to analyse it. I'm using reddit.com/r/wallpapers for this example. First, check for images in the HTML:

```
user@LJ $: wget https://www.reddit.com/r/wallpapers/ && grep
 ↪"img src=" index.html

--2018-01-28 20:13:39--  https://www.reddit.com/r/wallpapers/
```

```
Resolving www.reddit.com... 151.101.17.140
Connecting to www.reddit.com|151.101.17.140|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 27324 (27K) [text/html]
Saving to: 'index.html'

index.html                                          100%
[=================================================================>]
26.68K  --.-KB/s    in 0.1s

2018-01-28 20:13:40 (270 KB/s) - 'index.html' saved [169355]

</div></form><div class="bottom"><span class="age">a community
 ↪for <time title="Thu May 1 16:17:13 2008 UTC"
 ↪datetime="2008-05-01T16:17:13+00:00">9 years</time></span>
 ↪....Forever and ever......

--- SNIP ---
```

All the images have been returned in one long line, because the HTML for the images is also in one long line. You need to split this one long line into the separate image links. Enter Regex.

Regex is short for regular expression, a system used by many programs to allow users to match an expression to a string. It contains wild cards, which are special characters that match certain characters. For example, the * character will match every character. For this example, you want an expression that matches every link in the HTML file. All HTML links have one string in common. They all take the form href="LINK". Let's write a regex expression to match:

```
href="([^"#]+)"
```

Now let's break it down:

- href=“ — simply states that the first characters should match these.
- () — forms a capture group.
- [^] — forms a negated set. The string shouldn't match any of the characters inside.
- + — the string should match one or more of the preceding tokens.

Altogether the regex matches a string that begins href=“, doesn't contain any quotation marks or hashtags and finishes with a quotation mark.

This regex can be used with `grep` like this:

```
user@LJ $: grep -o -E 'href="([^"#]+)"' index.html

href="/static/opensearch.xml"
href="https://www.reddit.com/r/wallpapers/"
href="//out.reddit.com"
href="//out.reddit.com"
href="//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-57x57.png"

--- SNIP ---
```

The `-e` options allow for extended regex options, and the `-o` switch means `grep` will print only patterns exactly matching and not the whole line. You now have a much more manageable list of links. From there, you can use the same techniques from the first script to extract the links and filter for images. This looks like the following:

```
user@LJ $: grep -o -E 'href="([^"#]+)"' index.html | cut -d'"'
 ↪-f2 | sort | uniq | grep -E '.jpg|.png'
```

```
https://i.imgur.com/6DO2uqT.png
https://i.imgur.com/Ualn765.png
https://i.imgur.com/UO5ck0M.jpg
https://i.redd.it/s8ngtz6xtnc01.jpg
//www.redditstatic.com/desktop2x/img/favicon/
↪android-icon-192x192.png
//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-114x114.png
//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-120x120.png
//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-144x144.png
//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-152x152.png
//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-180x180.png
//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-57x57.png
//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-60x60.png
//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-72x72.png
//www.redditstatic.com/desktop2x/img/favicon/
↪apple-icon-76x76.png
//www.redditstatic.com/desktop2x/img/favicon/
↪favicon-16x16.png
//www.redditstatic.com/desktop2x/img/favicon/
↪favicon-32x32.png
//www.redditstatic.com/desktop2x/img/favicon/
↪favicon-96x96.png
```

The final `grep` uses regex again to match .jpg or .png. The | character acts as a boolean `OR` operator.

As you can see, there are four matches for actual images: two .jpgs and two .pngs. The others are Reddit default images, like the logo. Once you remove those images, you'll have a final list of images to set as a wallpaper. The easiest way to remove these images from the list is with `sed`:

```
user@LJ $: grep -o -E 'href="([^"#]+)"' index.html | cut -d'"'
 ↪-f2 | sort | uniq | grep -E '.jpg|.png' | sed /redditstatic/d

https://i.imgur.com/6DO2uqT.png
https://i.imgur.com/Ualn765.png
https://i.imgur.com/UO5ck0M.jpg
https://i.redd.it/s8ngtz6xtnc01.jpg
```

`sed` works by matching what's between the two forward slashes. The `d` on the end tells `sed` to delete the lines that match the pattern, leaving the image links.

The great thing about sourcing images from Reddit is that every subreddit contains nearly identical HTML; therefore, this small script will work on any subreddit.

Creating a Script

To create a script for Reddit, it should be possible to choose from which subreddits you'd like to source images. I've created a directory for my script and placed a file called "links" in the directory with it. This file contains the subreddit links in the following format:

```
https://www.reddit.com/r/wallpapers
https://www.reddit.com/r/wallpaper
https://www.reddit.com/r/NationalPark
https://www.reddit.com/r/tiltshift
https://www.reddit.com/r/pic
```

At run time, I have the script read the list and download these subreddits before stripping images from them.

Since you can have only one image at a time as desktop wallpaper, you'll want to narrow down the selection of images to just one. First, however, it's best to have a wide range of images without using a lot of bandwidth. So you'll want to download the web pages for multiple subreddits and strip the image links but not download the images themselves. Then you'll use a random selector to select one image link and download that one to use as a wallpaper.

Finally, if you're downloading lots of subreddit's web pages, the script will become very slow. This is because the script waits for each command to complete before proceeding. To circumvent this, you can fork a command by appending an ampersand (&) character. This creates a new process for the command, "forking" it from the main process (the script).

Here's my fully annotated script:

```
#!/bin/bash

DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
 ↪# Get the script's current directory

linksFile="links"

mkdir $DIR/downloads
cd $DIR/downloads

# Strip the image links from the html
function parse {
grep -o -E 'href="([^"#]+)"' $1 | cut -d'"' -f2 | sort | uniq
 ↪| grep -E '.jpg|.png' >> temp
grep -o -E 'href="([^"#]+)"' $2 | cut -d'"' -f2 | sort | uniq
 ↪| grep -E '.jpg|.png' >> temp
grep -o -E 'href="([^"#]+)"' $3 | cut -d'"' -f2 | sort | uniq
 ↪| grep -E '.jpg|.png' >> temp
```

```
grep -o -E 'href="([^"#]+)"' $4 | cut -d'"' -f2 | sort | uniq
 ↪| grep -E '.jpg|.png' >> temp
}

# Download the subreddit's webpages
function download {
rname=$( echo $1 | cut -d / -f 5  )
tname=$(echo t.$rname)
rrname=$(echo r.$rname)
cname=$(echo c.$rname)
wget --load-cookies=../cookies.txt -O $rname $1
 ↪&>/dev/null &
wget --load-cookies=../cookies.txt -O $tname $1/top
 ↪&>/dev/null &
wget --load-cookies=../cookies.txt -O $rrname $1/rising
 ↪&>/dev/null &
wget --load-cookies=../cookies.txt -O $cname $1/controversial
 ↪&>/dev/null &
wait # wait for all forked wget processes to return
parse $rname $tname $rrname $cname
}


# For each line in links file
while read l; do
   if [[ $l != *"#"* ]]; then # if line doesn't contain a
 ↪hashtag (comment)
        download $l&
   fi
done < ../$linksFile

wait # wait for all forked processes to return
```

```
sed -i '/www.redditstatic.com/d' temp # remove reddit pics that
 ↪exist on most pages from the list


wallpaper=$(shuf -n 1 temp) # select randomly from file and DL

echo $wallpaper >> $DIR/log # save image into log in case
 ↪we want it later

wget -b $wallpaper -O $DIR/wallpaperpic 1>/dev/null # Download
 ↪wallpaper image

gsettings set org.gnome.desktop.background picture-uri
 ↪file://$DIR/wallpaperpic # Set wallpaper (Gnome only!)


rm -r $DIR/downloads # cleanup
```

Just like before, you can set up a cron job to run the script for you at every reboot or whatever interval you like.

