

# LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community



WATCH:  
ISSUE  
OVERVIEW



DECEMBER 2016 | ISSUE 272

<http://www.linuxjournal.com>

## Provisioning Docker with Puppet



TRAIN YOUR  
COMPUTER  
WITH A MACHINE-  
LEARNING MODEL

WRITE A SHELL  
SCRIPT TO FIND  
THE MOON PHASE

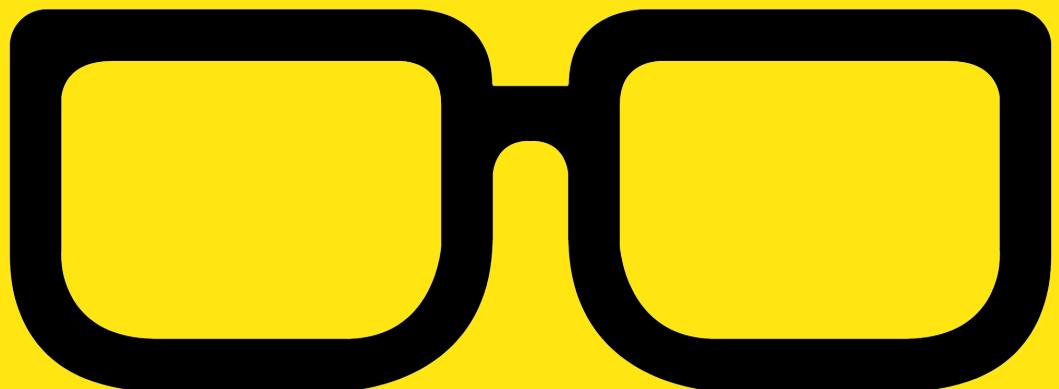
**Raspberry Pis**  
and IPv6 Networking

**Server Orchestration**  
with the MCollective Tool

EOF:  
PROTECTING YOUR  
PRIVACY ONLINE

Practical books  
for the most technical  
people on the planet.

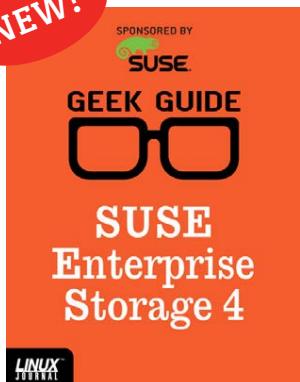
# GEEK GUIDES



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

<http://geekguide.linuxjournal.com>

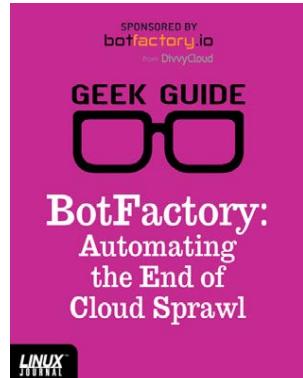
**NEW!**



## SUSE Enterprise Storage 4

**Author:**  
Ted Schmidt

**Sponsor:**  
SUSE



## BotFactory: Automating the End of Cloud Sprawl

**BotFactory:**  
Automating  
the End of  
Cloud Sprawl

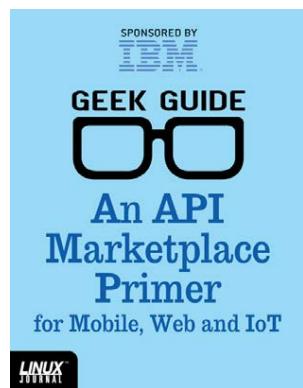
**Author:**  
John S. Tonello  
**Sponsor:**  
BotFactory.io



## Containers 101

**Author:**  
Sol Lederman

**Sponsor:** Puppet



## An API Marketplace Primer for Mobile, Web and IoT

**An API  
Marketplace  
Primer  
for Mobile,  
Web and IoT**

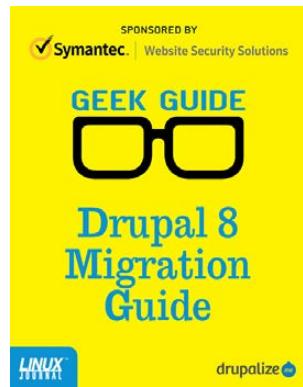
**Author:**  
Ted Schmidt  
**Sponsor:**  
IBM



## Public Cloud Scalability for Enterprise Applications

**Author:**  
Petros Koutoupis

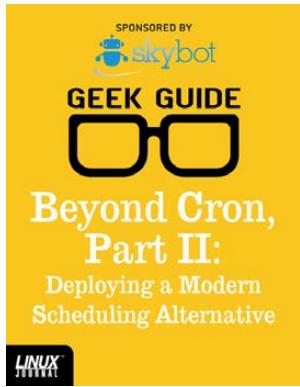
**Sponsor:**  
SUSE



## Drupal 8 Migration Guide

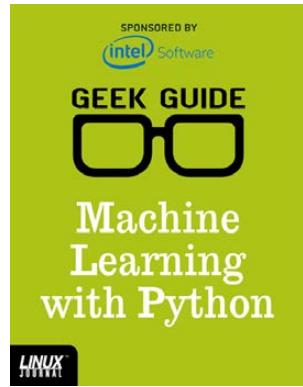
**Drupal 8  
Migration  
Guide**

**Author:**  
Drupalize.me  
**Sponsor:**  
Symantec



## Beyond Cron, Part II: Deploying a Modern Scheduling Alternative

**Author:**  
Mike Diehl  
**Sponsor:** Skybot



## Machine Learning with Python

**Machine  
Learning  
with Python**

**Author:**  
Reuven M. Lerner  
**Sponsor:**  
Intel

# CONTENTS

DECEMBER 2016  
ISSUE 272

## FEATURES

### 72 Provisioning Docker with Puppet

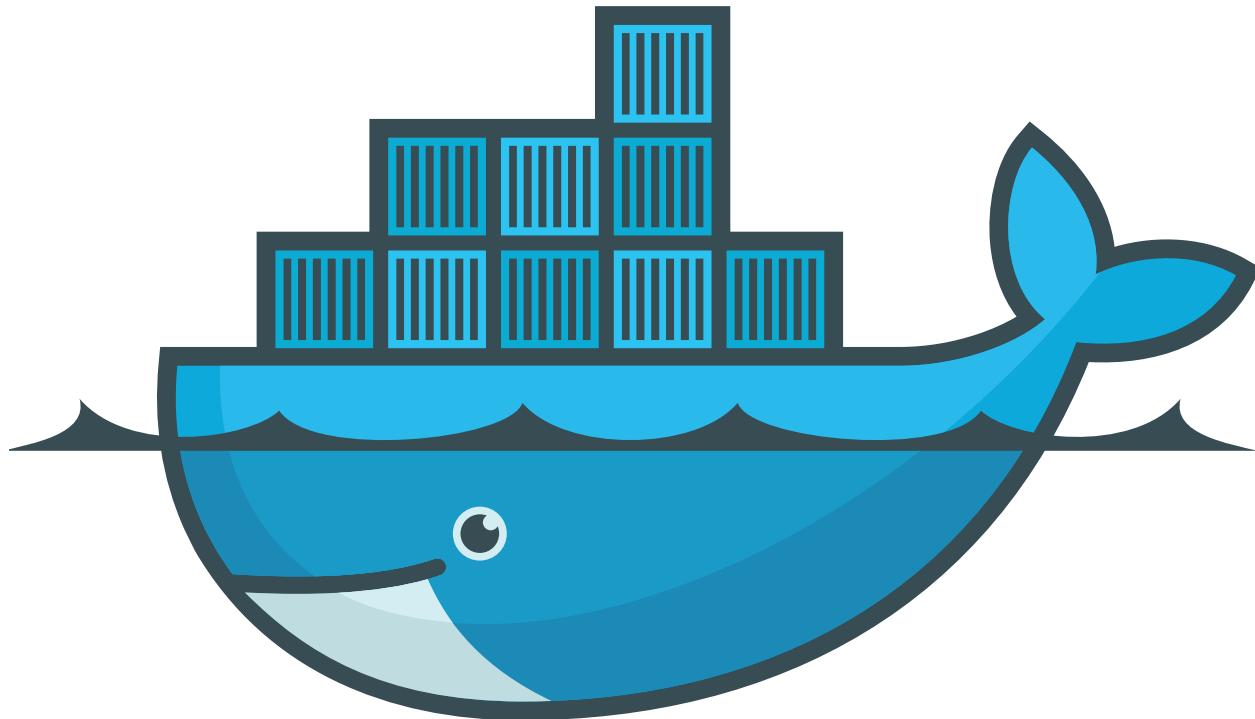
Learn how to automate the installation of Docker services onto selected servers using a little regular expression magic.

**Todd Jacobs**

### 82 Low Power Wireless: Routing to the Internet

How to get two Raspberry Pis to communicate over a 6LoWPAN network.

**Jan Newmarch**



## COLUMNS

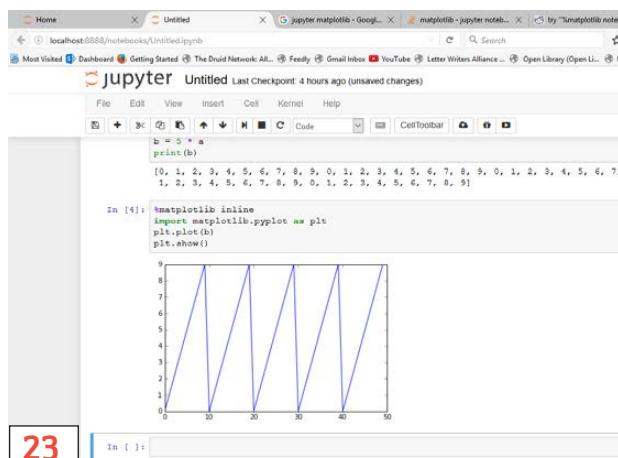
- 32 Reuven M. Lerner's At the Forge**  
Teaching Your Computer
- 42 Dave Taylor's Work the Shell**  
The Current Phase of the Moon
- 48 Kyle Rankin's Hack and /**  
Orchestration with MCollective
- 54 Shawn Powers' The Open-Source Classroom**  
The Family Dashboard in PHP
- 98 Doc Searls' EOF**  
Progress on Privacy

### ON THE COVER

- Provisioning Docker with Puppet, p. 72
- Raspberry Pis and IPv6 Networking, p. 82
- Server Orchestration with the MCollective Tool, p. 48
- Train Your Computer with a Machine-Learning Model, p. 32
- Write a Shell Script to Find the Moon Phase, p. 42
- EOF: Protecting Your Privacy Online, p. 98

## IN EVERY ISSUE

- 8 Current\_Issue.tar.gz**
- 10 Letters**
- 14 UPFRONT**
- 30 Editors' Choice**
- 64 New Products**
- 105 Advertisers Index**



The screenshot shows a Jupyter Notebook interface with several cells of code and their corresponding outputs. The first cell contains:

```
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

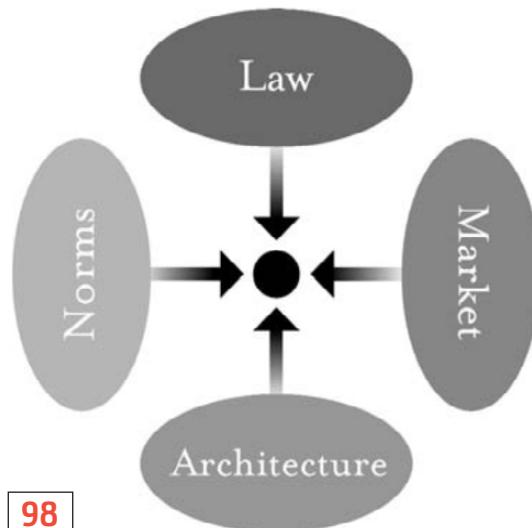
The second cell contains:

