LINUX JOURNAL™
Since 1994: The Original Magazine of the Linux Community

A Look at What's New in 3D PRINTING

JUNE 2015 | ISSUE 254 | www.linuxjournal.com

Develop a Multi-Container System with Docker and Weave

Inspect Network Traffic with tshark and Python

NETWORKING

PLUS

CAN YOU TRUST YOUR OPERATING SYSTEM?

DOING STUFF WITH DOCKER—HOW TO GET STARTED

MANIPULATE DATABASE RECORDS IN DJANGO

WATCH: ISSUE OVERVIEW

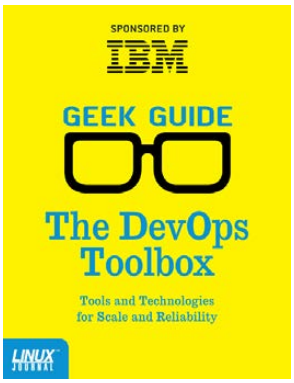
NEW!

Linux Journal eBook Series

GEEK GUIDES

FREE Download NOW!

The DevOps Toolbox:
Tools and Technologies for Scale and Reliability



By Bill Childers

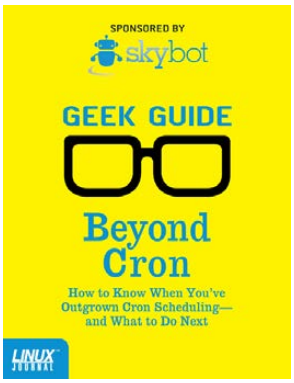
Introducing *The DevOps Toolbox: Tools and Technologies for Scale and Reliability* by Linux Journal Virtual Editor Bill Childers.

When I was growing up, my father always said, “Work smarter, not harder.” Now that I’m an adult, I’ve found that to be a core concept in my career as a DevOps engineer and manager. In order to work smarter, you’ve got to have good tools and technology in your corner doing a lot of the repetitive work, so you and your team can handle any exceptions that occur. More important, your tools need to have the ability to evolve and grow over time according to the changing needs of your business and organization.

In this eBook, I discuss a few of the most important tools in the DevOps toolbox, the benefits of using them and some examples of each tool. It’s important to not consider this a review of each tool, but rather a guide to foster thinking about what’s appropriate for your own organization’s needs.

Register today to receive your complimentary copy of The DevOps Toolbox:
http://linuxjournal.com/devops-toolbox-guide

Beyond Cron
How to Know When You’ve Outgrown Cron Scheduling—and What to Do Next



By Mike Diehl

If you’ve spent any time around UNIX, you’ve no doubt learned to use and appreciate cron, the ubiquitous job scheduler that comes with almost every version of UNIX that exists. Cron is simple and easy to use, and most important, it just works. It sure beats having to remember to run your backups by hand, for example.

But cron does have its limits. Today’s enterprises are larger, more interdependent, and more interconnected than ever before, and cron just hasn’t kept up. These days, virtual servers can spring into existence on demand. There are accounting jobs that have to run after billing jobs have completed, but before the backups run. And, there are enterprises that connect Web servers, databases, and file servers. These enterprises may be in one server room, or they may span several data centers.

Register today to receive your complimentary copy of Beyond Cron:
http://linuxjournal.com/beyond-cron-guide

---

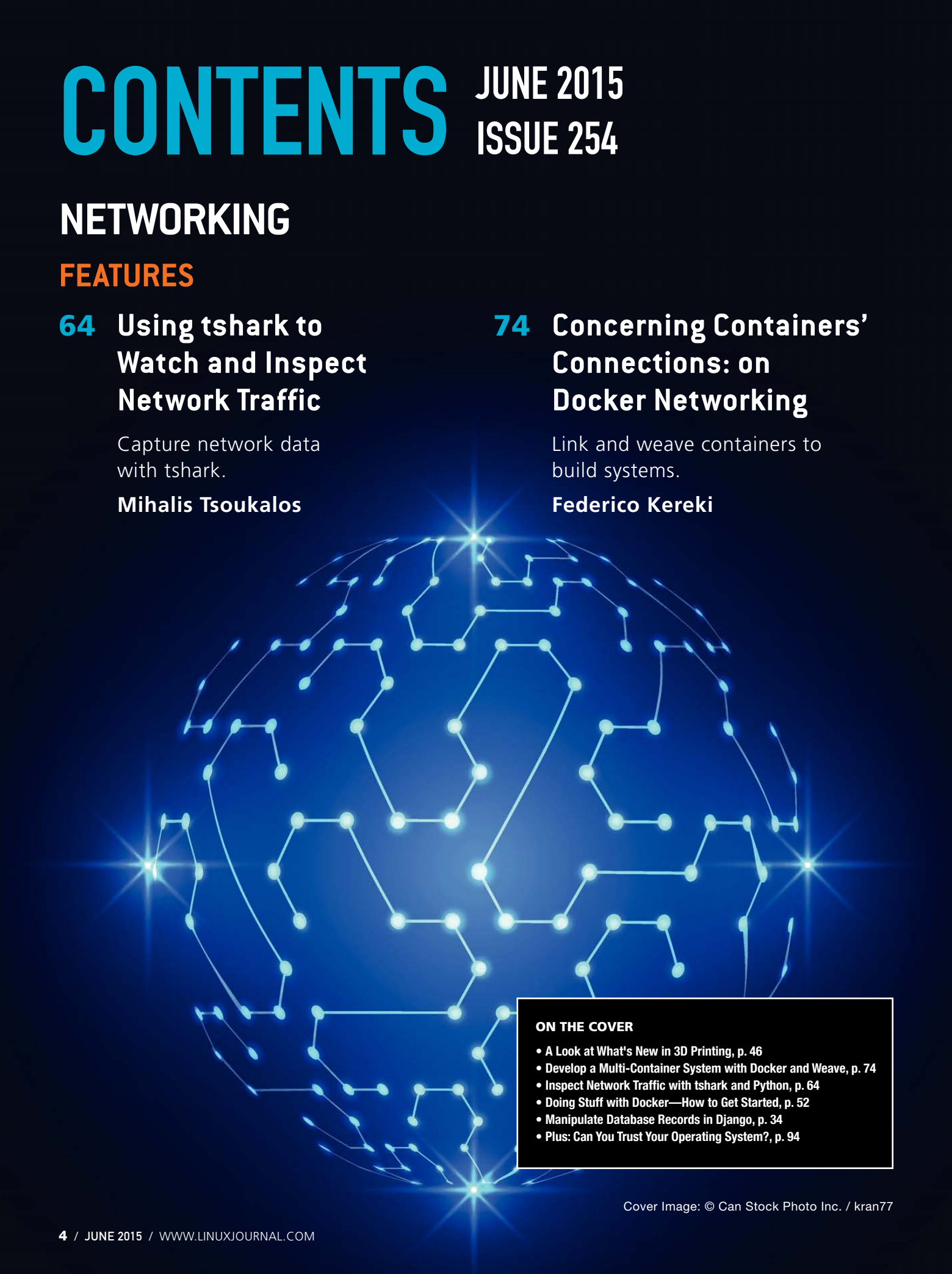
http://linuxjournal.com/geekguides

CONTENTS

JUNE 2015
ISSUE 254

NETWORKING

FEATURES

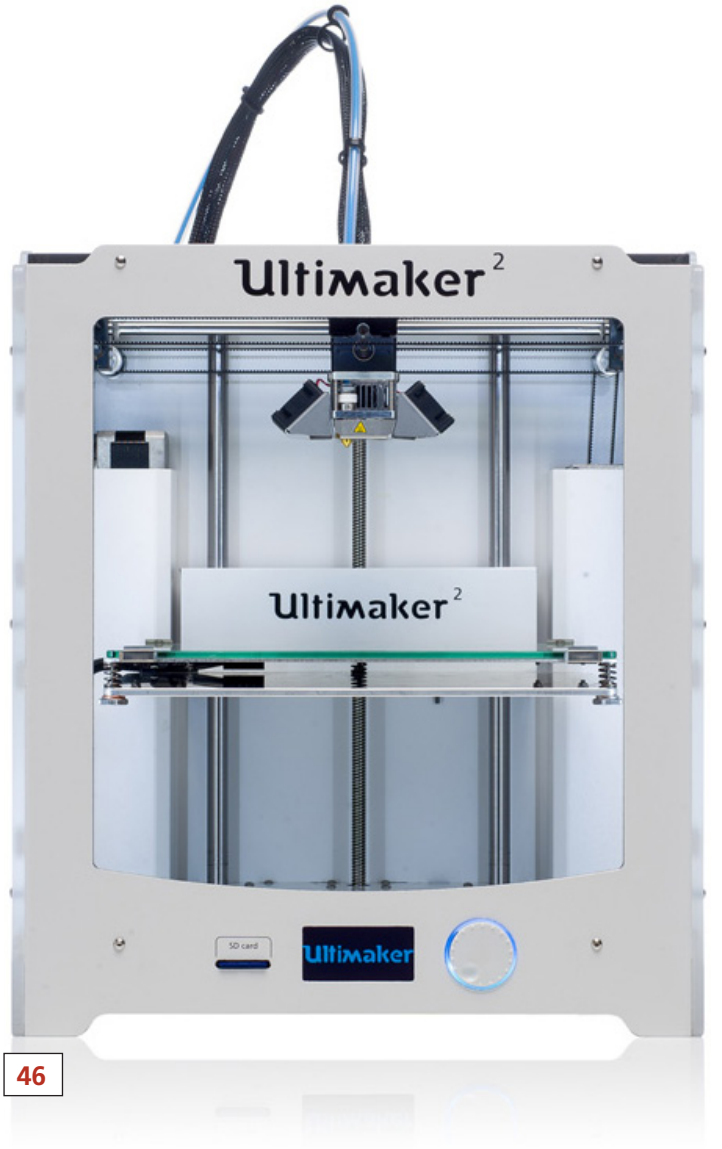


ON THE COVER

COLUMNS

IN EVERY ISSUE

22



46

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., 2121 Sage Road, Ste. 395, Houston, TX 77056 USA. Subscription rate is $29.50/year. Subscriptions start with the next issue.

LINUX JOURNAL

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

Contributing Editors

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

President	Carlie Fairchild publisher@linuxjournal.com
Publisher	Mark Irgang mark@linuxjournal.com
Associate Publisher	John Grogan john@linuxjournal.com
Director of Digital Experience	Katherine Druckman webmistress@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

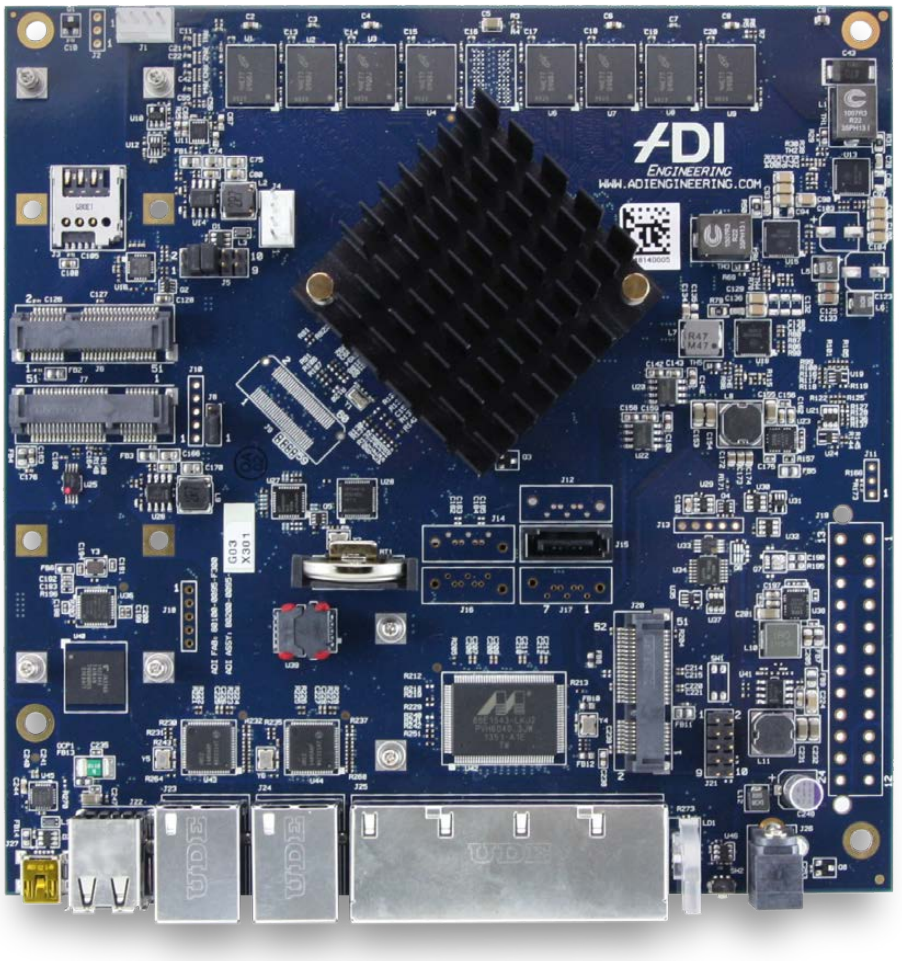
Linux Journal **is published by, and is a registered trade name of,**
Belltown Media, Inc.
PO Box 980985, Houston, TX 77098 USA

Advertising
E-MAIL: ads@linuxjournal.com
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

Subscriptions
E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
MAIL: PO Box 980985, Houston, TX 77098 USA

Network Communications Platform

Compact, low-power Intel® Atom™ C2000 family boards designed for Open Source



nanoITX from $198 miniITX from $254

RCC-VE 2440
C2358 2-core 1.7 GHz fanless
4GB on board DDR3L
4 Intel Gigabit Ethernet ports
4GB onboard flash

RCC-VE 4860
C2558 4-core 2.4 GHz fanless
8GB on board DDR3L
6 Intel Gigabit Ethernet ports
4GB onboard flash

RCC-VE 8860
C2758 8-core 2.4 GHz with fan
8GB on board DDR3L
6 Intel Gigabit Ethernet ports
64GB onboard flash

Other features including:
Intel QuickAssist
Desktop and 1U enclosures
1 x SATA
1 x mSATA socket
2 x miniPCIe, one with SIM

Quickstart instructions available for: CentOS®, Debian®, FreeBSD®, OpenBSD

Customize your NFV, Edge, Security, VoIP, CPE, wireless or other embedded appliance at:

Netgate®

7212 McNeil Drive Suite 204 Austin, TX 78729 +1.512.646.4100

SHAWN POWERS

Two Cups, One String

Whenever I watch episodes of *Battlestar Galactica*, it breaks my heart when they avoid Cylon hacking by disconnecting all networks. (It also annoys me how distorted the concept of networking is presented in the show, but I digress.) Anyone who has had their Wi-Fi go down knows just how much we depend on computer networks in our every day lives. In this issue, we focus on networking, because the most powerful computer in the world isn't nearly as awesome without the ability to search for cat videos on YouTube.

We start out with Reuven M. Lerner's column on Django models. No, that's not a runway of models showing off the new Django fashions; rather, Reuven explains how to manipulate database records from inside Django applications. He's followed by Dave Taylor, who delves into some insidious shell scripting issues that occur when dealing with data.

Kyle Rankin starts the first part of his series on 3D printing. Kyle has been interested in 3D printing since before it was "cool", and this month, you'll see how far the process has come in a few short years. 3D printing has hit the mainstream, and that's good news for those of us who were late to the game. We're way past the days of spending $1,000 to print a $0.17 plastic whistle, and Kyle brings us up to date.

My column this month is all about demystifying Docker. We've published some Docker articles in the past, but I never really understood Docker apart from what it was doing conceptually. I figured I probably wasn't alone, so I go through the process of learning how to use Docker in a project, and along the way I describe

VIDEO:
Shawn Powers runs through the latest issue.

Using Docker and Weave, Federico shows how to build complete systems of networked containers in a way that is efficient and easy to configure.

what it is, and how to use it. I hate intimidating technology, so hopefully this month we'll tame the Docker beast.

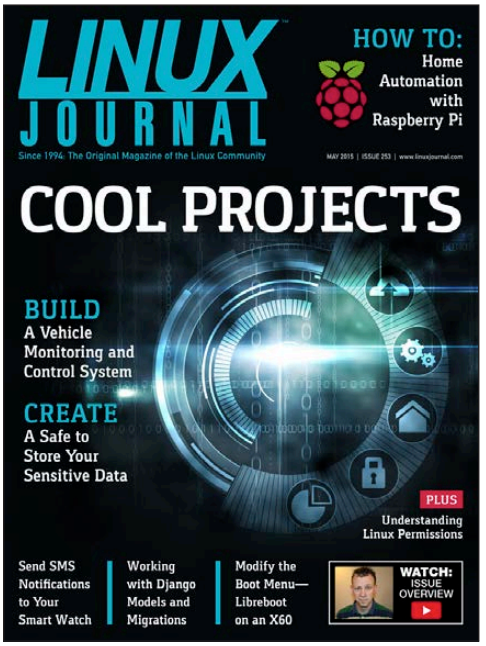
This is, of course, the networking issue. Federico Kereki follows my intro to Docker column with a more in-depth look at the networking aspect of containers. Using Docker and Weave, Federico shows how to build complete systems of networked containers in a way that is efficient and easy to configure. If you're not familiar with Docker, I recommend reading my column first, but you really don't want to miss Federico's awesome look at container-based networking.

Most networking-centric issues of a tech magazine would include an article on Wireshark—and rightly so. It's an amazing piece of software that does incredibly deep network inspections, and it does it all for free. Mihalis Tsoukalos covers tshark this month, which is a command-line version of Wireshark. Although tshark does all the same network capture and diagnoses of its brother, Wireshark, tshark does it without a GUI. This means that although you lose the pretty graphics, you gain the ability to script functionality and use tshark without the need to point and click. Mihalis shows how to store network capture data into MongoDB using all command-line tools.

Whether your networking prowess peaked when you tied two cups together with string or you pay your mortgage by managing a huge corporate network, this issue of *Linux Journal* should give you plenty of insight on all things network. We don't have any articles on how to keep Cylons out of your firewall, but apparently the only way to do that is to unplug your system anyway (sigh). This was a fun issue for us to put together, full of reviews, announcements, tech tips and more. We hope you enjoy it!■

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

letters



Legal to Open Word Documents?

Is it legal to open (patented) Microsoft Word documents if I have never bought a Microsoft-related product? Sure, OpenOffice.org will display such a document, but am I allowed to view it?

—Richel Bilderbeek

Good question. I'm not a lawyer (not even close), but it seems the problem would come with creating a Word document as opposed to reading it. So my follow-up question would be, is it safe for LibreOffice/OpenOffice to "save as" a Microsoft Word document? If we get any answers from lawyers, Richel, I'll try to follow up with an answer.—Shawn Powers

.NET?

Regarding Shawn Powers' "Non-Linux FOSS: .NET?" in the April 2015 UpFront section: it is not open source in the way that most people in the Linux world understand the term. A lot of this is similar to Sun's behavior regarding the Java programming language and JVM. Since the community has been burned by Oracle's attempt to reclassify retroactively what is open and what is not, one has to parse Microsoft's licensing terms carefully. In particular, if the .NET patents are transferred to another entity (even a third-party under Microsoft's control), some of the protections implied under the license become invalid. I am not a legal expert myself, but given recent developments with Oracle/Google, I am still wary of .NET. Like you, I think it's encouraging, but not enough to commit a serious project to the technology until it survives its first court challenge.

—Daniel Waites

Agreed. I always try to give the benefit of the doubt, but it still gives me the willies. Hopefully my paranoia is unfounded, but I think caution is wise.—Shawn Powers

Thank You

Hello everyone at *Linux Journal*! I am simply writing to say "Thank You." I appreciate what you do every day. I'm sure it seems like a thankless job sometimes, but I like to point out when someone does something right. Lord knows, there are enough people in our lives that tell us when we do something wrong.

—Don Brown

You rock, Don! Thanks for the encouragement; it's greatly appreciated.—Shawn Powers

Scanning on Linux Distributions

Having recently purchased a multifunction device—printer, copier, scanner and fax machine—I was troubled to find that scanners and connecting to and using them in Linux is really bothersome, if you can get them to work at all!

I realize that many Linux users, and readers of the excellent *LJ*, are mostly using it for development, exploration and trying esoteric software. That's great, but what about real-world users of the platform? I bought a Xerox 6505 mfd, and printing, (using Ubuntu) is simple, but scanning just does not work at all! I installed SANE, following the instructions on the Ubuntu site, and nothing! Yes, someone will say that Xerox is not on the "supported" list, but not being supported is not very helpful. Fortunately, I have a MacBook laptop, and all works well on MAC OS X. Why can't Linux be as easy to use?

—Alan Lewis

Ugh, I feel your pain. While printing has come a long way, you're right, scanning is still painful. I don't have a great answer, other than take comfort in the fact that you're not alone in your frustration (see Alan McConnell's letter below).—Shawn Powers

Pipes and Xargs

Great magazine (even though there is no longer a dead-tree version).

Regarding Shawn Powers' "Pipes and STDs" article in the April 2015 issue, the `xargs` example: it was a good example, but if I were using `find`, I think I would also use `find`'s `exec` instead of `xargs`:

```
find / -name "*.mp3' -exec rm '{}' \;
```

—Richard

Me too! In fact, when I used it as an example, I had to shake my head a little because like most things in Linux, there's always another way. The `-exec` *flag is a perfect example, and it would have been easier. I was just trying to come up with a simple example that would demonstrate how* `xargs` *works. Thank you for mentioning the other option though. It's a much more efficient way to accomplish the task.—Shawn Powers*

Canon Support for Scanning

I think it is time that the Linux community rise in its wrath. After trying a bunch of printers, including a couple HPs and Brothers, I finally got a Canon, because the Canon people gave good support to Linux for its printing capability. I had assumed that they would support Linux in their scanning function as well. I was wrong. Take a look at the message below. It is a response to the message I sent to Canon (that message is shown after the Canon reply below):

> Dear Alan McConnell:
>
> Thank you for contacting Canon product support. Scanning is not supported using Linux.
>
> Sincerely, Technical Support Representative

And here is the text of my original message to Canon:

> I have an MF4770n, bought two weeks ago. It prints fine. Now I need to get its scan function to work.
>
> I am running Linux, Debian Wheezy. I have libsane backends 1.0.24. When I run `sane-find-scanner`, it returns `found USB scanner (vendor=0x04a9, product=0x2774) at libusb:003:004`. However, `scanimage -L` returns: `no scanners were identified`.
>
> When I run `man sane-pixma`, I get a list of the models that work with this backend. Among them is imageCLASS MF4770n.
>
> I hope you can tell me what further to do, in order to use this MF4770n as a scanner.
>
> Thank you in advance.

I consider that I have bought this "all in one" printer under false pretenses. I don't know what recourse I have—it is past the

deadline to return it—except to publicize this problem and hope that other Linux users can be persuaded to get involved with this issue. Linux is no longer a toy to be brushed aside!
—Alan McConnell

Alan, I have nothing to add other than "GRRRRRRRR!" I share your frustration, and wish I had a solution.—Shawn Powers

Teeny Tiny Tablet?

Regarding Shawn Powers' "The Teeny Tiny $20 Tablet" article in the March 2015 issue: thanks for the article about the Boost LG LS620. I own such a device too, and I haven't even rooted the phone yet. All I have to do is download NoRoot Firewall (https://play.google.com/store/apps/details?id=app.greyshirts.firewall&hl=en), and after each reboot, I have to turn Wi-Fi on and off for one or two seconds and wait for the time-out of the activation. After this, I can turn on Wi-Fi until I have to reboot. I've disabled all preloaded apps that I don't need, and I always have Airplane Mode turned on. So, I don't see why you have tried to remove the cellular radio icon. This phone even supports 64GB Micro SD cards if you just reformat it with the phone or install an exfat driver (needs a rooted phone).

Why do you use Google Maps for off-line routing? I am using OSMAND (https://play.google.com/store/apps/details?id=net.osmand&hl=en). You can download ten Open Street Map maps for free or buy OSMAND+ if you need more. There are other apps, but I prefer open-source programs.
—Dirk Schwartzkopff

The radio icon thing is nothing more than my OCD driving me nuts. I've never had luck with off-line GPS apps on Android, but I'll give OSMAND a try. Thanks!—Shawn Powers

find|xargs

This is regarding Dave Taylor's article on `find|xargs` in the January 2015 issue: indeed, it's a combination of two really powerful commands that can do wonders in tandem. However, I wish to point out that find has a mini-xargs built in to it. If you terminate your `exec` clause with + (instead of ;), it will behave much like `xargs`.
—Mayuresh

Dave Taylor replies: *I have to admit, I didn't know that handy tip. Thanks, Mayuresh. Is it even documented?*

It's definitely in the category of "more fun with Linux".

Mayuresh replies: Yes Dave. Quoting "man find"

> If the list of arguments is terminated by a plus sign (+), then the pathnames for which the primary is evaluated are aggregated into sets, and utility will be invoked once per set, similar to xargs(1).

Indeed, it is in the "more fun with Linux" category.

WRITE *LJ* A LETTER

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

PHOTO OF THE MONTH

Remember, send your Linux-related photos to ljeditor@linuxjournal.com!