And, there you have it—a fully functional cat-image harvester. May your morning logins be greeted with many furry faces. Now go forth and discover new subreddits to gawk at and new websites to scrape for cool wallpapers. ■

Patrick Whelan is a first-year student at Edge Hill University in the UK. He is an aspiring developer, blogger and all-round hacker.

Creating an Adventure Game in the Terminal with ncurses

How to use `curses` functions to read the keyboard and manipulate the screen.

By Jim Hall

My previous article introduced the `ncurses` library and provided a simple program that demonstrated a few `curses` functions to put text on the screen. In this follow-up article, I illustrate how to use a few other `curses` functions.

An Adventure

When I was growing up, my family had an Apple II computer. It was on this machine that my brother and I taught ourselves how to write programs in AppleSoft BASIC. After writing a few math puzzles, I moved on to creating games. Having grown up in the 1980s, I already was a fan of the *Dungeons and Dragons* tabletop games, where you role-played as a fighter or wizard on some quest to defeat monsters and plunder loot in strange lands. So it shouldn't be surprising that I also created a rudimentary adventure game.

The AppleSoft BASIC programming environment supported a neat feature: in standard resolution graphics mode (GR mode), you could probe the color of a particular pixel on the screen. This allowed a shortcut to create an adventure game. Rather than create and update an in-memory map that was transferred to

Figure 1. A simple Tabletop Game Map with a Lake and Mountains

the screen periodically, I could rely on GR mode to maintain the map for me, and my program could query the screen as the player's character moved around the screen. Using this method, I let the computer do most of the hard work. Thus, my top-down adventure game used blocky GR mode graphics to represent my game map.

My adventure game used a simple map that represented a large field with a mountain range running down the middle and a large lake on the upper-left side. I might crudely draw this map for a tabletop gaming campaign to include a narrow path through the mountains, allowing the player to pass to the far side.

You can draw this map in `curses` using characters to represent grass, mountains

and water. Next, I describe how to do just that using `curses` functions and how to create and play a similar adventure game in the Linux terminal.

Constructing the Program

In my last article, I mentioned that most `curses` programs start with the same set of instructions to determine the terminal type and set up the `curses` environment:

```
initscr();
cbreak();
noecho();
```

For this program, I add another statement:

```
keypad(stdscr, TRUE);
```

The `TRUE` flag allows `curses` to read the keypad and function keys from the user's terminal. If you want to use the up, down, left and right arrow keys in your program, you need to use `keypad(stdscr, TRUE)` here.

Having done that, you now can start drawing to the terminal screen. The `curses` functions include several ways to draw text on the screen. In my previous article, I demonstrated the `addch()` and `addstr()` functions and their associated `mvaddch()` and `mvaddstr()` counterparts that first moved to a specific location on the screen before adding text. To create the adventure game map on the terminal, you can use another set of functions: `vline()` and `hline()`, and their partner functions `mvvline()` and `mvhline()`. These `mv` functions accept screen coordinates, a character to draw and how many times to repeat that character. For example, `mvhline(1, 2, '-', 20)` will draw a line of 20 dashes starting at line 1, column 2.

To draw the map to the terminal screen programmatically, let's define this `draw_map()` function:

```
#define GRASS     ' '
#define EMPTY     '.'
#define WATER     '~'
#define MOUNTAIN  '^'
#define PLAYER    '*'

void draw_map(void)
{
    int y, x;

    /* draw the quest map */

    /* background */

    for (y = 0; y < LINES; y++) {
        mvhline(y, 0, GRASS, COLS);
    }

    /* mountains, and mountain path */

    for (x = COLS / 2; x < COLS * 3 / 4; x++) {
        mvvline(0, x, MOUNTAIN, LINES);
    }

    mvhline(LINES / 4, 0, GRASS, COLS);

    /* lake */

    for (y = 1; y < LINES / 2; y++) {
        mvhline(y, 1, WATER, COLS / 3);
    }
}
```

In drawing this map, note the use of `mvvline()` and `mvhline()` to fill large chunks of characters on the screen. I created the fields of grass by drawing horizontal lines (`mvhline`) of characters starting at column 0, for the entire height and width of the screen. I added the mountains on top of that by drawing vertical lines (`mvvline`), starting at row 0, and a mountain path by drawing a single horizontal line (`mvhline`). And, I created the lake by drawing a series of short horizontal lines (`mvhline`). It may seem inefficient to draw overlapping rectangles in this way, but remember that `curses` doesn't actually update the screen until I call the `refresh()` function later.

Having drawn the map, all that remains to create the game is to enter a loop where the program waits for the user to press one of the up, down, left or right direction keys and then moves a player icon appropriately. If the space the player wants to move into is unoccupied, it allows the player to go there.

You can use `curses` as a shortcut. Rather than having to instantiate a version of the map in the program and replicate this map to the screen, you can let the screen keep track of everything for you. The `inch()` function and associated `mvinch()` function allow you to probe the contents of the screen. This allows you to query `curses` to find out whether the space the player wants to move into is already filled with water or blocked by mountains. To do this, you'll need a helper function that you'll use later:

```
int is_move_okay(int y, int x)
{
    int testch;

    /* return true if the space is okay to move into */

    testch = mvinch(y, x);
    return ((testch == GRASS) || (testch == EMPTY));
}
```

As you can see, this function probes the location at column y, row x and returns true if the space is suitably unoccupied, or false if not.