```
In [4]: %matplotlib inline
import matplotlib.pyplot as plt
plt.plot(b)
plt.show()
```

The output of the second cell is a line plot showing a sawtooth wave pattern.

23

98



# LINUX JOURNAL™

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



**ENJOY:**

**Timely delivery**

**Off-line reading**

**Easy navigation**

**Phrase search  
and highlighting**

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

**Embedded videos**

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

**SUBSCRIBE TODAY!**

# LINUX JOURNAL

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

## Contributing Editors

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

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

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

PO Box 980985, Houston, TX 77098 USA

## Editorial Advisory Panel

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

## Advertising

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

## Subscriptions

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

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

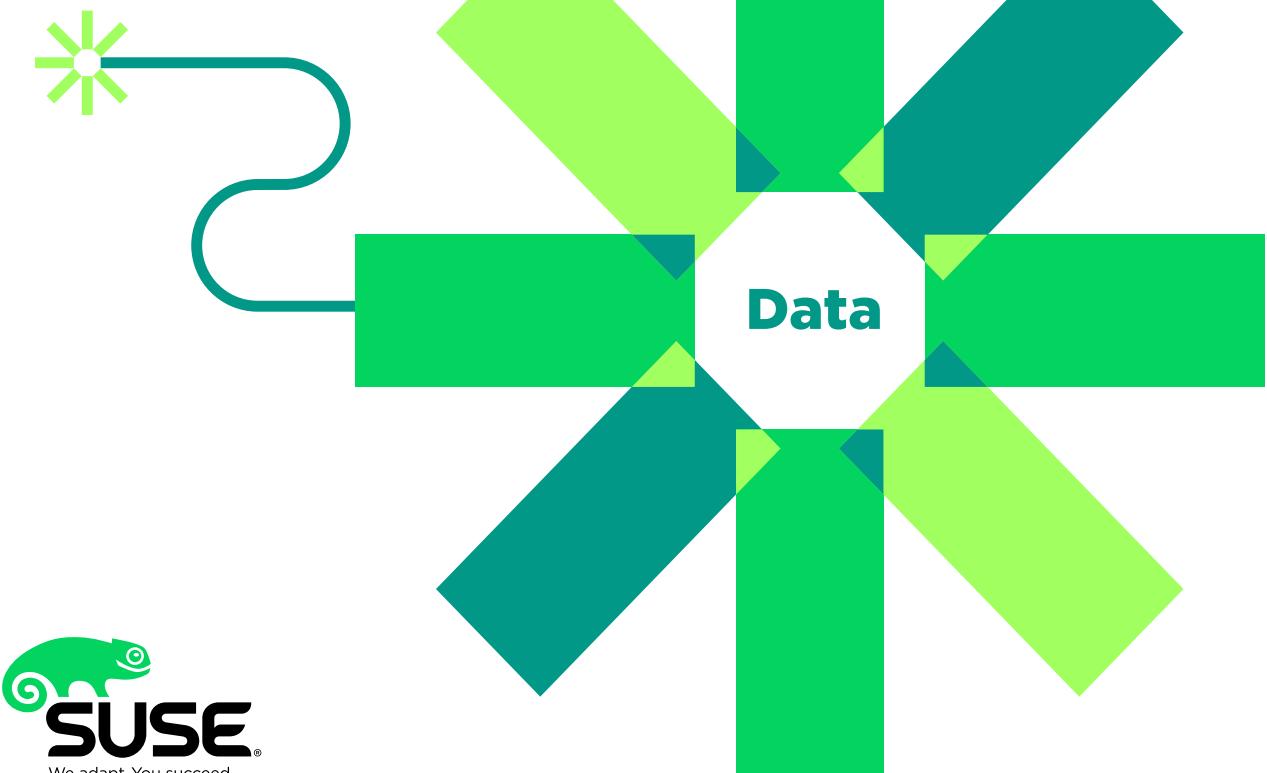
**STORAGE  
REDEFINED:**

# You ~~can~~ not keep up with data explosion.

Manage data expansion  
with SUSE Enterprise  
Storage.

SUSE Enterprise Storage, the leading  
open source storage solution, is highly  
scalable and resilient, enabling high-end  
functionality at a fraction of  
the cost.

[suse.com/storage](http://suse.com/storage)



# My Sysadmin Is a FOR/NEXT Loop



**SHAWN  
POWERS**

Technology always has promised to save us time by doing the things we can do more accurately and with greater efficiency. It has proven to live up to the details, but it completely missed the spirit of the concept. Rather than letting a Bash script do 12 hours of data entry in 20 seconds and then eating ice cream for two days, we've just crammed more work into the time that technology freed up. I realize it was inevitable, but at times, I still feel that as a generation, we were bamboozled. This month, we just give in to the inevitable and learn to do more and more things thanks to the "help" of technology.

We start off the issue with Reuven M. Lerner, who teaches us that computers even can replace the need for humans to judge burrito quality. Your burrito-tasting skills are not irreplaceable, and Reuven sadly proves it. Dave Taylor follows with a complex look at determining the phase of the moon. In ancient times, it took giant stone monuments and complicated stick alignment to determine the next lunar phase. Dave shows how a few lines of admittedly complicated code can figure it out in seconds. Thanks, Dave. I'll blame

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 email at [shawn@linuxjournal.com](mailto:shawn@linuxjournal.com). Or, swing by the #linuxjournal IRC channel on Freenode.net.



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

the next werewolf uprising on your script.

Kyle Rankin takes us a step beyond Puppet this month with a look at MCollective. Automating the setup of servers is a real time-saver, but sometimes managing them takes a bit more than configuration managers can handle. MCollective goes to the next step, and “orchestrates” server administration tasks. You’ll want to check it out if you manage more than a couple servers.

Even if you manage only a couple servers in your own home, however, you’ll want to read my column this month on how to create a PHP dashboard. I’ve covered PHP in my column before, but this time, I add input to the scripting in order to provide feedback and accomplish tasks. This article teaches how to do some powerful and dangerous things with PHP, but in the right situation, that power can be very useful.

In a move reminiscent of Reese’s adding peanut butter to chocolate, Todd A. Jacobs shows how to take Docker and Puppet, and integrate them for even more automation. During the past few years, we’ve learned just how powerful container systems like Docker can be for spinning up single-purpose servers. Todd walks through provisioning those containers with Puppet. It’s fascinating to see tasks that we used to do manually not only become automated, but also to evolve into something that can be managed with programmatic commands instead of elbow grease. If you want to take your automation to the next step, check out Todd’s article.

And finally, Jan Newmarch continues his series on low power wireless, which sounds like a step backward, but is really a huge advancement for near field communication. There are so many wirelessly connected devices in my house, the idea of efficient communication between them, using open standards, is incredible. If you are remotely interested in the Internet of Things, you’ll want to read his series.

We also have tech tips, product announcements, kernel updates and all the other things you expect in an issue of *Linux Journal*. Even though our magazine is digital, we still get to read it with our human eyes. But who knows, next month, we might have an article on a new AI that reads magazines for you and implants the learned information directly into your brain. That thought both excites and terrifies me! ■

# LETTERS

---



PREVIOUS  
Current\_Issue.tar.gz

NEXT  
UpFront



## How to Protect Against Hard Disk Firmware Hacking

As it has been known for at least a year, the intelligence agency, the Equation Group, is capable of reprogramming or reflashing a computer hard drive's firmware with malicious code. Needless to say, this is unsettling. I find it peculiar that I have read no articles about this malevolent act. You have been posting articles about hardening a server with specific encryption algorithms and hash message authentication protocols, but not against protecting your hard disk against hackers and such. My idea is to read the firmware of the hard disk upon purchase and then to compare this hash value with future firmware reads. If there is no way to do this, it could be an entertaining article to appear in your magazine. I would be thankful to read an article about this in the near future, and I am sure many of the Linux supporters around the world, will appreciate this as well.

—Vincent

**Kyle Rankin replies:** You are right that there aren't many articles out there about protecting a server against malicious hard disk firmware. The closest coverage we've had in Linux Journal was an approach you can use to protect against malicious motherboard firmware by using the completely Free Software Libreboot BIOS. See the Hack and / "Libreboot on an X60" series (in the March, April and May 2015 issues of LJ) and the follow-up "Flash ROMs with a Raspberry Pi" (November 2015).

With a combination of Libreboot as your BIOS and using only hardware that has free software firmware, you could use the same Raspberry Pi you used to flash your BIOS to pull a copy of the BIOS periodically while

*the laptop is off and compare checksums to confirm nothing has changed. Unfortunately, this approach validates only the motherboard firmware, not the hard disk firmware. So far, I haven't seen any successful projects that free up hard disk firmware like Libreboot and Coreboot has for the BIOS.*

## Firewalld Article in the September 2016 Issue

I wanted to thank you for one of the best technical articles I have ever read (see "Understanding Firewalld in Multi-Zone Configurations" by Nathan R. Vance and William F. Polik). It was very helpful, and I totally agree that firewalld is under-documented.

The only issue I had with the article was that if I copied and pasted any of the code onto the command line, it would not run.

—Mike Tarkowski

**William Polik and Nathan Vance reply:** We are glad you appreciated the additional documentation on firewalld. If the example commands do not run for you, you will want to check the following:

- Is firewalld installed on your system; verify with `which firewall-cmd`.
- Is firewalld running on your system (instead of iptables, for example); verify with `systemctl status firewalld`.
- Make sure commands are being run as root or with sudo.

Also note that this does not appear to be an



## At Your Service

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

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

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

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

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

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

issue with content of the article, but rather with embedding of hidden characters in the PDF formatting of the article. If you type the commands yourself, rather than copying and pasting, they work fine.

## **Hard Drive Rescue with a Raspberry Pi and Relay**

I just read the September 2016 issue of *LJ* and this article ("Hard Drive Rescue with a Raspberry Pi and Relay" by Andrew Nii Addo) is so awesome, it made my day! I never thought of using a Raspberry Pi to cycle a hard disk! it makes so much sense once I read the article.

This basically confirms my belief that *LJ* is a necessity and not a luxury.

Thanks for sharing your work in the article—great idea.

—Guru

## **Mars Lander**

Can you ask Dave Taylor to forward his code for the Mars lander to the ESA? Apparently, they've been having some problems with the reverse thrust settings on the Schiaparelli lander. (See Dave's Work the Shell column in the September, October and November 2016 issues.)

—David Terry

**Dave Taylor replies:** Sounds like a good plan, because a shell script is the best possible choice for our next interplanetary adventure vessel!

### **PHOTO OF THE MONTH**

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

### **WRITE LJ A LETTER**

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

**RETURN TO CONTENTS**



Where every interaction matters.

# break down your innovation barriers

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

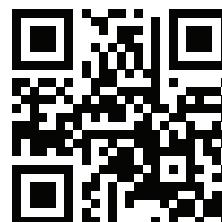
When you're presented with new opportunities, you want to focus on turning them into successes, not whether your IT solution can support them.

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

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

**Want more on cloud?**

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





PREVIOUS  
Letters

NEXT  
Editors' Choice



## diff -u

### What's New in Kernel Development

With the demise of the **big kernel lock** (BKL), various new locks have taken its place to cover various types of situations, some more rarefied than others. Recently **Waiman Long** implemented the **TO futex** (throughput-optimized futex), which prioritizes throughput over giving all processes a fair chance to claim the lock.

It's a strange concept—usually a UNIX-based system would make fairness to all users the highest priority. But the TO futex outperformed **wait-wake futexes** and **priority-inheritance futexes** on certain workloads, specifically those involving short, critical sections of code, with large numbers of threads competing for the same lock.

Sometimes, as **Thomas Gleixner** pointed out, it can be hard for developers to know which futex to choose, given so many options and each with such specific optimal use patterns, although Waiman has said he feels the TO futex may simply outperform the others well enough to be the preferred choice almost all the time.

It's hard to know for sure. We can assume that the many-eyes principle of open-source development implies that eventually any poor locking choice will be found and fixed. So maybe the added complexity will simply come out in the wash. On the other hand,

there's a real benefit in developers actually being able to understand what they're doing when they modify kernel code. So, keeping Linux locking simple may turn out to be preferable in the end.

Sometimes kernel developers will turn their attention to corner cases and pathological conditions, trying to smooth out behaviors that rarely if ever occur.

**Al Viro** gave that a shot recently, when he noticed that the **writev()** system call seemed to behave unintuitively when fed certain kinds of bad input. The writev() call writes a series of memory buffers to a file. But if one of the middle buffers is given an undefined address in the input, writev() still would write the first portion of data before giving up.

He felt this was a mistake. Instead of writing just a portion of the data—essentially creating an unpredictable state via behaviors that developers should definitely not rely on—he felt that no data (or at least a predictable amount of data) should be written, and the call should return the **EFAULT** error code.

When it comes to system calls, however, you can't just do whatever you want. There are various standards, specifically **POSIX**, that you either have to obey or have a good reason not to.

In this case, Al felt that POSIX was vague enough to let his preferred behavior sail through. As he understood the standard, the exact amount of data written when the error occurred was actually variable. And, that being the case, he figured why not just make the logic be “if some addresses in the buffer(s) we are asked to write are invalid, the write will be shortened by up to a PAGE\_SIZE from the first such invalid address”.

**Linus Torvalds** approved this arrangement, but **Alan Cox** objected. He noted that POSIX version 1003.1 said “Each iovec entry specifies the base address and length of an area in memory from which data should be written. The writev() function shall always write a complete area before proceeding to the next.”

So, instead of shortening the write by up to PAGE\_SIZE, Alan's reading of the standard required writing the whole amount of data and then failing on the upcoming invalid address.

But, Alan pointed out that passing an invalid address to writev() was not anything anyone would want to do and didn't have any clearly defined consequence. He felt it would be more useful to think about how to deal with realistic causes of writev() being passed an invalid address—for example, when the system ran out of disk space in the midst of the writev() call.

Linus felt that the disk-full scenario was a reasonable situation to care about. But, he also felt that the main point of the POSIX behavior—either expressed or implied—was to prevent weird situations where users could see later writes without being able to see earlier ones. He felt it was important to ensure that “you cannot do some fancy threaded thing where you do different iovec parts concurrently, because that could be seen by a reader (or more likely mmap) as doing the writes out of order.”

It's unclear which behavior might ultimately get into the kernel. These debates often pull from bizarre sources. For example, it could turn out that there's some kind of security issue with one or the other position on the issue, in which case whichever position successfully addressed the security concern would be the winner.

**Luis R. Rodriguez** tried to fix a possible race condition by having userspace recognize a given situation and alert the kernel. Not surprisingly, it was met with a strong rebuke from Linus Torvalds.

The race condition occurred if the user tried to read a file from the system's filesystem during bootup. If the read occurred at one time, the filesystem would be unavailable, but if the read occurred slightly later, it would.

The problem, as he saw it, was that only userspace could know whether certain filesystems already had been mounted. The obvious solution, he felt, was for userspace to alert the kernel to the filesystem availability, so the kernel could attempt to access the needed file only after that file was actually on a mounted filesystem.

Linus called this a “horrible hack” and a white flag of surrender. Not only that, but he said “it's broken nasty crap with a user interface, so we'll be stuck with it forever.”

He suggested instead that any drivers running into this problem

simply be fixed to not do that. But, **Dmitry Torokhov** didn't see how that could be accomplished. He said:

Some devices do need to have firmware loaded so we know their capabilities, so we really can't push the firmware loading into "open".... These devices we want to probe asynchronously and simply tell the firmware loader to wait for firmware to become available. The problem is we do not know when to give up, since we do not know where the firmware might be. But userspace knows and can tell us.

And, **Bjorn Andersson** piled on, saying there were actual real-world cases that would benefit from Luis' code.

But, Linus saw the entire concept as too broken to salvage, regardless of any use value it might have. He would rather bundle firmware directly into a kernel module, he said, than have the kernel depend on a userspace notification.

The whole discussion was useful mostly as an example of how kernel developers can stand up under scathing critiques from Linus and still pull useful technical feedback out of what he says. In this case, he left no room for doubt—alerts coming from userspace to the kernel would not be allowed under any circumstances.—Zack Brown

## THEY SAID IT

**"In the right light, at the right time, everything is extraordinary."**

—Aaron Rose

**There is no Them, there is only Us. Some of Us think this or some of Us think that, but we're all Us.**

—Lisa Williams

**I think people don't place a high enough value on how much they are nurtured by doing whatever it is that totally absorbs them.**

—Jean Shinoda Bolen

**He that will not sail until all dangers are over, will never put to sea.**

—Thomas Fuller

**In an industrial society which confuses work and productivity, the necessity of producing has always been an enemy of the desire to create.**

—Raoul Vaneigem

# Automatic Slack Notifications

Slack is an incredible communication tool for groups of any size (see my piece on it in the November 2016 issue). At the company I work for during the day, Slack has become more widely used than email or instant messaging. It truly has become the hub of company communication. So rather than have my servers send email, I've turned to Slack for delivering information to my users. Thankfully, Slack is extremely open to adding applications and integrations.

The screenshot shows the Slack App Directory with the URL <https://slack.com/apps/A0F984B7D/integrations/incoming-webhook>. The page is titled "Integration Settings" and contains the following fields:

- Post to Channel:** A dropdown menu set to "#general". Below it is a link to "or create a new channel".
- Webhook URL:** A text input field containing the URL <https://hooks.slack.com/services/T1YMD1U/B2V9XKQD8P/ERDQfC>. To its right are "Copy URL" and "Regenerate" buttons.
- Descriptive Label:** A text input field with placeholder text "Optional description of this integration".
- Customize Name:** A text input field containing "Incoming-webhook".
- Customize Icon:** A section with a placeholder icon, "Upload an image" and "Choose an emoji" buttons, and a note: "Change the icon that is used for messages from this integration."
- Preview Message:** A preview box showing a sample message from "incoming-webhook" at "10:09 AM". The message content is "This is what messages from this service will look like in Slack." Below the preview is a "Save Settings" button.

The simplest integration is called an “incoming webhook”, and it delivers messages to Slack channels (or individual users) by sending a POST to a specially formed URL. The first step is to find the custom integration area in Slack, which isn’t as clear as I’d like. On the website, click on your Slack group name at the top left, and pick “Apps & Integrations” from the drop-down menu. Then on the apps page, click “build” on the upper right. Finally, click “Make a Custom Integration” and select “incoming webhooks”.

From there it’s a matter of selecting where you want the notification to post, what icon to give it, what name to assign to your bot and so on. Once it’s saved, you can use curl to post a message to your unique URL (which is on the creation page, be sure to copy it to your clipboard):