InterMapper
A Division of HelpSystems

STRUGGLING TO FIND THE RIGHT NETWORK MONITORING SOFTWARE?

Maybe it's time for a checklist.

Choosing the right network monitoring software can be a daunting task. A quick list of features can be invaluable in helping you determine what capabilities are on your "must-have" list. If you're looking for a powerful, top-of-the-line monitoring solution, you need a tool that offers:

- **Constant device polling and real-time statistics**
- **In-depth mapping with highly granular detail and Layer 2 visibility**
- **Proactive alerting to notify you of potential issues before they cause outages**

Use our interactive checklist to compare InterMapper with other monitoring solutions.

START COMPARING

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

When you run a program as **setuid**, it runs with all the permissions of that user. And if the program spawns new processes, they inherit the same permissions. Not so with filesystem capabilities. When you run a program with a set of capabilities, the processes it spawns do not have those capabilities by default; they must be given explicitly.

This seemed unintuitive to **Christoph Lameter**, who posted a patch to change **capability inheritance** to match the behavior of setuid inheritance. This turned out to inspire some controversy.

For one thing, filesystem capabilities never were defined fully in the **POSIX** standard and appear only in a draft version of POSIX that later was withdrawn, so there can't really be any discussion of whether one form of capabilities is "more compliant" than another.

There are other problems, such as the need to make sure that any changes to capabilities don't break existing code and the need to make sure that any ultimate solution remains secure.

One problem with Christoph's idea was that it tied capability inheritance to the file itself, but as **Serge Hallyn** pointed out, capabilities were tied to both the file and the user executing the file. Ultimately, Christoph decided to adapt his code to that constraint, introducing a new capability that would list the inheritable capabilities available for the user to apply to a given file.

Yalin Wang recently made an abortive effort to have **/proc/stat** list all CPUs on a given system, not just the on-line ones. This would be a very useful feature, because many modern systems bring CPUs on- and off-line at a rapid pace. Often the number of CPUs actually in use is less important than the number available to be used.

He posted a patch to change /proc/stat accordingly, and **David Rientjes** pointed out that the **/sys/devices/cpu** file would be a better location for this. **Andrew Morton** also pointed out that **/proc/cpuinfo** would be a good location for this kind of data as well. So, there definitely was some support for Yalin's idea.

Unfortunately, it turned out that some existing code in the **Android** kernel

relied on the current behavior of those files—specifically desiring the number of on-line CPUs as opposed to the total number of CPUs on the system. With an existing user dependent on the existing behavior, it became a much harder sell to get the change into the kernel. Yalin would have to show a real need as opposed to just a sensible convenience, so his patch went nowhere.

John Stultz has been maintaining some timekeeping test patches on GitHub for several years now, and he finally wanted to get them into the kernel, so he could stop porting them forward continually. The test would do a variety of things, involving changing the system time in some way designed to induce a problem. He asked what he should do to make the patches acceptable to the kernel folks.

There were a bunch of generally supportive comments from folks like **Richard Chochran** and **Shuah Khan**, but Shuah requested some fairly invasive changes that would tie John's code to other testing code in the kernel. John said he'd be happy to

do that if it was required, but that one of his goals had been to keep the test files isolated, so any one of them could run independently of anything else on the system. In light of that, Shuah withdrew her suggestion.

Overall, it's not a controversial set of patches, and they'll undoubtedly get into the kernel soon.

One problem with making backups that guarantee filesystem consistency is that files on the system may change while they're being backed up. There are various ways to prevent this, but if another process already has an open file descriptor for a file, backup software just has to wait or risk copying an inconsistent version of the file.

Namjae Jeon posted some patches to address this problem by implementing **file freezing**. This would allow backup software to block writes to a given file temporarily, even if that file already had been opened by another process.

In addition to backup software, other tools like defragmenting software would benefit from Namjae's patches by preventing any changes to a file that was being reorganized on disk.

As **Jan Kara** pointed out, however, Namjae's code had some potential race conditions as well as other technical problems. **Dave Chinner** described the code as "terribly racy".

It's not clear what will happen with these patches. They seem to offer features that folks want, but the race conditions need to be resolved, and the code needs to be clean and clear enough that future fixes and enhancements will not be too likely to introduce new problems.—**ZACK BROWN**

They Said It

For to be free is not merely to cast off one's chains, but to live in a way that respects and enhances the freedom of others.
—Nelson Mandela

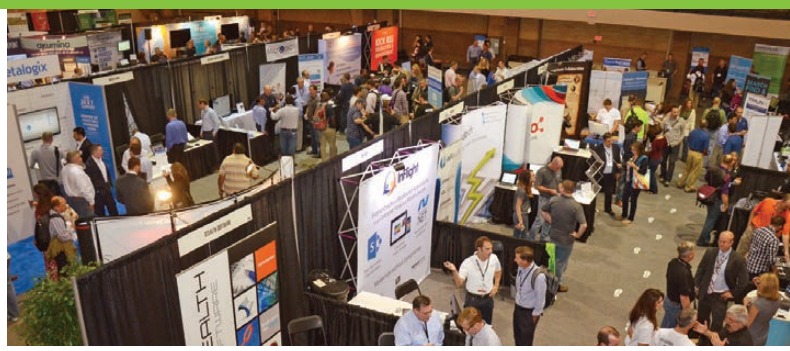
For the things we have to learn before we can do them, we learn by doing them.
—Aristotle

Words are a heavy thing...they weigh you down. If birds talked, they couldn't fly.
—Sy Rosen

You don't become great by trying to be great. You become great by wanting to do something, and then doing it so hard that you become great in the process.
—Randall Munroe

It's not the hours you put in your work that counts, it's the work you put in the hours.
—Sam Ewing

Learn what's new in SharePoint and Office 365!



SPTechCon
The SharePoint Technology Conference

August 24-27, 2015

BOSTON

Over 70 classes taught by expert speakers!

"This was a great conference that addresses all levels, roles and abilities. Great variety of classes, great presenters, and I learned many practical things that I can take back and start implementing next week."
—Kathy Mincey, Collaboration Specialist, FHI 360

SharePoint in the Cloud? On Premises? Or Both?

Come to SPTechCon Boston 2015 and learn about the differences between Office 365, cloud-hosted SharePoint, on-premises SharePoint, and hybrid solutions and build your company's SharePoint Roadmap!

Looking for SharePoint 2013 training?

Check out these targeted classes!

- Custom SharePoint 2013 Workflows that Use the SharePoint 2013 REST API
- SharePoint 2013 Farm Architecture and Visual Studio for Admin
- Creating a Branded Site in SharePoint 2013
- SharePoint's New Swiss Army Knife: The Content Search Web Part

Moving to Office 365?

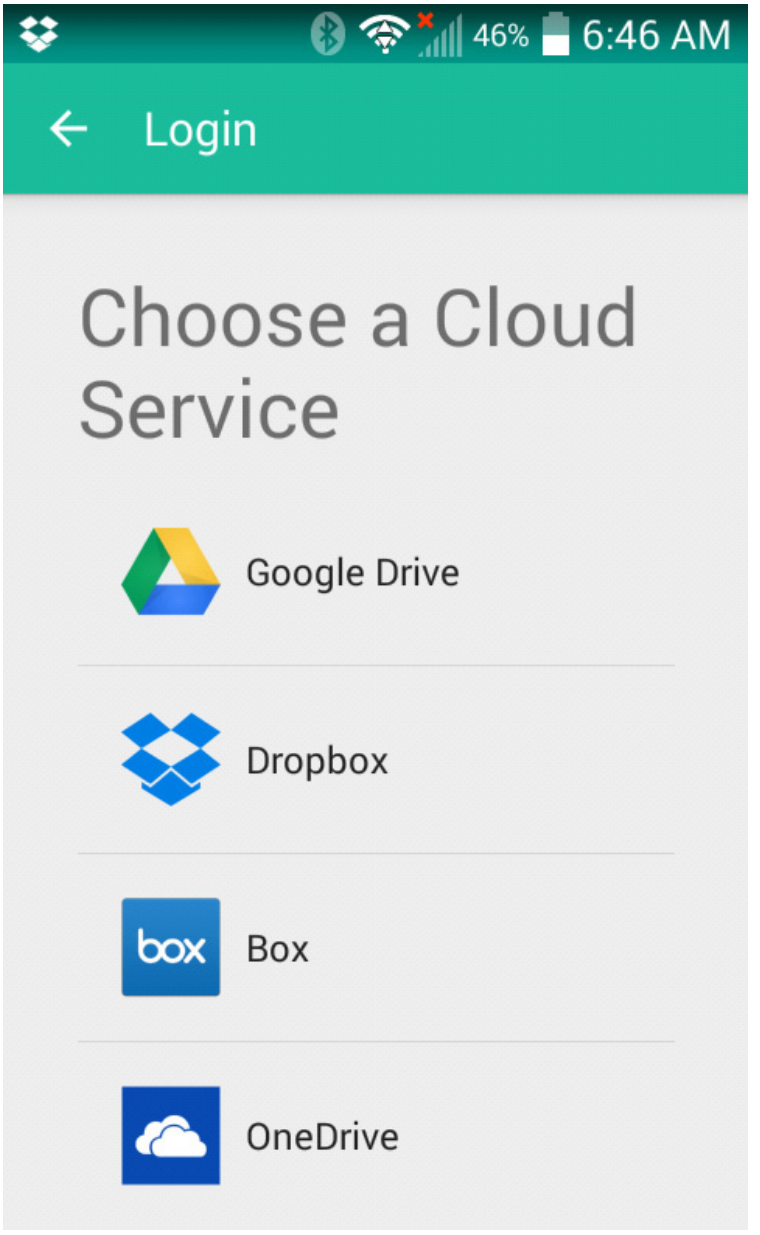
Here are some targeted classes for YOU!

- Baby-Stepping Into the Cloud with Hybrid Workloads
- Demystifying Office 365 Administration
- Document Management and Records Management for Office 365
- Office 365 Search in the Cloud

MASTER THE PRESENT, PLAN FOR THE FUTURE! REGISTER NOW! → www.sptechcon.com

A BZ Media Event

Android Candy: Cloud Bonding



Although the title might sound like some new-fangled tech jargon, I'm actually referring to a fairly simple Android app called "Unclouded." If you're a Dropbox user who also has things stored in Google Drive, Unclouded is a single interface to multiple file syncing backends. Sure, it's not horribly difficult to open multiple apps to work with your cloud-based files, but it can be inconvenient.

Unclouded also has some neat features like locating duplicate files taking up precious space in your cloud storage, and it can do previews for most media types. Unfortunately, the free version limits the number of accounts you can connect to, and it also disables useful things like sharing, moving, renaming and deleting files. Thankfully, the premium features are a few bucks at the most, and even without them, the app is elegant and useful. Check it out today at the Google Play Store: https://play.google.com/store/apps/details?id=com.cgollner.unclouded.

—SHAWN POWERS

Take your Android development skills to the next level!

Whether you're an enterprise developer, work for a commercial software company, or are driving your own startup, if you want to build Android apps, you need to attend AnDevCon!

AnDevCon
The Android Developer Conference

July 29-31, 2015
Sheraton Boston

Right after Google IO!

- Choose from more than 75 classes and in-depth tutorials
- Meet Google and Google Development Experts
- Network with speakers and other Android developers
- Check out more than 50 third-party vendors
- Women in Android Luncheon
- Panels and keynotes
- Receptions, ice cream, prizes and more (plus lots of coffee!)

Android is everywhere!
But AnDevCon is where you should be!

"There are awesome speakers that are willing to share their knowledge and advice with you."
—Kelvin De Moya, Sr. Software Developer, Intellisys

"Definitely recommend this to anyone who is interested in learning Android, even those who have worked in Android for a while can still learn a lot."
—Margaret Maynard-Reid, Android Developer, Dyne, Inc.

Register Early and Save at www.AnDevCon.com

A BZ Media Event #AnDevCon

The AtoMiC Toolkit!

If you're a cord cutter (and a nerd), you most likely have a server or two dedicated to serving and possibly retrieving videos from the Internet. Programs like Kodi and Plex are awesome for media delivery; however, there's more to a complete system than just playing the videos. Although the ethical and legal ramifications vary from country to country (and conscious to conscious), the unfortunate truth is that programs like Sickbeard and NZBDrone can be difficult to install and maintain.

The folks over at http://www.htpcbeginner.com created a set of Bash scripts designed to make the installation of media-related HTPC software painless. If applied to a freshly installed Ubuntu machine, the AtoMiC Toolkit installs the appropriate dependencies and software for most of the media-related software out there.

Like I always say when this topic comes up, using torrents and Usenet to download television episodes may not be legal where you live. Regardless of where you live, it might be ethically wrong to download them. Even if you just use programs like Sickbeard to organize the television shows you record with your own DVR, however, the AtoMiC Toolkit is a great way to get them up and running in short order. Check out the scripts at https://github.com/htpcBeginner/AtoMiC-ToolKit and learn how to install them at http://www.htpcbeginner.com/atomic-toolkit.—**SHAWN POWERS**

REGISTER TODAY!

24th USENIX Security Symposium

AUGUST 12–14, 2015 • WASHINGTON, D.C.

The USENIX Security Symposium brings together researchers, practitioners, system administrators, system programmers, and others interested in the latest advances in the security of computer systems and networks. The Symposium will include a 3-day technical program with more than 65 refereed paper presentations, invited talks, posters, a panel discussion on security and privacy research ethics, and Birds-of-a-Feather sessions. Featured speakers/sessions include:

Keynote Address by Richard Danzig, *member of the Defense Policy Board, The President's Intelligence Advisory Board, and the Homeland Security Secretary's Advisory Council*

Invited Talk: "Using Formal Methods to Eliminate Exploitable Bugs" by Katherine Fisher, *Tufts University*

Invited Talk: "Machine vs. Machine: Lessons from the First Year of Cyber Grand Challenge" by Mike Walker, *DARPA*

Invited Talk: "Preventing Security Bugs through Software Design" by Christopher Kern, *Google*

The following co-located events will precede the Symposium on August 10–11, 2015:

3GSE '15: 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education

CSET '15: 8th Workshop on Cyber Security Experimentation and Test

FOCI '15: 5th USENIX Workshop on Free and Open Communications on the Internet

HealthTech '15: 2015 USENIX Summit on Health Information Technologies
Safety, Security, Privacy, and Interoperability of Health Information Technologies

HotSec '15: 2015 USENIX Summit on Hot Topics in Security

JETS '15: 2015 USENIX Journal of Election Technology and Systems
(Formerly EVT/WOTE)

WOOT '15: 9th USENIX Workshop on Offensive Technologies

www.usenix.org/sec15

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Stay Connected...

twitter.com/USENIXSecurity
www.usenix.org/youtube
www.usenix.org/gplus
www.usenix.org/facebook
www.usenix.org/linkedin
www.usenix.org/blog

Physics Analysis Workstation

CERN is the European Laboratory for Particle Physics. It has been in the news quite a bit lately with the discovery of the Higgs Boson at the Large Hadron Collider. Something that many people may not know is that it also has a long tradition of developing software for scientific use. The HTML document format and the first browser both were developed there as a way of using rich documents that could include links between many different sources of information. It was so useful, it ended up sparking the World Wide Web. Along with such widespread software, CERN has been responsible for quite a bit of scientific software, especially physics software.

In this article, I take a look at a fairly large group of modules and libraries called the Physics Analysis Workstation (PAW, paw.web.cern.ch/paw). PAW contains several thousand subroutines and programs that are written in FORTRAN, C and even some assembly language code, which is built on top of a library called the CERN Program Library (CERNLIB).

You can download and install the code from the source located at the main Web site if you have any special needs, but considering the long list of required external libraries, I suggest you avoid that if possible. Packages should be available for your distribution. For Debian-based distros, you can install everything you need with the command:

```
sudo apt-get install paw
```

PAW also includes a large series of graphing and data visualization routines to help in data analysis. Sometimes you need to see what your data looks like in order to figure out what further analysis you need to investigate.

PAW actually is an interactive system, where you can apply commands against your data set. The original interface was a command-line one, but it now has collected several other interfaces that you can try out. If you open a terminal, type the command `paw` and press Enter, you are presented with a question as to which terminal type you want to use (Figure 1). The default is to use type 1, which opens an HIGZ graphic window where your plots will be displayed (Figure 2). If you are using PAW on a remote machine, you

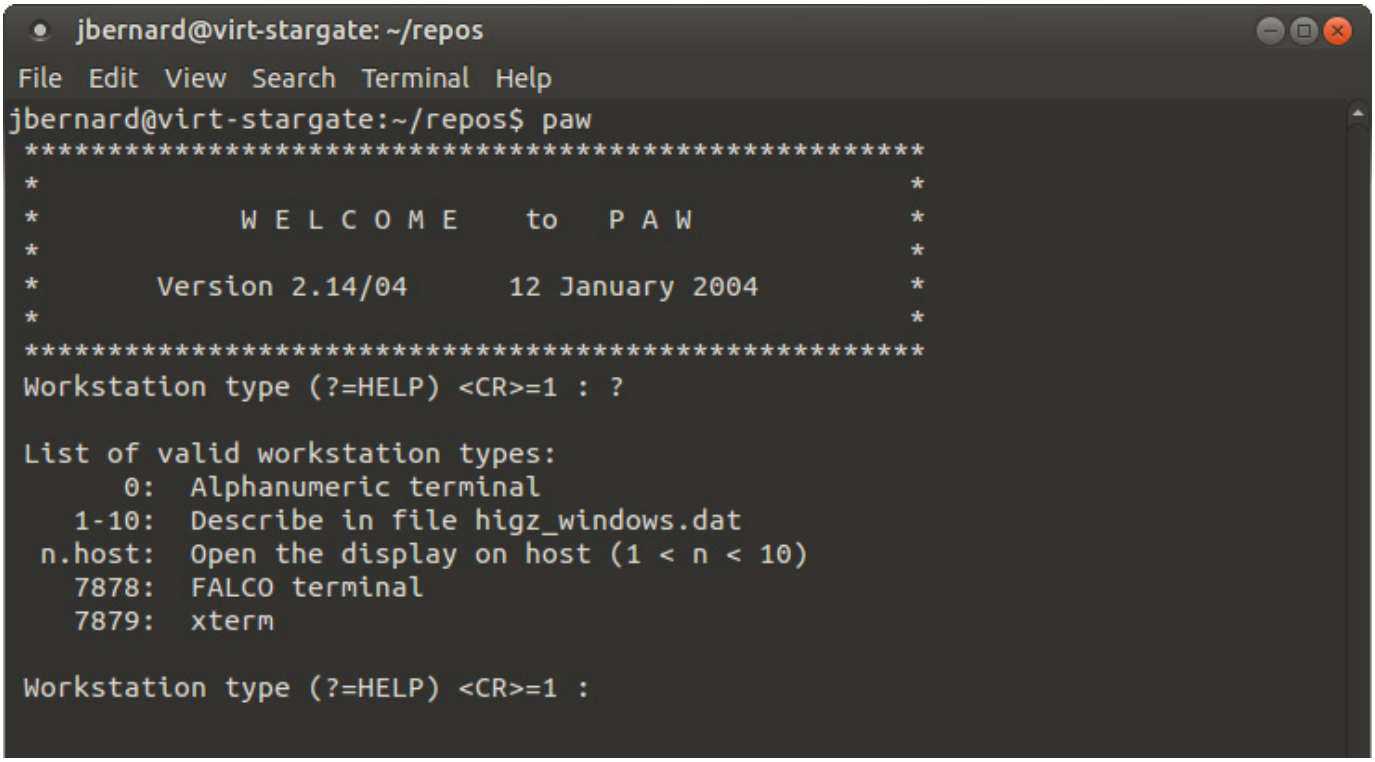


Figure 1. You can select the terminal type to use when you start PAW.

probably will want to use a different type. You can get a list by typing `?`. For a regular xterm, enter `7879`.

Once everything has finished loading, you are presented with a prompt that looks like this:

```
PAW >
```

Now you can start typing commands and doing data analysis. But, what commands can you use? Luckily, PAW includes a help system within the program that you can access by typing the `help` command, which pops up a list of topics.

Commands in PAW are grouped together in a tree structure, with the top-most level being the topics that pop up when you start the help system. There is also quite a bit of documentation available on the main Web site, including tutorials and a very large FAQ.

Because PAW is used for data analysis, let's start with what kinds of data you can use. PAW has three main data types: VECTORS, HISTOGRAMS and NTUPLES. VECTORS store arrays of reals or integers. PAW can handle up to three dimensions, or indexes, for these VECTORS. They can be manipulated by the group of VECTOR commands. Commands

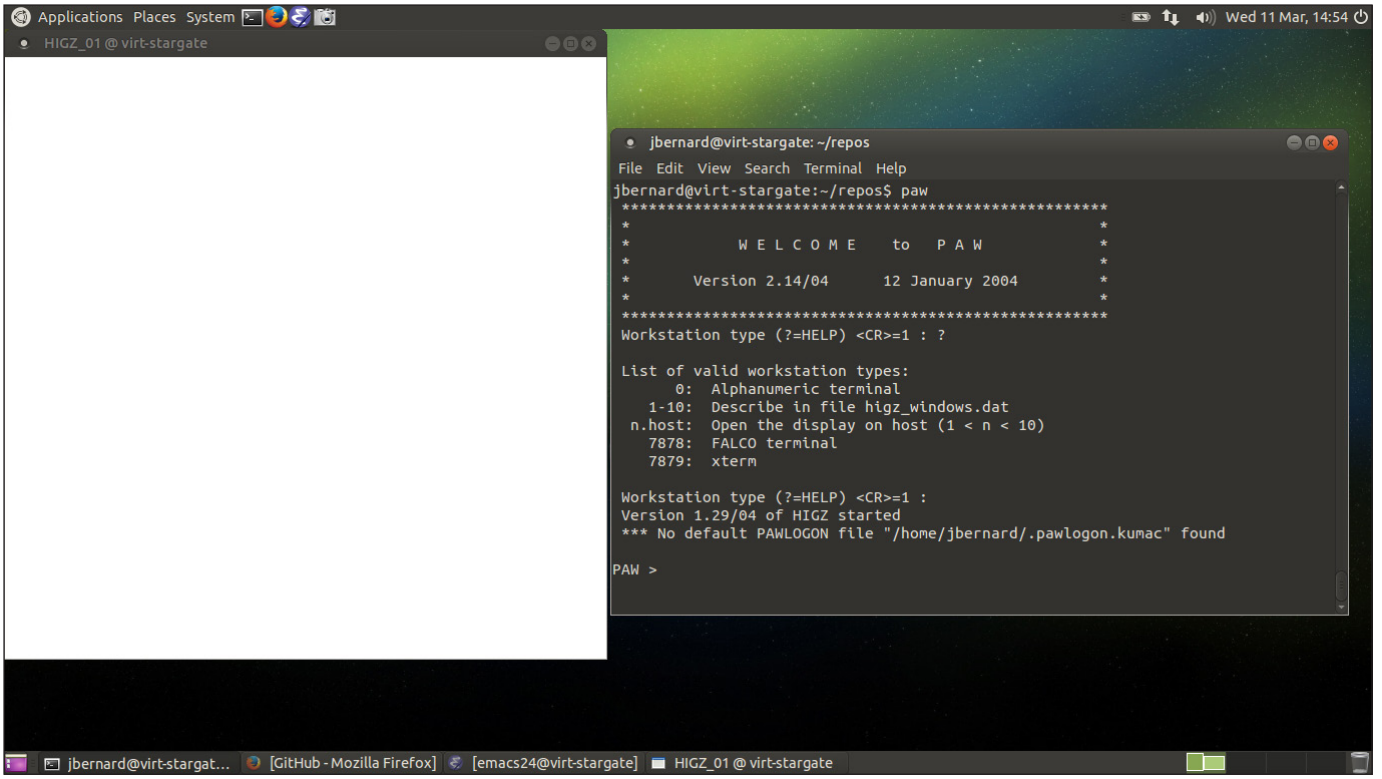


Figure 2. The default is to open a graphics window to draw your plots into, along with a command interface.

in PAW are not case-sensitive, but in most documentation, they are shown in uppercase. You also can use abbreviations for commands, as long as they can be matched uniquely to the full command text. So, you can create a new VECTOR of 20 elements with the command:

```
VECTOR/CREATE  vec1(20)
```