That makes it really easy to write a navigation loop: get a key from the keyboard and move the user's character around depending on the up, down, left and right arrow keys. Here's a simplified version of that loop:

```
    do {
        ch = getch();

        /* test inputted key and determine direction */

        switch (ch) {
        case KEY_UP:
            if ((y > 0) && is_move_okay(y - 1, x)) {
                y = y - 1;
            }
            break;
        case KEY_DOWN:
            if ((y < LINES - 1) && is_move_okay(y + 1, x)) {
                y = y + 1;
            }
            break;
        case KEY_LEFT:
            if ((x > 0) && is_move_okay(y, x - 1)) {
                x = x - 1;
            }
            break;
        case KEY_RIGHT
            if ((x < COLS - 1) && is_move_okay(y, x + 1)) {
                x = x + 1;
            }
            break;
```

```
        }
    }
    while (1);
```

To use this in a game, you'll need to add some code inside the loop to allow other keys (for example, the traditional WASD movement keys), provide a method for the user to quit the game and move the player's character around the screen. Here's the program in full:

```
/* quest.c */

#include <curses.h>
#include <stdlib.h>

#define GRASS     ' '
#define EMPTY     '.'
#define WATER     '~'
#define MOUNTAIN  '^'
#define PLAYER    '*'

int is_move_okay(int y, int x);
void draw_map(void);

int main(void)
{
    int y, x;
    int ch;

    /* initialize curses */

    initscr();
    keypad(stdscr, TRUE);
    cbreak();
```

```
    noecho();

    clear();

    /* initialize the quest map */

    draw_map();

    /* start player at lower-left */

    y = LINES - 1;
    x = 0;

    do {
     /* by default, you get a blinking cursor - use it to indicate
player */

     mvaddch(y, x, PLAYER);
     move(y, x);
     refresh();

     ch = getch();

     /* test inputted key and determine direction */

     switch (ch) {
     case KEY_UP:
     case 'w':
     case 'W':
         if ((y > 0) && is_move_okay(y - 1, x)) {
          mvaddch(y, x, EMPTY);
          y = y - 1;
         }
```

```
            break;
        case KEY_DOWN:
        case 's':
        case 'S':
            if ((y < LINES - 1) && is_move_okay(y + 1, x)) {
             mvaddch(y, x, EMPTY);
             y = y + 1;
            }
            break;
        case KEY_LEFT:
        case 'a':
        case 'A':
            if ((x > 0) && is_move_okay(y, x - 1)) {
             mvaddch(y, x, EMPTY);
             x = x - 1;
            }
            break;
        case KEY_RIGHT:
        case 'd':
        case 'D':
            if ((x < COLS - 1) && is_move_okay(y, x + 1)) {
             mvaddch(y, x, EMPTY);
             x = x + 1;
            }
            break;
        }
    }
    while ((ch != 'q') && (ch != 'Q'));

    endwin();

    exit(0);
}
```

```
int is_move_okay(int y, int x)
{
    int testch;

    /* return true if the space is okay to move into */

    testch = mvinch(y, x);
    return ((testch == GRASS) || (testch == EMPTY));
}

void draw_map(void)
{
    int y, x;

    /* draw the quest map */

    /* background */

    for (y = 0; y < LINES; y++) {
     mvhline(y, 0, GRASS, COLS);
    }

    /* mountains, and mountain path */

    for (x = COLS / 2; x < COLS * 3 / 4; x++) {
     mvvline(0, x, MOUNTAIN, LINES);
    }

    mvhline(LINES / 4, 0, GRASS, COLS);

    /* lake */
```

```
    for (y = 1; y < LINES / 2; y++) {
     mvhline(y, 1, WATER, COLS / 3);
    }
}
```

In the full program listing, you can see the complete arrangement of `curses` functions to create the game:

1) Initialize the `curses` environment.

2) Draw the map.

3) Initialize the player coordinates (lower-left).

4) Loop:

- Draw the player's character.
- Get a key from the keyboard.
- Adjust the player's coordinates up, down, left or right, accordingly.
- Repeat.

5) When done, close the `curses` environment and exit.

Let's Play

When you run the game, the player's character starts in the lower-left corner. As the player moves around the play area, the program creates a "trail" of dots. This helps show where the player has been before, so the player can avoid crossing the path unnecessarily.

To create a complete adventure game on top of this, you might add random

encounters with various monsters as the player navigates his or her character around the play area. You also could include special items the player could discover or loot after defeating enemies, which would enhance the player's abilities further.

But to start, this is a good program for demonstrating how to use the `curses` functions to read the keyboard and manipulate the screen.

Next Steps

This program is a simple example of how to use the `curses` functions to update and read the screen and keyboard. You can do so much more with `curses`, depending