```
curl -X POST --data "payload={"text": "Cool Message"}"  
→https://hooks.slack.com/services/YOURAPI/CODEHERE/TOPOST
```

That’s it! You can create a BASH script to make the process simple and integrate the notification system into your server scripts!  
—Shawn Powers



# Listen with Your Skull!

I listen to a lot of audiobooks. They're not the sort of thing you blast from your car speakers, because invariably when you pull up to a drive-thru window, it's at an awkward part of the book. Thankfully I don't read many books with sex scenes, but it's a bit embarrassing when it's a super-cheesy-sounding part of the book that plays while you're paying. But, I digress.

I don't like earbuds or headphones, because I prefer to hear what's going on around me. So when I'm driving, or walking with someone, I tend to leave one earbud in, and the other out. It's not perfect, but it works.

Recently, however, I discovered bone-conduction headphones. I really like the Trekz Titanium model (<https://aftershokz.com/products/trekz-titanium>), but they're fairly pricey, and others might work just as well.

The concept is that instead of putting something in your ears, the device vibrates the bones in your head, which in turn vibrate your inner ears and produce sound. It means your ears are completely open, so you can hear people talking to you or honking behind you. Thanks to

the sound transferring internally, however, you still can hear the audio really well too. In fact, if you want to drown out the outside world, you can put earplugs in and totally immerse yourself in audio. It feels weird to increase the volume by plugging your ears, but it works. In fact, if you want to try it, just hum, and while you're humming, plug your ears. That same sort of thing happens with bone conduction headsets. It's sort of magical.

Like I said, I use Trekz brand and really like them. I can take calls, listen to music or play/pause my audiobooks with ease. If you like the privacy of earbuds, but don't want to stick anything in your ears, give bone conduction a try. I thought it was a gimmick, but I'm happy to say I was wrong!—Shawn Powers

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



Available on the  
App Store

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



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

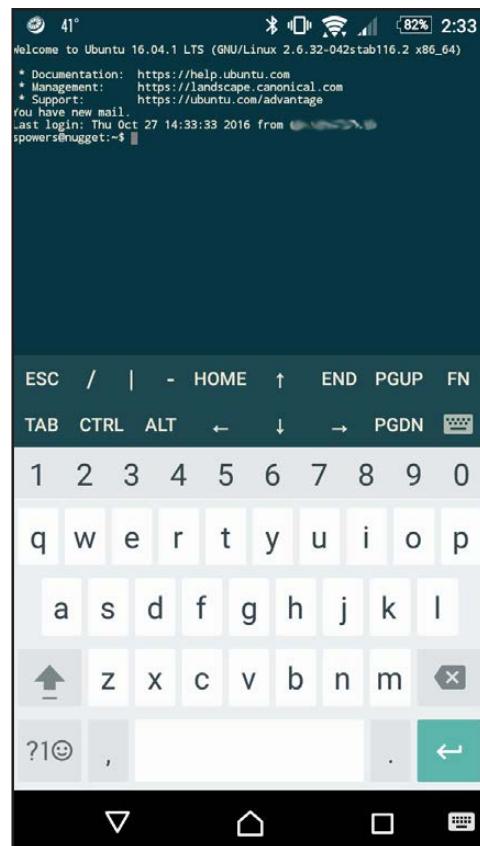
# Android Candy: Landing on the Moon, with your Thumbs

I do a lot of system administration with my thumbs. Yes, if I'm home, I grab a laptop or go to my office and type in a real terminal window. Usually, when things go wrong though, I'm at my daughters' volleyball match or shopping with my wife. Thankfully, most tasks can be done remotely via SSH. There are lots of SSH clients for Android, but my favorite is JuiceSSH.

Yes, part of my love for the app is that it has a cool icon in the shape of a lemon, but really, there's more to it than that. It has a plugin architecture that allows you to build functionality on top of SSH. It also allows you to execute code snippets on multiple connections with a click of a button.

The keyboard is designed in such a way that even vi users like myself can manage to edit files remotely. And thanks to the ability to import private SSH keys, I can connect to those servers where I have password authentication disabled. (For example, most cloud servers don't allow you to log in via password, they require you to use SSH keys, which is awesome.)

To be honest, I do so much work remotely with my phone, that I'm considering getting a foldable Bluetooth keyboard so I can actually do some typing in a pinch if needed. If I find a keyboard I like, I'll be sure to write about it in a future issue! You can get JuiceSSH from the Google Play Store.—Shawn Powers



# Pythonic Science in the Browser

In the past, if you wanted a friendly environment for doing Python programming, you would use Ipython. The Ipython project actually consists of three parts: the standard console interface, a Qt-based GUI interface and a web server interface that you can connect to with a web browser. The web browser interface, especially, has become the de facto way of doing scientific programming with Python. It has become so popular in fact, it has spun off as its own project, named Jupyter. In this article, I take a look at how to get the latest version up and running, and I discuss the kinds of things you can do with it once it is set up.

The first step is to install the latest version. Because it is under very active development, you probably will want to keep it updated on your system. `pip` is definitely the easiest way to do this. The following command will install Jupyter, if it isn't already installed, or it will update Jupyter to the latest version:

```
sudo pip install --upgrade jupyter
```

Be sure that you have a C compiler installed, along with the development package for Python. For example, on Debian-based systems, you can be sure you are ready by executing the following command:

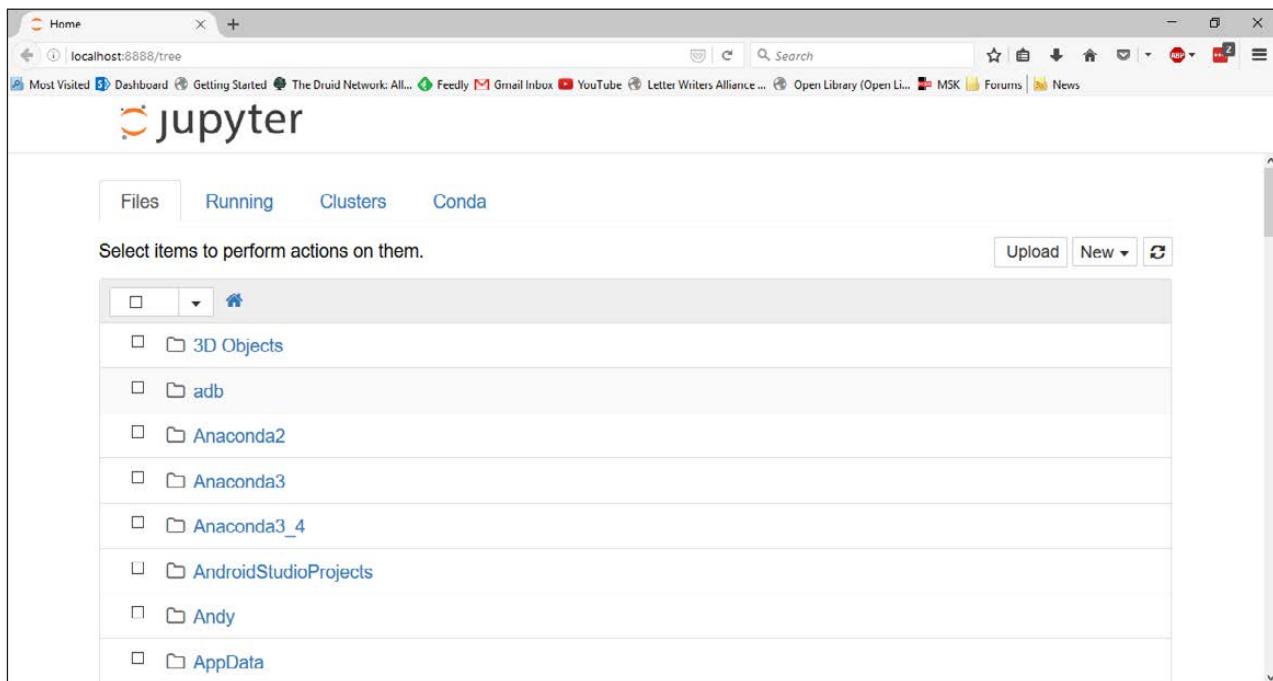
```
sudo apt-get install python-dev build-essentials
```

This should make sure you have everything you need installed.

To start Jupyter, open a terminal window and enter the command:

```
jupyter notebook --no-browser
```

This will start a web server, listening on port 8888, that will accept connections from the local machine. For security reasons, by default, it will ignore incoming connections from outside machines. If you want to



**Figure 1.** When you first enter Jupyter, you are presented with a file listing from the current working directory.

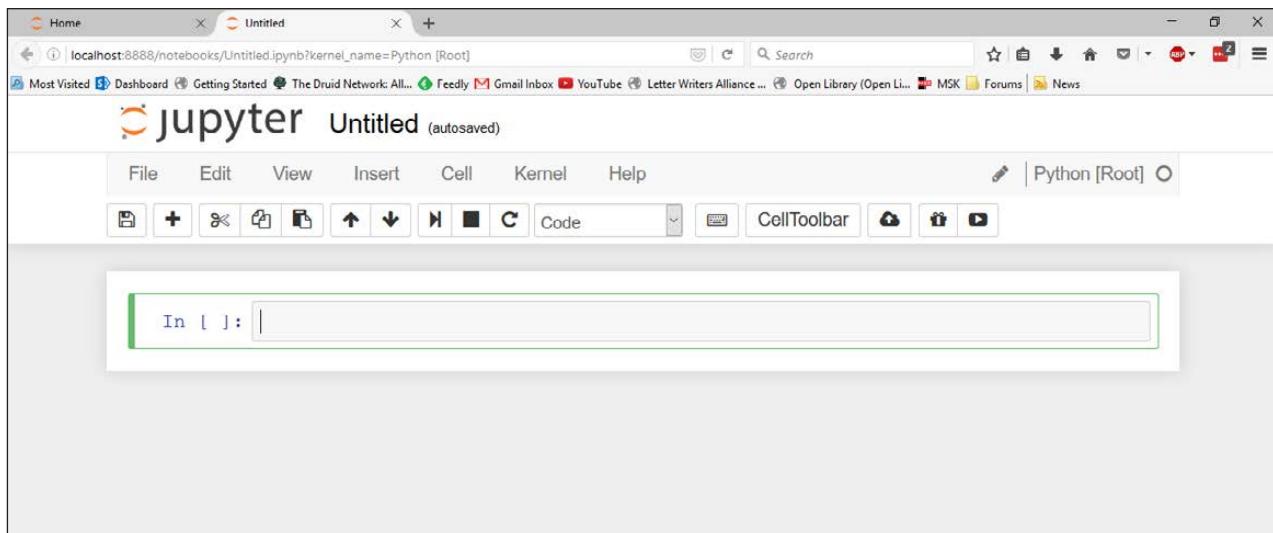
set it up to accept connections from outside machines, you can do so by adding an extra option:

```
jupyter notebook --no-browser --ip=*
```

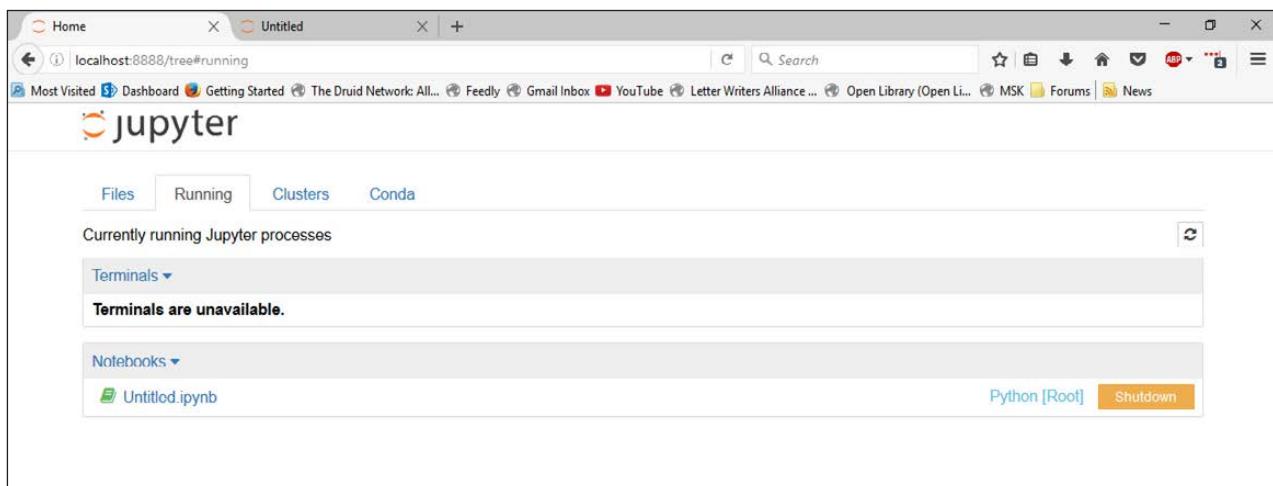
This makes your Jupyter server wide open, so it is strongly discouraged unless you are on a secure private network. Otherwise, you should have some sort of user authentication set up to manage who can use your system.

Once Jupyter is up and running, open a browser and point it to `http://localhost:8888`. Across the top, you will see a series of tabs for each section of the workspace. Most people will see only three: Files, Running and Clusters. If you are using the Anaconda Python distribution, you will get a fourth tab named Conda.

On startup, you will be located at the first tab, Files. This is simply a directory listing of the current working directory. You probably won't have any notebooks currently available, so you will need to create a new one. You can do that by clicking the drop-down list on the right-hand side of



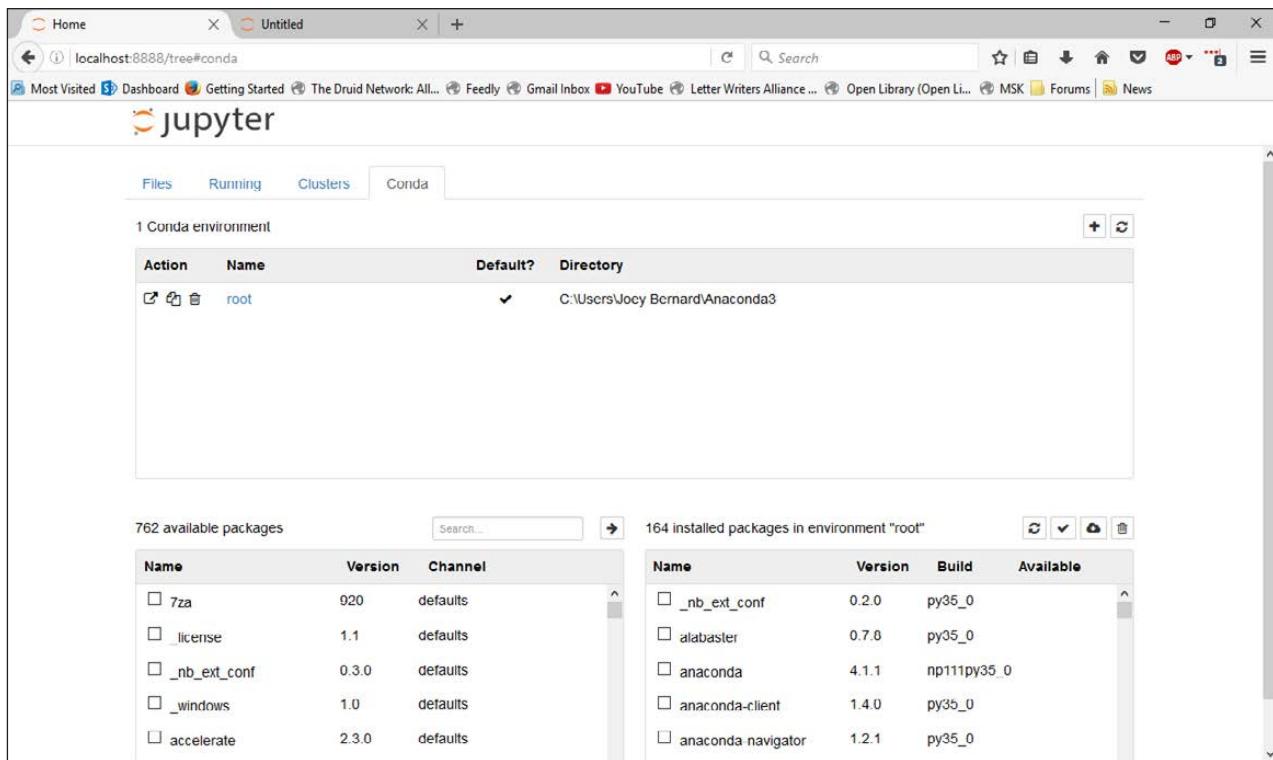
**Figure 2.** Clicking on the New tab will load a new, empty notebook within a new browser tab.



**Figure 3.** You can get a listing of all of the active notebooks and terminal sessions.

the screen labeled New and selecting the Python notebook entry on the menu. This will open a new browser tab and load a new empty notebook.

The second tab shows you all of the active notebooks running on this particular server. You can click the related link to open the selected notebook in a new browser tab or click the Shutdown button to halt the selected notebook. There is also a section for any active terminal sessions. The Clusters section gives you status information on the parallel Python engines that are configured on your system. By default, this isn't configured at all.

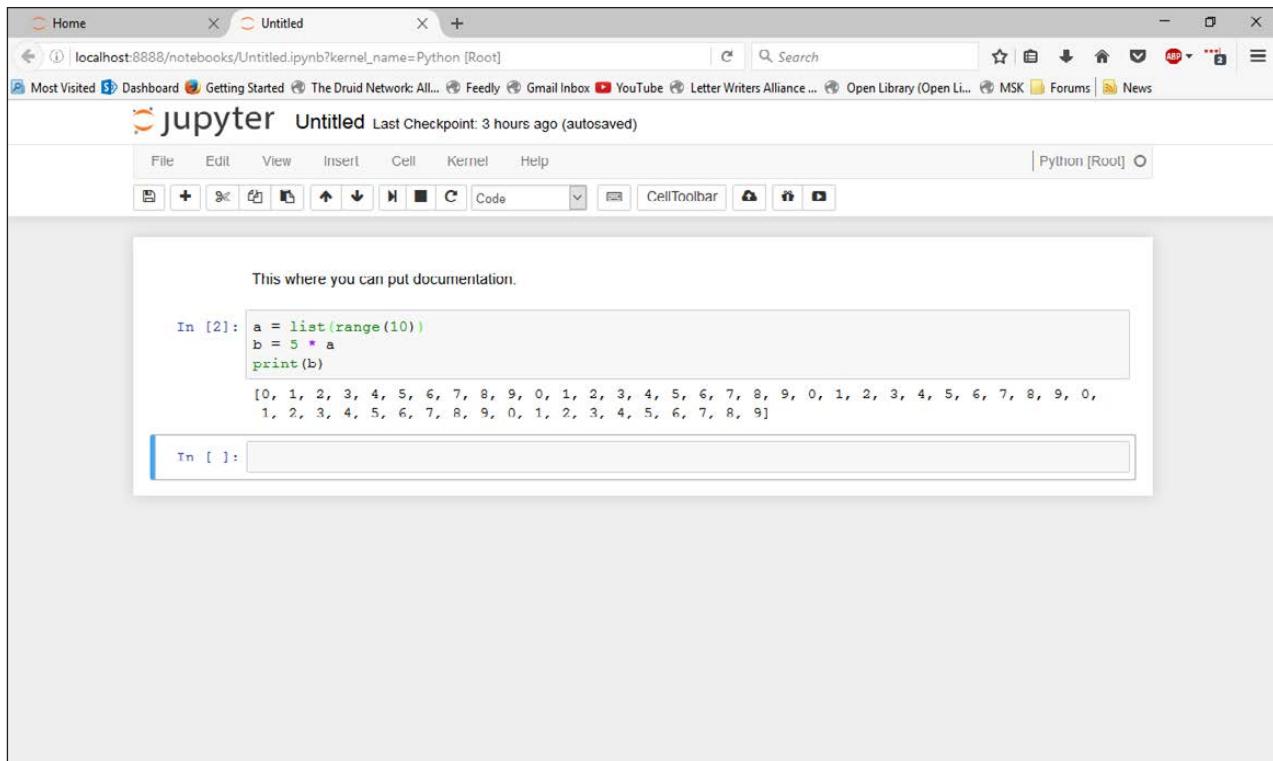


**Figure 4.** Jupyter lets you check the details of your Anaconda packages and environments.

If you are using Anaconda, you will get a fourth section labeled Conda. This section gives you details about what packages are available and what packages are installed. You even can manage your Conda environments, creating new ones, exporting existing ones or deleting environments that you are done with.

Let's take a look at what you can do with the notebook itself. The interface should feel familiar to people who have used applications like Maple or Mathematica. Your input is entered in sections called cells. Cells can be of different types, namely code, markdown text or headings. This way, you can have text cells describing the code sections, explaining what the code is doing and why. It's extremely useful when you're doing scientific calculations, because it allows you to include your documentation with your code, so everything stays synchronized and up to date.

When you start entering lines of code, pressing Enter takes you to a new line within the same cell. None of the code gets executed yet. When you are ready to run the cell, press the Shift and Enter keys



**Figure 5.** You get the output from your Python code displayed within the notebook.

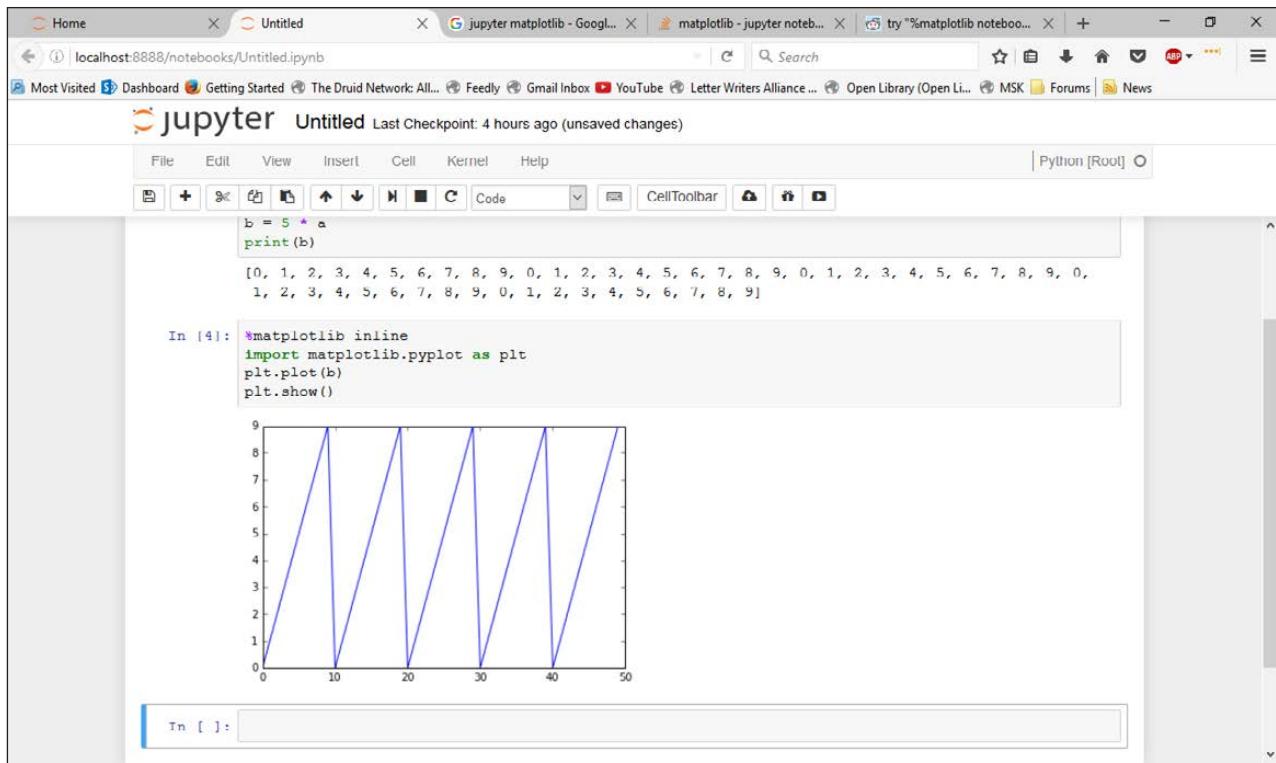
together. All of the code runs within the same Python engine, so results from one cell will be available to other cells later on.

You also can import extra Python modules, just like you do in any other Python environment. The most useful for visualizing data and results is matplotlib. If you import the matplotlib module and execute plotting commands, Jupyter can render the resulting graphs directly in the notebook.

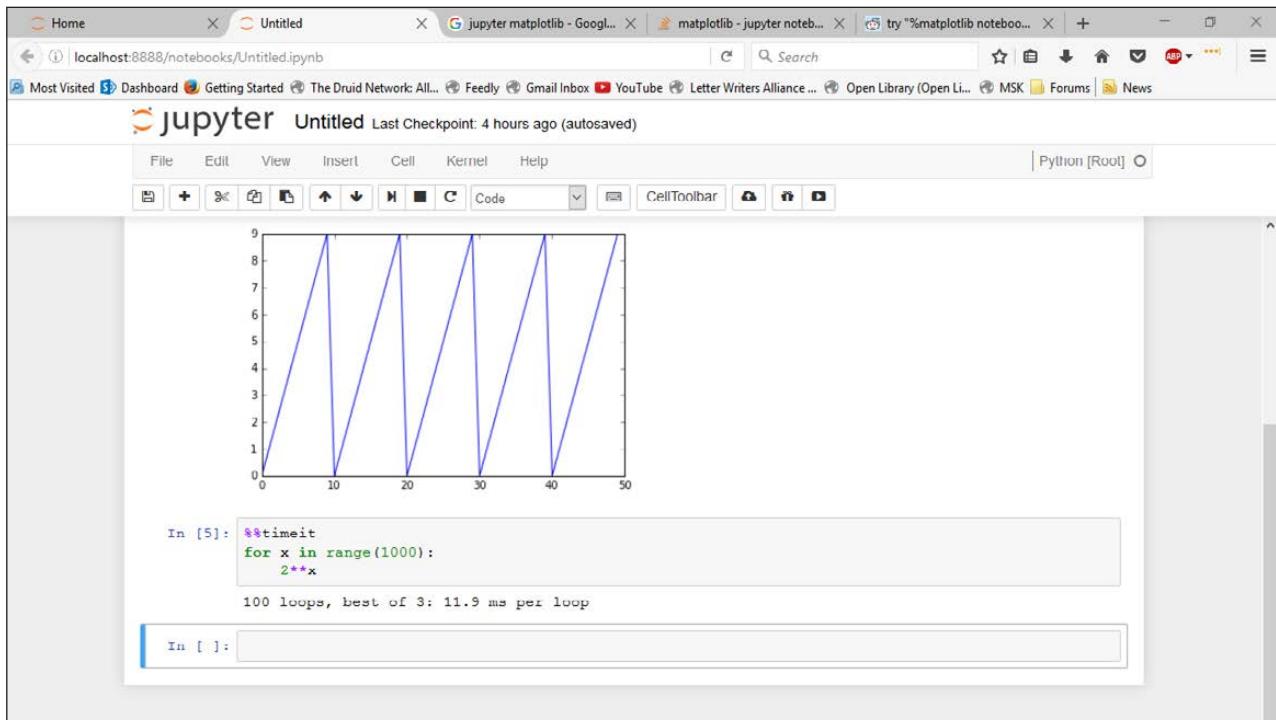
As you can see, you need to use an extra statement that starts with a % character to tell Jupyter to render the plot as an image within the notebook. Otherwise, the plots will be rendered within a new window. This new statement is called a magic. There is an entire library of magics available. For example, you can use the timeit magic to profile how long it takes to run the code within a cell.

If it is a section of code that has a short runtime, Jupyter automatically will run it several times to get an average runtime. This allows you to work on optimizing your code as well as developing it.

When you are ready to share your results, several different options



**Figure 6.** Jupyter can render matplotlib graphs directly in the notebook.



**Figure 7.** There are several magic statements available, such as the timeit magic to find runtimes of code cells.

are available. You always can simply share the Jupyter notebook. It is all stored in a single file with the filename ending .ipynb. The other options depend a bit on which Python modules you have installed on your system. The formats most often used are either as a single static HTML page or PDF document. If you are going to present your results, you even can export it as an HTML-based presentation where the cells are formatted as individual slides. All of these options are available under the File→Download As menu item.

Hopefully this article has shown some functionality you can make use of in your own code. Jupyter is especially useful for scientific exploration. You can try lots of different calculations and do different types of data analysis and see the results right away within the notebook. And when you reach a useful conclusion, you can share the notebook with others within the growing community of Jupyter users.

—Joey Bernard

## LINUX JOURNAL

on your  
e-Reader

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

**LEARN MORE**



# EDITORS' CHOICE



PREVIOUS  
UpFront



NEXT  
Reuven M. Lerner's  
At the Forge



## My Cup of Tea

Computer folks are known for their mass consumption of caffeinated beverages. Some prefer coffee; some prefer tea. (Some drink energy drinks too, but we won't talk about those folks.) I actually go between coffee and tea depending on the season. During the

summer, I drink mainly coffee. It can be freshly ground brewed coffee, a fancy coffee made with my espresso machine or even a quick K-cup abomination at 6am. But once fall sets in and the snow starts falling, I always switch to tea. I'm not sure why, but for me, the winter means tea.

Why would I mention my preferences in a tech magazine? Because if you're like me and drink tea, either seasonally or exclusively, you know it can be a pain. Sure there are teabags for people who don't care what their tea tastes like, but I prefer loose-leaf tea. And, it can be a pain to make. A couple years back I discovered the Adagio ingenuiTEA steeping device. It makes loose-leaf tea actually easier than teabags. You simply put the loose tea in the top, pour hot water directly on the leaves, and then after steeping, you let it drain through the filter into your cup.

Through the years, I've purchased several brands of the tea steepers (my current favorite is the Teaze Tea Infuser), and they all work in a similar manner. It's amazing how often I use my Teaze device, even when I have a \$250 Breville Perfect Tea maker on my office shelf. I still go to the \$15 device 95% of the time! It may seem like an odd recommendation for a tech magazine, but anyone in the tech industry knows that a proper beverage is as important as a proper operating system, so check one out today. If you search for "ingenuitea" on Amazon, you'll see a bunch of different brands. They're cheap, and awesome. In fact, it might be the first time I've ever given a non-tech-related item the Editors' Choice award, but if you like tea, I urge you to give it a try.—Shawn Powers

**RETURN TO CONTENTS**

# Teaching Your Computer

It's easier than you think to teach your computer what makes for a tasty burrito.



**REUVEN M.  
LERNER**

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

◀ PREVIOUS  
Editors' Choice

NEXT  
Dave Taylor's  
Work the Shell ▶

**AS I HAVE WRITTEN IN MY LAST TWO ARTICLES,** machine learning is influencing our lives in numerous ways. As a consumer, you've undoubtedly experienced machine learning, whether you know it or not—from recommendations for what products you should buy from various online stores, to the selection of postings that appear (and don't) on Facebook, to the maddening voice-recognition systems that airlines use, to the growing number of companies that offer to select clothing, food and wine for you based on your personal preferences.

Machine learning is everywhere, and although the theory and practice both can take some time to learn and internalize, the basics are fairly straightforward for people to learn.

The basic idea behind machine learning is that you build a model—a description of the ways the

inputs and outputs are related. This model then allows you to ask the computer to analyze new data and to predict the outputs for new sets of inputs. This is essentially what machine learning is all about. In “supervised learning”, the computer is trained to categorize data based on inputs that humans had previously categorized. In “unsupervised learning”, you ask the computer to categorize data on your behalf.

In my last article, I started exploring a data set created by Scott Cole, a data scientist (and neuroscience PhD student) who measured burritos in a variety of California restaurants. I looked at the different categories of data that Cole and his fellow eater-researchers gathered and considered a few ways one could pare down the data set to something more manageable, as well as reasonable.

Here I describe how to take this smaller data set, consisting solely of the features that were deemed necessary, and use it to train the computer by creating a machine-learning model.

### Machine-Learning Models

Let's say that the quality of a burrito is determined solely by its size. Thus, the larger the burrito, the better it is; the smaller the burrito, the worse it is. If you describe the size as a matrix  $X$ , and the resulting quality score as  $y$ , you can describe this mathematically as:

$$y = qX$$

where  $q$  is a factor describing the relationship between  $X$  and  $y$ .

Of course, you know that burrito quality has to do with more than just the size. Indeed, in Cole's research, size was removed from the list of features, in part because not every data point contained size information.

Moreover, this example model will need to take several factors—not just one—into consideration, and may have to combine them in a sophisticated way in order to predict the output value accurately. Indeed, there are numerous algorithms that can be used to create models; determining which one is appropriate, and then tuning it in the right way, is part of the game.

The goal here, then, will be to combine the burrito data and an algorithm to create a model for burrito tastiness. The next step will be to see if the model can predict the tastiness of a burrito based on its inputs.

The goal here, then, will be to combine the burrito data and an algorithm to create a model for burrito tastiness. The next step will be to see if the model can predict the tastiness of a burrito based on its inputs.

But, how do you create such a model?

In theory, you could create it from scratch, reading the appropriate statistical literature and implementing it all in code. But because I'm using Python, and because Python's scikit-learn has been tuned and improved over several years, there are a variety of model types to choose from that others already have created.

Before starting with the model building, however, let's get the data into the necessary format. As I mentioned in my last article and alluded to above, Python's machine-learning package (scikit-learn) expects that when training a supervised-learning model, you'll need a set of sample inputs, traditionally placed in a two-dimensional matrix called X (yes, uppercase X), and a set of sample outputs, traditionally placed in a vector called y (lowercase). You can get there as follows, inside the Jupyter notebook:

```
%pylab inline
import pandas as pd                      # load pandas with an alias
from pandas import Series, DataFrame       # load useful Pandas classes
df = pd.read_csv('burrito.csv')            # read into a data frame
```

Once you have loaded the CSV file containing burrito data, you'll keep only those columns that contain the features of interest, as well as the output score:

```
burrito_data = df[range(11,24)]
```

You'll then remove the columns that are highly correlated to one another and/or for which a great deal of data is missing. In this case, it means removing all of the features having to do with burrito size:

```
burrito_data.drop(['Circum', 'Volume', 'Length'], axis=1,  
➥inplace=True)
```

Let's also drop any of the samples (that is, rows) in which one or more values is NaN ("not a number"), which will throw off the values:

```
burrito_data.dropna(inplace=True, axis=0)
```

Once you've done this, the data frame is ready to be used in a model. Separate out the X and y values:

```
y = burrito_data['overall']  
x = burrito_data.drop(['overall'], axis=1)
```

The goal is now to create a model that describes, as best as possible, the way the values in X lead to a value in y. In other words, if you look at `x.iloc[0]` (that is, the input values for the first burrito sample) and at `y.iloc[0]` (that is, the output value for the first burrito sample), it should be possible to understand how those inputs map to those outputs. Moreover, after training the computer with the data, the computer should be able to predict the overall score of a burrito, given those same inputs.

## Creating a Model

Now that the data is in order, you can build a model. But which algorithm (sometimes known as a "classifier") should you use for the model? This is, in many ways, the big question in machine learning, and is often answerable only via a combination of experience and trial and error. The more machine-learning problems you work to solve, the more of a feel you'll get for the types of models you can try. However, there's always the chance that you'll be wrong, which is why it's often worth creating several different types of models, comparing them against one another for validity. I plan to talk more

## And, it's this myriad of choices and options that can lead to a data-science project being involved, and to incorporate your experience and insights, as well as brute-force tests of a variety of possible models.

about validity testing in my next article; for now, it's important to understand how to build a model.

Different algorithms are meant for different kinds of machine-learning problems. In this case, the input data already has been ranked, meaning that you can use a supervised learning model. The output from the model is a numeric score that ranges from 0 to 5, which means that you'll have to use a numeric model, rather than a categorical one.

The difference is that a categorical model's outputs will (as the name implies) indicate into which of several categories, identified by integers, the input should be placed. For example, modern political parties hire data scientists who try to determine which way someone will vote based on input data. The result, namely a political party, is categorical.

In this case, however, you have numeric data. In this kind of model, you expect the output to vary along a numeric range. A pricing model, determining how much someone might be willing to pay for a particular item or how much to charge for an advertisement, will use this sort of model.

I should note that if you want, you can turn the numeric data into categorical data simply by rounding or truncating the floating-point y values, such that you get integer values. It is this sort of transformation that you'll likely need to consider—and try, and test—in a machine-learning project. And, it's this myriad of choices and options that can lead to a data-science project being involved, and to incorporate your experience and insights, as well as brute-force tests of a variety of possible models.

Let's assume you're going to keep the data as it is. You cannot use a purely categorical model, but rather will need to use one that incorporates the statistical concept of "regression", in which you attempt to determine which of your input factors cause the output to correlate linearly with the outputs—that is, assume that the ideal is something like the " $y = qX$ " that you saw above; given that this isn't the case, how much influence did meat quality have vs. uniformity vs. temperature? Each of those factors affected the overall quality in some way, but some of them had more influence than others.

One of the easiest to understand, and most popular, types of models uses the K Network Neighbors (KNN) algorithm. KNN basically says that you'll take a new piece of data and compare its features with those of existing, known, categorized data. The new data is then classified into the same category as its K closest neighbors, where K is a number that you must determine, often via trial and error.

However, KNN works only for categories; this example is dealing with a regression problem, which can't use KNN. Except, Python's scikit-learn happens to come with a version of KNN that is designed to work with regression problems—the `KNeighborsRegressor` classifier.

So, how do you use it? Here's the basic way in which all supervised learning happens in scikit-learn:

1. Import the Python class that implements the classifier.
2. Create a model—that is, an instance of the classifier.
3. Train the model using the "fit" method.
4. Feed data to the model and get a prediction.

Let's try this with the data. You already have an X and a y, which you can plug in to the standard `sklearn` pattern:

```
from sklearn.neighbors import KNeighborsRegressor      # import classifier
KNR = KNeighborsRegressor()                          # create a model
KNR.fit(X, y)                                     # train the model
```

Without the `dropna` above (in which I removed any rows containing one or more NaN values), you still would have “dirty” data, and sklearn would be unable to proceed. Some classifiers can handle NaN data, but as a general rule, you’ll need to get rid of NaN values—either to satisfy the classifier’s rules, or to ensure that your results are of high quality, or even (in some cases) valid.

With the trained model in place, you now can ask it: “If you have a burrito with really great ingredients, how highly will it rank?”

All you have to do is create a new, fake sample burrito with all high-quality ingredients:

```
great_ingredients = np.ones(X.iloc[0].count()) * 5
```

In the above line of code, I took the first sample from X (that is, `X.iloc[0]`), and then counted how many items it contained. I then multiplied the resulting NumPy array by 5, so that it contained all 5s. I now can ask the model to predict the overall quality of such a burrito:

```
KNR.predict([great_ingredients])
```

I get back a result of:

```
array([ 4.86])
```

meaning that the burrito would indeed score high—not a 5, but high nonetheless. What if you create a burrito with absolutely awful ingredients? Let’s find the predicted quality:

```
terrible_ingredients = np.zeros(X.iloc[0].count())
```

In the above line of code, I created a NumPy array containing zeros, the same length as the X’s list of features. If you now ask the model to predict the score of this burrito, you get:

```
array([ 1.96])
```

The good news is that you have now trained the computer to predict the quality of a burrito from a set of rated ingredients. The other good news is that you can determine which ingredients are more influential and which are less influential.

At the same time, there is a problem: how do you know that KNN regression is the best model you could use? And when I say “best”, I ask whether it’s the most accurate at predicting burrito quality. For example, maybe a different classifier will have a higher spread or will describe the burritos more accurately.

It’s also possible that the classifier is a good one, but that one of its parameters—parameters that you can use to “tune” the model—wasn’t set correctly. And I suspect that you indeed could do better, since the best burrito actually sampled got a score of 5, and the worst burrito had a score of 1.5. This means that the model is not a bad start, but that it doesn’t quite handle the entire range that one would have expected.

One possible solution to this problem is to adjust the parameters that you hand the classifier when creating the model. In the case of any KNN-related model, one of the first parameters you can try to tune is `n_neighbors`. By default, it’s set to 5, but what if you set it to higher or to lower?

A bit of Python code can establish this for you:

```
for k in range(1,10):
    print(k)
    KNR = KNeighborsRegressor(n_neighbors=k)
    KNR.fit(X, y)
    print("\tTerrible: {}".format(KNR.predict([terrible_ingredients])))
    print("\tBest: {}".format(KNR.predict([great_ingredients])))
```

After running the above code, it seems like the model that has the highest high and the lowest low is the one in which `n_neighbors` is equal to 1. It’s not quite what I would have expected, but that’s why it’s important to try different models.

And yet, this way of checking to see which value of `n_neighbors` is the best is rather primitive and has lots of issues. In my next article,

**But as you can see, scikit-learn makes it easy—almost trivially easy, in fact—to create and experiment with different models.**

I plan to look into checking the models, using more sophisticated techniques than I used here.

## Using Another Classifier

So far, I've described how you can create multiple models from a single classifier, but scikit-learn comes with numerous classifiers, and it's usually a good idea to try several.

So in this case, let's also try a simple regression model. Whereas KNN uses existing, known data points in order to decide what outputs to predict based on new inputs, regression uses good old statistical techniques. Thus, you can use it as follows:

```
from sklearn.linear_model import LinearRegression
LR = LinearRegression()
LR.fit(X, y)
print("\tTerrible: {}".format(KNR.predict([terrible_ingredients])))
print("\tBest: {}".format(KNR.predict([great_ingredients])))
```

Once again, I want to stress that just because you don't cover the entire spread of output values, from best to worst, you can't discount this model. And, a model that works with some data sets often will not work with other data sets.

But as you can see, scikit-learn makes it easy—almost trivially easy, in fact—to create and experiment with different models. You can, thus, try different classifiers, and types of classifiers, in order to create a model that describes your data.

Now that you've created several models, the big question is which one is the best? Which one not only describes the data, but also does so well? Which one will give the most predictive power moving

forward, as you encounter an ever-growing number of burritos? What ingredients should a burrito-maker stress in order to maximize eater satisfaction, while minimizing costs?

In order to answer these questions, you'll need to have a way of testing your models. In my next article, I'll look at how to test your models, using a variety of techniques to check the validity of a model and even compare numerous classifier types against one another. ■

### RESOURCES

I used Python (<http://python.org>) and the many parts of the SciPy stack (NumPy, SciPy, Pandas, matplotlib and scikit-learn) in this article. All are available from PyPI (<http://PyPI.python.org>) or from SciPy.org (<http://scipy.org>).

I recommend a number of resources for people interested in data science and machine learning. One long-standing weekly e-mail list is “KD Nuggets” at <http://kdnuggets.com>. You also should consider the “Data Science Weekly” newsletter (<http://datascienceweekly.com>) and “This Week in Data” (<https://datareplicablog.com/category/this-week-in-data>), describing the latest data sets available to the public.

I am a big fan of podcasts and particularly love “Partially Derivative”. Other good ones are “Data Stories” and “Linear Digressions”. I listen to all three on a regular basis and learn from them all.

If you’re looking to get into data science and machine learning, I recommend Kevin Markham’s “Data School” (<http://dataschool.org>) and Jason Brownlie’s “Machine Learning Mastery” (<http://machinelearningmastery.com>), where he sells a number of short, dense, but high-quality ebooks on these subjects.

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

**RETURN TO CONTENTS**

# The Current Phase of the Moon

Phase of the moon? It turns out that's really complicated.



DAVE TAYLOR

---

Dave Taylor has been hacking shell scripts on UNIX and Linux systems for a really long time. He's the author of *Learning Unix for Mac OS X* and the popular shell scripting book *Wicked Cool Shell Scripts*. He can be found on Twitter as @DaveTaylor, and you can reach him through his tech Q&A site: <http://www.AskDaveTaylor.com>.

---

PREVIOUS



Reuven M. Lerner's  
At the Forge

NEXT



Kyle Rankin's  
Hack and /

## LADIES AND GENTLEMEN, WE'VE LEFT MARS.

Well, at least I'm done with the Martian lander from the past few months. I hope you had chance to experiment with it and find out that it's not too easy to land a craft on any planet!

While researching the Martian lander project, I bumped into another interesting scripting problem that relates to space. How do you ascertain the phase of the moon for a given date? There are formulas, of course, and you can do the math knowing that the lunar rotation is precisely—um...well, it's not quite that simple, actually.

### Sidereal versus Synodic Period

Sure, you can just say that the moon orbits the Earth every 27.3 days, but that's relative to the stars, the sidereal orbit. The period between moon phases (such as a full moon) is also known as a synodic orbit, and that's 29.5 days.

So the simple task of ascertaining whether it's a full moon already has some math involved. Then there's the issue of the moon's illumination level being relative to where you are on Earth too. That makes sense. A full moon in Punta Arenas, Chile, is different from that in Lapland (though not by much).

The long and short of it is that the math behind calculating the illumination level of the moon isn't quite as simple as it may seem. You could take a known date and time of a full moon (for example, November 14 at 8:52 am EST) and keep adding precisely 29.530 days or 42,523.20 minutes.

### Or You Can Scrape a Website!

But seriously, you also can let someone else do the work too, right? I mean, this column is just about a shell script, after all. So, let's see how Google does it! If you check Google to see the current phase of the moon, it actually references a website (<http://moongiant.com>), as shown in Figure 1.

Do a bit of digging at the Moon Giant site, and you can see that there are two basic forms of URL that produce the data desired: a

TODAY - Monday, October 3, 2016. The **Moon** today is in a Waxing Crescent **Phase**. A Waxing Crescent is the first **Phase** after the **New Moon** and is a great time to see the features of the **moon's** surface. During this **phase** the **Moon** can be seen in the wester sky after the sun dips below the horizon at sunset.

**Today's Moon Phase - Moongiant**  
[www.moongiant.com/phase/today/](http://www.moongiant.com/phase/today/)



Figure 1. Google reports the current phase of the moon.

specified date or just “today” as the date. Test it by going to this URL: <http://www.moongiant.com/phase/today>.

Specify a date, and the format gets a wee bit more complex: <http://www.moongiant.com/phase/MM/DD/YYYY>.

You can use this find out the phase of the moon on the day the new US President will be sworn in with: <http://www.moongiant.com/phase/01/20/2017>. (If you guessed it’s a full moon, well, you’re not right!)

### Phase of the Moon, VI

This means that you can quite easily write a succinct script that tells you the current illumination level of the moon by simply using curl or GET with the first of these three URLs:

```
url="http://www.moongiant.com/phase/today"
pattern="Illumination:"
phase=$( curl -s "$url" | grep "$pattern" | tr ',' ' \
' | grep "$pattern" | sed 's/[^\d]//g' )
echo $phase
```

A quick run of the script as I write this (on October 3, 2016), and the output is a rather confusing: “6”. Six. What does that mean? It’s actually just the illumination level with everything else scrubbed out of the output data.

A 6% illumination is close to a new moon, but not quite. The new moon was actually two days before, on October 1st.

The interesting part of the script is absolutely all in the phase= statement. Let’s unwrap it and look more closely:

```
curl -s "$url" |
grep "$pattern" |
tr ',' ' \
' | grep "$pattern" |
sed 's/[^\d]//g'
```

First off, if you aren’t familiar with curl, go read the man page. It’s a terrific, quite powerful utility that lets you debug web servers, send queries to web pages as if you were various web browsers, interact

with FTP servers and, of course, just grab a web page's source for further analysis. It's the latter skill I'm using for this task.

Once the source to the page is flowing in, the next step in the pipe is to extract the line that contains the illumination level. That turns out to be exactly "Illumination:", but unfortunately, it doesn't appear by itself on the HTML source line. In fact, it's quite a complex output line! That's the job of the next two lines actually.

The invocation to `tr` turns every comma into a hard return, effectively breaking up one really long line into a lot of shorter lines. Then `grep` is invoked a second time to extract the now further isolated illumination level indicator.

Finally, superfluous data is axed by having `sed` remove everything that's not a digit. The end result? Input like `Illumination: 6%` turns into "6", and that's stored in the variable `phase`. Got it?

Now the output can be enhanced:

```
echo "The moon's current illumination level: $phase%"
```

Slightly more understandable output!

### But What Phase Is It?

Phases of the moon aren't generally described by their illumination level, however, and require knowledge of the previous day's state too, since that's how you ascertain "waxing" or "waning".

Some are easy: 0% is a new moon, 25% is a quarter moon, 50% is a half moon, and 100% is a full moon. Or is it? Actually, there are eight phases to the moon, and 50% illumination is known as a "quarter moon", confusingly enough.

In fact, the phase depends on where in the new moon → new moon cycle it is, so that 50% illumination prior to a full moon is the "first quarter" phase, while 50% illumination subsequent to a full moon is the "last quarter" phase—crazy complicated.

Again, let's simplify, however. So skip the waxing and waning for now and instead use the following:

- 0–5% = new moon.

- 6–45% = crescent.
- 46–55% = quarter.
- 56–95% = gibbous.
- 96–100% = full moon.

Now let's code that. Most easily, that can be done with a chain of if-then-else statements:

```
if [ $phase -lt 5 ] ; then
    phasename="new"
elif [ $phase -lt 45 ] ; then
    phasename="crescent"
elif [ $phase -lt 55 ] ; then
    phasename="quarter"
elif [ $phase -lt 95 ] ; then
    phasename="gibbous"
else
    phasename="full"
fi
```

With the aesthetically pleasing results:

```
$ potm.sh
The moon is currently crescent with 11% illuminated.
```

Let's stop here for this article. In my next article, I'll add the ability to analyze whether it's waxing or waning (for example, compare yesterday's illumination level with today's to see if the moon is getting brighter or darker). ■

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

**RETURN TO CONTENTS**

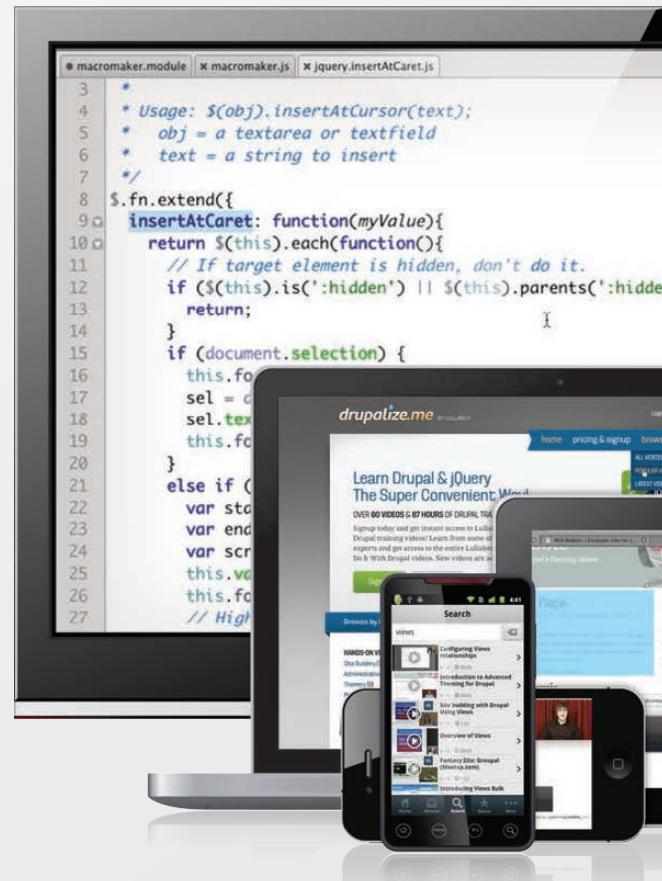


## Instant Access to Premium Online Drupal Training

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

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

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



# Orchestration with MCollective

Use MCollective to pick up where tools like Puppet leave off.



KYLE RANKIN

---

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

---

PREVIOUS  
Dave Taylor's  
Work the Shell

NEXT  
Shawn Powers'  
The Open-Source  
Classroom

---



I ORIGINALLY GOT INTO SYSTEMS ADMINISTRATION BECAUSE I LOVED LEARNING ABOUT COMPUTERS, and I figured that was a career that always would offer me something new to learn. Now many years later that prediction has turned out to be true, and it seems like there are new things to learn all the time. In particular, every now and then a new technology comes around that dramatically changes how sysadmins do their jobs. For instance, in the October 2012 issue of *LJ*, I wrote an article titled "How to Deploy a Server" where I described the progression of how sysadmins deployed servers from by-hand bespoke configuration, to images, to post-install scripts and finally with

configuration management.

So in this article, I'm going to expand on that concept to talk about how to use orchestration tools (in particular, MCollective) to manage orchestration tasks on servers post install. Many MCollective installation guides already exist, so I won't repeat that here; instead, my goal is to provide examples of how these tools can automate administration tasks further and to describe how I personally use them. And although I'm specifically discussing MCollective, these same concepts can be adapted and applied to any number of other orchestration tools.

These days, configuration management still is one of the most popular ways for sysadmins to configure a server, but over time, many administrators started pushing these tools past configuration management into what's being called orchestration. Orchestration refers to tools to help you push changes—in particular, software installation and updates—across your environment in a measured, staged way.

Although some administrators might be fine with pushing software updates randomly, if you want smooth upgrades, usually you want to follow an approach where you might update one server first, then if that succeeds, update a few more before updating the rest. Before you update software, you may want to notify upstream systems so they can stop sending traffic, and after you update the software, you may want to restart the service. This process is nothing new; it's just that in the past, administrators would do this by hand by logging in to machines one by one, or they would write custom scripts. With orchestration tools, you can perform these same steps from a centralized location.

The line between configuration management and orchestration is bit clearer with tools like Puppet and Chef than, say, with SaltStack or Ansible. Although Puppet and Chef can run in a masterless way, the default approach is to have clients check in to a master server periodically to see whether they comply with the central configuration and if not, to change until they do. Usually you have clients check in to the master in a somewhat randomized way or otherwise send them a trigger to apply changes.

Because tools like SaltStack and Ansible work on top of SSH, they already include an orchestration component that allows you to trigger certain kinds of changes from a central place in a staged way. Although you can make Puppet and Chef perform orchestration, many

administrators who try it end up becoming frustrated and replace the tool with something else instead of realizing that those tools are very capable of what they were built to do but just not as strong at orchestration.

I personally ran into a similar kind of frustrating situation myself with Puppet when I was trying to use it to stage software updates. Puppet works great for configuration management, but wasn't ideal for orchestration in my experience. Instead of throwing away Puppet, I simply supplemented it with a very powerful tool, MCollective, that was expressly intended for orchestration and integrates well with Puppet.

MCollective lets you send out commands that query the value of particular Puppet facts, start and stop services, query and update software, and even start Puppet itself. MCollective also can restrict which servers run the command with the same facts from Facter that Puppet uses. So, for instance, you could send out a command that executes only on machines running a particular Linux distribution.

Although many orchestration tools exist, most of them take a glorified "SSH for loop approach", and the end result is some centralized admin host that has SSH root access everywhere and runs commands one server at a time. MCollective has a strong security model where your commands are restricted to specific plugins that exist on each client, and when you run a command from your admin node, it is signed with your user's local key and sent to a central job queue. Each client checks whether the command is intended for it, and if so, it picks up the job off the queue, validates the signature, and if the plugin is installed, only then will it execute the command. With this security model, attackers can't compromise the job queue and inject new jobs, because they can't sign them, and if attackers compromise the administrative node, they are restricted to whatever plugins you have enabled. Also, because MCollective uses a job queue, commands run in parallel, so a command sent to 50 servers should return about as fast as a command sent to one.

Instead of describing every default MCollective plugin and its arguments, a better way to illustrate MCollective as an orchestration tool is to walk through how it helps automate what is (unfortunately) a pretty frequent task for sysadmins these days: patching a security hole in OpenSSL. The basic steps an administrator would have to perform by hand on each server would be the following:

- Check what version of OpenSSL is installed on a server. Proceed with the rest of the steps if it isn't up to date.
- Update OpenSSL.
- Confirm that the OpenSSL package is now the patched version.
- Restart any services on the host (like Apache, nginx or PostgreSQL) that use OpenSSL so they load the new library.

Although you certainly could use a configuration management tool to make sure that you always are running the latest version of OpenSSL, the process of restarting any services that use OpenSSL is probably not something you want to occur at random the next time the client checks in. Here's how you could perform the above steps using MCollective from a central admin host.

The `package` plugin allows you to query packages on a system, and this particular command polls all of the hosts in your environment at the same time and returns the version of the OpenSSL package each of them has:

```
mco package openssl status
```

You also can use the `package` plugin command to update packages, and this particular command updates OpenSSL on every host in your environment to the latest version:

```
mco package openssl update
```

In the output, it will return with a complete tally of how many hosts have OpenSSL installed and at what version.

The `service` plugin lets MCollective start, stop, restart and query the state of init services on a system. This particular command restarts the nginx service on every host in your environment at the same time:

```
mco service nginx restart
```

Any hosts that don't have an nginx service will safely do nothing. You could replace nginx in the above command with any other init service on your system.

So there you have it. With three commands, I could patch OpenSSL and restart nginx across the entire environment. If I had just needed to patch bash (such as back in the days of the Shellshock vulnerability), I could have done it with a single mco package bash update command.

Of course, most administrators won't want to apply a command (especially a restart command) across every server at the same time. Instead, you want to stage things to parts of your environment at a time. The simplest way to do this is with the -I argument that lets you apply a command to a particular server. So for instance, you could reboot nginx only on web1.example.com like this:

```
mco service nginx restart -I web1.example.com
```

MCollective allows you to apply very sophisticated filters to your commands so that they apply only to particular groups of hosts with the -W argument. For example, if you wanted to update OpenSSL only on hosts running Debian 8.5, you could type:

```
mco package openssl update -W "operatingsystem=Debian  
→operatingsystemrelease=8.5"
```

What's more, because these filters can be based on Facter facts, you don't have to maintain and update local lists of server categories like back in the bad old days of SSH for loop scripts. So for instance, if you spin up a new Debian 8.5 server in AWS, the next MCollective command you run that happens to reference the distribution version, fact will return this server in the results without your having to do anything. You even can use the mco find command to return a list of all of the servers that match a particular fact:

```
mco find -W "operatingsystem=Debian operatingsystemrelease=8.5"
```

You can use any facts that show up in the output from the facter

command, and if you use Puppet, you also can take advantage of any custom facts from Puppet. So for example, the way that I take advantage of this is to split up my hosts into different high-availability groups based on the number in a host's hostname. In my case, when I create a host in AWS, I divide the availability zones into three groups, and the number in the hostname reflects one of those groups. So all hosts with a 1, 4 or 7 in their hostname, for instance, would be in one availability zone; 2s, 5s and 8s would be in another; and 3s, 6s and 9s in another. I then set a custom fact in Puppet I called hagroup to a, b or c, based on which of these three groups the host is in. So if I wanted to update OpenSSL across all servers but only restart nginx in a fault-tolerant way, I might do something like this:

```
mco package openssl update  
mco service nginx restart -w hagroup=c
```

This way, I restart nginx only in a third of my environment. If there were some kind of problem, the other two-thirds of the environment would be fine. Then I would wait for all the nginx hosts in that group to return, and repeat the `nginx restart` command for `hagroup=b` and then finally `hagroup=a`. When I'm updating software that possibly could crash or packages that automatically restart the service after an update, I also limit the package update command to a particular hagroup.

What's nice about MCollective is that because you can limit it based on facts that are set automatically on each system, it's particularly easy to create shell scripts that wrap around a group of MCollective commands to perform common sysadmin tasks (like, say, upgrading OpenSSL) that apply in a consistent but fast and automated way. You also can extend MCollective with your own custom plugins that are relatively easy to write.

In my next article, I plan to describe how I wrapped a series of MCollective commands, including some custom plugins we wrote in-house, to automate all of the steps you would normally do by hand to upgrade in-house software on production systems. ■

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

**RETURN TO CONTENTS**

# The Family Dashboard in PHP

Tired of explaining how to log in over the phone?  
Make a dashboard!



**SHAWN  
POWERS**

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

---

PREVIOUS  
Kyle Rankin's  
Hack and /

NEXT  
New Products

---

I'VE WRITTEN A LITTLE ABOUT PHP BEFORE, because I think it's a great utility language for writing quick things you need to do. Plus, it allows you to use a web browser as your interface, and everyone has a web browser. That makes it very convenient for my family, because I can make simple web interfaces for the various things I normally have to do from the command line. (This is extremely useful when I'm gone to a conference and the Plex server needs to be rebooted, or any of a dozen other things need to be done that are hard to explain over the phone.)

My "Family Dashboard" will look different from yours, but the concept is pretty simple. PHP allows you

to execute local functions on the server, and so as long as you can create a bash script that does what you need it to do, it can be launched from the “dashboard” you create for your family. Here’s a sample dashboard file I’ve created, so you can see how simple it is to create a custom page that does what you need it to do (see Figure 1 for a screenshot of the dashboard in action):

```
<html><head><title>My Dashboard</title></head>
<body>
<h3>You need to enter some commands and possibly options,
  ↪or just press a button:<br />
<button onclick="window.location='lj.php?command=weather&
  ↪option=houston'">Weather</button>
<button onclick="window.location='lj.php?command=bing'">Bing
  ↪Photo</button>
<button onclick="window.location='lj.php?command=uname'">Kernel
  ↪Name</button>
<button onclick="window.location='lj.php?command=time'">Unix
  ↪Time</button>
</h3>

<?php

$command = $_GET['command'];
$option = $_GET['option'];

switch ($command)
{
    case "weather":
        echo file_get_contents("http://wttr.in/$option");
        break;
    case "time":
        echo time() . " <-- that's how I read time! I'm a robot!";
        break;
    case "bing":
        $json = json_decode(file_get_contents("http://www.bing.com/
```

```
→HPIImageArchive.aspx?format=js&idx=0&n=1&mkt=en-US"), TRUE);  
    $url = "http://bing.com" . $json['images'][0]['url'];  
    echo "Here is the image of the day:\n";  
    echo "<img src=$url />";  
    break;  
case "uname":  
    echo shell_exec("uname -a");  
    break;  
default:  
    echo "<h1>Press a button!</h1>";  
  
}  
  
?  
</body></html>
```

First off, copy and paste that code into a file called lj.php and save it onto your local web server. The server needs to have PHP active, but I'll leave that as an exercise for the reader to set up. I've written about installing a LAMP stack before, so it shouldn't be too challenging to get a web server running with PHP support (see my article "PHP for Non-Developers" in the December 2014 issue or at <http://www.linuxjournal.com/content/php-non-developers>). Also, naming the file "lj.php" is only important because if you look at the code, it references itself. If you name it something different, just change the references in the HTML/PHP code.



Figure 1. My dashboard is simple, but it's just a front end for the code beneath.

Before learning how the code works, test it out and watch it work. If you can't host the file yourself, but want to see it in action, you can use my server for testing. Just head over to <http://snar.co/php-dashboard>, and it should redirect you to a hosted version of this file. Click the buttons, and see if you can figure out what's going on. Can you get the local forecast for your area?

### What's with the GET and Switch Stuff?

It's possible to create a separate PHP file for every action you need to accomplish. That is a lot of PHP files, however, and it still doesn't give you the ability to receive input to use in the PHP file itself. I want my family to have a single URL, and I want all my code in a single file. It's just easier that way. First I'll explain what the `$_GET` variable does.

As you click the buttons on the page, you should look at the address bar on your browser. When you click on the weather button, for instance, you should see this in the address bar:  
`http://your.server.here/lj.php?command=weather&option=houston`.

That stuff at the end is how you tell the PHP script what information you want it to display. All the variables you assign are put into an array called `$_GET`. So in the weather example above, I've assigned two variables. To reference them inside the PHP script, you use the `$_GET` array. So in the URL above, these two variables are assigned:

```
$_GET['command'] = "weather";
$_GET['option'] = "houston";
```

And, you can use those variables in your PHP code. Notice that I've actually assigned those two variables to standard variable names, so that it's easier to reference them later. You can change what variables are sent to the PHP script by changing the information in the URL. That allows the script to be dynamic and provide output based on the input you give it. In fact, the only reason pressing those buttons works is that it loads the page with the arguments already in place! See if you can get your local weather now by changing the "option" variable in the URL and loading the page. Cool, huh?

## More Than Just Weather

Since you're able to send your PHP script variables via the URL, that means your dashboard can do much more than just show the weather. Based on the variables, you can call different commands with the `switch` construct in PHP. It's like a `CASE` statement in other languages, and the logic is pretty straightforward.

You run the `switch` statement on the `$command` variable assigned from the `$_GET` array. If the variable matches any of the options listed as a "case", it executes the code in that section, then you `break`; out of the `switch` construct. If the `$command` variable doesn't match any of the case options, the `switch` executes the `default:` section at the end. In this example, it's a message to press a button.

Let's look at each section to see what's going on when you press a button (or manually enter the command in the URL).

## The Part before the PHP

If you put standard HTML into a PHP file, and don't enclose it between `<?PHP ?>` tags, it just sends it to the web browser as HTML code. So the top of the `lj.php` file is just plain HTML. The text is shown in `<h3>` tags, and the buttons are created with a little bit of JavaScript that allows them to load the URL specified. If the buttons and JavaScript make you uncomfortable, it's okay to make standard text links that point where you want them. I just like buttons because they look cool.

It's important to realize that the buttons aren't doing anything other than loading the page with `$_GET` variables assigned in the URL. The buttons themselves don't execute code, and aren't anything fancy. You can type the URL out by hand and achieve the same thing. Your family will appreciate it if you make them links or buttons though, because clicking is much easier than typing long, complicated URLs!

## Weather

If you click the weather button, or enter the URL by hand to send the `$_GET['command']` and `$_GET['option']` variables to the script with `weather` as the command, the `switch` statement will execute the code inside the `"weather":` section.

This is a really simple command that just echoes (prints on the screen)

the results from fetching the web page. The `file_get_contents` function in PHP will get the contents of a local file or a file on the internet. In this instance, you create the URL with your `$option` variable. If you clicked the button, you'll notice `$option` is set to "houston", but you can change the URL by hand in order to get your local weather. It will accept city names, ZIP codes and even airport codes.

The weather section of the script is the only one that looks at the `$option` variable, but it's possible to assign as many variables as you want from the URL. If you assign a variable and it isn't used, there's no harm, it's just ignored.

### The Time?

The "time" section doesn't return what you'd expect for a time button to return. In fact, I labeled the button that loads that page "Unix Time", because I used the `time()` function in PHP, which displays the number of seconds that have elapsed since January 1, 1970. That might not seem like a terribly useful number, but it's very convenient when programming, because you don't have to parse out hours, minutes and so on. You can click (or refresh) the page a few times, and you should see the number increment.

UNIX time (sometimes called Epoch Time) is fun to play with, and although this example isn't terribly useful, I wanted to include it so you could see how the `time()` command works, along with the `echo` command. If you look, there is a single period after the `time()` function. That concatenates the two items into a single string and displays it all together. If you click the button, you'll see what I mean.

### Bing? How Dare You Load a Microsoft Page!

The Bing photograph of the day is always awesome (Figure 2). Really, Microsoft does a great job of procuring incredible photos, and I love to see them. Since the URL is always different, this was a great way to show how to load JSON into a variable and then extract an array element. Don't let the scary looking code intimidate you; JSON is really cool. Basically, you load the JSON from that long Bing URL and put it into a PHP array. Then, you form the URL for the photo from the contents of that array. Here's a snippet of code you can use to see

the array in a more readable form:

```
<?php  
  
$json = json_decode(file_get_contents("http://www.bing.com/  
➥HPImageArchive.aspx?format=js&idx=0&n=1&mkt=en-US"), TRUE);  
  
echo "<pre>";  
print_r($json);  
echo "</pre>";  
  
?>
```

You need to enter some commands and possibly options, or just press a button:

[Weather](#) [Bing Photo](#) [Kernel Name](#) [Unix Time](#)

Here is the image of the day:



Figure 2. The Bing photos are always so cool.

If you don't have a server, head over to <http://snar.co/php-json> to see the results of the PHP file. You can see where I got the information to build the URL for the image, and in the `switch` statement, you can see it just loads the image based on that URL. Isn't JSON cool?

### Local Scripts

This part of the `switch` statement is powerful, but also a little scary. If you click on the "Kernel Name" button, you can see it executes the code in the `uname` section of the `switch` statement. Using the `shell_exec` command, you can execute a file on the local server and show the results in the browser window. This is powerful because it means you can have your family execute local bash scripts by clicking on a button. But it's a little scary, because you're executing local commands on your server by clicking a button!

The script is executed with the permission of the web browser, so, for example, in Ubuntu running Apache, the `www-data` user would be executing the command. If that user doesn't have permission to do something in the script, the script will fail. This is one of those "with great power comes great responsibility" things. It can be incredibly useful, but also incredibly dangerous, especially if your server is exposed to the internet!

### Troubleshooting

Whenever I write PHP code, I make mistakes. Usually it's a forgotten semicolon or a mismatched bracket. It can be very annoying when you load the page, and it's suddenly just blank instead of showing you an error. In the last article I wrote about PHP, I showed how to turn on PHP errors so you could see in the web browser what's going wrong. I don't do that anymore, because it's annoying to see PHP warnings when things are working fine. So what I do now is run `php` from the command line. If the code is broken, it will show errors on your command line, and you won't have to worry about turning error logging on and off in your web browser. For example, in the example `lj.php` file, just go to the folder where it's stored and type:

```
php lj.php
```

And the server will dump the HTML to your command line as if it were a web browser. If there's an error, it will tell you what you did wrong. I like that method of error checking much more than getting error notifications in my web browser, but if you prefer to see them on the browser, look back to my PHP article from the December 2014 issue and see how to activate error logging.

Just like last time, I'm giving you only a taste of the sorts of things you can accomplish with PHP and a little ingenuity. If you come up with an interesting dashboard of your own, I'd love to see it, even if it's just a screenshot. (Don't expose your dashboard to the internet, especially if it controls your local server with `shell_exec` statements!) Feel free to email me at [shawn@linuxjournal.com](mailto:shawn@linuxjournal.com), but be sure to put "DASHBOARD" in the subject line, or I might assume it's spam. I get so much darn spam!■

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

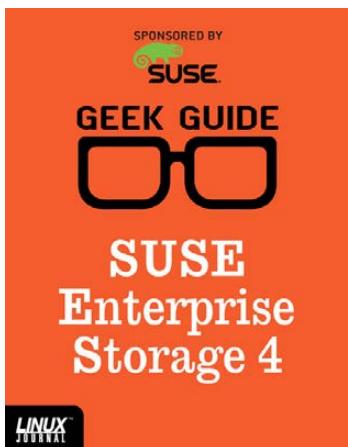
**RETURN TO CONTENTS**

# Linux Journal eBook Series

# GEEK GUIDES

Practical books for the  
most technical people on the planet.

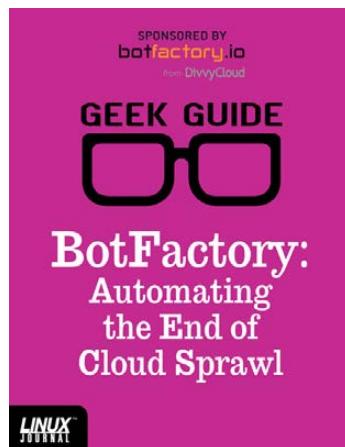
**FREE**  
Download  
**NOW!**



## SUSE Enterprise Storage 4

**Author:**  
Ted Schmidt

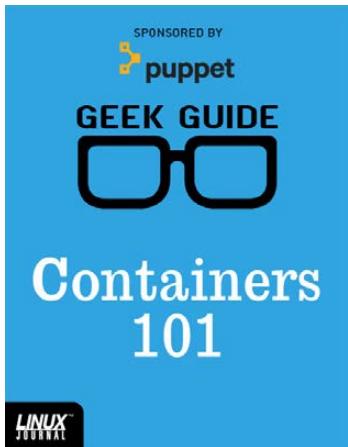
**Sponsor:**  
SUSE



## BotFactory: Automating the End of Cloud Sprawl

**Author:**  
John S. Tonello

**Sponsor:**  
BotFactory.io



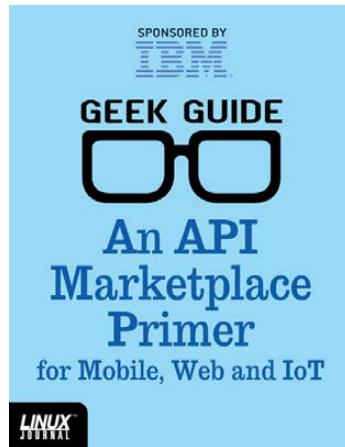
## Containers 101

**Author:**

Sol Lederman

**Sponsor:**

Puppet



## An API Marketplace Primer for Mobile, Web and IoT

**Author:**  
Ted Schmidt

**Sponsor:**  
IBM

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

# NEW PRODUCTS

---

PREVIOUS



Shawn Powers'  
The Open-Source  
Classroom

NEXT



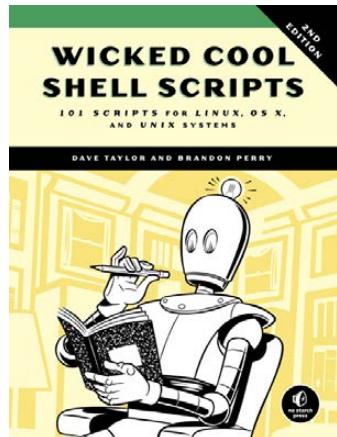
Feature:  
Provisioning Docker  
with Puppet

---

## Dave Taylor and Brandon Perry's *Wicked Cool Shell Scripts* (No Starch Press)

The new second edition of Dave Taylor and Brandon Perry's classic *Wicked Cool Shell Scripts* features a smorgasbord of classic favorite scripts and 23 brand-new ones. Subtitled *101 Scripts for Linux, OS X, and UNIX Systems*, Taylor and Perry's guide features a collection of useful, customizable and fun shell scripts for solving common problems and personalizing one's computing environment. Each chapter contains ready-to-use scripts and explanations of how they work, why one would use them and suggestions for changing and expanding them. Highlights of these not just useful but also wicked cool scripts include a disk backup utility that keeps files safe from a system crash, a password manager, a weather tracker, several games, a ZIP code lookup tool, a Bitcoin address information retriever, as well as tools for cloud services, bulk file management and image processing and editing. Whether users want to save time managing their systems or just find new ways to goof off, these scripts are just the ticket.

<http://nostarch.com>





## USMobile, Inc.'s Scrambl3

The special sauce in USMobile, Inc.'s Scrambl3, the mobile app that facilitates "the world's most private calls and messages", is a set of open-source components that create a top-secret-grade VPN, encryption algorithms and internet protocols. USMobile says that, for myriad reasons, Scrambl3 stands head and shoulders above WhatsApp and Viber for security and privacy. For instance, new Scrambl3 Android and iOS users are asked to provide only a user name and password sans verification of a cell-phone number, access to private cell-phone contacts and email addresses, all of which is "a hassle and a violation of your personal privacy". Scrambl3 provides many advantages since it does not rely on a cell-phone number. For example, Scrambl3 can be used on Wi-Fi-connected tablets, and attackers cannot listen in on user calls and texts by exploiting the public telephone system's SS7 security flaw. Finally, Scrambl3 users privately exchange their user names and add them to their respective Scrambl3 Black Book contact listing, making unwanted calls or messages impossible on users' private networks.  
<http://scrambl3.com>



## Permabit Technology Corporation's Albireo VDO for Ubuntu Server

In perfect alignment with its self-described identity as “the data reduction expert”, Permabit Technology Corporation recently announced availability of its Albireo Virtual Data Optimizer (VDO) 6 for Canonical’s Ubuntu Server. VDO data reduction enables enterprise hybrid cloud data centers and cloud service providers to reduce their storage footprint, increase data density and avoid costly data-center expansions, resulting in “massive savings on data-center investment”. Permabit says its move to Ubuntu Server 14.04 LTS—and imminently 16.04 LTS, as well—is the only modular data reduction solution available for the Linux block storage stack. The move occurred due to Ubuntu’s place in the forefront of large cloud infrastructure deployments and its deep involvement in the OpenStack project. VDO leverages Permabit’s patented deduplication, HIOPS Compression and thin provisioning technologies.

<http://permabit.com>



## CloudBees Jenkins Enterprise

Although open-source software excels at innovation and leverages the immense power of talented developers dedicated to solving difficult problems, the focus is rarely on enterprise capabilities, asserts CloudBees, the hub of enterprise Jenkins and DevOps. Fortunate for Jenkins developers, CloudBees, Inc., has announced CloudBees Jenkins Enterprise, a Jenkins distribution aimed directly at enterprises that “ensures the highest levels of testing and verification, providing smooth upgrades and the most reliable and stable Jenkins foundation for software development and DevOps teams”. This enterprise distribution of Jenkins is possible due to CloudBees’ new comprehensive testing and verification process called the CloudBees Assurance Program. Consisting of a trio of engineering, QA and machine resources, the program is dedicated to verifying the stability, security, inter-compatibility and upgradability of the Jenkins’ core along with the curated set of the most popular third-party open-source Jenkins extensions that round out the distribution. As DevOps needs rapidly evolve, enterprises require confidence that they are implementing the most feature-rich, reliable and secure Jenkins-based continuous delivery platform possible, which they now have at their disposal.

<http://cloudbees.com>



## **Jetico's BestCrypt Container Encryption for Linux**

For users in search of a commercially supported encryption tool for Linux with a backdoor-free guarantee, Jetico recommends its recently updated BestCrypt Container Encryption for Linux 3.0. Jetico's BestCrypt Container Encryption automatically encrypts any selected files or folders on an active computer, shared workstation or network storage in Linux, Windows and Mac OS environments so that nobody can gain access without the right password or keys. Jetico says that BestCrypt is easy to install, easy to use and totally transparent—meaning it actually gets used. The new version 3.0 of BestCrypt features keyfile support, one-click installation and access to binary packages for popular Linux distributions and a graphics and usability makeover. Jetico adds that while drive or disk encryption safeguards from physical threats, like lost or stolen devices, it fails to protect online storage or computers connected to the internet. Jetico's BestCrypt ensures that encrypted files stored in the cloud can be accessed on Linux.

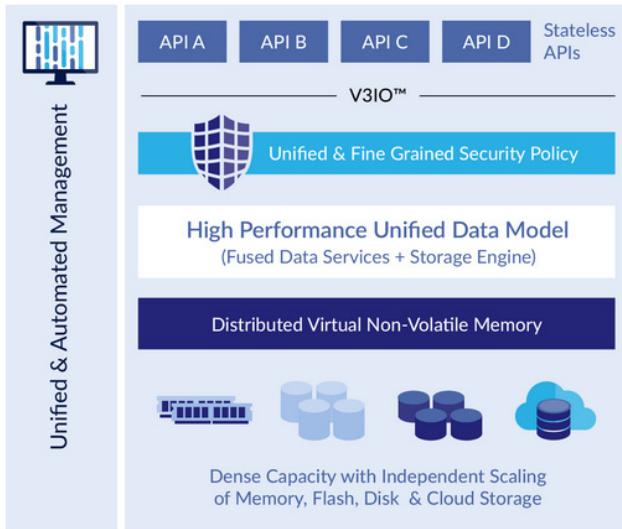
<http://jetico.com>



## Applied Expert Systems, Inc.'s CleverView for TCP/IP on Linux

One of the most important characteristics of the contemporary data center, notes Applied Expert Systems, Inc. (AES), is that an ever-increasing amount of the traffic is between servers. Realizing the resulting need to facilitate improved server-to-server communications, AES developed CleverView for TCP/IP on Linux v2.5 with KVM Monitoring. CleverView gives IT staff access to current and historical server performance and availability details from not only their browser desktops but also their cell phones via the CLEVER Mobile for Linux app. The highlight of this version 2.5 is the new ability to monitor KVM guest support providing clear and concise information on availability and performance. KVMView shows CPU count, memory used, max memory and CPU used with the ability to drill down into the TCP/IP statistics for the selected KVM Guest. The new enterprise-wide metrics provide a crisp, clear and concise view allowing trend, pattern and anomaly identification. Reports provide for more effective decision-making to meet today's dynamic anywhere-anytime service demand.

<http://aesclever.com>



## iguazio's Enterprise Data Cloud

The description of iguazio's new flagship Enterprise Data Cloud platform is bold and simple: the world's fastest, simplest and lowest-cost enterprise data cloud. iguazio adds that unleashing the full potential of megatrend applications and analytics for big data, IoT and cloud-native applications, it has pioneered a new service-driven approach to enterprise data management, redesigning the entire data stack to accelerate performance and bridge the enterprise skill gap. iguazio's Enterprise Data Cloud, asserts the firm, is the only secure data platform-as-a-service deployed either on-premises or in hybrid cloud architectures, with self-service portals and APIs for developers and operators. The new unified platform delivers a breakthrough in application performance and efficiency. With only four data appliances, enterprises can store up to 10 petabytes per rack, with costs starting at \$0.03 per gigabyte per month. The platform delivers 10 million transactions per second and throughput of 50 gigabytes per second with sub-100 microsecond application latencies, across streaming, NoSQL, objects or files. iguazio's "revolutionary stack" supports simultaneous high-performance access through multiple industry-standard and Amazon-compatible APIs.

<http://iguaz.io>



## **GENIVI Alliance's GENIVI Vehicle Simulator**

By providing a realistic simulated driving experience, the new GENIVI Vehicle Simulator (GVS) assists adopters to develop and test the user interface of an open in-vehicle infotainment (IVI) system safely, thereby identifying and executing necessary design changes quickly and efficiently. The open-source, extensible driving simulator was developed under the auspices of the GENIVI Alliance by Elements Design Group and the Jaguar Land Rover Open Software Technology Center. Key features of GVS' realistic driving experience include obstacle triggering, infraction logging and infraction review.

<http://projects.genivi.org/gvs>

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

**RETURN TO CONTENTS**

# PROVISIONING DOCKER WITH PUPPET

Docker containers are great, but Docker hosts and instances still need to be managed. Configuration management tools like Puppet can work hand in hand with Docker, and their powerful domain-specific languages (DSLs) make light work of things that are tricky or impossible to do in Docker itself.

TODD JACOBS



PREVIOUS  
New Products

NEXT

Feature:  
Low Power Wireless:  
Routing to the Internet



DevOps and containerization are currently all the rage in agile IT shops. However, there are valid differences of opinion about the appropriate demarcation between containerization and configuration management technologies in a well designed DevOps toolchain. Both technologies have specific use cases, although there is also a great deal of overlap between them. In many cases, the technologies complement one another and are intended to work together. This article focuses on building synergy between Docker and Puppet, and shows how the two technologies can work together to provide a more robust DevOps environment than either tool can manage alone.

One way of looking at Docker is as a modern take on the IT practice of deploying “golden images” onto a server. In the simplest cases, Docker images bundle up a service and its runtime dependencies into an easily deployable, self-contained unit. A static image is a great solution for reliably deploying services to the data center, but on the other hand, the lack of configurability can lead to needlessly bloated service catalogs as new images are built for each different hard-coded configuration.

Docker supports a limited number of features for enabling configuration changes at build time and runtime, but actively maintaining complex configurations is not what Docker is designed to do. You certainly can tweak Docker images with environment variables, command-line options, layered images, Docker volumes with configuration data or custom scripting, but all of those options come at the cost of additional complexity. Without a robust domain-specific language and ongoing convergence to a defined state, using Docker

**Docker supports a limited number of features for enabling configuration changes at build time and runtime, but actively maintaining complex configurations is not what Docker is designed to do.**

features to support dynamic configurations often leads to the very problems that dedicated configuration management tools like Puppet were designed to solve.

Why focus on just Docker and Puppet, when there are other tools in that space? Although there certainly are alternatives, when one thinks of containerization, Docker is far and away the current mindshare leader. The race is a little tighter in the configuration management space, but Puppet is definitely a strong enterprise-class solution that has been widely deployed for more than a decade. I focus on Docker and Puppet for the remainder of this article in order to avoid gross generalizations that may not apply equally to every containerization or configuration management tool currently available.

### Use Cases for Integrating Docker and Puppet

If you're deploying services into a Docker-based infrastructure, why add Puppet to the toolchain? After all, one of the most common uses of Docker is to deploy preconfigured services out to the cloud. If the services are preconfigured, how would using a configuration management tool like Puppet improve the process?

There are three core use cases to consider. The first is using Puppet to provision the Docker service on a host, so that it is available to manage Docker instances. The second is provisioning specific Docker instances, such as a containerized web service, onto a managed host. The third is managing complex or dynamic configurations inside Docker containers using Puppet agents baked in to the Docker image. In this article, I address only the first use case, but I plan to address the others in future articles.

### Provisioning Docker with Puppet

Docker is a great tool, but it isn't installed by default on most Linux distributions. When your cloud or data center has tens of thousands of nodes running Linux, how do you install the Docker dæmon on only the nodes that need it?

In a homogeneous server farm, you might use a template or image that already has Docker on it, and whatever process spins up the node will ensure that Docker is available for running containers. However, in a

more heterogeneous environment, only *some* of the nodes will be hosting Docker containers, and without a configuration management system, you can't easily add or remove Docker from existing hosts in a well controlled, automated way. Managing packages, files and services on existing nodes is what Puppet was made for!

Imagine that you have an Ubuntu server with no special characteristics or running services other than SSH. As a best practice, this server already should have the Puppet agent installed in order to avoid yet another bootstrap problem, but as long as you already have a Puppet master in your environment, placing this new node under configuration management is easy.

The specifics of doing this may vary slightly by Linux distribution, Puppet version and whether you are using Puppet open source or Puppet Enterprise. Although the commands may vary, the steps will be very similar.

First, you'll configure some basic items on the server. Second, you'll install the Puppet agent, register the client with the server and kick off the provisioning process.

### Server-Side Settings

**Automating Which Nodes Get Docker Installed** At scale, it can be very helpful to apply certain configurations automatically to nodes based on the node's hostname. To see this in action, create a Puppet environment named "example":