This new VECTOR is named "vec1". Then you can add elements to your new vector with this command:

```
VECTOR/INPUT vec1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

The command takes a vector name and a list of values to add. This is fine if you are dealing with just a small set of data. If you have larger data sets stored in files, you can use the command `VECTOR/READ`. This command takes a filename, and it also can take several other options, like the format of the elements, and loads the data into the given VECTORS.

The optional format string is similar to those used in reading and writing data in FORTRAN code, so a refresher course may be a good idea if it has been some time since you have used FORTRAN.

You can output data to a file with the inverse VECTOR/WRITE command.

To visualize your data, use the VECTOR/DRAW command. The options available allow you to select whether to draw a histogram, a smooth curve or a bar chart. You also can draw this visualization over the top of another graph.

You can get a list of all of the VECTORS that have been created with the VECTOR/LIST command, and you can clean up unneeded data with the VECTOR/DELETE command.

Once you have loaded your data and taken a look at it, you may have an idea of how the different parts are related to each other. You can use the VECTOR/FIT command to take a function, defined by you with a subroutine, and try to fit the data to it. You also can include a set of associated errors when issuing the command.

The HISTOGRAM group of commands within PAW gives you a larger selection of plotting and analysis tools to apply to your data. The commands are broken down into subgroups that give you commands to create histograms, 2D plots and apply histogram operations to histograms. You can use the GET_VECT and PUT_VECT command subgroups to interact with the VECTOR object that you created above. You also can use FUNCTION commands to create functions that are used in commands that do data fitting, among other areas.

The NTUPLE group of commands are used to manipulate ntuple objects. Ntuples essentially are lists of lists, and you can think of them as matrices. In the PAW documentation, each row is called an event, and each column is called a variable. There are functions to merge data together or make cuts of subsets. Ntuples have their own plot commands that allow you to plot different variables against each other in various forms. If you have lots of data to deal with, you can use the CHAIN command to chain together multiple ntuples to create data sets of essentially unlimited size.

Although PAW is no longer under active development, there still is more than enough really useful code here to keep any scientist busy. If you are doing any work involving data analysis or modeling, especially in C or FORTRAN, it would be well worth your time to do a quick search of the available modules and subroutines in PAW to see if there is anything you can use to make your work progress more quickly. I cover only a very small portion of the functionality available in this article, so be sure to do a bit of a deeper dive to see what you can mine for your own work.

—JOEY BERNARD

Gettin' Sticky with It

In last month's issue, I talked about Linux permissions (see "It's Better to Ask Forgiveness..." in the May 2015 UpFront section). I could have covered SUID, GUID and sticky bit in the same article, but it seemed like a lot to cover in one sitting. So in this article, I describe the special permissions on a Linux system. Where standard permissions are fairly intuitive, the special permissions don't make a lot of sense at first. Once you understand what they do, however, they're really not too complicated.

But There's No Room for More Permissions! When you learned to set read, write and execute bits on files and folders, you probably realized that you used all the available "spots" for permissions. So when manipulating special permissions, you sort of re-use existing permission bits. It functions just like any other permission attribute, but they're represented a bit oddly.

Every section of the permissions string (user, group, other) has an additional "special" permission bit that can be set just like `rwx`. The indication for whether those bits are set is shown on the execute section of the string. For example:

- If the SUID (Set User ID) permission is set, the execute bit on the user section shows an s instead of an x.
- If the GUID (Group User ID) permission is set, the execute bit on the group section shows an s instead of an x.
- If the sticky bit is set, the execute bit on the other section shows a t instead of an x.

Confused yet? Here are a few examples:

- `-rwsrw-rw-` — SUID is set on this file.
- `drw-rwsrw-` — GUID is set on this folder.
- `drw-rw-r-t` — sticky bit is set on this folder.
- `-rwSr--r--` — SUID is set on this file, *but* the user execute bit is not.

Note that in the last example the S is uppercase. That's the way you can tell whether the execute bit underneath is set. If the SUID bit is

lowercase, it means the execute bit is set. If it's uppercase, it means the SUID bit is set, but the executable bit is not.

What Do They Do? Unlike standard permissions, special permissions change the way files and folders function, as opposed to controlling access. They also function differently depending on whether they're assigned to files or folders. Let's take a look at them one at a time.

SUID: the SUID bit is applied to executable programs. Once it is set, the program executes with the permissions and abilities of the user who owns the file. As you can imagine, this can be an enormous security risk! If a file is owned by root and has the SUID bit set, anyone who executes it has the same permissions as the root user. As scary as it sounds, there are a few valid use cases for such things. One perfect example is the `ping` program. In order to access the network hardware required to ping hosts, a user needs to have root access to system. In order for all users to be able to use `ping`, it's set with the SUID bit, and everyone can execute it with the same system permission that root has. Check it out on your system by typing `ls -l /bin/ping`. You should see the SUID bit set!

Setting the SUID bit on folders has no effect.

GUID: the GUID set on executable files has a similar effect to SUID, except that instead of using the permissions of the user who owns the file, it executes with the permissions of the group membership. This isn't used very often, but in certain multi-user environments, it might be desirable.

Mainly, GUID is used on a folder. If the GUID bit is set on a folder, files created inside that folder inherit the same group membership of the folder itself. This is particularly useful in group collaborations. Normally when someone creates a file, it has the group membership of that user's primary group. Inside a GUID folder, the user still owns the file, but the group membership is set automatically so others in the group can access the files.

Sticky bit: first off, I have no idea why the sticky bit is represented by a t instead of an s. I've searched high and low, and asked many people. No one seems to know. Maybe a *Linux Journal* reader knows the answer and will enlighten me. (If so, I'll include it in the Letters to the Editor section.) Anyway, the sticky bit is another special permission that is used on folders. In fact, it has no effect at all

if it's set on a file.

Folders that have the sticky bit set add a layer of protection for files created within them. Normally in a folder accessible by multiple people, anyone can delete anyone else's files. (Even if they don't have write access to the files!) With the sticky bit set, only the user who owns the file can delete it. It seems like a subtle thing, but when you consider a folder like the /tmp folder on a multi-user Linux system, you can see how important the sticky bit can be! In fact, if it weren't for the sticky bit, the /tmp folder on your system would be like the Wild Wild West, and nefarious gunslingers could delete other people's files willy nilly. You can see the sticky bit set on your system by typing `ls -l / | grep tmp`.

Assigning Special Permissions

Applying the special permissions to a file or folder is exactly like assigning regular permissions. You use the chmod tool—for example:

- `chmod u+s file.txt` — adds the SUID permission to file.txt.
- `chmod g-s file.txt` — removes the GUID permission from file.txt.
- `chmod o+t folder` — adds the sticky bit to the "folder" directory.

Special permissions can be assigned right alongside regular permissions as well, so things like this are perfectly fine:

```
chmod ug+rw,u+s,ugo-x file.txt
```

And just like standard permissions, it's possible (and often preferable) to assign special permissions using octal notation. In order to do that, you use the fourth field. When assigning permissions like this:

```
chmod 755 file.txt
```

there's a fourth field that if left off, is assumed to be zero. So this is actually the same as the above example:

```
chmod 0755 file.txt
```

That preceding zero is the field that assigns special permissions. If you leave it off, it's assumed to be zero, and no special permissions are assigned. Knowing it's there, however, should make it fairly easy to understand how to use it. If you read last month's article on permissions that included understanding octal notation, just apply that concept to special permissions. Figure 1 shows how it breaks down.

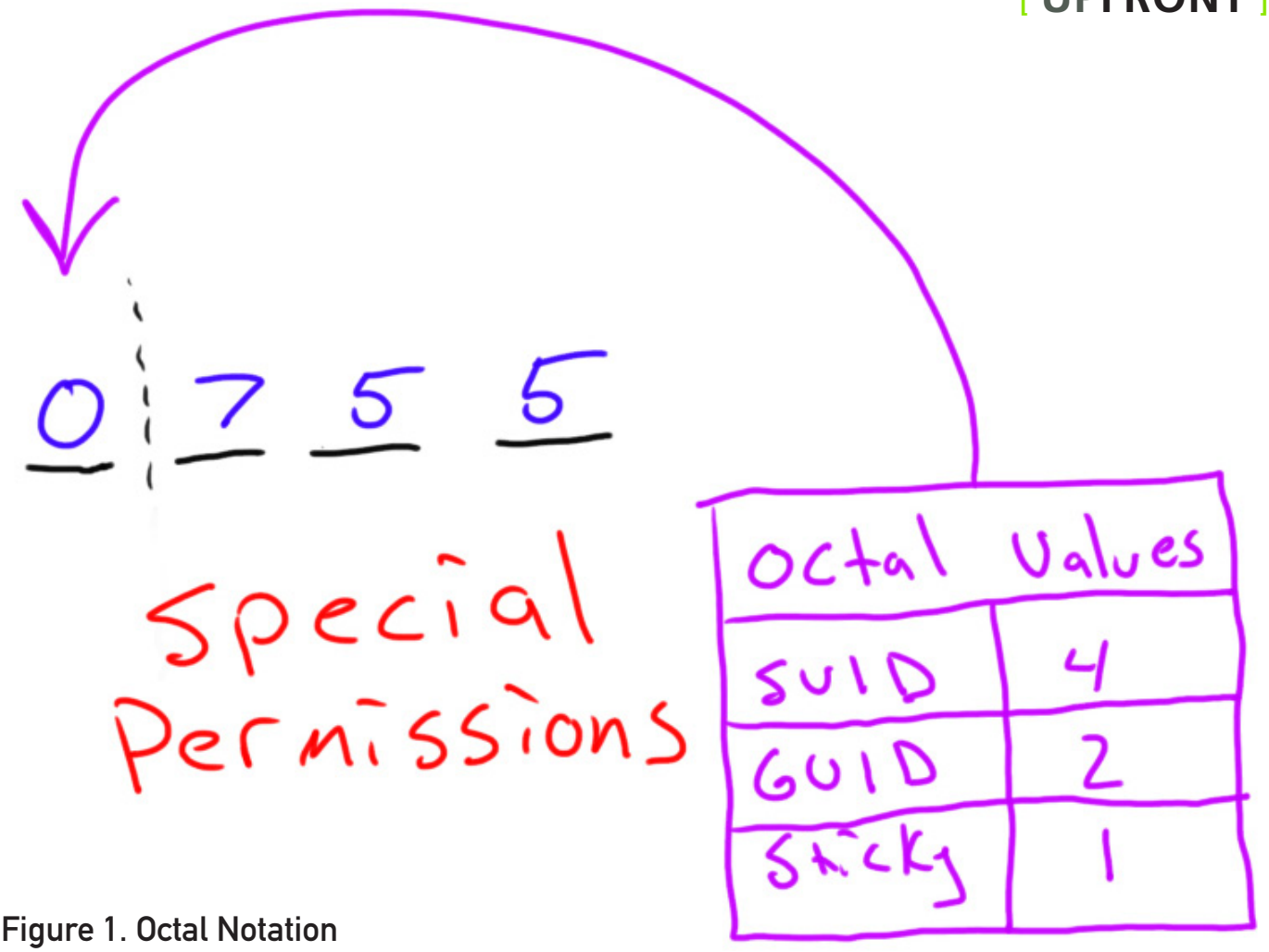


Figure 1. Octal Notation

So in order to assign a folder read/write access for user and groups along with the GUID bit, you would type:

```
chmod 2770 foldername
```

And, the resulting permission string (seen by typing `ls -l`) would show the following (note the lowercase s— remember what that means?):

```
drwxrws--- foldername
```

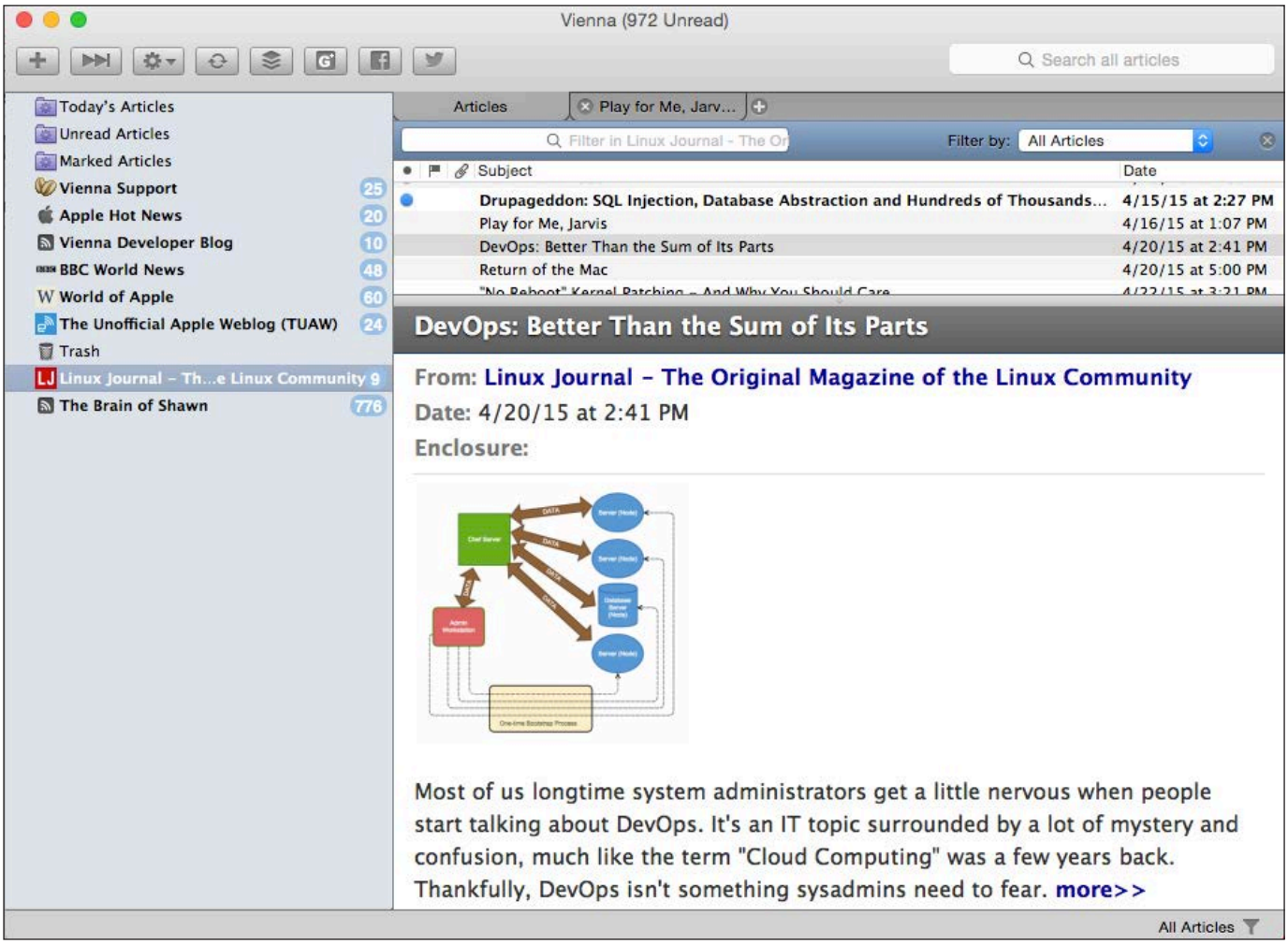
Just like standard permissions, if you want to set multiple special permissions, you just add the values. In order to set SUID and sticky bit, you would set the fourth octal field to 5. Usually, only a single special permission is set on any particular file or folder, but with octal notation, you have the option to set them in any way you see fit.

Hopefully these two articles clear up any misconceptions about Linux permissions. More complicated access controls are available with ACLs, but for most use cases, the standard permission strings are all you need to control access to files and folders on your system.

—SHAWN POWERS

Non-Linux FOSS: Vienna, Not Just for Sausages

Although the technology itself has been around for a while, RSS is still the way most people consume Web content. When Google Reader was ended a few years back, there was a scramble to find the perfect alternative. You may remember my series of articles on Tiny Tiny RSS, Comma Feed and a handful of other Google Reader wannabes. I don't



mention standalone RSS readers very often, however, because I don't like being tied to a single computer for reading Web sites. That's where syncing comes into play.

Vienna (http://www.vienna-rss.org) is an open-source RSS feed reader for OS X. Because it's written in Cocoa, it's available only for Macs. There are many alternatives for Linux and Windows, but the RSS reader options for OS X are surprisingly few.

The interface for Vienna is about like what you'd expect from an RSS reader. The view is customizable, and you can open complete stories in tabs to see the original Web site if you so desire. The real beauty of Vienna, however, is under the hood.

The Open Reader API (http://rss-sync.github.io/Open-Reader-API/rssconsensus) is a protocol that aims to be vendor-neutral and completely open. Google Reader used to be the back end that everyone used for RSS feed syncing, and since its demise, people sort of re-invented the wheel in their own way. The Open Reader API is one solution that may catch on. It's already supported by BazQux (http://bazqux.com) and FeedHQ (http://feedhq.org), and although adoption has been slow, hopefully it becomes the standard protocol for RSS syncing.

Luckily, if you're an OS X user, you can take advantage of the protocol right now with Vienna. Thanks to its great interface and open attitude, Vienna gets this month's Editors' Choice award. I think it's the first time we've given a non-Linux program the Editors' Choice honor, but its great interface and commitment to open standards makes us proud.

—SHAWN POWERS

REUVEN M. LERNER

Django Models

How to read, write and manipulate your database records in Django.

In my last article, I continued looking at the Django Web framework, showing how you can create and modify models. As you saw, Django expects you to describe your models using Python code. The model description is then transformed into SQL and compared with any previous version of the model that might have existed. Django then creates a "migration", a file that describes how you can move from one version of the model definition to the next. A migration is a fantastic tool, one that allows developers to move their database forward (and backward) in defined chunks. Migrations make it easier to collaborate with others and upgrade existing applications.

The thing is, migrations have little or nothing to do with the day-to-day application that you want to run. They are useful for the creation and maintenance of your application's models, but in your application, you're going to want to use the models themselves.

So in this article, I look at Django's ORM (object-relational mapper). You'll see how how Django allows you to perform all the traditional CRUD (create-read-update-delete) actions you need and expect within your application, so that you can use a database to power your Web application.

For the purposes of this article, I'll be using the "atfapp" application within the "atfapp" project that I created in last month's article. The model, of an appointment calendar, is defined

The easiest and best way to get your hands dirty with Django models is to use the Django interactive shell—meaning, the Python interactive shell within the Django environment.

as follows in atfapp/models.py:

```
class Appointment(models.Model):
    starts_at = models.DateTimeField()
    ends_at = models.DateTimeField()
    meeting_with = models.TextField()
    notes = models.TextField()
    minutes = models.TextField()
    def __str__(self):
        return "{} - {}: Meeting with {}
        ➥({})".format(self.starts_at,
                       self.ends_at,
                       self.meeting_with,
                       self.notes)
```

As you can see, the above model has four fields, indicating when the meeting starts, ends, with whom you are meeting and notes for before the meeting starts. The first two fields are defined to be DateTime fields in Django, which is translated into an SQL TIMESTAMP time in the database.

Creating a New Appointment

The easiest and best way to get your hands dirty with Django models is to use the Django interactive shell—meaning, the Python interactive shell within the Django environment. Within your project, just type:

```
django-admin shell
```

and you'll be placed in the interactive Python interpreter—or if you have it installed, in IPython. At this point, you can start to interact with your project and its various applications. In order to work with your Appointment object, you need to import it. Thus, the first thing I do is write:

```
from atfapp.models import Appointment
```

This tells Django that I want to go into the "atfapp" package—and since Django applications are Python packages, this means the "atfapp" subdirectory—and then import the "Appointment" class from the models.py module.

The important thing to remember is that a Django model is just a Python

class. The ORM magic occurs because your class inherits from models.Model and because of the class attributes that you use to define the columns in the database. The better you understand Python objects, the more comfortable you'll feel with Django models.

If you want to create a new appointment object, you can do what you normally would do with a Python object:

```
>>> a = Appointment()
```

Sure enough, if you ask "a" about itself, it'll tell you:

```
>>> type(a)
atfapp.models.Appointment
```

The first thing you might try to do is save your new appointment to the database. You can do this with the "save" method:

```
>>> a.save()
```

However, as you'll quickly discover if you try to do this, you get an exception—an `IntegrityError`, as the exception is named, which looks like this:

```
IntegrityError: NOT NULL constraint failed:
atfapp_appointment.starts_at
```

Here, Django is mixing Python and SQL to tell you what went wrong. You defined your model such that it requires a `starts_at` column, which is translated into a `NOT NULL` constraint within the database. Because you have not defined a `starts_at` value for your appointment object, your data cannot be stored in the database.

Indeed, if you simply get the printed representation of your object, you'll see that this is the case:

```
>>> a
<Appointment: None - None: Meeting with  ()>
```

The above output comes from the `__str__` instance method, which you can see was defined above. The new object has `None` values for `starts_at`, `ends_at` and `meeting_with`. Note that you don't have `None` values for `meeting_with` and notes. That's because the former are defined as `DateTimeField`, whereas the latter are defined as `TextField`.

By default, Django models are defined such that their columns in the database are `NOT NULL`. This is a good thing, I think. `NULL` values cause all sorts of problems, and it's better to have to name them explicitly. If you want a field to allow `NULL` values, you need to pass the

`null=True` option, as in:

```
starts_at = models.DateTimeField(null=True)
```

However, I'm not interested in `NULL` values for starting and ending times. Thus, if you want to store your appointment, you'll need to supply some values. You can do that by assigning to the fields in question:

```
>>> from datetime import datetime
>>> a.starts_at = datetime.now()
>>> a.ends_at = datetime(2015, 4, 28, 6,43)
```

Once you've done that, you can save it:

```
>>> a.save()
```

Another way to create your model would be to pass the parameters at creation time:

```
>>> b = Appointment(starts_at=datetime.now(),
        ends_at=datetime.now(),
        meeting_with='VIP', notes='Do not be late')
```

Reading Your Appointment Back

Now that you have two appointments, let's try to read them back and see what you can do with them. Access to the objects you have created in the database is done through the "objects" attribute, known as a "manager" in Django. The "all" method on objects gives you all of your objects back:

```
>>> len(Appointment.objects.all())
2
```

You can use your column names as attributes on each object:

```
>>> for a in Appointment.objects.all():
    print "{}: {}".format(a.starts_at, a.notes)

2015-04-28 05:59:21.316011+00:00:
2015-04-28 07:14:07.872681+00:00: Do not be late
```

`Appointment.objects.all()` returns an object known in Django as a QuerySet. A QuerySet, as you can see above, is iterable. And, if you call `len()` on it, or even if you ask for its representation (for example, in the Python shell), you'll see it displayed as a list. So you might think that you're talking about a list here, which potentially means using a great deal of memory.

But, the Django development folks have been quite clever about things, and a QuerySet is actually an iterator—meaning that it tries as hard as possible not to retrieve a large number of records into memory at once, but to use "lazy loading" to wait until the information is

truly needed. Indeed, just creating a QuerySet has no effect on the database; only when you actually try to use the QuerySet's objects does the query run.

It's nice to be able to get all of the records back, but what's even more useful and important is to be able to select individual records and then to order them.

For this, you can apply the "filter" method to your manager:

```
>>> for a in Appointment.objects.filter(meeting_with='VIP'):
        print a.starts_at
```

Now you know when your appointments with a VIP will be starting. But, what if you want to search for a range of things, such as all of the appointments since January 1st, 2015?

Django provides a number of special methods that perform such comparisons. For each field that you have defined in your model, Django defines `__lt`, `__lte`, `__gt` and `__gte` methods that you can use to filter query sets. For example, to find all of the appointments since January 1st, 2015, you can say:

```
>>> Appointment.objects.filter(starts_at__gte=datetime(2015,1,1))
```

As you can see, because you have a `starts_at` field name, Django accepts a `starts_at__gte` keyword, which is turned into the appropriate operator. If you pass more than one keyword, Django will combine them with AND in the underlying SQL.

QuerySets can be filtered in more sophisticated ways too. For example, you might want to compare a field with `NULL`. In that case, you cannot use the = operator in SQL, but rather, you must use the IS operator. Thus, you might want to use something like this:

```
>>> Appointment.objects.filter(notes__exact=None)
```

Notice that `__exact` knows to apply the appropriate comparison, based on whether it was given `None` (which is turned into SQL's `NULL`) or another value.

You can ask whether a field contains a string:

```
>>> Appointment.objects.filter(meeting_with__contains='VIP')
```

If you don't care about case sensitivity, you can use `icontains` instead:

```
>>> Appointment.objects.filter(meeting_with__icontains='VIP')
```

Don't make the mistake of adding % characters to the front and back of the string for which you're searching.

Django will do that for you, turning the `icontains` filter parameter into an SQL `ILIKE` query.

You even can use slice notation on a QuerySet in order to get the effects of `OFFSET` and `LIMIT`. However, it's important to remember that in many databases, the uses of `OFFSET` and `LIMIT` can lead to performance issues.

Django, by default, defines an "id" field that represents a numeric primary key for each record stored. If you know the ID, you can search based on that, using the `get` method:

```
>>> Appointment.objects.get(pk=2)
```

If there is a record with this primary key, it'll be returned. If not, you'll get a `DoesNotExist` exception.

Finally, you also can sort the records that are returned using the `order_by` method. For example:

```
>>> Appointment.objects.filter
➥(starts_at__gte=datetime(2015,1,1)).order_by('id')
```

What if you want to reverse the ordering? Just preface the name of the column with a - sign:

```
>>> Appointment.objects.filter
➥(starts_at__gte=datetime(2015,1,1)).order_by('-id')
```

You can pass multiple arguments to `order_by` if you want to order (ascending or descending) by a combination of columns.

One nice feature of Django's QuerySets is that every call to `filter` or `order_by` returns a new QuerySet object. In this way, you can make your calls to `filter` all at once or incrementally. Moreover, you can create one QuerySet and then use that as the basis for further QuerySets, each of which will execute (when necessary) its query independently.

A big problem with creating dynamic queries is that of SQL injection—that users can, through the use of manipulation, force their own SQL to be executed, rather than what you intended. Using Django's QuerySets basically removes this threat, because it checks and appropriately quotes any parameters it receives before passing their values along to SQL. Really, there's no excuse nowadays for SQL injection to be a problem—please think twice (or three times) before trying to work around Django's safeguards.

Updating and Deleting

Updating the fields of a Django model is trivially easy. Modify one or more attributes, as you would with any other Python object and then save the

updated object. Here, I load the first (unordered) record from the database before updating it:

```
>>> a = Appointment.objects.first()
>>> a.notes = 'blah blah'
>>> a.save()
```

Note that if you change the "id" attribute and then save your object, you'll end up creating a new record in the database! Of course, you shouldn't be changing the "id" of an object in any event, but now you can consider yourself warned as well.

To delete an object, just use the `delete` method on the instance. For example:

```
>>> len(Appointment.objects.all())
2

>>> a = Appointment.objects.first()
>>> a.delete()

>>> len(Appointment.objects.all())
>>> 1
```

As you can see, in the above example, I found that there is a total of two records in my database. I load the first and then delete it. Following that call—no need for saving or otherwise approving this action—you can see that the record is removed.

Conclusion

In my next article, I'll finish this series on Django with a discussion of the different types of relationships you can have across different models. I'll look at one-to-one, one-to-many and many-to-many relationships, and how Django lets you express and work with each of them.■

Reuven M. Lerner is a Web developer, consultant and trainer. He recently completed his PhD in Learning Sciences from Northwestern University. You can read his blog, Twitter feed and newsletter at http://lerner.co.il. Reuven lives with his wife and three children in Modi'in, Israel.

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

Resources

The main site for Django is **http://DjangoProject.com**, and it has a great deal of excellent documentation, including a tutorial. Several pages are dedicated to QuerySets and how you can create and manipulate them.

Information about Python, in which Django is implemented, is at **http://python.org**.

DAVE TAYLOR

When Is a Script Not a Script?

Dave receives a half-written script from a reader and realizes it's easily replaced with `find`—or is it? The problem might be more subtle than it first appears.

I received a very interesting script from reader Jeremy Stent via e-mail, and our subsequent conversation is something other script writers should consider too.

First off, here's the script he sent in:

```
function recurse_dir()

{
        for f in * ; do
          if [ -d "${f}" ] ; then
            pushd "${f}"
            recurse_dir
            popd
          fi
        done
}

pushd ~/dir
recurse_dir
popd
```

It's an interesting little script, and in case you aren't sure what's going on, it basically is recursively stepping through a directory tree. It's not actually doing anything, not even pushing any output, just recursing.

Of course, it'd be easy to add output or commands, but I was a bit baffled about the script's purpose when I received it. It's also hard to understand why there are so many `pushd`/`popd` invocations as well.

The original e-mail message actually was about how to deal with tricky filenames that contain spaces or punctuation, but that's usually just managed by ensuring that every

time you reference a filename, you include quotes. Doing so breaks the "for" statement, however, as is easily understood if you think about the fact that Bash uses white space (space, tab) as the field separator (aka "FS").

So if the current directory contains a file called "hello world", the "for" loop will offer up values of the "f" variable "hello", then "world", both of which are invalid filenames. This is one of the many reasons Linux is really clumsy with modern filenames, whether they contain punctuation or white space.

Still, here's how I responded to the query e-mail:

> That's an interesting script you're trying to build. I'm not clear why you're using push/pop as you traverse the directories too. Why not just have `cd ${f}` followed by `cd ..` to get back up a level and simplify things?
>
> In terms of difficult filenames, yeah, Linux wasn't really written to deal with filenames that start with a dash, have a space or other punctuation. The best you can do is experiment to see if the commands you're using accept -- as a way to delineate that you're done with command arguments, and quote the directory names themselves, as you've done.

Where the entire dialog got interesting was with his response, when he finally explained what he was trying to do: "My end intent is to remove the execute bit from files in a directory tree. Running `rsync` from a Windows box sometimes sets execute on files that I do not want. I do not want to remove the execute bit from directories, so I write a script like this."

Ah, I realized what he was trying to do, and the answer is actually quite straightforward: use `find`, not a shell script.

In fact, the `find` command is more than capable of traversing a filesystem, identifying non-directory files and changing their permissions to remove an execute bit that's presumably erroneously set.

(I say "presumably erroneously set", because there are actually a number of situations where a non-directory should retain its execute permission, including any shell, Perl or Ruby script and any compiled program, whether written in C++, Pascal or Fortran. In fact, blindly removing execute permission is problematic across any large piece of the Linux filesystem.)

On the assumption that the writer does want to proceed by removing the executable permission on files in a subsystem of the file tree, it's easily

done with:

```
find . -type f -exec chmod -x {} ;
```

To understand that, start with the benign alternative:

```
find . -type f -exec echo {} ;
```

This simple invocation of `find` will give you a quick list of every non-directory file in the current directory and any subdirectory below.

If you do dig in to the `find` man page, don't be misled by one of the other predicates: `-perm` lets you test permissions, not change them. So if you wanted to limit things to only those files that were executable, `-perm +x` would make sense.

Sidetracks, We Have Sidetracks

This problem of trying to debug a complex shell script when a simple Linux command invocation will do the trick is not uncommon, and it's one of the challenges for all developers. Unless you're in a shell programming class, the goal is what should dictate the solution path, not the tools. In other words, just because you happen to have a desire to learn more about shell script programming, doesn't mean that it's always the best and smartest solution for a given challenge.

And in terms of the original script, here's an interesting variation: what if you used `find` to generate a list of all files, then probed to see if you could ascertain whether a given file is associated with program source code (for example, it finds "hello" and then tests to see if "hello.c" exists) or if it's a shell script (information obtainable through the `file` command)?

Here's my first stab at this:

```
for filename in $(find . -type f -print) ; do
 if [ -x $filename ] ; then
    echo "File $filename is executable:"
    if [ ! -z "$(file $filename | grep "shell script")" ] ; then
      echo "  It's okay, it appears to be a shell script."
    elif [ -f "${filename}.c" -o -f "${filename}.cxx" ] ; then
      echo "  It's okay, there's a corresponding source file."
    else
      echo "  >> might be erroneously marked executable."
    fi
  fi
done
```

You can see that I'm using the `find` command to generate a list of every file from the current spot in the filesystem downward, so if there are lots of directories, this might generate quite a list. If there are too many, the shell can complain that it has run out of buffer, but that's a problem I'll sidestep in the interest of simplicity.

To test whether the executable

file is a shell script, there are two basic output formats for the `file` command, as demonstrated here:

```
test.sh: POSIX shell script, ASCII text executable
test.sh: POSIX shell script text executable
```

In either case, a simple test for "shell script" does the trick, as you can see if you closely examine the second conditional statement.

To see if the executable file is associated with a C or C++ source file, those are tested against the ".c" and ".cxx" filename suffixes in the `elif` statement. Keep in mind that `-o` is a logical `OR`, so the test is literally "if the .c file exists OR the .cxx file exists".

A quick run of this script produces this output:

```
$ sh test.sh
File ./taylor-trust.pdf is executable:
  >> might be erroneously marked executable.
File ./hello is executable:
  It's okay, there's a corresponding source file.
File ./plus is executable:
  It's okay, there's a corresponding source file.
File ./test.sh is executable:
  It's okay, it appears to be a shell script.
```

You can see that the script has recognized correctly that test.sh is a shell script (the last file tested), that "hello" and "plus" are both associated with source files (one a C program and the other a C++ program), but that the file taylor-trust.pdf is probably erroneously marked as executable.

In fact, PDF files shouldn't be executable, so the output is exactly as desired. It's a simple matter to add a `chmod -x` where the error message about erroneous executable files is located in the script source.

By focusing too closely on the script, you could have spent a lot of time debugging something unneeded. That initial problem was solved more easily with a single invocation to `find`. Thinking about it more, however, it's clear that a more sophisticated algorithm is required to ensure that getting rid of the execute permission could be a problem, so a more sophisticated set of tests is required—and easily solved.■

Dave Taylor has been hacking shell scripts for more than 30 years—really. He's the author of the popular *Wicked Cool Shell Scripts* (10th anniversary update coming very soon from O'Reilly and NoStarch Press) and can be found on Twitter as @DaveTaylor and more generally at his tech site: http://www.AskDaveTaylor.com.

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

KYLE RANKIN

What's New in 3D Printing, Part I: Introduction

In the kickoff article to a multipart 3D printing series, Kyle introduces some of the innovations that have taken place in 3D printing during the past three years.

Three years ago, I wrote a series of articles titled "Getting Started with 3D Printing" that discussed the current state of the hobbyist 3D printing market from both the hardware and software angles. This is an incredibly fast-moving industry, and a lot has changed since I wrote those columns. So much has changed in fact, that this first article will serve just to introduce what likely will be a three- or four-part series on the current state of 3D printing. In my next articles, I'll dive deeper into particular 3D printing topics, so consider this article as an overview and sneak peek to those topics. 3D printing is a big topic, and this is *Linux Journal*, so I'm going to approach this topic from a Linux-using open-source perspective and stick to tools that work in Linux.

Open Source in 3D Printing

One of the things that has interested me most as I've followed the 3D printing industry is just how similar it is to the story of Linux distributions. In my articles from three years ago, I discussed all of the open-source underpinnings that have built the hobbyist 3D printing movement, starting with the RepRap 3D printer—an open-source 3D printer designed to be able to build as many of its parts as possible. Basically every other 3D printer you see today can trace its

roots back to the RepRap line. Now that commercial interests have taken the lead in the hobby though, it is no longer a given that you will be able to download the hardware plans for your 3D printer to make improvements, even though most of those printers got their initial designs from RepRaps. That said, you still can find popular 3D printers that value their open-source roots, and in my follow-up article on hardware, I will highlight popular 3D printers and point out which ones still rely on open hardware and open-source software.

On the subject of open-source software, many 3D printers still depend on open-source software to run. Open-source 3D printing software works well, so I can see why many companies would prefer to focus on their hardware and use the common, popular and capable open-source options. That said, some 3D printers on the market, particularly those from larger companies, ship with their own proprietary software that you must use with the printers.

The Hardware

The hardware side of the 3D printing world probably has changed the most during the past few years. Three years ago, many of the popular 3D printers still primarily were purchased in kit form, and in some cases, they required not just assembly with screwdrivers, wrenches and calipers, but also might have even required some soldering. Many of the printers also heavily followed the RepRap approach of having many 3D-printed parts. Those printers that veered from the RepRap approach still often used laser-cut wood. In either case, the end result were 3D printers that looked and felt much more like a hobbyist electronics project than a consumer product. These days, most of the popular printers look

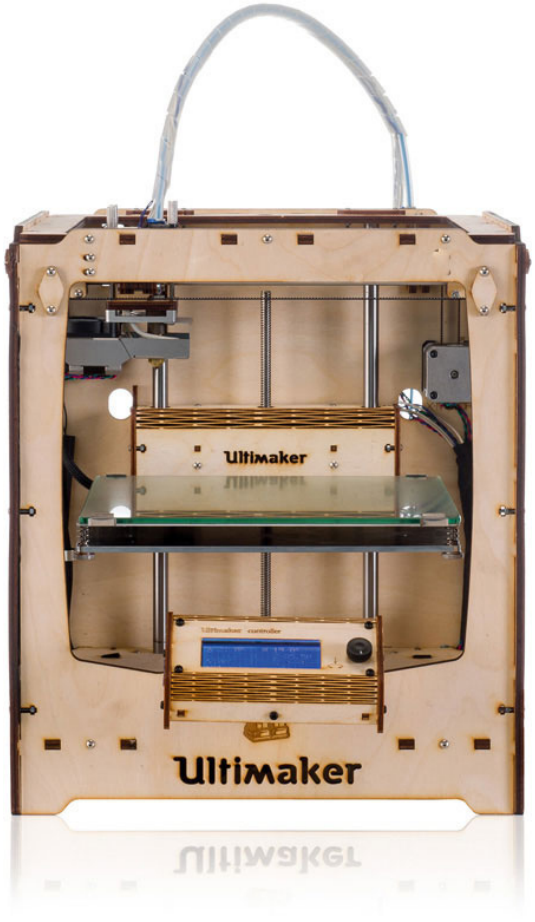


Figure 1. The Original Ultimaker

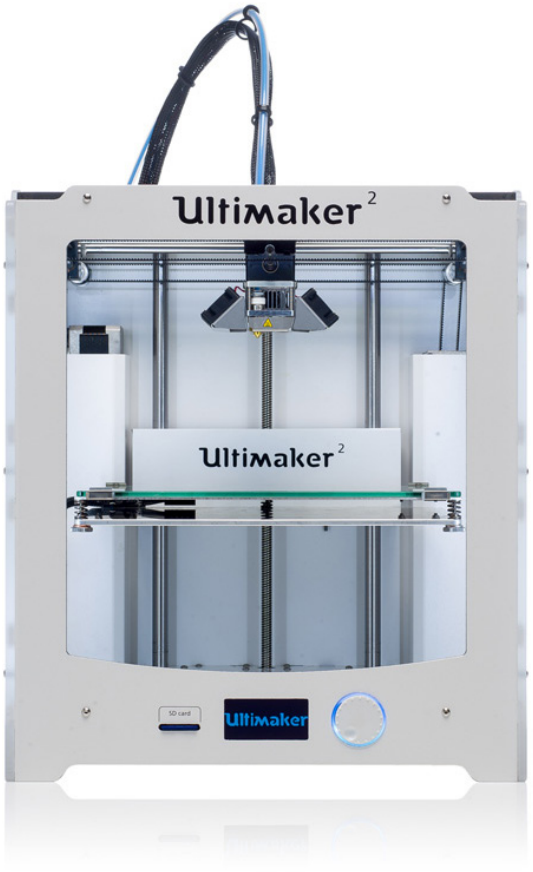


Figure 2. The Current Ultimaker 2

more like a consumer appliance than a hobbyist project. Wires and electronics are hidden. The cases themselves are made of painted wood, metal or glass, and if there are any plastic parts on the printer, they're more likely to be injection-molded than printed.

Calibration of 3D printers three years ago still was largely a manual affair. Leveling the bed might have involved a pair of calipers or a feeler gauge as you adjusted screws in each corner of the bed. Adjusting the Z axis on the printer also typically involved adjusting a screw somewhere on the printer. In some cases, you might even have had to adjust stepper motor controls on your 3D printer electronics with a screwdriver to dial in the proper voltage. As many printers were kits, a large part of the assembly process involved squaring, centering and calibrating hardware as you built the printer. These days, a lot of engineering effort has gone into automating as much of the calibration as possible. Some printers automatically sense the print bed and level it in software. Finer Z axis adjustments often can be made in software along with more exotic adjustments like stepper motor voltage. Most of the printers are sold assembled these days, and most of the calibration already has been done.

Extruder design three years ago mostly was based on the Wade's extruder design from Thingiverse, and it incorporated a number of 3D-printed parts, including gears. Although everyone was eyeing the multiple extruder support that commercial printers had, it still was at the prototype phase at best. Most hot ends had .5mm tips, and .3mm layer heights were the norm. These days, extruders have moved away from 3D printed parts with large gears into machined parts that directly drive filament into the hot end. A dual-extruder option now is

available on many of the higher-end hobbyist printers. That said, most hot ends these days still extrude with .3mm to .5mm tips, although the average printer is expected to be able to extrude at a .1mm or .2mm layer height.

The change in printable materials is one of the latest and most exciting areas of innovation in 3D printing. Three years ago, ABS and PLA plastic were your only real options. Now you have a huge variety of choices: glow-in-the-dark PLA; water-soluble PVA; a number of different types of nylon filament with different strength and flexibility profiles; flexible Ninjaflex filament that behaves more like rubber than plastic; metallic PLA filament with embedded copper, brass or bronze dust that can be polished and finished much like the pure metal counterparts; and even PLA filament with carbon fiber or bamboo. A cutting-edge category in consumer 3D printers even has emerged that prints in liquid resin and allows a new level of fine detail.

The Software

Although the improvements in 3D printing software during the past three years may not be as dramatic as the hardware changes, that doesn't make them any less interesting. The general software workflow three years ago involved downloading or building a 3D model in STL format and then loading it in a slicing tool, such as the open-source Slic3r software that you configured manually with your printer's capabilities, the filament you were using and the overall settings for the print. The slicer would slice the model into individual layers and then convert each layer into a series of GCODE commands, such as stepper motor movements that the printer understood. That GCODE then was loaded into a second piece of software, like the open-source Printrun software that was able to communicate with your printer, provide you with manual controls and send your GCODE to the printer so it could start printing. These tools worked, but they required quite a bit of in-depth knowledge of your printer's individual quirks.

Although Slic3r and Printrun still exist, these days, other open-source projects, such as Cura, are becoming the preferred open-source tools. Cura combines the slicing, communication and manual control of your printer in one interface, and it also adds nice 3D visualizations of the object so it's easier to rotate, manipulate and resize. It also ships with printer profiles for quite a few popular printers along with a wizard

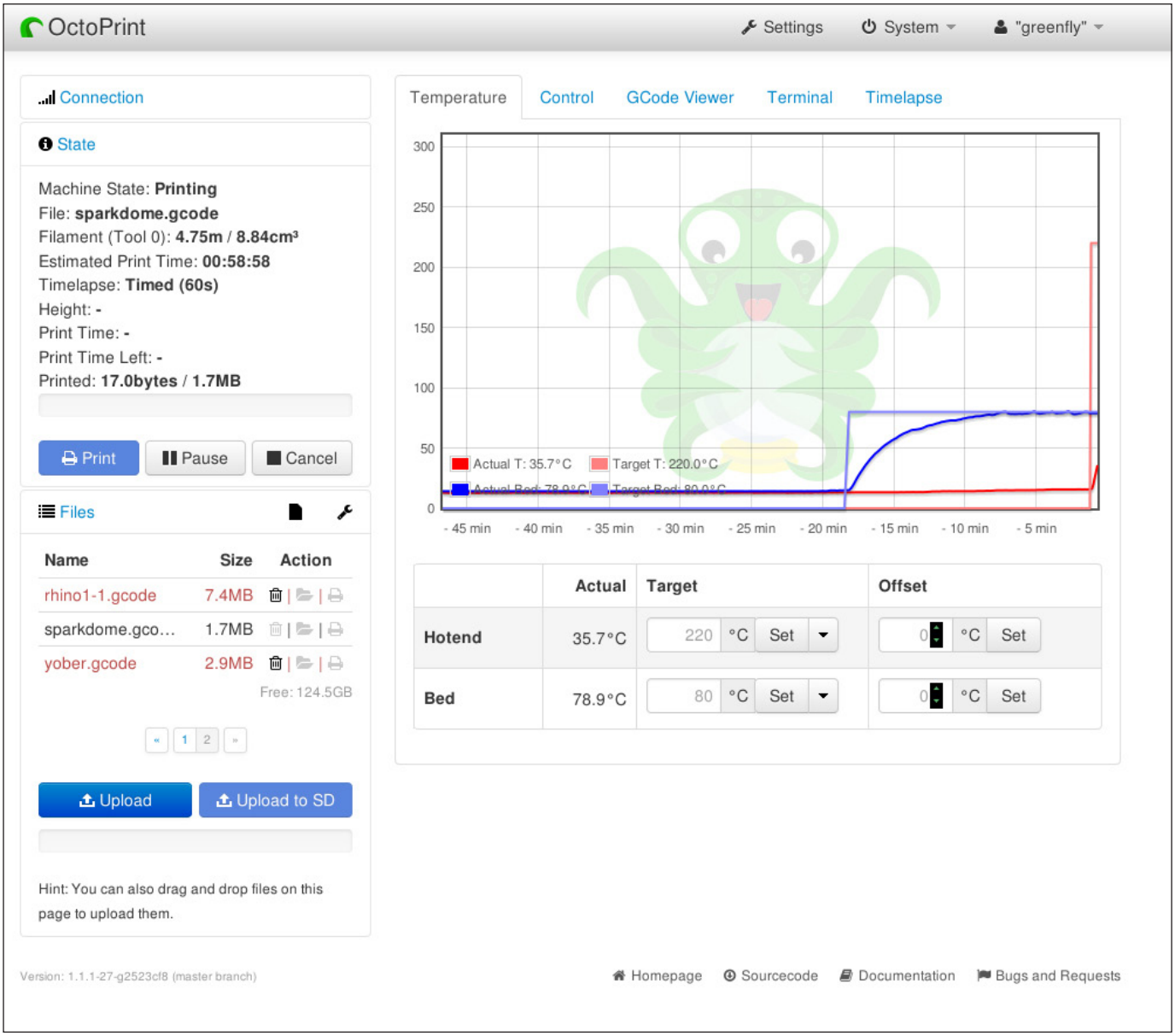


Figure 3. OctoPrint

that runs the first time you start it, so it's much easier to set up your printer the first time.

Another interesting innovation on the open-source software front is a program called OctoPrint that provides a Web-based interface to control your printer remotely. It can run on a regular computer but is geared to run from a Raspberry Pi. It supports both the Raspberry Pi camera as well as most modern Webcams that run in Linux, so you not only can watch your printer print over the network, but you also easily can generate time-lapse movies of your prints to watch

over and over again.

As you can see, a lot has changed since the last time I discussed 3D printing with Linux. In my next column, I'll discuss the hardware side of 3D printing in more detail, followed by an article on open-source 3D printing software. I'll finish the series with a column that walks through setting up OctoPrint on a Raspberry Pi step by step. My hope is that by the end of the series if you still were holding out on buying a 3D printer, you'll be convinced that now is the time to get started, and you'll have a good idea of what's out there so you can begin immediately.■

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

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

SHAWN POWERS

Doing Stuff with Docker

Don't be afraid of Docker!

I have a drawer in my office full of screws, braces, gaskets, washers and countless other "extra" pieces from various things I've built through the years. It seems that every time I assemble a bookshelf or put together a toy for my girls, there always are parts left over. If you're the type of person who reads directions, you might argue that I simply missed some steps along the way and don't really have extra pieces after all. You might be right, but I still prefer to learn by doing, even if that's a messy way to go about it.

In this article, I talk about doing stuff with Docker. *Linux Journal* has covered the Linux container system before in depth (Dirk Merkel wrote an incredible article for the March 2014 issue that explained the entire system in fine detail, and Federico Kereki has a great article this issue as well). I don't cover all the intricate workings of Docker here; I just explain how to use it. If you learn along the way, well, let's call it a bonus—just like all those bonus parts I have leftover when I build things!

What It Actually Does

If you're already familiar with the concept of Linux containers, Docker will be a no-brainer. The only thing Docker does is provide a convenient interface for creating and managing containers. If you're like me, the concept of containers makes about as much sense as feathers on a frog. Fear not, once you get it, it makes sense (the containers, not the flying frogs).

Hardware virtualization is pretty easy to understand. Every VM gets a virtualized set of hardware, and it behaves just like bare-metal hardware off a shelf behaves. You install an operating system and so on and so on. With containers, it's more like *The Matrix* for applications. Applications are all running on the same computer, but they don't realize it, because their environments are completely

In fact, containers are so flexible, you can run an application that depends on CentOS inside a container hosted on Ubuntu.

separated from each other.

The main advantage of using containers is that they're more efficient. Because all applications run on the same system, only one OS is installed, and only one set of hardware (real or virtual) is used. The isolation of the apps means they can have different dependencies, even dependencies that conflict with other apps! If you have one Web application that requires PHP version 4 and one that requires PHP version 5, normally you'd need to set up two separate machines. With containers, you just package the application and its dependencies together, and they interact independently from the rest of the apps in other containers!

In fact, containers are so flexible, you can run an application that depends on CentOS inside a container hosted on Ubuntu. You just package the required CentOS files in the container with the app, and it has no idea it's actually running on Ubuntu, because it sees all the CentOS files it needs inside its container!

If that's all a little too confusing, here's my simplified version. Traditional hardware virtualization (VMware and so on) virtualizes the hardware. Containers virtualize only the software environment in which an application runs.

So Where Does Docker Fit In?

Everything I just described concerns containers in general. There are multiple ways to manipulate containers on Linux. Docker is one of those ways. Arguably it's the best way, but at the very least, it's the most popular way. If you're a VMware user, think of Linux containers as being ESXi and Docker being like VSphere. It's a way to create, interact and manage Linux containers.

Like most things in the Open Source world, the best thing about Docker is the community of users who use it. Not only does Docker provide a great user interface for using containers, but the community also has created hundreds (maybe thousands) of pre-made environments for running specific applications inside Docker. In this

article, I walk through installing one of those images—specifically, the first Docker container I ever installed: Plexmediaserver.

Docker Jargon

Although I'm not going to delve into the low-level Docker stuff here, it's still important to understand the concepts regarding what Docker actually does. The two main Docker bits I cover in this article are "images" and "containers".

Images are downloaded from the Internet or built locally. These images are stored on the Docker server, but are not directly executed. They're basically a collection of the dependencies, the application and any other things required to create a running container. It's like a cake mix. All the ingredients are packaged nicely, waiting for you to mix them up and bake them. Pre-built images are available from the Docker Hub, which is a community-driven repository of images anyone can download.

Containers are what you get when you deploy an image. A container is the actual running application nestled inside its own environment. When you unpack an image and start a container, it takes all the ingredients in that "cake mix" and extracts them into an isolated environment, then executes the app. Unlike a cake mix, however, it's possible to create multiple containers from a single image. Once you have an image, it's a simple one-line command to start up the application in a container of its own.

Installing Docker

Most Linux distributions (along with Windows and OS X) can run Docker. I cover the method for installing on Ubuntu 14.04 here, but a quick Google search will show you how to install Docker anywhere. In order to install the most recent version of Docker on your system, simply type:

```
wget -qO- https://get.docker.com/ | sh
```

Normally, installing an application using a script is horrible, horrible advice. In this case, however, the folks at Docker have created a script that does things properly. If you're running Ubuntu or Debian, it will create the proper repositories and install the correct dependencies using APT. In fact, the same `wget` command probably will work on a CentOS or Red Hat system as well. It just detects your system type and installs repos using the YUM tools. I've tested it only in Ubuntu 14.04, however, so if you want to

experiment elsewhere, things might behave slightly differently.

Once the installer is finished, type:

```
docker -v
```

and you should see Docker return the current version.

A Few More Docker Concepts

Before downloading an image and starting a container, it's important to know how Docker containers access data. See, when a container is created, it's purposefully isolated from the rest of the system. The filesystem that the app inside the container sees is a virtualized filesystem to which only it has access. If your application is a standalone app that doesn't require any external data, that's fine. In this case (and most cases), however, you need your container to have access to shared folders. It's certainly possible to create a container with an NFS client and mount directories internally, but Docker provides a really simple way to share folders with containers. When you start a container, you specify what folders you want to have accessible from inside the running container, and it "maps" that folder on the fly without any complicated NFS or Samba configuration required.

Docker also allows for several networking options with containers. By default, Docker tries to create a bridged network interface intelligently and start each container with a unique private IP. You can then redirect ports on your firewall to the appropriate container IP address, or connect directly to the private IP from within your network. While that allows for a very robust and complex network infrastructure for Docker, it also makes things frustratingly complex for people just starting out. In this example here, you'll use the "host" feature of Docker, which allows the container to share an IP with the host system. In production, there potentially are security concerns with this method, but especially at first, it's a great way to use Docker.

Checking Out the Goods

Although it's possible to create Docker images from scratch and build them on your local system, the best way to start is by downloading an image someone else already created. You can browse those images by heading over to https://hub.docker.com, or you can search the same repository directly from the command line. If you think of an app you'd like to run in Docker, the first thing I suggest is to check the Docker Hub and see if someone else

```
spowers@docker:~$ sudo docker search plex
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
timhaak/plex                        This is a Dockerfile to set up "Plex Media...   40                   [OK]
needo/plex                                                                          21                   [OK]
binhex/arch-plex                    Arch Linux base running AUR compiled versi...   5                    [OK]
maxexcloo/plexmediaserver           Docker service container with Plex Media S...   3                    [OK]
jrarseneau/plexmediaserver          Plex Media Server (Regular & PlexPass vers...   2                    [OK]
nicholasvmoore/docker-plex          Plex fully functional within Docker             1                    [OK]
lanrat/plex                                                                         1                    [OK]
nicocoffo/docker-plex-media-server                                                  1                    [OK]
gionn/plex                          A Plex Media Server on base ubuntu 14.04 w...   1                    [OK]
dperson/plex                                                                        1                    [OK]
wernight/plex-media-server          Plex Media Server - Your media on all your...   1                    [OK]
scottcase/plex                                                                      0                    [OK]
```

Figure 1. There's actually a huge listing, but the "cream" floats to the top.

already has "dockerized" the app for you. (That's what you're going to do with Plex.)

It's possible to log in to Docker Hub from the command line using the `docker` program, but you don't have to have an account in order to use existing images. You need to have an account only if you want to host or upload images. In order to see if Plex has been dockerized by someone else, type:

```
sudo docker search plex
```

You should see a huge list of images uploaded by multiple people. It's very likely that they all work, but I recommend using images that have the largest number of "stars" rating them as favorites. Figure 1 shows the first few lines of my search query. Notice that the timhaak/plex image has 40 stars. Let's use that one.

It's So Simple, Shawn Can Do It!

In order to download the image to your local system, type:

```
sudo docker pull timhaak/plex
```

You should see the process as it downloads all the files so you can create your own container from the downloaded image. Remember, downloading the image doesn't create a container, it just downloads the "cake mix" so you can start up your own instance.

Once it downloads all the information it needs, you can type:

```
sudo docker images
```

You should get a listing of all the images stored on your local system, and you should see the timhaak/plex image listed. You'll probably also see a "debian" image that has been

downloaded automatically as well. The plex image builds on top of the debian image, so it downloads that too. When you start the container, it won't create a separate debian container, it will pull what it needs (as defined by the plex image) from the debian image and include it in the running container.

In my case, I need to have the Plex app be able to access my video files. I also want the log files to be accessible from outside the container, so I can see what's going on from the outside. I created a shared folder on my host computer called /mnt/docker/plex, and I have my videos stored on /mnt/videos. Once those places have been created (again, not always necessary, but in this particular case, I need to access the videos!), the last step is creating the container. Here is the command I use (I'll go over it piece by piece afterward):

```
sudo docker run -d --net="host" \
-v /mnt/docker/plex:config \
-v /mnt/videos:data \
-p 32400:32400 \
timhaak/plex
```

I used the backslashes because it's a really long command, but it can all be typed on a single line since it's really just a single command. Here's the breakdown:

- `sudo docker run` — This tells Docker to create and execute a container.
- `-d` — This is a flag specifying that I want the container to run as a dæmon in the background.
- `--net="host"` — This specifies that the container will be sharing the host's IP address.
- `-v /mnt/docker/plex:config` — This tells Docker to create a folder inside the container located at /config that is mapped to the host system's /mnt/docker/plex folder.
- `-v /mnt/videos:data` — Another shared folder, this maps the /data folder inside the container to the /mnt/videos folder on the host system.
- `-p 32400:32400` — Here the single port 32400 from inside the container is mapped to the host system's 32400 port. That makes Plex accessible from other computers.
- `timhaak/plex` — This specifies the image to use when creating the container.

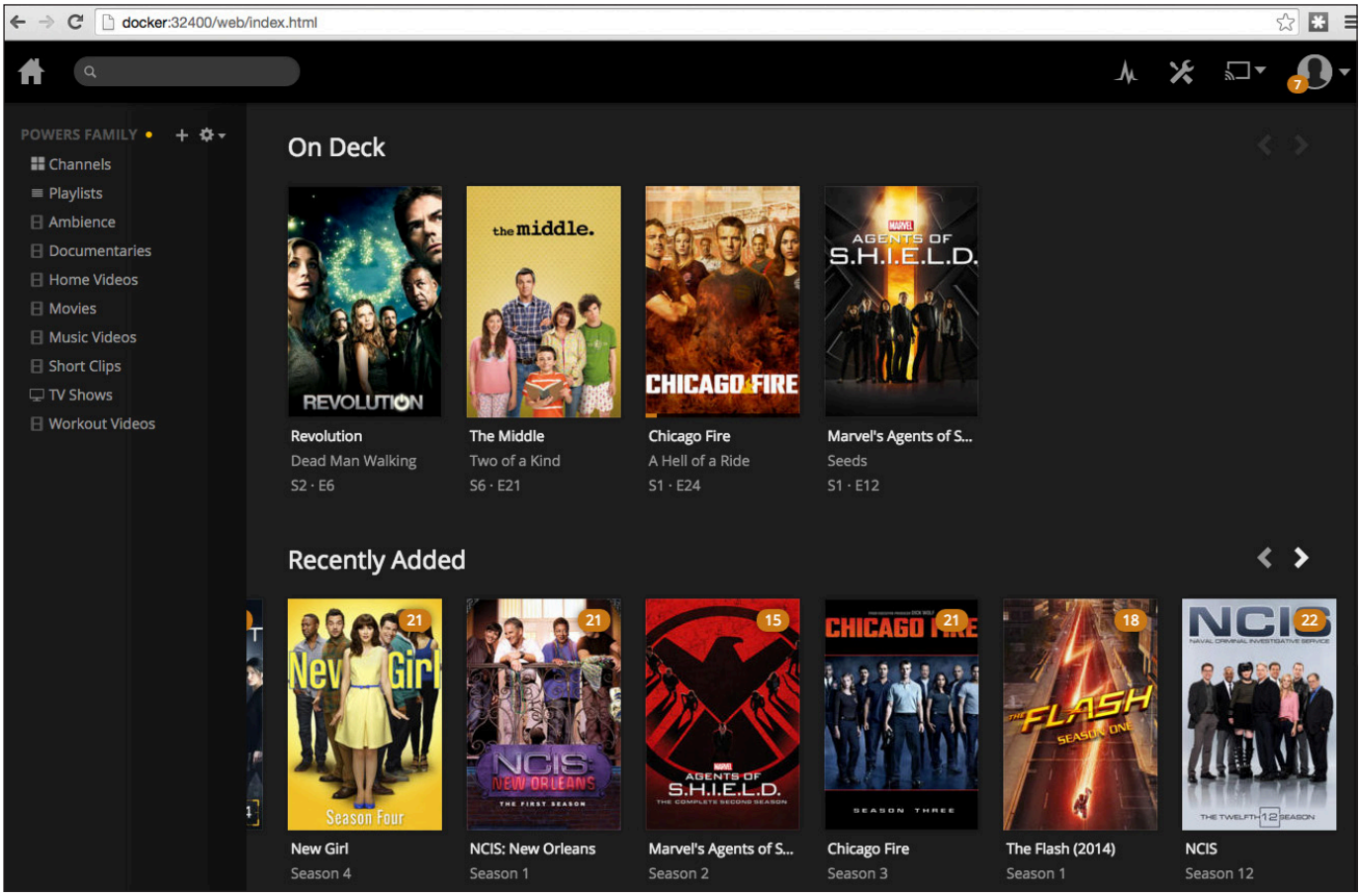


Figure 2. Don't judge me on the shows my family watches!

Test It!

As long as you don't get any errors, you should be returned to the command-line prompt. Head over to a Web browser and visit http://host-ip:32400/web/, and see if you can connect to the Plex server! (Note: host-ip in that URL is the IP address of your host system.) Figure 2 shows my Plex server running from a container.

Of course, my screenshot shows my Plex server after it has been configured. The first time you visit the server, you'll need to configure it for your own system. Still, it should be that easy to get the container running.

Managing Containers

Much like running:

```
sudo docker images
```

shows you the images on your system, you can see the containers on your

Docker has an official GUI called Kitematic that works about like you'd expect a GUI to work.

system by typing:

```
sudo docker ps -a
```

If you leave off the `-a`, it will show you only running containers on your system. Once you see the containers that are running, you can start, stop, restart or destroy (delete) them using the `docker` command. So running:

```
sudo docker restart CONTAINER_ID
```

will restart the container specified by the ID. You also can specify the container you want to manipulate by referring to its funny name listed in the "NAMES" column of the `ps -a` results. For instance, mine is called "sad_babbage", but yours will be some other two-word name.

Where to Go from Here?

There are tons more things you can do with Docker. You can create custom images. You can build your own images from scratch. You can automate the creation and destruction of containers on the fly. In this article, you probably learned just enough to understand what Docker is and how to use it. Hopefully you're inspired to learn more.

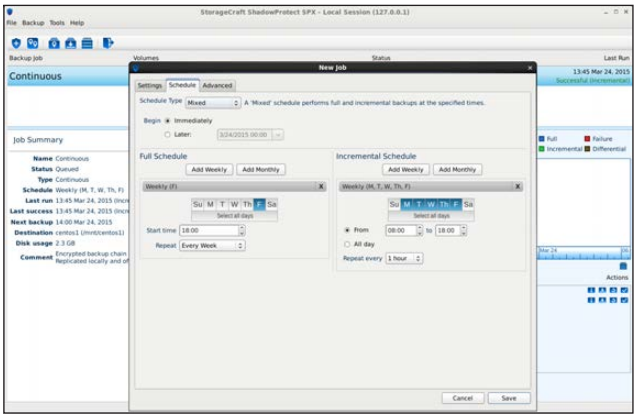
If you prefer not to use the command line to deal with images and containers, there also are a few GUI tools. Docker has an official GUI called Kitematic that works about like you'd expect a GUI to work. You can manipulate images and containers by pointing and clicking instead of typing on the command line.

However you choose to use Docker, the important part is not to be afraid of the technology. Even if you never plan to use it in production, I urge you to play with it a bit. Linux containers and Docker specifically are really efficient ways to utilize your resources. Plus, it's really fun!■

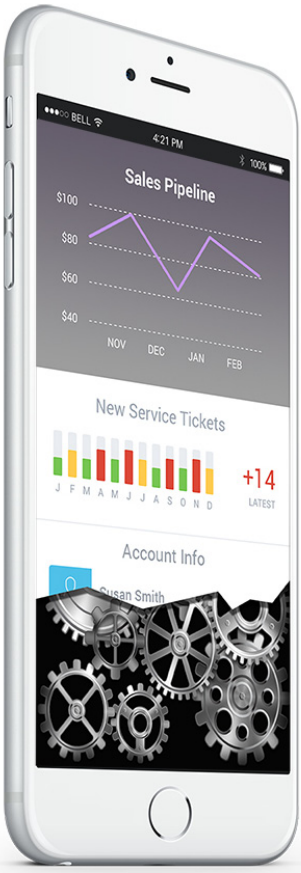
Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing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

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

StorageCraft ShadowProtect SPX

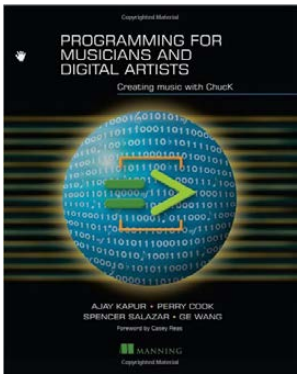


The same backup and disaster-recovery technologies that brought StorageCraft long-term success outside the Linux space are now available to Linux users everywhere. The recently revealed StorageCraft ShadowProtect SPX solution allows Linux users to back up, protect, migrate and recover virtual and physical Linux servers reliably. SPX features, announced StorageCraft, enable quick and efficient sector-level back up of a complete Linux system, including the OS, applications, settings, services and data. In the case of disaster, IT administrators can recover their systems and regain access to their systems and data within minutes. Supported Linux flavors include Ubuntu 12.04, Red Hat Enterprise Linux 6 and CentOS 6.
http://www.storagecraft.com



AnyPresence's JustAPIs

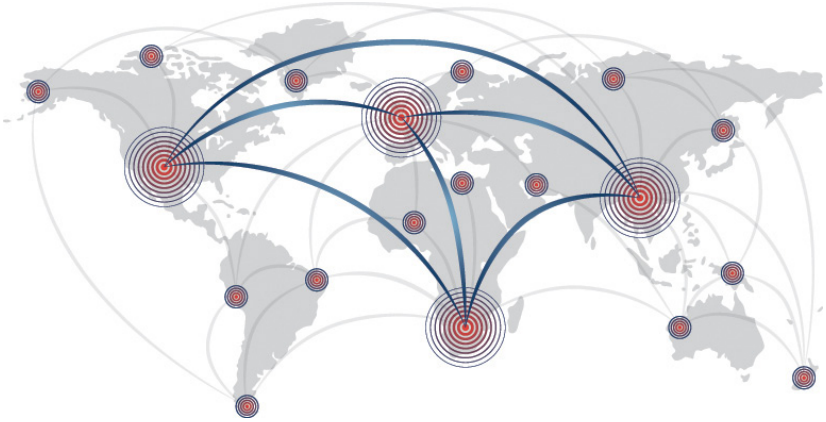
JustAPIs from AnyPresence is designed with a singular focus: to solve the API building challenge for the enterprise app developer in an elegant manner. The JustAPIs solution, which enables the building and deploying of contemporary RESTful APIs, is targeted at IT organizations and enterprise developers who need to define custom API workflows within the corporate firewall, complementing existing MBaaS, MEAP/MADP or app development frameworks. JustAPIs gives individuals a quick, easy way to define and deploy APIs with specific signatures, which either can be standalone services with JavaScript-based business logic or connect to existing legacy and SOAP-based Web services in the enterprise. AnyPresence says that JustAPIs rises above traditional API management solutions that historically focus on enterprise-wide API governance and often are too expensive and cumbersome for the app-specific needs on which this "revolutionary new solution" focuses.
http://www.anypresence.com



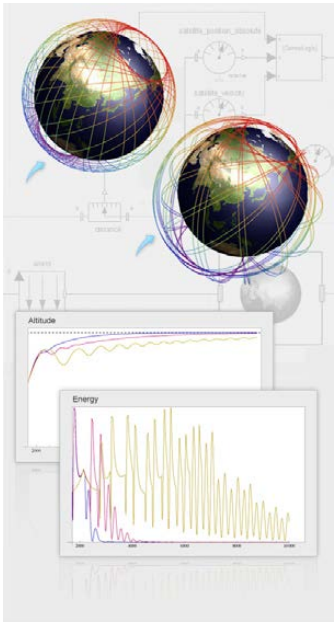
Ajay Kapur, Perry Cook, Spencer Salazar and Ge Wang's *Programming for Musicians and Digital Artists* (Manning)

The world of digital music offers endless opportunities for creativity. Channeling your inner Philip Glass—and doing so without sacrificing your Linux-enthusiast principles—is a snap with the new book *Programming for Musicians and Digital Artists*. Subtitled *Creating music with ChucK*, this book presents a complete introduction to programming in the ChucK open-source music language. Readers will learn the basics of digital sound creation and manipulation while mastering the ChucK language. ChucK provides precise control over time, audio computation and user interface elements like track pads and joysticks. While moving example by example through this easy-to-follow book, readers create meaningful and rewarding digital compositions and "instruments" that make sound and music in direct response to program logic, scores, gestures and other systems connected via MIDI or the network. Because this book utilizes the vocabulary of sound, ChucK is easy to learn even for artists with little or no exposure to computer programming.
http://www.manning.com/kapur

2ndQuadrant's Bi-Directional Replication



Bi-Directional Replication (BDR) is an open-source PostgreSQL replication solution developed by 2ndQuadrant, an independent sponsor and developer of PostgreSQL. The upgraded BDR 0.9.0 adds a range of significant new features and enhancements, such as dynamic SQL-level configuration of connections between nodes, dynamic configuration (no restarting any nodes during the node join or removal process), easy node removal, UDR (Uni-Directional Replication), replication sets to specify sets of tables that each node should receive changes on and expanded documentation.
http://www.2ndquadrant.com



Wolfram Research's SystemModeler

Reliability analysis is critical to product development, illuminating where to concentrate engineering efforts, where failure might happen and how warranties should be priced. These are just a few of the benefits of Wolfram Research's updated SystemModeler 4.1, an intuitive modeling and simulation environment for cyber-physical systems. New feature highlights include full capabilities for importing models, importing from tools based on the FMI standard, importing of subsystems from other tools, model exchange without exposing intellectual property, construction of hierarchical models containing reliability block diagrams and fault trees and greatly improved speeds in the GUI. A sampling of industries that might benefit from SystemModeler's reliability analysis tool are aerospace, automotive, pharmaceuticals, systems biology and electrical engineering.
http://www.wolfram.com

BLACKDUCK

Black Duck Software's Black Duck Hub

A critical component of security management in today's enterprises involves identifying and tracking vulnerabilities in open-source code. To tackle this task, two natural partners—Black Duck Software and Risk Based Security—have joined forces to develop Black Duck Hub, a new solution that combines powerful open-source discovery with greater vulnerability intelligence to ensure higher levels of security in open-source software. Black Duck Hub helps customers identify security-related issues faster, prioritize remediation activity and implement proactive controls to avoid the use of vulnerable components. The power of the partnership between Black Duck and Risk Based Security is evident in the latter partner's VulnDB, a resource that extends the commonly used National Vulnerability Database by an additional 35,000 vulnerabilities, resulting in actionable intelligence for more than 119,000. The result, says Black Duck, is the ability for customers to take control of software and application security proactively.
http://www.blackducksoftware.com

Super Talent's mSATA SJ2 SSD

Industrial and embedded applications where expansion options for storage are limited is right where Super Talent's new mSATA SJ2 Solid State Drive (SSD) belongs. Available in capacities from 16GB to 128GB, the mSATA SJ2 SSD with SATA-III interface offers extremely fast speeds of up to 480MB/sec reads and 160MB/sec writes for mobile solutions. A small form factor and high reliability are other features that Super Talent notes about the mSATA SJ2. Target applications include aerospace, casino gaming, embedded systems and the medical industry.
http://www.supertalent.com

Symple PC

The founder of Symple LLC and inspiration behind his firm's new Symple PC, Jason Spisak, makes at least two fine points. First, there are millions of off-lease PCs gathering dust that are more than capable of running Linux, and we have a responsibility (as stewards of our finite planet) to re-use them and prevent e-waste. Second, thanks to a convergence of technological advancements, the present is an ideal time to speed the adoption of open-source into schools, nonprofits, call centers and Web-enabled businesses. Enter the Symple PC, a re-manufactured Ubuntu Linux Web workstation priced under $100. "This little marvel", as the company calls it, is "lovingly made in the USA from recycled and re-manufactured materials". The case—with 50% less mass than conventional ones—is made from recycled ABS plastic, the parts are recycled, and the carton has no new fiber content, among other planet-friendly pluses. Under the hood, users currently will find Ubuntu Linux orchestrating resources on at least 2GB of RAM, 2.8GHz of desktop-class processing power and at least a 80GB SATA hard drive. To encourage the closing of the product loop, a $10 Environmental Credit is offered for any Symple PC that is returned toward the purchase of a new unit.
http://symplepc.com

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

Using tshark to Watch and Inspect Network Traffic

Learn how to store network information in MongoDB using tshark and Python.

MIHALIS TSOUKALOS

Most of you probably have heard of Wireshark, a very popular and capable network protocol analyzer. What you may not know is that there exists a console version of Wireshark called tshark. The two main advantages of tshark are that it can be used in scripts and on a remote computer through an SSH connection. Its main disadvantage is that it does not have a GUI, which can be really handy when you have to search lots of network data.

You can get tshark either from its Web site and compile it yourself or from your Linux distribution as a precompiled package. The second way is quicker and simpler. To install tshark on a Debian 7 system, you just have to run the following command as root:

```
# apt-get install tshark
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libc-ares2 libcap2-bin libpam-cap libsmi2ldbl
  libwireshark-data libwireshark2
  libwiretap2 libwsutil2 wireshark-common
Suggested packages:
  libcap-dev snmp-mibs-downloader wireshark-doc
The following NEW packages will be installed:
  libc-ares2 libcap2-bin libpam-cap libsmi2ldbl
  libwireshark-data libwireshark2
  libwiretap2 libwsutil2 tshark wireshark-common
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 15.6 MB of archives.
After this operation, 65.7 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
...
```

To find out whether tshark is installed properly, as well as its version, execute this command:

```
$ tshark -v
TShark 1.8.2
...
```

Note: this article assumes that you already are familiar with network data, TCP/IP, packet capturing and maybe Wireshark, and that you want to know more about tshark.

About tshark

tshark can do anything Wireshark can do, provided that it does not require a GUI. It also can be used as a replacement for tcpdump, which used to be the industry standard for network data capturing. Apart from the capturing part, where both tools are equivalent, tshark is more powerful than tcpdump; therefore, if you want to learn just one tool, tshark should be your choice.

As you can imagine, tshark has many command-line options. Refer to its man page for the full list.

Capturing Network Traffic Using tshark

The first command you should run is `sudo tshark -D` to get a list of the available network interfaces:

```
$ sudo tshark -D
1. eth0
2. nflog (Linux netfilter log (NFLOG) interface)
3. any (Pseudo-device that captures on all interfaces)
4. lo
```

If you run tshark as a normal user, you most likely will get the following output, because normal users do not have direct access to network interface devices:

```
$ tshark -D
tshark: There are no interfaces on which a capture can be done
```

The simplest way of capturing data is by running `tshark` without any parameters, which will display all data on screen. You can stop data capturing by pressing Ctrl-C.

The output will scroll very fast on a busy network, so it won't be helpful at all. Older computers could not keep up with a busy network, so programs like tshark and tcpdump used to drop network packets. As modern computers are pretty powerful, this is no longer an issue.

Saving and Reading Network Data Using Files

The single-most useful command-line parameter is `-w`, followed by a filename. This parameter allows you to save network data to a file in order to process it later. The following tshark command captures 500 network packets (`-c 500`) and saves them into a file called LJ.pcap (`-w LJ.pcap`):