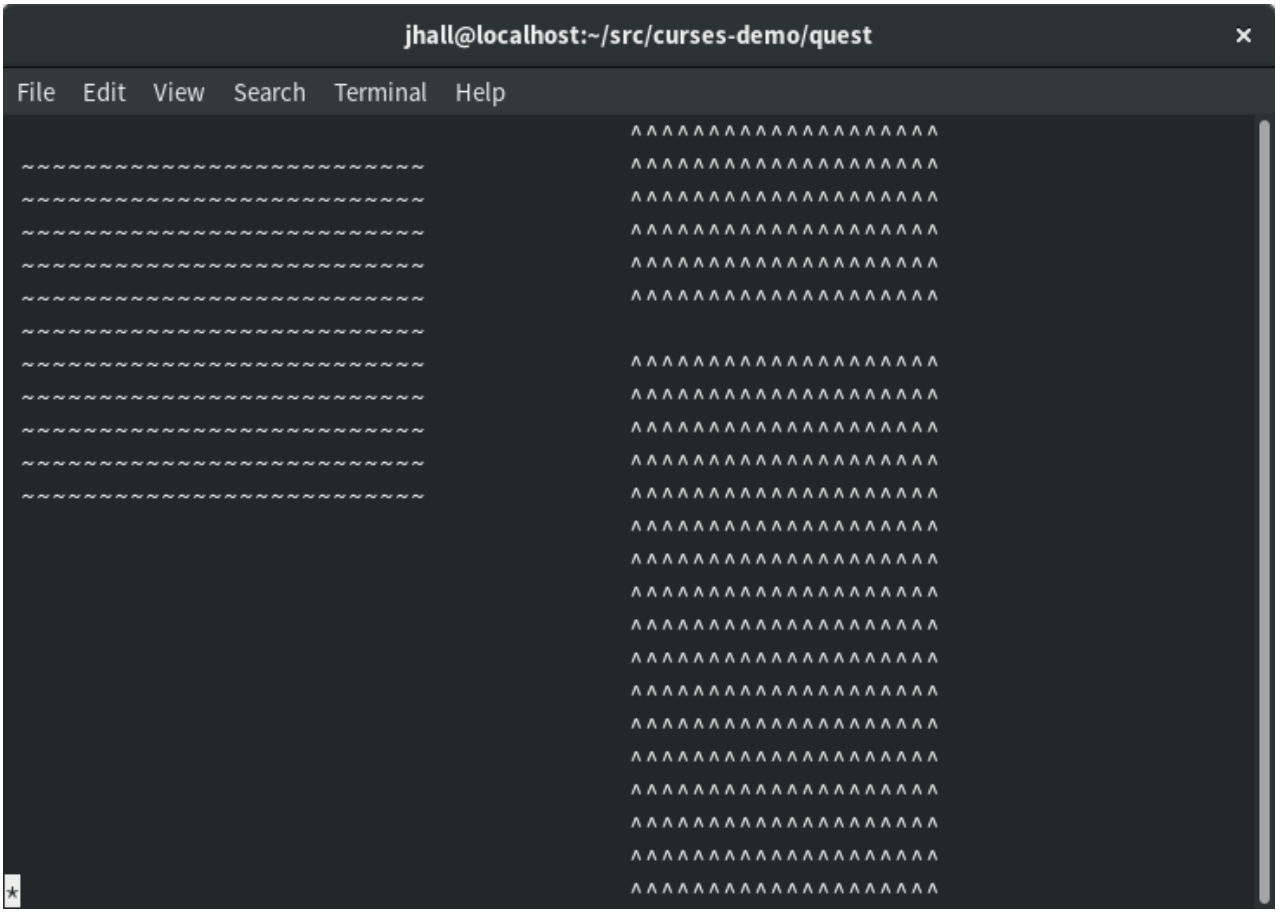


Figure 2. The player starts the game in the lower-left corner.

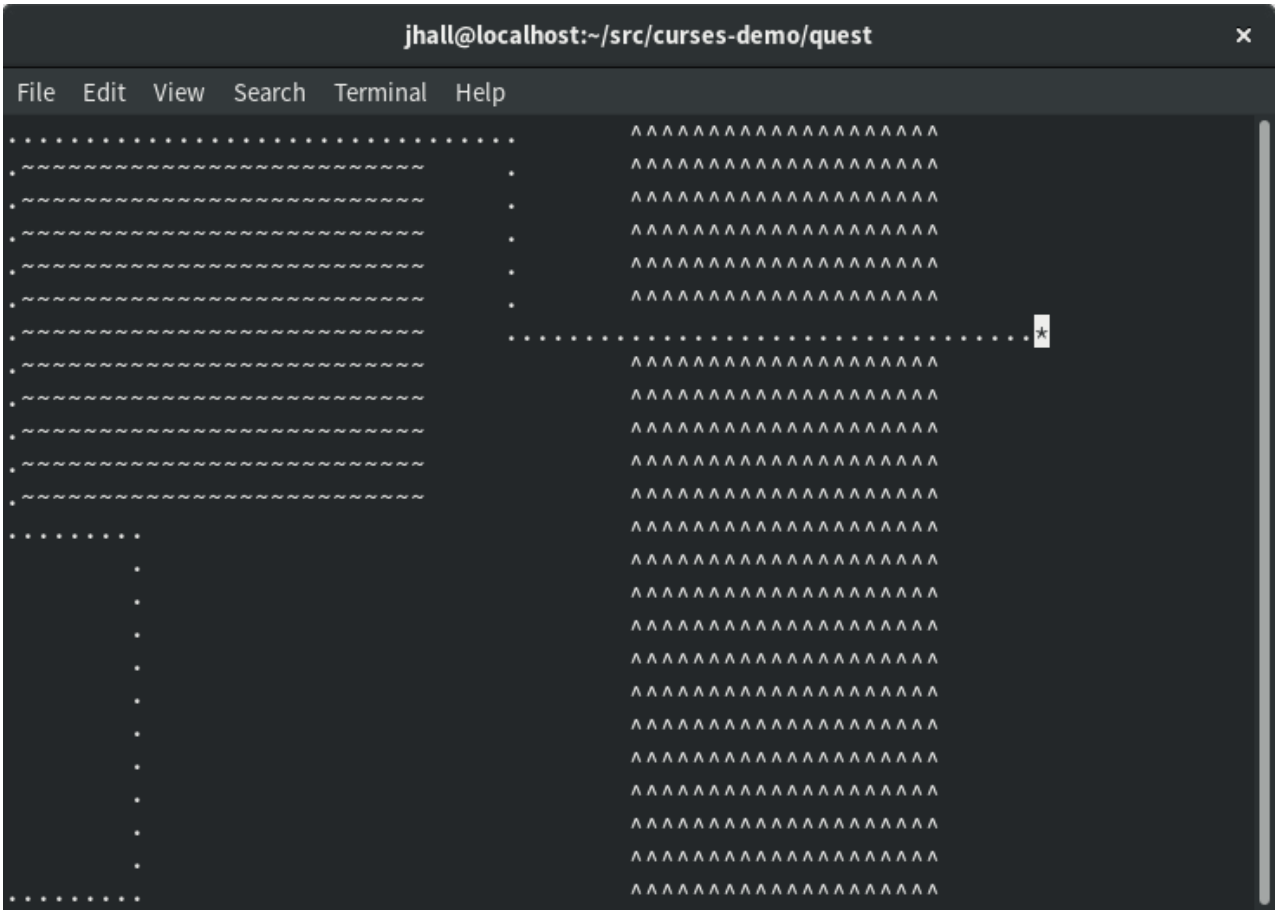


Figure 3. The player can move around the play area, such as around the lake and through the mountain pass.

on what you need your program to do. In a follow up article, I plan to show how to update this sample program to use colors. In the meantime, if you are interested in learning more about `curses`, I encourage you to read Pradeep Padala's NCURSES Programming HOWTO at the Linux Documentation Project. ■

Jim Hall is an advocate for free and open-source software, best known for his work on the FreeDOS Project, and he also focuses on the usability of open-source software. Jim is the Chief Information Officer at Ramsey County, Minnesota.

ONNX: the Open Neural Network Exchange Format

An open-source battle is being waged for the soul of artificial intelligence. It is being fought by industry titans, universities and communities of machine-learning researchers world-wide. This article chronicles one small skirmish in that fight: a standardized file format for neural networks. At stake is the open exchange of data among a multitude of tools instead of competing monolithic frameworks.

By Braddock Gaskill

The good news is that the battleground is Free and Open. None of the big players are pushing closed-source solutions. Whether it is Keras and Tensorflow backed by Google, MXNet by Apache endorsed by Amazon, or Caffe2 or PyTorch supported by Facebook, all solutions are open-source software.

Unfortunately, while these projects are *open*, they are not *interoperable*. Each framework constitutes a complete stack that until recently could not interface in any way with any other framework. A new industry-backed standard, the Open Neural Network Exchange format, could change that.

Now, imagine a world where you can train a neural network in Keras, run the trained model through the NNVM optimizing compiler and deploy it to production on MXNet. And imagine that is just one of countless combinations of interoperable deep learning

tools, including visualizations, performance profilers and optimizers. Researchers and DevOps no longer need to compromise on a single toolchain that provides a mediocre modeling environment and so-so deployment performance.

What is required is a standardized format that can express any machine-learning model and store trained parameters and weights, readable and writable by a suite of independently developed software.

Enter the Open Neural Network Exchange Format (ONNX).

The Vision

To understand the drastic need for interoperability with a standard like ONNX, we first must understand the ridiculous requirements we have for existing monolithic frameworks.

A casual user of a deep learning framework may think of it as a language for specifying a neural network. For example, I want 100 input neurons, three fully connected layers each with 50 ReLU outputs, and a softmax on the output. My framework of choice has a domain language to specify this (like Caffe) or bindings to a language like Python with a clear API.

However, the specification of the network architecture is only the tip of the iceberg. Once a network structure is defined, the framework still has a great deal of complex work to do to make it run on your CPU or GPU cluster.

Python, obviously, doesn't run on a GPU. To make your network definition run on a GPU, it needs to be compiled into code for the CUDA (NVIDIA) or OpenCL (AMD and Intel) APIs or processed in an efficient way if running on a CPU. This compilation is complex and why most frameworks don't support both NVIDIA and AMD GPU back ends.

The job is still not complete though. Your framework also has to balance resource allocation and parallelism for the hardware you are using. Are you running on a Titan X card with more than 3,000 compute cores, or a GTX 1060 with far less than half as many? Does your card have 16GB of RAM or only 4? All of this affects how the computations must be optimized and run.

And still it gets worse. Do you have a cluster of 50 multi-GPU machines on which to train your network? Your framework needs to handle that too. Network protocols, efficient allocation, parameter sharing—how much can you ask of a single framework?

Now you say you want to deploy to production? You wish to scale your cluster automatically? You want a solid language with secure APIs?

When you add it all up, it seems absolutely insane to ask one monolithic project to handle all of those requirements. You cannot expect the authors who write the perfect network definition language to be the same authors who integrate deployment systems in Kubernetes or write optimal CUDA compilers.

The goal of ONNX is to break up the monolithic frameworks. Let an ecosystem of contributors develop each of these components, glued together by a common specification format.

The Ecosystem (and Politics)

Interoperability is a healthy sign of an open ecosystem. Unfortunately, until recently, it did not exist for deep learning. Every framework had its own format for storing computation graphs and trained models.

Late last year that started to change. The Open Neural Network Exchange format initiative was launched by Facebook, Amazon and Microsoft, with support from AMD, ARM, IBM, Intel, Huawei, NVIDIA and Qualcomm. Let me rephrase that as *everyone but Google*. The format has been included in most well known frameworks *except Google's TensorFlow* (for which a third-party converter exists).

This seems to be the classic scenario where the clear market leader, Google, has little interest in upending its dominance for the sake of openness. The smaller players are banding together to counter the 500-pound gorilla.

Google is committed to its own TensorFlow model and weight file format, SavedModel, which shares much of the functionality of ONNX. Google is building

its own ecosystem around that format, including TensorFlow Server, Estimator and Tensor2Tensor to name a few.

The ONNX Solution

Building a single file format that can express all of the capabilities of all the deep learning frameworks is no trivial feat. How do you describe convolutions or recurrent networks with memory? Attention mechanisms? Dropout layers? What about embeddings and nearest neighbor algorithms found in fastText or StarSpace?

ONNX cribs a note from TensorFlow and declares everything is a graph of tensor operations. That statement alone is not sufficient, however. Dozens, perhaps hundreds, of operations must be supported, not all of which will be supported by all other tools and frameworks. Some frameworks may also implement an operation differently from their brethren.

There has been considerable debate in the ONNX community about what level tensor operations should be modeled at. Should ONNX be a mathematical toolbox that can support arbitrary equations with primitives such as sine and multiplication, or should it support higher-level constructs like integrated GRU units or Layer Normalization as single monolithic operations?

As it stands, ONNX currently defines about 100 operations. They range in complexity from arithmetic addition to a complete Long Short-Term Memory implementation. Not all tools support all operations, so just because you can generate an ONNX file of your model does not mean it will run anywhere.

Generation of an ONNX model file also can be awkward in some frameworks because it relies on a rigid definition of the order of operations in a graph structure. For example, PyTorch boasts a very pythonic imperative experience when defining models. You can use Python logic to lay out your model's flow, but you do not define a rigid graph structure as in other frameworks like TensorFlow. So there is no graph of operations to save; you actually have to run the model and *trace* the operations. The trace of operations is saved to the ONNX file.

Conclusion

It is early days for deep learning interoperability. Most users still pick a framework and stick with it. And an increasing number of users are going with TensorFlow. Google throws many resources and real-world production experience at it—it is hard to resist.

All frameworks are strong in some areas and weak in others. Every new framework must re-implement the full "stack" of functionality. Break up the stack, and you can play to the strengths of individual tools. That will lead to a healthier ecosystem.

ONNX is a step in the right direction.

Note: the ONNX GitHub page is here.

Disclaimer

Braddock Gaskill is a research scientist with eBay Inc. He contributed to this article in his personal capacity. The views expressed are his own and do not necessarily represent the views of eBay Inc. ■

Braddock Gaskill has 25 years of experience in AI and algorithmic software development. He also co-founded the Internet-in-a-Box open-source project and developed the libre Humane Wikipedia Reader for getting content to students in the developing world.

Oracle Patches Spectre for Red Hat

Red Hat's Spectre remediation currently requires new microcode for a complete fix, which leaves most x86 processors vulnerable as they lack this update. Oracle has released new retpoline kernels that completely remediate Meltdown and Spectre on all compatible CPUs, which I show how to install and test on CentOS here.

By Charles Fisher

The Red Hat community has patiently awaited a retpoline kernel implementation that remediates CVE-2017-5715 (Spectre v2) and closes all Meltdown and Spectre vulnerabilities that have captured headlines this year.

Red Hat's initial fixes rely upon microcode updates for v2 remediation, a decision that leaves the vast majority of AMD64-capable processors in an exploitable state. Intel's new microcode has proven especially problematic; it performs badly and the January 2018 versions were plagued with stability issues that crashed many systems. It is a poor solution to a pressing problem.

Google's retpolines—an ingenious approach to v2—essentially play out the exploit within the Linux kernel in any and all vulnerable code. This method allows the kernel to retain complete control over speculative execution hardware in all architectures upon which it runs. It is faster and more generic than Intel's microcode and seems in every way a superior solution.

Oracle appears to be the first major member of the Red Hat community that has delivered a Linux kernel with retpoline support. The latest version of the

Unbreakable Enterprise Kernel preview release 5 (UEKr5) now offers complete remediation for Meltdown and Spectre regardless of CPU microcode status. The UEKr5 is based on the mainline Linux kernel's long-term support branch v4.14 release. *In late-breaking news, Oracle has issued a production release of the UEKr4 that also includes retpolines, details below.* For corporate users and others with mandated patch schedules over a large number of servers, the UEK now seems to be the only solution for complete Spectre coverage on all CPUs. The UEK brings a number of other advantages over the "Red Hat-Compatible Kernel" (RHCK), but this patch response is likely to drive Oracle deeply into the Red Hat community should they remain the single source.