```
$ tshark -c 500 -w LJ.pcap
```

The second-most useful parameter is `-r`. When followed by a valid filename, it allows you to read and process a previously captured file with network data.

Capture Filters

Capture filters are filters that are applied during data capturing; therefore, they make tshark discard network traffic that does not match the filter criteria and avoids the creation of huge capture files. This can be done using the `-f` command-line parameter, followed by a filter in double quotes.

The most important TCP-related Field Names used in capture filters are tcp.port (which is for filtering the source or the destination TCP port), tcp.srcport (which is for checking the TCP source port) and tcp.dstport (which is for checking the destination port).

Generally speaking, applying a filter after data capturing is considered more practical and versatile than filtering during the capture stage, because most of the time, you do not know in advance what you want to inspect. Nevertheless, if you really know what you're doing, using capture filters can save you time and disk space, and that is the main reason for using them.

Remember that filter strings always should be written in lowercase.

Display Filters

Display filters are filters that are applied after packet capturing; therefore, they just "hide" network traffic without deleting it. You always can remove the effects of a display filter and get all your data back.

Display Filters support comparison and logical operators. The `http.response.code == 404 && ip.addr == 192.168.10.1` display filter shows the traffic that either comes from the 192.168.10.1 IP address or goes to the 192.168.10.1 IP address that also has the 404 (Not Found) HTTP response code in it. The `!bootp && !ip` filter excludes BOOTP and IP traffic from the output. The `eth.addr == 01:23:45:67:89:ab && tcp.port == 25` filter displays the traffic to or from the network device with the 01:23:45:67:89:ab MAC address that uses TCP port 25 for its incoming or outgoing connections.

When defining rules, remember that the `ip.addr != 192.168.1.5` expression does not mean that none of the `ip.addr` fields can contain the 192.168.1.5 IP address. It means that one of the `ip.addr` fields should not contain the 192.168.1.5 IP address! Therefore, the other `ip.addr` field value can be equal to 192.168.1.5! You can think of it as "there exists one `ip.addr` field that is not 192.168.1.5". The correct way of expressing it is by typing `!(ip.addr == 192.168.1.5)`. This is a common misconception with display filters.

Also remember that MAC addresses are truly useful when you want to track a given machine on your LAN, because the IP of a machine can change if it uses DHCP, but its MAC address is more difficult to change.

Display filters are extremely useful tools when used correctly, but you still have to interpret the results, find the problem and think about the possible solutions yourself. It is advisable that you visit the display filters reference site for TCP-related traffic at http://www.wireshark.org/docs/dfref/t/tcp.html. For the list of all the available field names related to UDP traffic, see http://www.wireshark.org/docs/dfref/u/udp.html.

Exporting Data

Imagine you want to extract the frame number, the relative time of the frame, the source IP address, the destination IP address, the protocol of the packet and the length of the network packet from previously captured network traffic. The following tshark command will do the trick for you:

```
$ tshark -r login.tcpdump -T fields -e frame.number -e
➥frame.time_relative -e ip.src -e ip.dst -e
➥frame.protocols -e frame.len -E header=y -E
➥quote=n -E occurrence=f
```

The `-E header=y` option tells tshark first to print a header line. The `-E quote=n` dictates that tshark not include the data in quotes, and the `-E occurrence=f` tells tshark to use only the first occurrence for fields that have multiple occurrences.

Having plain text as output means that you easily can process it the UNIX way. The following command shows the ten most popular IPs using input from the `ip.src` field:

```
$ tshark -r ~/netData.pcap -T fields -e ip.src | sort
➥| sed '/^\s*$/d' | uniq -c | sort -rn
➥| awk {'print $2 " " $1'} | head
```

Two Python Scripts That Use tshark

Now, let's look at two Python scripts that read tshark's text output and process it. I can't imagine doing the same thing with a GUI application, such as Wireshark!

Listing 1 shows the full Python

Listing 1. checkIP.py

```
# Programmer: Mihalis Tsoukalos
# Date: Tuesday 28 October 2014

import socket
import sys
import re

def valid_ip(address):
    try:
        socket.inet_aton(address)
        return True
    except:
        return False

# Counters for the IPs
total = 0
valid = 0
invalid = 0

# Read the file from stdin, line by line
for line in sys.stdin:
        line = line.rstrip('\n')
        if valid_ip(line):
            valid = valid + 1
            # print "The IP is valid!"
        else:
            # print "The IP is not valid!"
            invalid = invalid + 1
        total = total + 1

# Present the total number of IPs checked
print "Total number of IPs checked:",
total
print "Valid IPs found:", valid
print "Invalid IPs found:", invalid
```

code of the first script that checks the validity of an IP address.

The purpose of the checkIP.py Python script is just to find invalid IP addresses, and it implies that the network data already is captured with tshark. You can use it as follows:

```
$ tshark -r ~/networkData.pcap -T fields -e ip.src
 ➥| python checkIP.py
Total number of IPs checked: 1000
Valid IPs found: 896
Invalid IPs found: 104
```

Listing 2 shows the full code of the second Python script (storeMongo.py).

The Python script shown in Listing 2 inserts network data into a MongoDB database for further processing and querying. You can use any database you want. The main reason I used MongoDB is because I like the flexibility it offers when storing structured data that may have some irregular records (records with missing fields).

The name of the Python script is storeMongo.py, and it assumes that

Listing 2. store Mongo.py

```
# Programmer: Mihalis Tsoukalos
# Date: Tuesday 28 October 2014
#
# Description: This Python script reads input from
# tshark, parses it and stores it in a MongoDB database

import sys
import pymongo
import re

# The number of BSON documents written
total = 0

# Open the MongoDB connection
connMongo = pymongo.Connection('mongodb://localhost:27017')
# Connect to database named LJ (Linux Journal)
db = connMongo.LJ
# Select the collection to save the network packet
traffic = db.netdata

# Read the file from stdin, line by line
for line in sys.stdin:
        line = line.rstrip('\n')
        parsed = line.split("\t")
        total = total + 1

        # Construct the "record to be inserted
        netpacket = {
                'framenumber': parsed[0],
                'sourceIP': parsed[1],
                'destIP': parsed[2],
                'framelength': parsed[3],
                'IPlength': parsed[4]
                }

        # Store it!
        net_id = traffic.insert(netpacket)

connMongo.close()

# Present the total number of BSON documents written
print "Total number of documents stored: ", total
```

the network data already is captured using either tshark or tcpdump. The next shell command runs the Python script with input from tshark:

```
$ tshark -r ~/var/test.pcap -T fields -e frame.number
 ➥-e ip.src -e ip.dst -e frame.len -e
 ➥ip.len -E header=n -E quote=n -E occurrence=f
 ➥| python storeMongo.py
Total number of documents stored:  500
```

The text output of the tshark command is similar to the following:

```
5    yy.xx.zz.189    yyy.74.xxx.253  66      52
6    197.224.xxx.145 yyy.74.xxx.253  86      72
7    109.xxx.yyy.253 zzz.224.xxx.145 114     100
8    197.xxx.zzz.145 zzz.xxx.xxx.253 86      72
9    109.zzz.193.yyy 197.224.zzz.145 114     100
```

Currently, all numerical values are stored as strings, but you easily can convert them to numbers if you want. The following command converts all string values from the IPlength column to their respective integer values:

```
> db.netdata.find({IPlength : {$exists : true}}).forEach(
 ➥function(obj) { obj.IPlength = new NumberInt(
 ➥obj.IPlength ); db.netdata.save(obj); } );
```

Now you can start querying the MongoDB database. The following commands find all "records" (called documents in NoSQL terminology) that contain a given destination IP address:

```
> use LJ
switched to db LJ
> db.netdata.find({ "destIP": "192.168.1.12" })
...
>
```

The next command finds all entries with a frame.len value that is less than 70:

```
> use LJ
switched to db LJ
> db.netdata.find({ "framelength": {"$lt" : "70" }})
...
>
```

The next command finds all entries with an IPlength value greater than 100 and less than 200:

```
> use LJ
switched to db LJ
> db.netdata.find({ "IPlength": {"$lt" : "200", "$gt": "100" }})
...
>
```

What you should remember is not the actual commands but the fact that you can query the database of your choice, using the query language you want and find useful information without the need to re-run tshark and parse the network data again.

After you test your queries, you can run them as cron jobs. *La vie est belle!*

Examining an Nmap ping Scan Using tshark

Next, let's examine the network traffic that is produced by Nmap when it performs a ping scan. The purpose of the ping scan is simply to find out whether an IP address is up. What is important for Nmap in a ping scan is not the actual data of the received packets but, put simply, the actual existence of a reply packet. Nmap ping scans inside a LAN are using the ARP protocol; whereas hosts outside a LAN are scanned using the ICMP protocol. The performed scan pings IP addresses outside the LAN.

The following Nmap command scans 64 IP addresses, from 2.x.yy.1 to 2.x.yy.64:

```
# nmap -sP 2.x.yy.1-64
Starting Nmap 6.00 ( http://nmap.org ) at 2014-10-29 11:55 EET
Nmap scan report for ppp-4.home.SOMEisp.gr (2.x.yy.4)
Host is up (0.067s latency).
Nmap scan report for ppp-6.home.SOMEisp.gr (2.x.yy.6)
Host is up (0.084s latency).
...
Nmap scan report for ppp-64.home.SOMEisp.gr (2.x.yy.64)
Host is up (0.059s latency).
Nmap done: 64 IP addresses (35 hosts up) scanned in 3.10 seconds
```

The results show that at execution time only 35 hosts were up, or to be 100% precise, only 35 hosts answered the Nmap scan. Nmap also calculates the round-trip time delay (or latency). This gives a pretty accurate estimate of the time needed for the initial packet (sent by Nmap) to go to the target device plus the time that the response packet took to return back to Nmap.

The following tshark command is used for the capturing and is terminated with Ctrl-C:

```
# tshark -w nmap.pcap
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
2587 ^C
18 packets dropped
# ls -l nmap.pcap
-rw------- 1 root root 349036 Oct 29 11:55 nmap.pcap
```

Now, let's analyze the generated traffic using tshark. The following command searches for traffic to or from the 2.x.yy.6 IP address:

```
$ tshark -r nmap.pcap -R "ip.src == 2.x.yy.6 || ip.dst == 2.x.yy.6"
712  3.237125000 109.zz.yyy.253 -> 2.x.yy.6
 ➥ICMP 42 Echo (ping) request  id=0xa690, seq=0/0, ttl=54
1420 5.239804000 109.zz.yyy.253 -> 2.x.yy.6
 ➥ICMP 42 Echo (ping) request  id=0x699a, seq=0/0, ttl=49
1432 5.240111000 109.zz.yyy.253 -> 2.x.yy.6
 ➥TCP 58 41242 > https [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1441 5.296861000     2.x.yy.6 -> 109.zz.yyy.253 ICMP 60
 ➥Timestamp reply      id=0x0549, seq=0/0, ttl=57
```

tshark allows you to display useful statistics about a specific protocol.

As you can see, the existence of a response packet (1441) from 2.x.yy.6 is enough for the host to be considered up by Nmap; therefore, no additional tests are tried on this IP.

Now, let's look at the traffic for an IP that is considered down:

```
$ tshark -r nmap.pcap -R "ip.src == 2.x.yy.2 || ip.dst == 2.x.yy.2"
708  3.236922000 109.zz.yyy.253 -> 2.x.yy.2
 ➥ICMP 42 Echo (ping) request  id=0xb194, seq=0/0, ttl=59
1407 5.237255000 109.zz.yyy.253 -> 2.x.yy.2
 ➥ICMP 42 Echo (ping) request  id=0x24ed, seq=0/0, ttl=47
1410 5.237358000 109.zz.yyy.253 -> 2.x.yy.2
 ➥TCP 58 41242 > https [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1413 5.237448000 109.zz.yyy.253 -> 2.x.yy.2
 ➥TCP 54 41242 > http [ACK] Seq=1 Ack=1 Win=1024 Len=0
1416 5.237533000 109.zz.yyy.253 -> 2.x.yy.2
 ➥ICMP 54 Timestamp request    id=0xf7af, seq=0/0, ttl=51
1463 5.348871000 109.zz.yyy.253 -> 2.x.yy.2
 ➥ICMP 54 Timestamp request    id=0x9d7e, seq=0/0, ttl=39
1465 5.349006000 109.zz.yyy.253 -> 2.x.yy.2
 ➥TCP 54 41243 > http [ACK] Seq=1 Ack=1 Win=1024 Len=0
1467 5.349106000 109.zz.yyy.253 -> 2.x.yy.2
 ➥TCP 58 41243 > https [SYN] Seq=0 Win=1024 Len=0 MSS=1460
```

As the ICMP packet did not get a response, Nmap makes more tries on the 2.x.yy.2 IP by sending an HTTP and an HTTPS packet, still without any success. This happens because Nmap adds intelligence to the standard ping (ICMP protocol) by trying some common TCP ports in case the ICMP request is blocked for some reason.

The total number of ICMP packets sent can be found with the help of the following command:

```
$ tshark -r nmap.pcap -R "icmp" | grep "2.x" | wc -l
233
```

Displaying Statistics for a Specific Protocol

tshark allows you to display useful statistics about a specific protocol. The following command displays statistics about the HTTP protocol using an existing file with network data:

```
$ tshark -q -r http.pcap -R http -z http,tree
```

```
================================================
 HTTP/Packet Counter      value    rate     percent
------------------------------------------------
 Total HTTP Packets         118   0.017749
  HTTP Request Packets       66   0.009928    55.93%
   GET                       66   0.009928   100.00%
  HTTP Response Packets      52   0.007822    44.07%
   ???: broken                0   0.000000     0.00%
   1xx: Informational         0   0.000000     0.00%
   2xx: Success              51   0.007671    98.08%
    200 OK                   51   0.007671   100.00%
   3xx: Redirection           0   0.000000     0.00%
   4xx: Client Error          1   0.000150     1.92%
    404 Not Found             1   0.000150   100.00%
   5xx: Server Error          0   0.000000     0.00%
  Other HTTP Packets          0   0.000000     0.00%

================================================
```

All the work is done by the `-z` option, which is for calculating statistics, and the `-q` option, which is for disabling the printing of information per individual packet. The `-R` option discards all packets that do not match the specified filter before doing any other processing.

Here's another useful command that shows protocol hierarchy statistics:

```
$ tshark -nr ~/var/http.pcap -qz "io,phs"
```

Try it yourself to see the output!

Summary

If you have an in-depth understanding of display filters and a good knowledge of TCP/IP and networks, with the help of tshark or Wireshark, network-related issues will not longer be a problem.

It takes time to master tshark, but I think it will be time well spent.■

Mihalis Tsoukalos is a UNIX administrator, a programmer (UNIX and iOS), a DBA and a mathematician. You can reach him at http://www.mtsoukalos.eu or via Twitter: @mactsouk.

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

Resources

tshark: http://www.wireshark.org/docs/man-pages/tshark.html

Wireshark: http://www.wireshark.org

Display Filters Reference: http://www.wireshark.org/docs/dfref

Internetworking with TCP/IP, Volume I, Douglas E. Comer, Prentice Hall

CONCERNING CONTAINERS' CONNECTIONS: ON DOCKER NETWORKING

Use Docker and Weave to build container-based systems.

FEDERICO KEREKI

Containers can be considered the third wave in service provision after physical boxes (the first wave) and virtual machines (the second wave). Instead of working with complete servers (hardware or virtual), you have virtual operating systems, which are far more lightweight. Instead of carrying around complete environments, you just move applications, with their configuration, from one server to another, where it will consume its resources, without any virtual layers. Shipping over projects from development to operations also is simplified—another boon. Of course, you'll face new and different challenges, as with any technology, but the possible risks and problems don't seem to be insurmountable, and the final rewards appear to be great.

Docker is an open-source project based on Linux containers that is showing high rates of adoption. Docker's first release was only a couple years ago, so the technology isn't yet considered mature, but it shows much promise. The combination of lower costs, simpler deployment and faster start times certainly helps.

In this article, I go over some details of setting up a system based on several independent containers, each providing a distinct, separate role, and I explain some aspects of the underlying network configuration. You can't think about production deployment without being aware of how connections are made, how ports are used and how bridges and routing are set up, so I examine those points as well, while putting a simple Web database query application in place.

Basic Container Networking

Let's start by considering how Docker configures network aspects. When the Docker service dæmon starts, it configures a virtual bridge, `docker0`, on the host system (Figure 1). Docker picks a subnet not in use on the host and assigns a free IP address to the bridge. The first try is 172.17.42.1/16, but that could be different if there are conflicts. This virtual bridge handles all host-containers communications.

When Docker starts a container, by default, it creates a virtual interface on the host with a unique name, such as `veth220960a`, and an address within the same subnet. This new interface will be connected to the `eth0` interface on the container itself. In order to allow connections, iptables rules are added, using a `DOCKER`-named chain. Network address translation (NAT) is used to forward traffic to external hosts, and the host machine must be set up to forward IP packets.

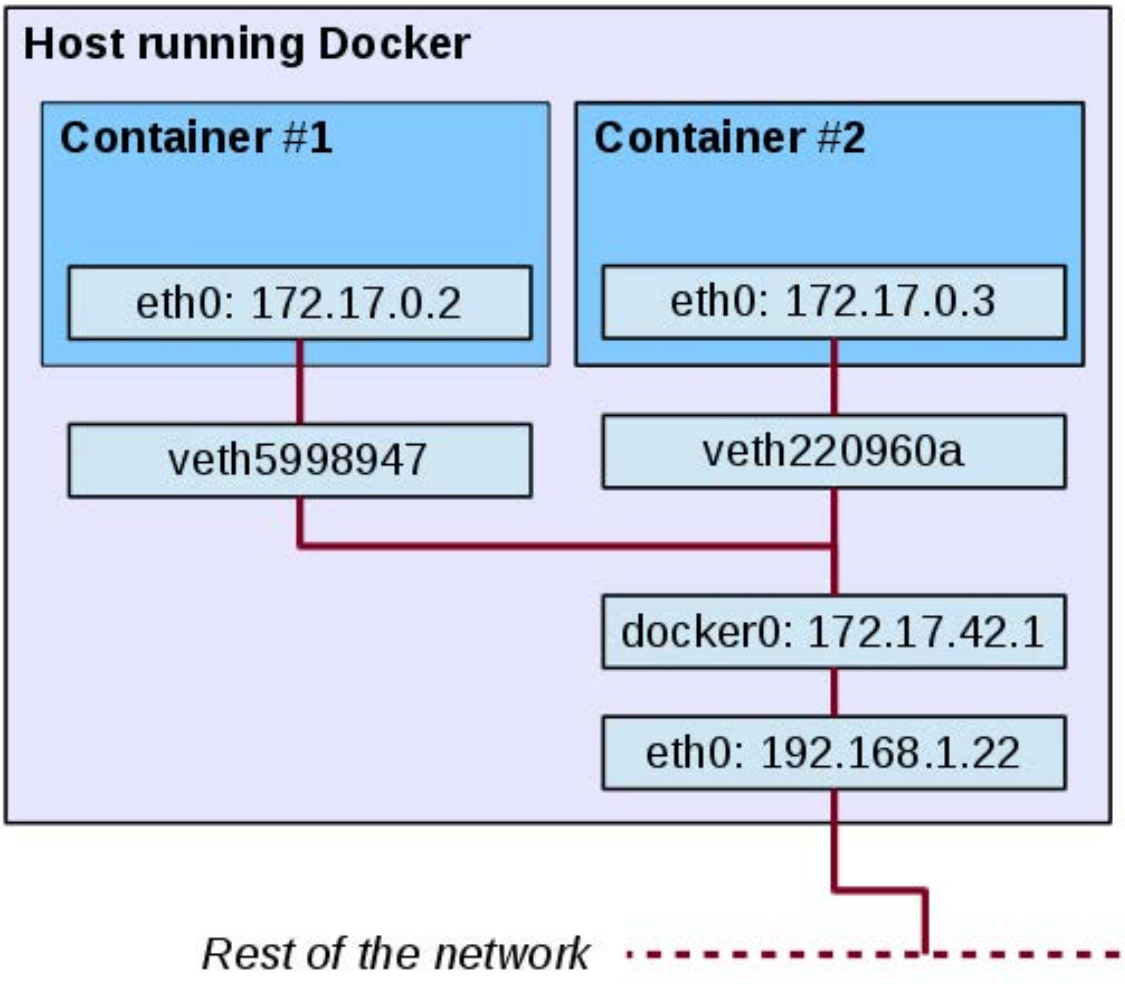


Figure 1. Docker uses a bridge to connect all containers on the same host to the local network.

The standard way to connect a container is in "bridged" mode, as described previously. However, for special cases, there are more ways to do this, which depend on the `-net` option for the `docker run` command. Here's a list of all available modes:

- `-net=bridge` — The new container uses a bridge to connect to the rest of the network. Only its exported public ports will be accessible from the outside.
- `-net=container:ANOTHER.ONE` — The new container will use the network stack of a previously defined container. It will share its IP address and port numbers.
- `-net=host` — This is a dangerous option. Docker won't separate the container's network from the host's. The new container will have full access to the host's network stack. This can cause problems and security risks!

Listing 1. The last three lines show Docker's special mount trick, so containers get information from Docker-managed host files.

```
root@4de393bdbd36:/var/www/html# findmnt -o TARGET,SOURCE
TARGET                  SOURCE
/                       /dev/mapper/docker-8:2-25824189-4de...822[/rootfs]
|-/proc                 proc
| |-/proc/sys           proc[/sys]
| |-/proc/sysrq-trigger proc[/sysrq-trigger]
| |-/proc/irq           proc[/irq]
| |-/proc/bus           proc[/bus]
| `-/proc/kcore         tmpfs[/null]
|-/dev                  tmpfs
| |-/dev/shm            shm
| |-/dev/mqueue         mqueue
| |-/dev/pts            devpts
| `-/dev/console        devpts[/2]
|-/sys                  sysfs
|-/etc/resolv.conf      /dev/sda2[/var/lib/docker/containers/4de...822/resolv.conf]
|-/etc/hostname         /dev/sda2[/var/lib/docker/containers/4de...822/hostname]
`-/etc/hosts            /dev/sda2[/var/lib/docker/containers/4de...822/hosts]
```

- `-net=none` — Docker won't configure the container network at all. If you want, you can set up your own iptables rules (see Resources if you're interested in this). Even without the network, the container could contact the world by shared directories, for example.

Docker also sets up each container so it will have DNS resolution information. Run `findmnt` inside a container to produce something along the lines of Listing 1. By default, Docker uses the host's /etc/resolv.conf data for DNS resolution. You can use different nameservers and search lists with the `--dns` and `--dns-search` options.

Now that you have an idea about how Docker sets up networking for individual containers, let's develop a small system that will be deployed via containers and then finish by working out how to connect all the pieces together.

Designing Your Application: the World Database

Let's say you need an application that will let you search for cities that include a given text string in their names. (Figure 2 shows a sample run.) For this example, I used the geographical information at GeoNames (see Resources) to create an appropriate database. Basically, you work with countries (identified by their ISO 3166-1 two-letter codes, such as "UY" for "Uruguay") and cities (with a name, a pair of coordinates and the country to which they belong). Users will be able to enter part of the city name and get all the matching cities (not very complex).

How should you design your mini-system? Docker is meant to package single applications, so in order to take advantage of containers, you'll run separate containers for each required role. (This doesn't necessarily imply that only a single process may run on a container. A container should

Figure 2. This sample application finds these cities with DARWIN in their names.

Listing 2. The Dockerfile to create the database server also pulls down the needed geographical data.

```
FROM mysql:latest
MAINTAINER Federico Kereki fkereki@gmail.com

RUN     apt-get update && \
        apt-get -q -y install wget unzip && \
        wget 'http://download.geonames.org/export/dump/countryInfo.txt' && \
        grep -v '^#' countryInfo.txt >countries.txt && \
        rm countryInfo.txt && \
        wget 'http://download.geonames.org/export/dump/cities1000.zip' && \
        unzip cities1000.zip && \
        rm cities1000.zip

RUN     echo "\
        CREATE DATABASE IF NOT EXISTS world;    \
        USE world;                              \
        DROP TABLE IF EXISTS countries;         \
        CREATE TABLE countries (                \
                id CHAR(2),                     \
                ignore1 CHAR(3),                \
                ignore2 CHAR(3),                \
                ignore3 CHAR(2),                \
                name VARCHAR(50),               \
                capital VARCHAR(50),            \
                PRIMARY KEY (id));              \
        LOAD DATA LOCAL INFILE 'countries.txt'  \
                INTO TABLE countries            \
                FIELDS TERMINATED BY '\t';      \
        DROP TABLE IF EXISTS cities;            \
        CREATE TABLE cities (                   \
                id NUMERIC(8),                  \
                name VARCHAR(200),              \
                asciiname VARCHAR(200),         \
                alternatenames TEXT,            \
                latitude NUMERIC(10,5),         \
                longitude NUMERIC(10,5),        \
                ignore1 CHAR(1),                \
                ignore2 VARCHAR(10),            \
                country CHAR(2));               \
        LOAD DATA LOCAL INFILE 'cities1000.txt' \
                INTO TABLE cities               \
                FIELDS TERMINATED BY '\t';      \
        " > mydbcommands.sql

RUN     echo "#!/bin/bash \n                    \
        mysql -h localhost -u root -p\$MYSQL_ROOT_PASSWORD <mydbcommands.sql \
        " >loaddata.sh && \
        chmod +x loaddata.sh
```

fulfill a single, definite role, and if that implies running two or more programs, that's fine. With this very simple example, you'll have a single process per container, but that need not be the general case.)

You'll need a Web server, which will run in a container, and a database server, in a separate container. The Web server will access the database server, and end users will need connections to the Web server, so you'll have to set up those network connections.

Start by creating the database container, and there's no need to start from scratch. You can work with the official MySQL Docker image (see Resources) and save a bit of time. The Dockerfile that produces the image can specify how to download the required geographical data. The `RUN` commands set up a loaddata.sh script that takes care of that. (For purists: a single longer `RUN` command would have sufficed, but I used three here for clarity.) See Listing 2 for the complete Dockerfile file; it should reside in an otherwise empty directory. Building the `worlddb` image itself can be done from that directory with the `sudo docker build -t worlddb .` command.

The `sudo docker images` command verifies that the image was created. After you create a container based on it, you'll be able to initialize the database with the `./loaddata.sh` command.

Searching for Data: Your Web Site

Now let's work on the other part of the system. You can take advantage of the official PHP Docker image, which also includes Apache. All you need is to add the `php5-mysql` extension to be able to connect to the database server. The script should be in a new directory, along with search.php, the complete code for this "system". Building this image, which you'll

Listing 3. The Dockerfile to create the Apache Web server is even simpler than the database one.