CentOS Installation

CentOS, Scientific and Oracle Linux are all based off the upstream provider Red Hat. CentOS is a popular variant and is likely the best demonstration for loading the UEK on a peer distro.

It appears that Red Hat itself has been laying groundwork for a retpoline kernel. A January 2018 gcc compiler update implies a successful backport:

```
# rpm -q --changelog gcc | head -2
* Wed Jan 24 2018 Jeff Law ... 4.8.5-16.2
- retpoline support for spectre mitigation (#1538487)
```

Oracle announced a yum repository for the UEKr5 on February 25, 2018. There are detailed instructions for users of Oracle Linux to add this repo, but the installation procedure on CentOS is slightly different. Alternately, the RPMs can be manually downloaded and installed from the repository. Below I describe proceeding with the yum repo.

First, define the repo for yum with the exact contents that are found in Oracle's installation instructions:

```
echo '[ol7_developer_UEKR5]
```

```
name=Oracle Linux $releasever UEK5 Development Packages
 ↪($basearch)
baseurl=http://yum.oracle.com/repo/OracleLinux/OL7/
↪developer_UEKR5/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1' > /etc/yum.repos.d/public-yum-ol7_uek.repo
```

Note that the GPG entry above will not normally exist on CentOS. If you want to install it, the contents of the file must be:

```
# cat /etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.5 (GNU/Linux)

mQENBEwtJWoBCACpiY2rGA6T0ceBi92X88/QclytVBjtDRohOVzs3pmIPh3ULqsW
G323nmyKbKQBBSjh9OnuO9Y09VG8mzr/w9YV0Ix4cI9/HDTERZ2+TR5u+VNn5J5h
yvwQSN/FEK6oH2+mz7O0yUNleN7UltR4MOEkHIoAhIvv+1AQQKN3OM8oalz+3gv/
zz9rAoQczQzT7QWOtBrsRMZgBrKXY/TpJrpVSO3Hx8CnbhKGtLxCCrxZ5v7hh1ht
3CRAr2+h5bDA9KP6vBZWKEs7NgcvtZFDY6EJc7AoApr3phX9hHE2+snTxe82DkFT
uA69C8wLyjPCoSy+tcaCqJKOZelNy9RN/WKRABEBAAG0RE9yYWNsZSBPU1MgZ3Jv
dXAgKE9wZW4gU291cmNlIFNvZnR3YXJlIGdyb3VwKSA8YnVpbGRAb3NzLm9yYWNs
ZS5jb20+iQE8BBMBAgAmAhsDBgsJCAcDAgQVAggDBBYCAwECHgECF4AFAlNhkUEF
CSaOl9cACgkQcvl7dOxVHwNHFQf9G2ZI5ZH8T1VASvQteTyUR7uNgqXEbJhi9zZO
7EO26+wzkj9EGLmH1COdUApZ1cINsYfGGgUJT5mfcRV3yYZbwc4AZJbJe0z7C5Yu
ZLs5W0ryV4bzIqcWnVphIAOwmxMxIVGz8Cr1Dsyyal7ORgYzdfOppYetwtZ+J+Wn
/oVgFkh9v19l/CltBkRh2SqgUZYfCoELo7hUzLNh7cw2av8rcSUKSH3ra9MvpYfS
ANCnouzbgKix1gD6niT3pm1s5II3VuQ2vEcJunnoRFci9FzLXelTuL00MvuxERr7
Fsqm1/D2JfKDbE09qy5bLWrWaTM6zOFQKN7F2edY2uaukLT6/w==
=Djed
-----END PGP PUBLIC KEY BLOCK-----
```

Assuming the GPG key has not been loaded, install the kernel-uek package with a flag

to skip the GPG check:

```
# yum install --nogpgcheck kernel-uek
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: www.gtlib.gatech.edu
 * extras: www.gtlib.gatech.edu
 * updates: www.gtlib.gatech.edu
Resolving Dependencies
--> Running transaction check
---> Package kernel-uek.x86_64 0:4.14.26-1.el7uek will be installed
--> Processing Dependency: linux-firmware >= 20180113-60.git65b1c68
     for package: kernel-uek-4.14.26-1.el7uek.x86_64
--> Running transaction check
---> Package linux-firmware.noarch
      0:20170606-58.gitc990aae.el7_4 will be updated
---> Package linux-firmware.noarch
      0:20180113-60.git65b1c68.el7 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package         Arch    Version                     Repository
================================================================================
Installing:
 kernel-uek      x86_64  4.14.26-1.el7uek            ol7_developer_UEKR5
Updating for dependencies:
 linux-firmware  noarch  20180113-60.git65b1c68.el7  ol7_developer_UEKR5

Transaction Summary
================================================================================
Install  1 Package
```

```
Upgrade             ( 1 Dependent package)

Total size: 108 M
Is this ok [y/d/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : linux-firmware-20180113-60.git65b1c68.el7.noarch       1/3
  Installing : kernel-uek-4.14.26-1.el7uek.x86_64                     2/3
  Cleanup    : linux-firmware-20170606-58.gitc990aae.el7_4.noarch     3/3
  Verifying  : kernel-uek-4.14.26-1.el7uek.x86_64                     1/3
  Verifying  : linux-firmware-20180113-60.git65b1c68.el7.noarch       2/3
  Verifying  : linux-firmware-20170606-58.gitc990aae.el7_4.noarch     3/3

Installed:
  kernel-uek.x86_64 0:4.14.26-1.el7uek

Dependency Updated:
  linux-firmware.noarch 0:20180113-60.git65b1c68.el7

Complete!
```

After the installation on a clean CentOS system where all past kernel updates have been removed, the following kernel packages will be present:

```
# rpm -qa | grep ^kernel | sort
kernel-3.10.0-693.21.1.el7.x86_64
kernel-devel-3.10.0-693.21.1.el7.x86_64
kernel-headers-3.10.0-693.21.1.el7.x86_64
kernel-tools-3.10.0-693.21.1.el7.x86_64
kernel-tools-libs-3.10.0-693.21.1.el7.x86_64
```

```
kernel-uek-4.14.26-1.el7uek.x86_64
```

Reboot to this kernel. The GRUB menu entry will appear as:

```
CentOS Linux (4.14.26-1.el7uek.x86_64) 7 (Core)
```

After booting into CentOS-uek, run the spectre-meltdown-checker.sh script by Stephane Lesimple. It should confirm retpoline support:

```
# ./spectre-meltdown-checker.sh
Spectre and Meltdown mitigation detection tool v0.35

Checking for vulnerabilities on current system
Kernel is Linux 4.14.26-1.el7uek.x86_64 #2 SMP
 Tue Mar 13 01:14:49 PDT 2018 x86_64
CPU is Intel(R) Core(TM)2 Duo CPU     E6550  @ 2.33GHz

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available:  NO
    * CPU indicates IBRS capability:  NO
  * Indirect Branch Prediction Barrier (IBPB)
    * PRED_CMD MSR is available:  NO
    * CPU indicates IBPB capability:  NO
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available:  NO
    * CPU indicates STIBP capability:  NO
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability:  NO
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability:  NO
  * CPU explicitly indicates not being vulnerable to Meltdown (RDCL_NO):
     NO
```

```
  * CPU microcode is known to cause stability problems:
     NO  (model 15 stepping 11 ucode 0xba)
* CPU vulnerability to the three speculative execution attacks variants
  * Vulnerable to Variant 1:  YES
  * Vulnerable to Variant 2:  YES
  * Vulnerable to Variant 3:  YES

CVE-2017-5753 [bounds check bypass] aka 'Spectre Variant 1'
* Mitigated according to the /sys interface:
     YES  (kernel confirms that the mitigation is active)
* Kernel has array_index_mask_nospec:
     YES  (1 occurrence(s) found of 64 bits array_index_mask_nospec())
* Kernel has the Red Hat/Ubuntu patch:  NO
> STATUS:  NOT VULNERABLE  (Mitigation: __user pointer sanitization)

CVE-2017-5715 [branch target injection] aka 'Spectre Variant 2'
* Mitigated according to the /sys interface:
     YES  (kernel confirms that the mitigation is active)
* Mitigation 1
  * Kernel is compiled with IBRS/IBPB support:  YES
  * Currently enabled features
    * IBRS enabled for Kernel space:  UNKNOWN
    * IBRS enabled for User space:  UNKNOWN
    * IBPB enabled:  UNKNOWN
* Mitigation 2
  * Kernel compiled with retpoline option:  YES
  * Kernel compiled with a retpoline-aware compiler:
     YES  (kernel reports full retpoline compilation)
> STATUS:  NOT VULNERABLE  (Mitigation: Full generic retpoline)

CVE-2017-5754 [rogue data cache load] aka 'Meltdown' aka 'Variant 3'
* Mitigated according to the /sys interface:
     YES  (kernel confirms that the mitigation is active)
```

```
* Kernel supports Page Table Isolation (PTI):  YES
* PTI enabled and active:  YES
* Running as a Xen PV DomU:  NO
> STATUS:  NOT VULNERABLE  (Mitigation: PTI)

A false sense of security is worse than no security at all,
 see --disclaimer
```

On the day of this article's submission (March 15, 2018), Oracle issued a new UEKr4 including retpoline support. Production Oracle Linux has now completely mitigated Meltdown and Spectre:

```
# ./spectre-meltdown-checker.sh
Spectre and Meltdown mitigation detection tool v0.35

Checking for vulnerabilities on current system
Kernel is Linux 4.1.12-112.16.4.el7uek.x86_64
     #2 SMP Mon Mar 12 23:57:12 PDT 2018 x86_64
CPU is Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available:  YES
    * CPU indicates IBRS capability:  YES  (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * PRED_CMD MSR is available:  YES
    * CPU indicates IBPB capability:  YES  (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available:  YES
    * CPU indicates STIBP capability:  YES
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability:  NO
```

```
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability:  NO
  * CPU explicitly indicates not being vulnerable to Meltdown (RDCL_NO):
  * NO
  * CPU microcode is known to cause stability problems:
     NO  (model 45 stepping 7 ucode 0x713)
* CPU vulnerability to the three speculative execution attacks variants
  * Vulnerable to Variant 1:  YES
  * Vulnerable to Variant 2:  YES
  * Vulnerable to Variant 3:  YES

CVE-2017-5753 [bounds check bypass] aka 'Spectre Variant 1'
* Mitigated according to the /sys interface:
     YES  (kernel confirms that the mitigation is active)
* Kernel has array_index_mask_nospec:  NO
* Kernel has the Red Hat/Ubuntu patch:  YES
> STATUS:  NOT VULNERABLE  (Mitigation: lfence)

CVE-2017-5715 [branch target injection] aka 'Spectre Variant 2'
* Mitigated according to the /sys interface:
     YES  (kernel confirms that the mitigation is active)
* Mitigation 1
  * Kernel is compiled with IBRS/IBPB support:  YES
  * Currently enabled features
    * IBRS enabled for Kernel space:  NO
    * IBRS enabled for User space:  NO
    * IBPB enabled:  YES
* Mitigation 2
  * Kernel compiled with retpoline option:  YES
  * Kernel compiled with a retpoline-aware compiler:
     YES  (kernel reports full retpoline compilation)
> STATUS:  NOT VULNERABLE
     (Mitigation: Full generic retpoline, IBRS_FW, IBPB)
```

```
CVE-2017-5754 [rogue data cache load] aka 'Meltdown' aka 'Variant 3'
* Mitigated according to the /sys interface:
     YES  (kernel confirms that the mitigation is active)
* Kernel supports Page Table Isolation (PTI):  YES
* PTI enabled and active:  YES
* Running as a Xen PV DomU:  NO
> STATUS:  NOT VULNERABLE  (Mitigation: PTI)

A false sense of security is worse than no security at all,
     see --disclaimer
```