```
FROM php:5.6-apache
MAINTAINER Federico Kereki fkereki@gmail.com

COPY search.php /var/www/html/

RUN     apt-get update && \
        apt-get -q -y install php5-mysql && \
        docker-php-ext-install mysqli
```

name “worldweb”, requires the `sudo docker build -t worldweb .` command (Listing 3).

The search application search.php is simple (Listing 4). It draws a basic form with a single text box at the top,

Listing 4. The whole system consists of only a single search.php file.

```
<html>
<head>
<title>Cities Search</title>
</head>
<body>
<form action="search.php">
Search for: <input type="text" name="searchFor"
 ➥value="<?php echo $_REQUEST["searchFor"]; ?>">
<input type="submit" value="Go!">
<br><br>
<?php
if ($_REQUEST["searchFor"]) {
  try {
    $conn = mysqli_connect("MYDB", "root", "ljdocker", "world");
    $query = "SELECT countries.name, cities.name,
     ➥cities.latitude, cities.longitude ".
      "FROM cities JOIN countries ON cities.country=countries.id ".
      "WHERE cities.name LIKE ? ORDER BY 1,2";
    $stmt = $conn->prepare($query);

    $searchFor = "%".$_REQUEST["searchFor"]."%";
    $stmt->bind_param("s", $searchFor);
    $stmt->execute();
    $result = $stmt->get_result();

    echo "<table><tr><td>Country</td><td>City</td><td>Lat</td>
➥<td>Long</td></tr>";
    foreach ($result->fetch_all(MYSQLI_NUM) as $row) {
      echo "<tr>";
      foreach($row as $data) {
        echo "<td>".$data."</td>";
      }
      echo "</tr>";
    }
    echo "</table>";

  } catch (Exception $e) {
    echo "Exception " . $e->getMessage();
  }
}
?>
</form>
</body>
</html>
```

plus a "Go!" button to run a search. The results of the search are shown just below that in a table. The process is easy too—you access the database server to run a search and output a table with a row for each found city.

Both images are ready, so let's get your complete "system" running.

Linking Containers

Given the images that you built for this example, creating both containers is simple, but you want the Web server to be able to reach the database server. The easiest way is by linking the containers together. First, you start and initialize the database container (Listing 5).

Now, start the Web container, with `docker run -it -d -p 80:80 --link MYDB:MYDB --name MYWEB worldweb`. This command has a couple interesting options:

- `-p 80:80` — This means that port 80 (the standard HTTP port) from the container will be published as port 80 on the host machine itself.
- `--link MYDB:MYDB` — This means that the MYDB container (which you started earlier) will be accessible from the MYWEB container, also under the alias MYDB. (Using the database container name as the alias is logical, but not mandatory.) The MYDB container won't be visible from the network, just from MYWEB.

In the MYWEB container, /etc/hosts includes an entry for each linked

Listing 5. The database container must be started first and then initialized.

```
# su -
# docker run -it -d -e MYSQL_ROOT_PASSWORD=ljdocker
 ➥--name MYDB worlddb
fbd930169f26fce189a9d6020861eb136643fdc9ee73a4e1f114e0bfd0fe6a5c
# docker exec -it MYDB bash
root@fbd930169f26:/# dir
bin   cities1000.txt  dev    etc   lib
 ➥loaddata.sh  mnt   opt   root  sbin
 ➥srv  tmp  var
boot  countries.txt   entrypoint.sh  home  lib64  media
 ➥mydbcommands.sql  proc  run   selinux  sys  usr
root@fbd930169f26:/# ./loaddata.sh
Warning: Using a password on the command line interface
 ➥can be insecure.
root@fbd930169f26:/# exit
```

container (Listing 6). Now you can see how search.php connects to the database. It refers to it by the name given when linking containers (see the `mysqli_connect` call in Listing 4). In this example, MYDB is running at IP 172.17.0.2, and MYWEB is at 172.17.0.3.

The environment variables basically provide all the connection data for each linkage: what container it links to, using which port and protocol, and how to access each exported port from the destination container. In this case, the MySQL container just exports the standard 3306 port and uses TCP to connect. There's just a single problem with some of these variables. Should you happen to restart the MYDB container, Docker won't update them (although it would update the /etc/hosts information), so you must be careful if you use them!

Examining the iptables configuration,

Listing 6. Linking containers in the same server is done via /etc/hosts entries.

```
# su -
# docker exec -it MYWEB bash
root@fbff94177fc7:/var/www/html# cat /etc/hosts
172.17.0.3     fbff94177fc7
127.0.0.1      localhost
...
172.17.0.2     MYDB

root@fbff94177fc7:/var/www/html# export
declare -x MYDB_PORT="tcp://172.17.0.2:3306"
declare -x MYDB_PORT_3306_TCP="tcp://172.17.0.2:3306"
declare -x MYDB_PORT_3306_TCP_ADDR="172.17.0.2"
declare -x MYDB_PORT_3306_TCP_PORT="3306"
declare -x MYDB_PORT_3306_TCP_PROTO="tcp"
...
```

Listing 7. Docker adds iptables rules to link containers' ports.

```
# sudo iptables --list DOCKER
Chain DOCKER (1 references)
target     prot opt source       destination
ACCEPT     tcp  --  anywhere     172.17.0.3   tcp dpt:http
ACCEPT     tcp  --  172.17.0.3   172.17.0.2   tcp dpt:mysql
ACCEPT     tcp  --  172.17.0.2   172.17.0.3   tcp spt:mysql
```

you'll find a `DOCKER` new chain (Listing 7). Port 80 on the host machine is connected to port 80 (`http`) in the MYWEB container, and there's a connection for port 3306 (`mysql`) linking MYWEB to MYDB.

If you need to have circular links (container A links to container B, and vice versa), you are out of luck with standard Docker links, because you can't link to a non-running container! You might want to look into docker-dns (see Resources), which can create DNS records dynamically based upon running containers. (And in fact, you'll be using DNS later in this example when you set up containers in separate hosts.) Another possibility would imply creating a third container, C, to which both A and B would link, and through which they would be interconnected. You also could look into orchestration packages and service registration/discovery packages. Docker is still evolving in these areas, and new solutions may be available at any time.

You just saw how to link containers together, but there's a catch with this. It works only with containers on the same host, not on separate hosts. People are working on fixing this restriction, but there's an appropriate solution that can be used for now.

Weaving Remote Containers Together

If you had containers running on different servers, both local and remote ones, you could set up everything so the containers eventually could connect with each other, but it would be a lot of work and a complex configuration as well. Weave (currently on version 0.9.0, but quickly evolving; see Resources to get the latest version) lets you define a virtual network, so that containers can connect to each other transparently (optionally using encryption for added security), as if they were all on the same server. Weave behaves as a sort of giant switch, with all your containers connected in the same virtual network. An instance must run on each host to do the routing work.

Locally, on the server where it runs, a Weave router establishes a network bridge, prosaically named weave. It also adds virtual Ethernet connections from each container and from the Weave router itself to the bridge. Every time a local container needs to contact a remote one, packets are forwarded (possibly with "multi-hop" routing) to other Weave routers, until they are delivered by the (remote) Weave router to the remote container. Local traffic isn't affected; this forwarding applies only to remote

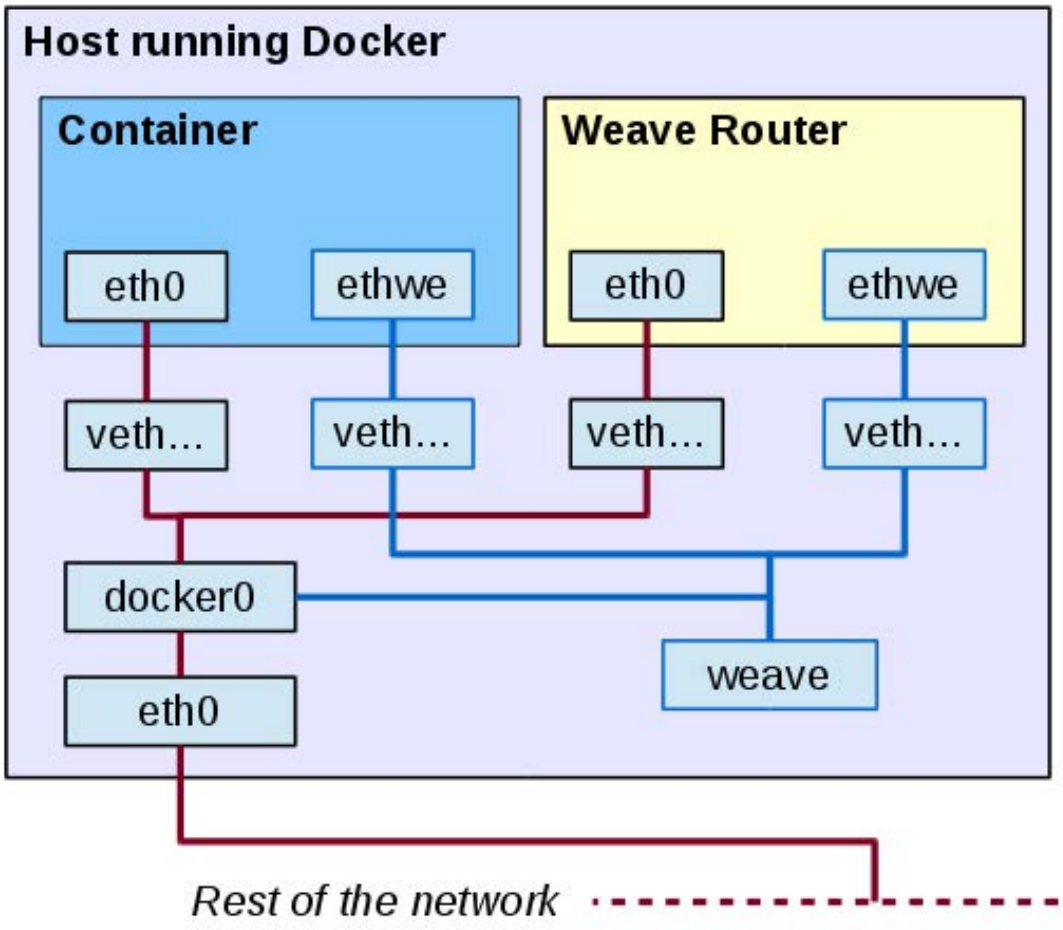


Figure 3. Weave adds several virtual devices to redirect some of the traffic eventually to other servers.

containers (Figure 3).

Building a network out of containers is a matter of launching Weave on each server and then starting the containers. (Okay, there is a missing step here; I'll get to that soon.) First, launch Weave on each server with `sudo weave launch`. If you plan to connect containers across untrusted networks, add a password (obviously, the same for all Weave instances) by adding the `-password some.secret.password` option. If all your servers are within a secure network, you can do without that. See the sidebar for a list of all the available `weave` command-line options.

When you connect two Weave routers, they exchange topology information to "learn" about the rest of the network. The gathered data is used for routing decisions to avoid unnecessary packet broadcasts. To detect possible changes and to work around any network problems that might pop up, Weave routers routinely monitor connections. To connect two routers, on a server, type the `weave connect`

`weave` Command-Line Options

- `weave attach` — Attach a previously started running Docker container to a Weave instance.
- `weave connect` — Connect the local Weave instance to another one to add it into its network.
- `weave detach` — Detach a Docker container from a Weave instance.
- `weave expose` — Integrate the Weave network with a host's network.
- `weave hide` — Revert a previous `expose` command.
- `weave launch` — Start a local Weave router instance; you may specify a password to encrypt communications.
- `weave launch-dns` — Start a local DNS server to connect Weave instances on distinct servers.
- `weave ps` — List all running Docker containers attached to a Weave instance.
- `weave reset` — Stop the running Weave instance and remove all of its network-related stuff.
- `weave run` — Launch a Docker container.
- `weave setup` — Download everything Weave needs to run.
- `weave start` — Start a stopped Weave instance, re-engaging it to the Weave topology.
- `weave status` — Provide data on the running Weave instance, including encryption, peers, routes and more.
- `weave stop` — Stop a running Weave instance, disengaging it from the Weave topology.
- `weave stop-dns` — Stop a running Weave DNS service.
- `weave version` — List the versions of the running Weave components; today (April 2015) it would be 0.9.0.

`the.ip.of.another.server` command. (To drop a Weave router, do `weave forget ip.of.the.dropped.host`.) Whenever you add a new Weave router to an existing network, you don't need to connect it to every previous router. All you need to do is provide it with the address of a single existing Weave instance in the same network, and from that point on, it will gather all topology information on its own. The rest of the routers similarly will update their own information in the process.

Let's start Docker containers, attached to Weave routers. The containers themselves run as before; the only difference is they are started through Weave. Local network connections work as before, but connections to remote containers are managed by Weave, which encapsulates (and encrypts) traffic and sends it to a remote Weave instance. (This uses port 6783, which must be open and accessible on all servers running Weave.) Although I won't go into this here, for more complex applications, you could have several independent subnets, so containers for the same application would be able to talk among themselves, but not with containers for other applications.

First, decide which (unused) subnet you'll use, and assign a different IP on it to each container. Then, you can `weave run` each container to launch it through Docker, setting up all needed network connections. However, here you'll hit a snag, which has to do with the missing step I mentioned earlier. How will containers on different hosts connect to each other? Docker's `--link` option works only within a host, and it won't work if you try to link to containers on other hosts. Of course, you might work with IPs, but maintenance for that setup would be a chore. The best solution is using DNS, and Weave already includes an appropriate package, WeaveDNS.

WeaveDNS (a Docker container on its own) runs over a Weave network. A WeaveDNS instance must run on each server on the network, with the `weave launch-dns` command. You must use a different, unused subnet for WeaveDNS and assign a distinct IP within it to each instance. Then, when starting a Docker container, add a `--with-dns` option, so DNS information will be available. You should give containers a hostname in the `.weave.local` domain, which will be entered automatically into the WeaveDNS registers. A complete network will

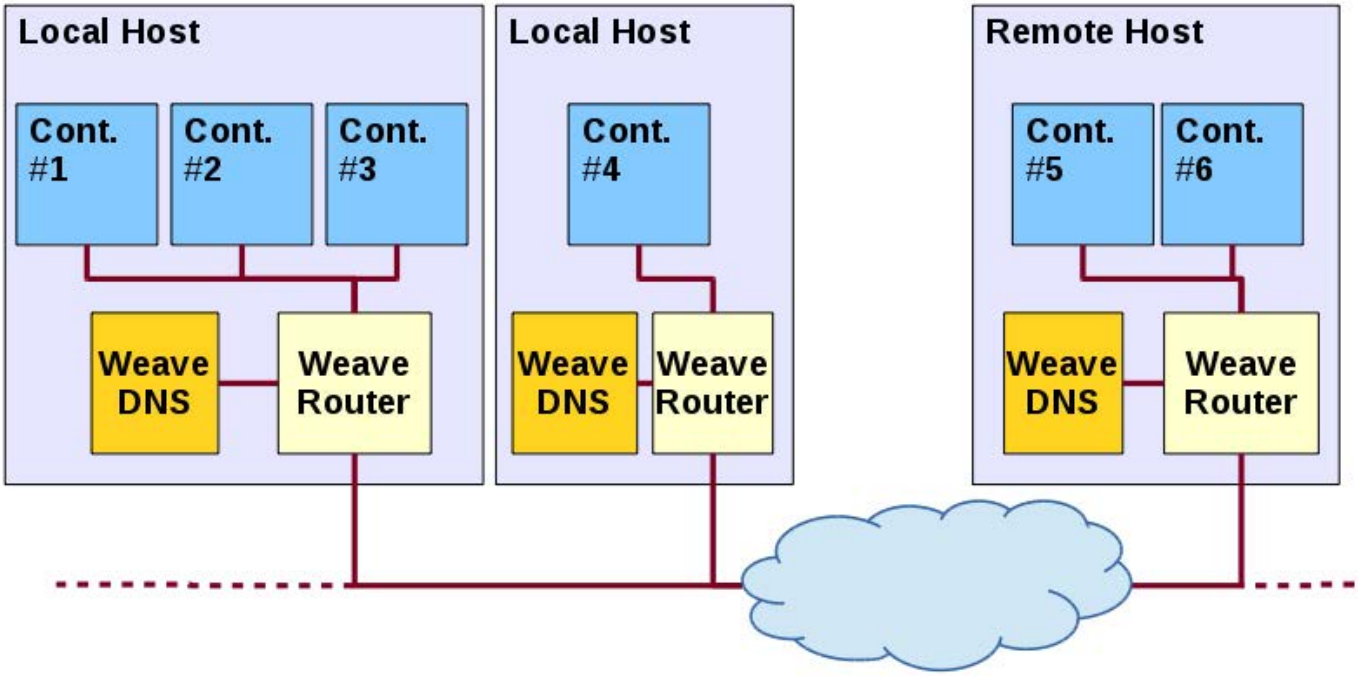


Figure 4. Using Weave, containers in local and remote networks connect to each other transparently; access is simplified with Weave DNS.

Listing 8. Getting the Weave network to run on two servers.

```
> # At 192.168.1.200 (OpenSUSE 13.2 server)
> su -
$ weave launch
$ weave launch-dns 10.10.10.1/24
$ C=$(weave run --with-dns 10.22.9.1/24 -it -d -e
 ➥MYSQL_ROOT_PASSWORD=ljdocker -h MYDB.weave.local --name MYDB worlddb)
$ # You can now enter MYDB with "docker exec -it $C bash"

> # At 192.168.1.108 (Linux Mint virtual machine)
> su -
$ weave launch
$ weave launch-dns 10.10.10.2/24
$ weave connect 192.168.1.200
$ D=$(weave run --with-dns 10.22.9.2/24 -it -d -p 80:80 -h
 ➥MYWEB.weave.local --name MYWEB worldweb)
```

look like Figure 4.

Now, let's get your mini-system to run. I'm going to cheat a little, and instead of a remote server, I'll use a virtual machine for this example. My main box (at 192.168.1.200)

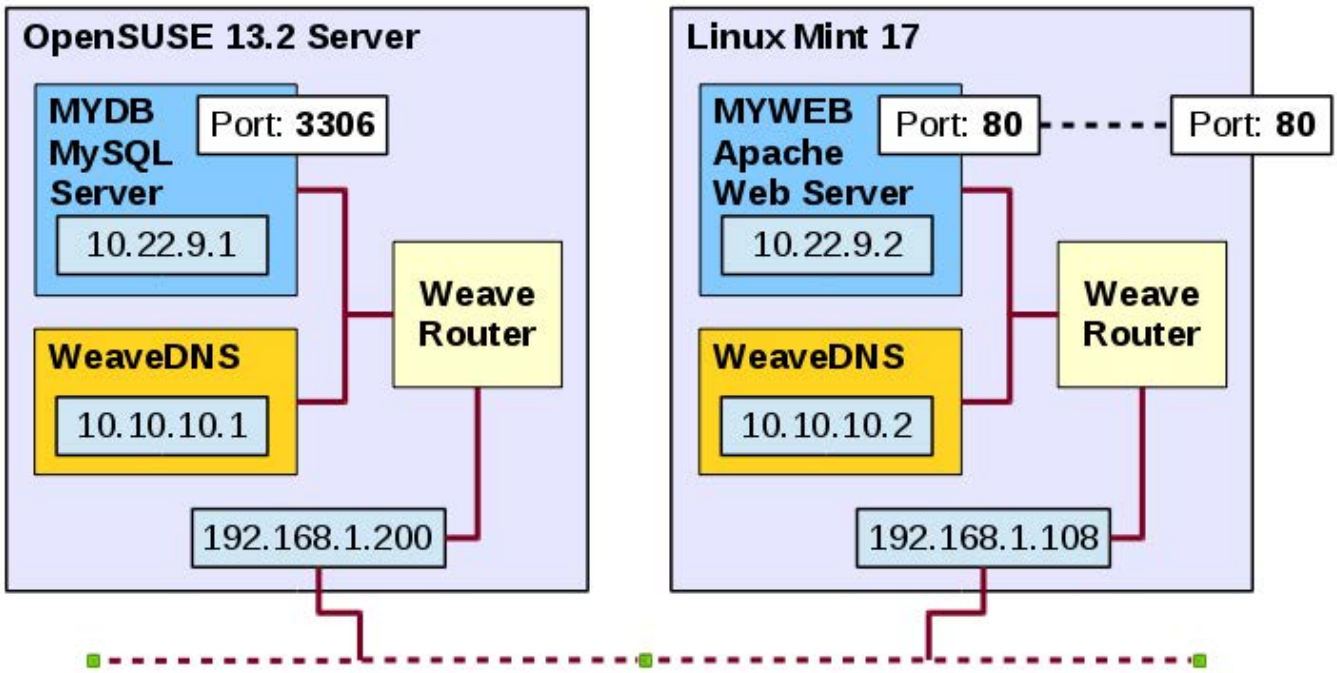


Figure 5. The final Docker container-based system, running on separate systems, connected by Weave.

runs OpenSUSE 13.2, while the virtual machine (at 192.168.1.108) runs Linux Mint 17, just for variety. Despite the different distributions, Docker containers will work just the same, which shows its true portability (Listing 8).

The resulting configuration is shown in Figure 5. There are two hosts, on 192.168.1.200 and 192.168.1.108. Although it's not shown, both have port 6783 open for Weave to work. In the first host, you'll find the MYDB MySQL container (at 10.22.9.1/24 with port 3306 open, but just on that subnet) and a WeaveDNS server at 10.10.10.1/24. In the second host, you'll find the MYWEB Apache+PHP container (at 10.22.9.2/24 with port 80 open, exported to the server) and a WeaveDNS server at 10.10.10.2/24. From the outside, only port 80 of the MYWEB container is accessible.

Because port 80 on the 192.168.1.108 server is directly connected to port 80 on the MYWEB server, you can access http://192.168.1.108/search.php and get the Web page you saw earlier (in Figure 2). Now you have a multi-host Weave network, with DNS services and remote Docker containers running as if they resided at the same host—success!

Conclusion

Now you know how to develop a multi-container system (okay, it's not very large, but still), and you've learned some details on the internals of Docker (and Weave) networking. Docker is still maturing, and surely even better tools will appear to simplify configuration, distribution and deployment of larger and more complex applications. The current availability of networking solutions for containers shows you already can begin to invest in these technologies, although be sure to keep up with new developments to simplify your job even further.■

Federico Kereki is a Uruguayan systems engineer with more than 25 years of experience doing consulting work, developing systems and teaching at universities. He is currently working as a UI Architect at Globant, using a good mixture of development frameworks, programming tools and operating systems—and FLOSS, whenever possible! He has written several articles on security, software development and other subjects for *Linux Journal*, IBM developerWorks and other Web sites and publications. He also wrote the *Essential GWT* book, in which you can find some security concerns for Web applications. You can reach Federico at fkereki@gmail.com.

Resources

Get Docker itself from http://www.docker.com. The actual code is at https://github.com/docker/docker.

For more detailed documentation on Docker network configuration, see https://docs.docker.com/articles/networking.

The docker-dns site is at https://www.npmjs.com/package/docker-dns, and its source code is at https://github.com/bnfinet/docker-dns.

The official MySQL Docker image is at https://registry.hub.docker.com/_/mysql. If you prefer, there also are official repositories for MariaDB (https://registry.hub.docker.com/_/mariadb). Getting it to work shouldn't be a stretch.

The Apache+PHP official Docker image is at https://registry.hub.docker.com/_/php.

Weave is at http://weave.works, and the code itself is on GitHub at https://github.com/weaveworks/weave. For more detailed information on its features, go to https://zettio.github.io/weave/features.html.

WeaveDNS is on GitHub at https://github.com/weaveworks/weave/tree/master/weavedns.

For more on articles on Docker in *Linux Journal*, read the following:

- David Strauss' "Containers—Not Virtual Machines—Are the Future Cloud": http://www.linuxjournal.com/content/containers—not-virtual-machines—are-future-cloud.
- Dirk Merkel's "Docker: Lightweight Linux Containers for Consistent Development and Deployment": http://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment.
- Rami Rosen's "Linux Containers and the Future Cloud": http://www.linuxjournal.com/content/linux-containers-and-future-cloud.

The geographical data I used for the example in this article comes from GeoNames http://www.geonames.org. In particular, I used the countries table (http://download.geonames.org/export/dump/countryInfo.txt) and the cities (with more than 1,000 inhabitants) table (http://download.geonames.org/export/dump/cities1000.zip), but there are larger and smaller sets.

WEBCASTS

Learn the 5 Critical Success Factors to Accelerate IT Service Delivery in a Cloud-Enabled Data Center

Today's organizations face an unparalleled rate of change. Cloud-enabled data centers are increasingly seen as a way to accelerate IT service delivery and increase utilization of resources while reducing operating expenses. Building a cloud starts with virtualizing your IT environment, but an end-to-end cloud orchestration solution is key to optimizing the cloud to drive real productivity gains.

> http://lnxjr.nl/IBM5factors

Modernizing SAP Environments with Minimum Risk—a Path to Big Data

Sponsor: SAP | Topic: Big Data

Is the data explosion in today's world a liability or a competitive advantage for your business? Exploiting massive amounts of data to make sound business decisions is a business imperative for success and a high priority for many firms. With rapid advances in x86 processing power and storage, enterprise application and database workloads are increasingly being moved from UNIX to Linux as part of IT modernization efforts. Modernizing application environments has numerous TCO and ROI benefits but the transformation needs to be managed carefully and performed with minimal downtime. Join this webinar to hear from top IDC analyst, Richard Villars, about the path you can start taking now to enable your organization to get the benefits of turning data into actionable insights with exciting x86 technology.

> http://lnxjr.nl/modsap

WHITE PAPERS

White Paper: JBoss Enterprise Application Platform for OpenShift Enterprise

Sponsor: DLT Solutions

Red Hat's® JBoss Enterprise Application Platform for OpenShift Enterprise offering provides IT organizations with a simple and straightforward way to deploy and manage Java applications. This optional OpenShift Enterprise component further extends the developer and manageability benefits inherent in JBoss Enterprise Application Platform for on-premise cloud environments.

Unlike other multi-product offerings, this is not a bundling of two separate products. JBoss Enterprise Middleware has been hosted on the OpenShift public offering for more than 18 months. And many capabilities and features of JBoss Enterprise Application Platform 6 and JBoss Developer Studio 5 (which is also included in this offering) are based upon that experience.

This real-world understanding of how application servers operate and function in cloud environments is now available in this single on-premise offering, JBoss Enterprise Application Platform for OpenShift Enterprise, for enterprises looking for cloud benefits within their own datacenters.

> http://lnxjr.nl/jbossapp

WHITE PAPERS

Linux Management with Red Hat Satellite: Measuring Business Impact and ROI

Sponsor: Red Hat | **Topic:** Linux Management

Linux has become a key foundation for supporting today's rapidly growing IT environments. Linux is being used to deploy business applications and databases, trading on its reputation as a low-cost operating environment. For many IT organizations, Linux is a mainstay for deploying Web servers and has evolved from handling basic file, print, and utility workloads to running mission-critical applications and databases, physically, virtually, and in the cloud. As Linux grows in importance in terms of value to the business, managing Linux environments to high standards of service quality — availability, security, and performance — becomes an essential requirement for business success.

> http://lnxjr.nl/RHS-ROI

Standardized Operating Environments for IT Efficiency

Sponsor: Red Hat

The Red Hat® Standard Operating Environment SOE helps you define, deploy, and maintain Red Hat Enterprise Linux® and third-party applications as an SOE. The SOE is fully aligned with your requirements as an effective and managed process, and fully integrated with your IT environment and processes.

Benefits of an SOE:

SOE is a specification for a tested, standard selection of computer hardware, software, and their configuration for use on computers within an organization. The modular nature of the Red Hat SOE lets you select the most appropriate solutions to address your business' IT needs.

SOE leads to:

- Dramatically reduced deployment time.
- Software deployed and configured in a standardized manner.
- Simplified maintenance due to standardization.
- Increased stability and reduced support and management costs.
- There are many benefits to having an SOE within larger environments, such as:
  - Less total cost of ownership (TCO) for the IT environment.
  - More effective support.
  - Faster deployment times.
  - Standardization.

> http://lnxjr.nl/RH-SOE

EOF

VINAY GUPTA

A Machine for Keeping Secrets?

Some lessons from the public past about the private future we won't have unless we take a new approach.

[I can't begin to describe all the things Vinay Gupta does. Fortunately, he does, at http://re.silience.com. There his leadership in many involvements are on display, where you can treat yourself to many hours of productive reading, listening and viewing—many involving breeds of Linux. After getting a little hang time with Vinay in London recently, I invited him to treat us to a guest EOF on any topic of his choice. He took the bait, and here it is.—Doc Searls]

The Lesson of Ultra and Mincemeat

The most important thing that the British War Office learned about cryptography was how to keep a secret: Enigma was broken at Bletchley Park early enough in World War II to change the course of the war—and of history. Now here's the thing: only if the breakthrough (called Ultra, which gives you a sense of its importance) was secret could Enigma's compromise be used to defeat the Nazis. Breaking Enigma was literally the "zero-day" that brought down an empire. Zero-day is a bug known only to an attacker. Defenders (those creating/protecting the software) have never seen the exploit and are, therefore, largely powerless to respond until they have done analysis. The longer the zero-day is kept secret, and its use undiscovered, the longer it represents absolute power.

Like any modern zero-day sold on the black market, the Enigma compromise had value only if it remained secret. The stakes were higher, but the basic template of the game—secret compromise, secret exploitation, doom on discovery—continues to be one basic form of

the computer security game to this day. The allies went to extraordinary lengths to conceal their compromise of the Enigma, including traps like Operation Mincemeat (planting false papers on a corpse masquerading as a drowned British military officer). The Snowden revelations and other work has revealed the degree to which this game continues, with many millions of taxpayer dollars being spent keeping illicit access to software compromises available to the NSA, GCHQ and all the rest. The first rule is not to reveal success in breaking your enemy's security by careless action; the compromise efforts that Snowden revealed had, after all, been running for many years before the public became aware of them.

Who Does Software Serve?

I would like to posit a fundamental problem in our attitude toward computer security. For a long time we basically have assumed that computers are tools much like any other. Pocket calculators and supercomputer clusters all share the same von Neumann architecture (another artifact of WWII). But the truth is that the computer also has been, from its very first real implementation, a machine for keeping and seeking secrets. This history applies not just to the Enigma machines that the British subverted to help defeat the Nazis, but also to IBM's Hollerith tabulators, used by the Nazis to identify Jews from census databases.

This is why the general utility model of computing we now use is notoriously difficult to secure. At a conceptual level, all programs are assumed to be direct representatives of the user (or superuser). This is fundamentally a mistake, a conceptual error that cannot be repaired by any number of additional layers piled on top of the fundamental error: software serves its authors, not its users. Richard M. Stallman, of course, understands this clearly but focuses mainly on freeing the source code, giving technical users control of their software. But beyond the now-rusty saw of "with enough eyes, all bugs are shallow", the security community as a whole has not gone back to basics and assigned the intentionality of software correctly: to its authors, rather than to its users. Once we admit that software works for those who wrote it, rather than the hapless ones running it, many of the problems of managing computer security get much clearer, if not easier! Furthermore, there is always the gremlin: discordia manifested as bugs. Software behaviors that no human

The fact that in the 21st century we still download and run programs that have arbitrary access to all of our personal files, data and often deep access to our operating systems is frankly madness.

intended are not only common, but ubiquitous. In these cases, software serves neither the user nor the author, but silently adds to the entropy of the universe all by itself.

Imagine if all the people that wrote the software you use every day were made visible. If you run a fully-free computer, right down to the BIOS, you would generally expect to see a group of people who are fully on your side. But then there is the router, and the firmware in your mouse and your telephone's baseband processor, and indeed the epic maze of software that powers the electrical grid to which your devices must connect, and so on. In truth, we do not like or trust many of the people writing the software on which our lives depend in so many ways. The fact that in the 21st century we still download and run programs that have arbitrary access to all of our personal files, data and often deep access to our operating systems is frankly madness. I'm not discussing sandboxing or virtual environments—these may be answers, but let us first clearly state the question: who does this machine serve?

The machine serves the authors of the software, not the person choosing to run it. If you have recently handed over permissions you were not entirely happy with while installing software on an Android phone, you have felt a sense of "No, I do not want you to do that—that's your desire, not mine!" Often we do not entirely trust those authors, their software or the hardware on which it runs. We literally cannot trust our possessions. Nobody wants to carry a snitch in their pocket, and yet we all do.

In an ideal world, all of our systems (and perhaps not only technological ones) would obey the Principle of Least Privilege. Rather than granting large, abstract powers to code (or other systems) and trusting there to be no bugs, we could grant powers in a more narrow way. Consider the all-too-typical "programs can see the

entire filesystem" permission we grant to nearly all software dæmons: when something goes wrong, it results in disasters like Squid deleting your root filesystem when restarting. Why does Squid need the power to do that? Why even hold those keys?

So What Happens If We Choose Not to Trust Everybody?

There was a path not taken: capability-based operating systems. Capability-based operating systems really are machines for keeping secrets. They assume that all code is written by people we do not trust, and that the code may contain damaging bugs, if not outright assaults. "All code is untrusted code" creates a completely different role for the operating system in protecting users from the tools they themselves have downloaded. This is a realistic model of what software is like, an explicit model of distrust, unlike the vague trust we feel when installing software many other people are using, thinking "with enough eyes all bugs are shallow, so I'm sure this will be fine." That's not a great model of trust! Capability-based systems assume that all code may be evil, even code the user writes (bugs!), so it is, by default, untrusted in the most profound way.

A bare program can do nothing—no network, no filesystem access, nothing until it is granted permissions, and the operating system provides a smooth interface for an extremely granular approach to granting and managing these permissions. This is not like the Android model, where the application has access to high-level constructs like "your address book"; rather, this extends all the way from that level down to a low-level file-by-file access control model.

In an object capability model, a program cannot open a directory or search for files without a go-ahead from a user, although usually that go-ahead is implicit. For example, passing an open file handle as a command-line argument would grant the relevant program access to that file. A shell could manage those open file handles seamlessly for the user, opening files and passing their handles in a way that is seamless and transparent to the user. Without that permission, all attempts to access a file simply will be met by failure; as far as the software is concerned, that resource simply does not exist.

To get to this security position, one has to be very clear about the politics of software. Why was this code written? Who does it serve? Toward whose advantage does it

work? *Cui bono?* Even if the only illicit advantage is a bug or two serving only the increase of entropy in the universe, we must admit that, when we get right down to it, if you did not write the software yourself, it's pretty much like giving somebody the keys to your house. But, it does not have to be this way.

This line of argument gives me an uneasy feeling every time I write it down using a modern Linux machine, knowing full well that every single thing I've used `apt-get install` to put on my computer could relaying my key presses, because once I install it, it acts as if it were me, whether I want that behavior or not, moment by moment.

The computer is a machine for keeping and seeking secrets.

Is There an Evolutionary Upgrade Path?

I'm not suggesting that we throw out everything that has been done and start again. My suspicion is that to a very substantial degree, with a concerted effort, ideas from the capability-based systems could be comprehensively re-integrated into Linux. Security-Enhanced Linux uses these terms, but without having the full object capability model available. Post-Snowden, now fully aware of how pervasive advanced persistent threat type attacks are on our machines, it seems like it should be possible to start reconsidering what we think we know about software and security for the new operating environment in which we find ourselves. But, can we work out from the long-established SELinux project to those goals?

This is not a straightforward proposition for two reasons: the current limitations of SELinux and the problem of who wrote SELinux.

SELinux currently builds on top of Linux's POSIX capabilities, which are a way of dividing up the power of root into a set of compartments, avoiding the use of setuid. This is important because, in the event of a privilege escalation bug, the illicitly gained privileges aren't the full power of root, but a constrained subset of those powers: notionally, under SELinux, breaking "sudo tail /log/stuff" won't give you access to install new software in the network stack or any other unrelated thing. You might be able to read what you should not, but you can't write to a damn thing. However, the POSIX capability model in SELinux is (confusingly) not the fully blown object capabilities model, because it does not allow for delegation and (as far as I can

tell from the docs!) applies only to superuser privileges. It comes from a different theoretical base.

In a full-blown object capability system with delegation, like the research operating systems lineage of GNOSIS, KeyKos (used in production systems), EROS, CapROS and Coyotos, a program (let's say a ported version of GIMP) is run and is blind. It can't see the filesystem, the network stack or anything else; it exists in the void. A user opens a filesystem browser and passes a file to the program, and along for the ride go a necessary set of access keys that are passed invisibly by the operating system. These can be implemented as cryptographic tokens, a little like Kerberos, or as an operating-system-level grant of permissions. Now GIMP can see that file. It can pass the token to the operating system like a filename or handle, which then will open/close the file, and so on. Furthermore, however, when permitted, it can pass that token to another program. Want to run an external filter that only exists as a command-line utility? GIMP can pass that token over to an external utility; the authority to see the file is a transferable asset. And, this model extends across computers. A token for, say, Wi-Fi access can be passed from one machine to another as a delegated authority, and authorities can be chained and combined. Something can act on your behalf (it has the token) without being you as far as the software is concerned.

Say a printer requires network access from one user, and a file to print from another. Normally this is a little tricky. You usually wind up with one user e-mailing the file to another, because the printer expects to work for a single individual: authentication is authorization. In an object capabilities system, the printer (or device, or program) simply assembles capabilities until it has what it needs to do the job. This completely breaks the model in which people are (all too commonly) passing passwords, which have infinite power, to people that they actually want to do one specific job on a remote machine. The granularity of control is so much finer, and delegation fits our real-world security use cases so much better, than the password identity model. You may still use a password to log in, but after that, it's delegated capabilities to manage untrusted software (and untrusted people) all the way down. Doesn't that sound like a better way of doing business in our unsafe times?

Now for the second problem: who wrote SELinux?

NSA Security-Enhanced Linux is a

The NSA team behind SELinux released it under a FOSS license at year end 2000. Now we need to ask ourselves, what is it?

set of patches to the Linux kernel and some utilities to incorporate a strong, flexible mandatory access control (MAC) architecture into the major subsystems of the kernel. It provides an enhanced mechanism to enforce the separation of information based on confidentiality and integrity requirements, which allows threats of tampering and bypassing of application security mechanisms to be addressed and enables the confinement of damage that can be caused by malicious or flawed applications. It includes a set of sample security policy configuration files designed to meet common, general-purpose security goals.

The NSA team behind SELinux released it under a FOSS license at year end 2000. Now we need to ask ourselves, what is it? We have strong reason to suspect from the Snowden documents that long-term attempts to compromise open and academic security work are part of the NSA's mandate—for example, subverting the National Institute for Standards and Technology cryptography credentialing process by introducing flawed algorithms and getting NIST to sign off on them as credible standards. And, as bitter experience with OpenSSL has shown us (Heartbleed) "with enough eyes, all bugs are shallow" in fact buys us very little security. OpenSSL was extremely under-funded ($2,000 per year!) until the Heartbleed bug brought the world's focus to the plight of OpenSSL's underpaid development team. GPG's development team has been similarly underfunded. This is not working.

So now we have to look at SELinux in much the same light as (sadly) the Tor project—FOSS security tools funded by deeply untrusted sources with a long history of coercive undermining of security, privacy and user control of their own computers. Do we have enough eyes to be able to trust the code under these circumstances? SELinux is one of only four systems that can provide this kind of control under Linux (the others being AppArmor, Smack and Tomoyo) using the same underlying POSIX

capabilities. Are eyeballs the right metric? Is that enough eyeballs?

These ongoing questions cut to the heart of our security development processes. I hope in the next few years we find a good way of funding the necessary security work that we, and increasingly the entire world, depend on day-in, day out.

Enter Capsicum. Capsicum is a fairly serious push to integrate deeply a full implementation of capability-based security into FreeBSD. There is an ongoing effort to create Capsicum for Linux, and work is continuing. This seems like a sensible and obvious approach to providing users with an appropriate level of security for the post-Snowden environment we now know we operate in. Because any flawed piece of software assumes full permissions as the user or as the superuser, depending on whether it was a user agent like a browser or a dæmon that got compromised (roughly speaking), we have a challenge. Either perfectly tighten every bolt on an entire starship and miss not a single one, or install bulkheads and partition the space into safe areas, so that, if there is a compromise, it is not systemic.

Bolt-tightening approaches to security are certainly necessary, but I cannot see any way to offer users comprehensive privacy and security on devices that act as secure end points without capability-based operating system concepts coming to Linux in a big way, and right now, that appears to mean Capsicum is the only game in town. This is a matter of some urgency. End-point security weaknesses are really starting to have systemic effects. Let me explain.

I would be much more comfortable if I did not have to trust the thousands of apps on my laptop as much as I do today, and I have very specific reasons for my unease: private key management. Right now I work for Ethereum, a FOSS project producing software to build a global-distributed metacomputer that includes a blockchain database. It's a bit like bitcoin, but it uses the database to store executable software in the form of "contracts" (little scripts you trust to manage your assets).

I think Ethereum is pretty cool. We expect to see an awful lot of very interesting use cases for the platform. Many people may wind up deeply relying on services using that software. For example, comprehensive solutions to the increasing mess that is DNS and issuing SSL certificates could come out of a global-distributed database with scripting: register a domain on the blockchain and

self-publish your certificates using the same keys you used to pay for the domain name registration. Simple. Namecoin already has given some sense of what is possible, and I have no doubt there is far more to come.

There is more at risk than individual users being compromised and having their contracts spoofed. In a distributed system, there is a monoculture risk. If we have individual users being hacked because their laptops slip a version behind bleeding-edge security patches, that's bad enough. We have all heard tales of enormous numbers of bitcoins evaporating into some thief's pockets. But if we have only three major operating systems, run by >99% of our users, consider the risk that a zero-day exploit could be used to compromise the entire network's integrity by attacking the underlying consensus algorithms. If enough computers on the network say 2+2 = 5, the nature of blockchains is that 2+2 not only equals 5, but it always will equal five.

Huge disruption to everyday life could result from an error like this if blockchain technology winds up being the solution to DNS and SSL namespace issues (a conclusion I consider likely and that I may write up in future for this journal). We could lose basic connectivity to a large part of the Internet in the event that the consensus protocols are attacked by compromised machines. If a zero-day was used to construct malware that abused or just published private keys, that also could have disastrous effects not only for individual users, but also for the decentralized databases as a whole. If blockchains turn out to be vital to the Internet of Things (IBM has an Ethereum-based project, ADEPT, looking at blockchains and the IoT), then even if the blockchain itself and our software are secure, we have hostages to fortune in the form of the PCs being used to manage the keys and the code on which all of this value and utility is stored.

There is an urgent problem that users are starting to store very real value on their machines, not simply in the form of indirect access to value via banking Web sites, but as direct access to their own private keys and a political role in the consensus algorithms on which the entire blockchain is formed. This is all getting a lot more systemic and potentially serious than having somebody read one's e-mail or private journal.

Right now the weakest link in the ongoing adoption of blockchain technology is operating system security. The fear that one will get

hacked discourages many users from using software on their own machines to do their computation (to use the Stallman model). Instead, third-party Web sites operating theoretically more secure wallets are common—essentially people are storing their bitcoin and similar "in the cloud" because they do not trust their own PCs enough to store value in a decentralized fashion. This negates many of the decentralization benefits of blockchain-based approaches. This is clearly a massive problem when viewed from Stallman's usual mode of analysis.

Surely at this point in history it's time for us to return computing to its roots. The computer is a machine for keeping my secrets: my banking details, my cryptocurrency holdings, my private keys controlling software acting on my behalf on the blockchain, or in the Internet of things, or in virtual reality, or in any other setting. It's becoming increasingly necessary that users can actually store value on their own machines, and right now, playing whackamole with zero-day exploits is not a good enough security model for this revolution to continue. We have to return to the hard question of how do I stop other people from telling my computer what to do without first asking me?

Encryption without secure endpoints

isn't going to help very much, and right now, operating system security is the weakest link. I look forward to your ideas about how we might address these issues in an ongoing fashion—both as a question of awareness raising and funding models, and for the long, hard quest for genuine security for average users. Ordinary people should be able to store value on their home computers without feeling that they have automatically left the front door open with the keys in the lock.

How can we provide people with an equivalent level of protection for their bank accounts or their bitcoin holdings? This is the real challenge meeting cryptocurrencies, blockchains and even the Internet of Things. If we cannot trust the users' devices, how can we give them all this access to and power over users' lives?

The revolution is stalling for ordinary users because they cannot trust their operating systems to protect their private keys and thereby their accounts. What now?

Acknowledgements

I'd like to thank a few people for their input: Alan Karp of HP Labs, and Ben Laurie and David Drysdale of Google (and Capsicum).

And thanks to Doc too, for inviting me to do this.■

Vinay Gupta is the release coordinator for Ethereum, a FOSS scriptable blockchain platform. He was a cypherpunk and coder in the 1990s. His main area of personal interest is using technology to end poverty globally, using an engineering-led approach. This work has taken him through disaster relief (Hexayurt Project, an emergency shelter that was one of the first Open Hardware projects), critical infrastructure and state failure modelling (Simple Critical Infrastructure Maps), and cryptography/governance (CheapID). His work is in use for Burning Man (>2000 hexayurts constructed per year), military humanitarians (STAR-TIDES project) and with poverty activists around the world.

Resources

British War Office: https://en.wikipedia.org/wiki/War_Office

Enigma: http://www.bbc.co.uk/history/topics/enigma

Bletchley Park: http://www.bletchleypark.org.uk/content/hist/worldwartwo/industrialisation.rhtm

Ultra: https://en.wikipedia.org/wiki/Ultra

"How Zero-Day Exploits Are Bought & Sold": http://null-byte.wonderhowto.com/inspiration/zero-day-exploits-are-bought-sold-0159611

Operation Mincemeat: https://en.wikipedia.org/wiki/Operation_Mincemeat

The Man Who Never Was (a film about Operation Mincemeat): https://en.wikipedia.org/wiki/The_Man_Who_Never_Was

"NSA purchased zero-day exploits from French security firm Vupen":
http://www.zdnet.com/article/nsa-purchased-zero-day-exploits-from-french-security-firm-vupen

IBM and the Holocaust: https://en.wikipedia.org/wiki/IBM_and_the_Holocaust

Principle of Least Privilege: http://en.wikipedia.org/wiki/Principle_of_least_privilege

"restarting a testing build of squid results in deleting all files in a hard-drive": https://bugzilla.redhat.com/show_bug.cgi?id=1202858

Capability-Based Security: https://en.wikipedia.org/wiki/Capability-based_security

From Objects to Capabilities: Capability Operating Systems: http://erights.org/elib/capability/ode/ode-capabilities.html

Security-Enhanced Linux: https://en.wikipedia.org/wiki/Security-Enhanced_Linux

POSIX Capabilities: https://friedhoff.org/posixfilecaps.html

"Using POSIX capabilities in Linux, part one (avoiding the use of setuid)":
http://archlinux.me/brain0/2009/07/28/using-posix-capabilities-in-linux-part-one

EROS (The Extremely Reliable Operating System): http://www.eros-os.org/eros.html

CapROS (The Capability-Based Reliable Operating System): http://www.capros.org

The Coyotos Secure Operating System: http://www.coyotos.org

"Explain Like I'm 5: Kerberos": http://www.roguelynn.com/words/explain-like-im-5-kerberos

Who Wrote SELinux?: https://www.nsa.gov/research/selinux

Patch: https://en.wikipedia.org/wiki/Patch_(computing)

Linux Kernel: https://en.wikipedia.org/wiki/Linux_kernel

Mandatory Access Control (MAC): https://en.wikipedia.org/wiki/Mandatory_access_control

"Tech Titans Launch 'Core Infrastructure Initiative' to Secure Key Open Source Components":
http://www.securityweek.com/tech-titans-launch-core-infrastructure-initiative-secure-key-open-source-components

Heartbleed: https://en.wikipedia.org/wiki/Heartbleed

"The Internet Is Being Protected by Two Guys Named Steve":
http://www.buzzfeed.com/chrisstokelwalker/the-internet-is-being-protected-by-two-guys-named-st#.earzPzxNAB

"US government increases funding for Tor, giving $1.8m in 2013":
http://www.theguardian.com/technology/2014/jul/29/us-government-funding-tor-18m-onion-router

Clipper Chip: https://en.wikipedia.org/wiki/Clipper_chip

Google Transparency Report: https://www.google.com/transparencyreport/userdatarequests/US

"Capsicum: practical capabilities for UNIX": https://lwn.net/Articles/482858

Capsicum for Linux: https://www.cl.cam.ac.uk/research/security/capsicum/linux.html

Linux Kernel with Capsicum Support: https://github.com/google/capsicum-linux

Ethereum: https://ethereum.org

Smart Contract: https://en.wikipedia.org/wiki/Smart_contract

Dapps for Beginners (Ethereum contract tutorials): https://dappsforbeginners.wordpress.com

Namecoin: https://namecoin.info

"A history of bitcoin hacks": http://www.theguardian.com/technology/2014/mar/18/history-of-bitcoin-hacks-alternative-currency

Device democracy—Saving the future of the Internet of Things: http://public.dhe.ibm.com/common/ssi/ecm/en/gbe03620usen/GBE03620USEN.PDF

Endpoint Security: http://searchmidmarketsecurity.techtarget.com/definition/endpoint-security

peer1 hosting

Where every interaction matters.

break down your innovation barriers

power your business to its full potential

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

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

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

Want more on cloud?
Call: 844.855.6655 | go.peer1.com/linux | Vew Cloud Webinar:

Public and Private Cloud | Managed Hosting | Dedicated Hosting | Colocation