The yum repo entry for the UEKr4 is below:

```
[ol7_UEKR4]
name=Latest Unbreakable Enterprise Kernel Release 4 for Oracle Linux
$releasever ($basearch)
baseurl=http://yum.oracle.com/repo/OracleLinux/OL7/UEKR4/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

The UEK is a popular choice in enterprise Linux, and it brings many new features to RPM distributions. It is the easiest and most stable method to implement a v4 performance-tuned kernel in a supported manner on a Red Hat-derivative OS (which usually runs v3.10 at best).

The UEK is especially notable as it works with Ksplice (as does Oracle's RHCK), which allows for online kernel patches and upgrades without rebooting, and Ksplice can even extend this ability to select userland libraries (glibc and OpenSSL). This service is offered for free to Fedora and Ubuntu users, and as a paid support option for both CentOS (after a scripted conversion to Oracle Linux) and Red Hat. Ksplice is the oldest and most mature high-availability option for Linux security upgrades.

Conclusion

While the UEKr5 remains in preview release status, it is likely not appropriate to use it on critical production servers. The UEKr4 is now a production option for those who require immediate remediation. Users on peer distributions should certainly begin testing the UEKr5 now if their timetables permit. Those with critical systems should consider commercial (paid) support and potentially implement Ksplice for high availability.

Many customers of peer distros would likely prefer to keep their systems "pure" from a single supplier. Still, the community has long been making use of EPEL, ElRepo, Nux and others. Oracle simply offers another repository with useful packages. The EPEL maintainer Fedora was launched as an organization by Red Hat; administrators who forbid third-party yum repositories likely do not understand the historical relationships between these organizations.

There is also occasional hostile sentiment toward Oracle for various reasons. Still, Oracle remains an important contributor to the Linux community, and its software meets the needs of a large base of customers. Quality software with useful features should not be rejected because of organizational bias.

In any case, this is a major accomplishment for Oracle Linux, and all members of the Red Hat community stand to benefit from these efforts regardless of their level of participation. It is unlikely that Oracle will forever maintain the technical advantages discussed here, but it has earned and deserve accolades for these advances.

Disclaimer

The views and opinions expressed in this article are those of the author and do not necessarily reflect those of *Linux Journal*. ■

Charles Fisher has an electrical engineering degree from the University of Iowa and works as a systems and database administrator for a Fortune 500 mining and manufacturing corporation.

How the EU's Copyright Reform Threatens Open Source—and How to Fight It

Open source is under attack from new EU copyright laws.

By Glyn Moody

Glyn Moody has been writing about the internet since 1994, and about free software since 1995. In 1997, he wrote the first mainstream feature about GNU/Linux and free software, which appeared in *Wired*. In 2001, his book *Rebel Code: Linux And The Open Source Revolution* was published. Since then, he has written widely about free software and digital rights. He has a blog, and he is active on social media: @glynmoody on Twitter or identi.ca, and +glynmoody on Google+.

Open Source and copyright are intimately related. It was Richard Stallman's clever hack of copyright law that created the General Public License (GPL) and, thus, free software. The GPL requires those who copy or modify software released under it to pass on the four freedoms. If they don't, they break the terms of the GPL and lose legal protection for their copies and modifications. In other words, the harsh penalties for copyright infringement are used to ensure that people can share freely.

Despite the use of copyright law to police the GPL and all the other open source licenses, copyright is not usually so benign. That's not surprising: copyright is an intellectual monopoly. In general, it seeks to prevent sharing—not to promote it. As a

result, the ambitions of the copyright industry tend to work against the aspirations of the Open Source world.

One of the clearest demonstrations of the indifference of the copyright world to the concerns of the Free Software community can be found in Europe. Proposals for reform of copyright law in the European Union contain one element that would have a devastating effect on open-source coding there. Article 13 of the grandly titled "Directive Of The European Parliament And Of The Council on copyright In The Digital Single Market" contains the following key requirement:

> Information society service providers that store and provide to the public access to large amounts of works or other subject-matter uploaded by their users shall, in cooperation with rightholders, take measures to ensure the functioning of agreements concluded with rightholders for the use of their works or other subject-matter or to prevent the availability on their services of works or other subject-matter identified by rightholders through the cooperation with the service providers. Those measures, such as the use of effective content recognition technologies, shall be appropriate and proportionate.

Translated into practical terms, this means that sites with major holdings of material uploaded by users will be required to filter *everything* before allowing it to be posted. The problems with this idea are evident. It represents constant surveillance of people's online activities on these sites, with all that this implies for loss of privacy. False positives are inevitable, not least because the complexities of copyright law cannot be reduced to a few algorithmic rules that can be applied automatically. That, and the chilling effect it will have on people's desire to upload material, will have a negative impact on freedom of expression and undermine the public domain.

The high cost of implementing upload filters—Google's ContentID system required 50,000 hours of coding and $60 million to build—means that a few big companies will end up controlling the market for censorship systems. Their oligopoly power potentially gives them the ability to charge high prices for their services, which will impose burdens on companies in the EU and lead to fewer online startups in the

region. Other problems with the idea include the important fact that it seems to go against existing EU law.

Article 13 has been drawn up mainly to satisfy the barely disguised desire of the European copyright industry to attack successful US companies like Google and Facebook. But the upload filter is a very crude weapon, and it will affect many others who—ironically—will be less able than internet giants to comply with the onerous requirement to censor. For example, it is likely that Wikipedia will be caught by the new rule. After all, it hosts huge amounts of "subject-matter" that is uploaded by users. As a post on the Wikimedia blog pointed out: "it would be absurd to require the Wikimedia Foundation to implement costly and technologically impractical automated systems for detecting copyright infringement."

But for readers of *Linux Journal*, perhaps the most troubling aspect of Article 13 is how it would affect open source. Another class of websites that "provide to the public access to large amounts of works or other subject-matter uploaded by their users" are collaborative software platforms and code repositories. As a site called Savecodeshare.eu, set up by the Free Software Foundation Europe and OpenForum Europe (full disclosure: I am an unpaid Fellow of the related OpenForum Academy), explains:

> As a result of this ongoing copyright review, every user of a code sharing platform, be they an individual, a business or a public administration, is to be treated as a potential copyright infringer: their content, including entire code repositories, will be monitored and blocked from being shared online at any time. This restricts the freedom of developers to use specific software components and tools that in return leads to less competition and less innovation. Ultimately this can result in software that is less reliable and a less resilient software infrastructure for everybody.

As a detailed white paper on the problem from the same organization explains, at risk are key websites, such as Software Heritage, GitHub, GitLab, GNU Savannah and SourceForge. It points out that under the proposed law, these sites would be directly responsible for a wide range of user actions, including uploading unauthorized copies

of software or other material and using code in ways that are not compliant with their original licenses. Since it will be hard if not impossible for many of these sites to prevent those things from occurring, it is quite likely some will block all access from European users and shut down their operations in the EU. Open-source projects may be forced to do the same. In the white paper, KDE board member Thomas Pfeiffer is quoted as saying:

> To what extent KDE is directly impacted by the proposed regulation fully depends on whether the above setup would make us "information society service providers storing and giving access to large amounts of works and other subject-matter uploaded by their users". If yes, then we would probably need to move most of our infrastructure and organization outside of the EU.

Figure 1. Services Affected by Article 13 (Image Courtesy of EDiMA)

Clearly, the potential impact of the Copyright Directive's Article 13 on the way open source is created around the world is considerable. The good news is that the new EU law has not yet been finalized and is still in a state of flux. The bad news is, it's not getting better.

For example, the politician overseeing the passage of the new law through the European Parliament has recently proposed an amendment that would exempt major sites from upload filters on the condition that they sign formal licensing agreements with all the copyright owners of the material they host. For code repositories, that's an impossible task given the volume of the files that are uploaded—there is no collecting society for coders, as there is for musicians or authors, say, to give blanket permission for all the hosted material. Any form of Article 13 with either upload filters or mandatory licensing is unworkable for key sites in the Open Source world.

Its fundamental flaws mean that Article 13 must be removed from the Copyright Directive. A multilingual site called Savethememe.net has been set up to make it easy to speak directly to members of the European Parliament, who will have a decisive vote on the proposals. Another course of action is to spread the word about how harmful Article 13 would be for the entire open-source ecosystem, and the negative knock-effects this would have on global innovation and society. The more that programmers are aware of the problem and make noise about it everywhere, the more the code repositories worst affected can point to the mounting global concern about the impact of the upload filters, and the greater the impact of their calls to drop Article 13 completely. ■

Send comments or feedback
via http://www.linuxjournal.com/contact
or email ljeditor@linuxjournal.com.