```
sudo mkdir -p \
/etc/puppet/code/environments/example/manifests
```

```
sudo touch \
/etc/puppet/code/environments/example/manifests/site.pp
```

**At scale, it can be very helpful to apply certain configurations automatically to nodes based on the node's hostname.**

And, assign the docker class to any Puppet client that has “docker” in its hostname:

```
# /etc/puppet/code/environments/example/manifests/site.pp

# Use a regular expression to assign the docker
# class to any node that contains "docker" in its
# hostname.
node /docker/ {
  include docker
}
```

**Autosigning Client Certificates** By default, Puppet operates in client/server mode and uses SSL certificates on both the client and server sides for authentication as well as transport security. If your Puppet master is not on the same machine as the client, you can make your life easier (but a little less secure) by allowing Puppet to sign client certificates from whitelisted hosts automatically. For example, depending on your naming conventions for subdomains and hostnames, one or more of the following entries could be adapted to whitelist Docker nodes:

```
# /etc/puppet/autosign.conf

docker-host-001.example.com
*.docker-hosts.localdomain
*.local
```

Autosigning client certificates reduces security in exchange for a dramatic boost in convenience and productivity. Within a secure network, this often is worth the modest risk. However, you also can configure more secure policy-based autosigning with Puppet, but doing so is well outside the scope of this article.

In the short term, if you prefer not to use autosigning, you can pass the `--waitforcert` flag to the Puppet agent and then manually approve unsigned client certificates on the Puppet master while the

clients wait. For small numbers of servers, the manual nature of this process is offset by the fact that it has to be done only once per client. However, as a process, it simply scales poorly. Policy-based autosigning is definitely the right way to go for the security-conscious enterprise.

**Defining a Docker Manifest** Now Puppet is ready to do its real job: using a declarative domain-specific language to place each node into a known state. Create the following manifest for Ubuntu to tell your designated Docker nodes how to install and start the Docker service:

```
# /etc/puppet/code/environments/example/manifests/docker.pp

class docker {
  package { 'docker':
    name    => 'docker.io',
    ensure  => present,
  } ->
  service { 'docker':
    ensure => running,
  }
}
```

Note that on recent Ubuntu versions, the Docker package was renamed docker.io to avoid conflicts with an unrelated package that is also named docker, but the service script and process name are still docker. This is potentially confusing, but the Puppet manifest above handles this situation with ease. The manifest should be tweaked for other distributions or replaced with a suitable module from the Puppet Forge that selects the appropriate package name based on the client's distribution and OS version.

## Client-Side Settings

Remember, you're managing the Puppet agent manually here because it's not yet part of your default OS installation. Once you have a configured Puppet master, and the Puppet agent is baked in to your default OS installation process, the client installation (and in many

cases the certificate management process) will be automatic.

**Installing the Puppet Agent** You've configured the basic elements of the Puppet server to support Docker. Now you need to configure the client. First, install the Puppet agent:

```
sudo apt-get --assume-yes install puppet-agent
```

Next, you have to assign the agent to a Puppet environment. By default, all Puppet agents are assigned to *production*, but it's a best practice to perform these sorts of experiments within a dedicated environment that won't impact your real production systems. There are ways to set the client's environment server-side, but they are out of scope for this exercise. Instead, you'll use the agent's configuration file and command-line options to ensure that you're using manifests defined for a dedicated, non-production environment:

```
# This command will configure the correct agent
# environment.
sed -i '$a\\n[agent]\\nenvironment = example' \
/etc/puppet/puppet.conf
```

If the command above worked properly, the `puppet.conf` file should now look similar to this:

```
# /etc/puppet/puppet.conf

[main]
ssldir = /var/lib/puppet/ssl

[master]
vardir = /var/lib/puppet
cadir = /var/lib/puppet/ssl/ca
dns_alt_names = puppet

[agent]
environment = example
```

Once the Puppet agent is installed, and assuming the existence of a Puppet master named `puppet` that has been configured to autosign client certificates, this system will install Docker on new nodes automagically any time the hostname matches the regular expression defined in the `site.pp` node list. Simply ensure that each node that should run Docker contains “`docker`” in its hostname and you’re done!

However, it’s often wise to kick off the first agent run manually, especially so that you can spot any problems with server connectivity or SSL certificates quickly. You already defined a Puppet environment named “`example`”, and the agent will attempt to contact a server named “`puppet`” unless directed otherwise by `puppet.conf` or on the command line. The following commands show how to define both the server and the environment explicitly, and it will override any configuration file settings or defaults if necessary:

```
# Running agent with puppet.conf and/or default
# values.
sudo puppet agent --test
```

```
# Overriding the server and environment values.
sudo puppet agent \
--test \
--server ubuntu-yakkety.localdomain \
--environment example
```

Once the Puppet agent has completed its run, you quickly can validate that the Docker service is running properly—for example:

```
# Show verbose status of the Docker service.
$ sudo service docker status

# Count of running Docker processes.
$ pgrep -c docker
1
```

## Putting It All in Context

At this point, you may be thinking that all this Puppet stuff seems like a lot more work than simply running `apt-get -y install docker.io` on each new Linux machine in your data center. In the short term, using Puppet to install Docker certainly requires more work up front. However, this will really pay off in the long term when you have large numbers of nodes to manage and you're attempting to provision them in a fully automated way. In addition, don't overlook the value of how easily Puppet can automate critical patches related to Docker, or the assurance that Puppet will enforce (and report any changes to) the expected status of the Docker service every time the Puppet agent runs.

In my next article, I'll expand on Docker/Puppet integration further to install, start or stop Docker containers across the data center based on centrally managed roles and profiles. If you aren't already convinced that Docker and Puppet make a powerful one-two combination, you won't want to miss the next installment.■

---

**Todd A. Jacobs** is a frequent contributor to *Linux Journal*, a Stack Exchange enthusiast, and a practicing DevOps and Automation Architect with a special focus on automated security. He currently lives in Baltimore with his beautiful wife, toddler-aged son, and two geriatric but lovable dogs.

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

**RETURN TO CONTENTS**

# SUPERMICRO® MARKETPLACE

Powered by Silicon Mechanics



Broad  
Selection



Zero  
Defects



3-Year  
Warranty

Your Source for  
Supermicro Platform Technology

Configure Now



Talk to a Supermicro Expert!  
866.352.1173

# LOW POWER WIRELESS: ROUTING TO THE INTERNET

This article continues the series begun last month by getting two Raspberry Pis to communicate over a 6LoWPAN network.

It looks at how to make them talk to other IPv6 hosts on different network segments, necessary to get IoT data off the sensors and onto the internet.

JAN NEWMARCH

---

PREVIOUS



Feature:  
Provisioning Docker  
with Puppet

NEXT  
Doc Searls' EOF



In the first article in this series (in the November 2016 issue), I described how to configure two Raspberry Pis to talk using the low power wireless protocol 6LoWPAN over IEEE 802.15.4 with the OpenLabs wireless module. As an example, I showed Python code running a server on one RPi to deliver CPU temperature data to a client on another.

### What's Next?

But, that isn't the real point of using 6LoWPAN. You could have done the same thing using Zigbee, Bluetooth Low Energy, Z-Wave or some other low power networking system. The point of using 6LoWPAN is that it creates and sends IPv6 packets. This potentially brings it into the wider internet world where IP packets can be routed across multiple hosts without having to decode and re-code the packets.

The 6LoWPAN network generates IPv6 packets. The internet is very slowly moving across to IPv6, but much of it is still IPv4. If you have to deal with an IPv4 network, these are your choices:

- Decode the packets on the RPi and use the data in them to talk to IPv4 hosts thereafter. That's what you could have done after following along with the first article in this series—you had decoded a packet and then could manipulate it or send it on.
- Tunnel across the IPv4 network to another IPv6 network and carry on from there.
- Use NAT'ing techniques to convert IPv6 packets into IPv4 packets automatically. This isn't easy and is beyond the scope of this article. (If you are interested, look up NAT64.)

I'm going to assume you can continue to use IPv6, so the RPi is in an IPv6 6LoWPAN network on one side and an IPv6 Ethernet/Wi-Fi network on the other.

This article looks at how to get IPv6 packets from a 6LoWPAN network and route them into an IPv6 network (and of course, back the other way). As in the first article, if you're following along, you will use Raspberry Pis with the same OpenLabs modules. And just like in the first article, there are a number of problems to be resolved along the way.

The goal is to get suitable IPv6 addresses generated on all the 6LoWPAN devices and for one of these devices to act as a router (an “edge router”) between the 6LoWPAN network and some other IPv6 network. This article actually ends up really being about routing between IPv6 link local networks, joining them into a global IPv6 network.

### IPv6 Address Types

IPv6 has several different types of addresses, just like IPv4 does. Link local addresses are visible only on a single link, and you can't route them. They are like link local IPv4 addresses, and they're in the address range `fe80::/10`. There also are site local addresses in the range `fec0::/10`, but these are deprecated. Multicast addresses are in the `ff00::/8` range. The loopback address is `::1/128`. Every other address is a global address.

### Getting a Fixed Link Local Address

Whenever you reboot the OpenLabs module, it gives itself a new MAC address. This is used to generate the link local address, and later you will see this used as part of the process to generate a global address. For the gateway, you will need a fixed global address, or external clients won't know how to find it. So for the gateway, you should fix the MAC address to ensure that you get a “known” global address.

General IPv6 addresses also are difficult to read and remember—16 hexadecimal numbers. There are special simplification rules for addresses with zeros in them, so I will exploit those here so you get simple addresses (for this article only, of course). You will set simple addresses on the gateway (needed) and on the sensor (convenient).

The MAC address `02:0:0:0:0:1` generates the IPv6 host address `::1`, which is about as simple as you can get. Set that on the gateway with:

```
ip link set dev wpan0 address 02:0:0:0:0:1
```

to give IPv6 host part `::1`, and for convenience on the sensor with:

```
ip link set dev wpan0 address 02:0:0:0:0:2
```

to give IPv6 host part `::2`.

**LINK LOCAL PACKETS AREN'T ROUTABLE—THAT IS, YOU CAN'T SEND PACKETS FROM ONE LINK TO ANOTHER LINK. TO ROUTE PACKETS FROM ONE LINK TO ANOTHER, THEY MUST HAVE A UNIQUE GLOBAL OR UNIQUE LOCAL ADDRESS.**

## Getting a Routable IPv6 Address

When any host starts its networking, it is assigned an IPv6 link local address automatically, based on its MAC address, which you have just set on the gateway and sensor RPis. Routing tables also are set up on the local link, so hosts on the same link can talk to each other directly.

It's easy to tell which addresses are link local addresses; they start with the prefix `fe80::`. The `ifconfig` program on any Linux/UNIX box will show something like this:

```
inet6 addr: fe80::84f1:df50:eb27:97ff/64 Scope:Link
```

Because you've fixed the MAC address on the gateway, it's just:

```
inet6 addr: fe80::1/64 Scope:Link
```

Link local packets aren't routable—that is, you can't send packets from one link to another link. To route packets from one link to another, they must have a unique global or unique local address.

Unique global addresses will be given to you by your internet provider, or you can buy them from an organization like ARIN. Australia's internet providers are way behind and very few of them support IPv6. I don't want to buy one when I can't use it yet. But, unique local addresses are

good enough. I can route them across my private network for free, across all my network segments. I probably won't ever want to broadcast my temperature data across the whole internet anyway—at most I would process it on my own network or send it to a particular cloud service.

The <http://unique-local-ipv6.com> website generates random unique local /48 prefixes, such as:

```
fd28:e5e1:869::/48
```

That leaves you 80 bits (128–48) for any subnets you want to create and unique addresses within those subnets. So you can specify any 80 bits you want, or easier, any 20 hex digits. I'm going to cheat a bit and simplify this to prefix `fd28::/64` as a 64-bit prefix. Use this on the gateway explicitly by setting:

```
ip addr add fd28::1/64 dev lowpan0
```

## Packet Forwarding

The RPi you are going to use as the gateway must have two NICs. Well, this one does: the 6LoWPAN device and the Ethernet device. But just like any router in any UNIX system, it has to be configured for packet forwarding between the NICs. This is really easy: edit the `/etc/sysctl.conf` file and uncomment the line:

```
net.ipv6.conf.all.forwarding=1
```

Then reboot, and it's an IPv6 router.

## Router Advertisements

You now have one fixed routable address that will be used for external clients to talk to the gateway/router. You also have a fixed link local address for hosts on this local link to talk to the gateway. At present, you have only one other RPi in the network, but your 6LoWPAN network might consist of hundreds or even thousands of nodes, and they need to be configured too, ending up with routable addresses so that external clients can get and set information on the sensors/actuators. But, you

don't want to be assigning addresses manually to every one of them!

The answer is *stateless address autoconfiguration* using *router solicitation* and *router advertisements*. You have to set up and configure router advertising, but then it becomes a no-brainer. This is the IPv6 equivalent of DHCP.

A new IPv6 node attempting to join a network will send out a router solicitation message using IPv6 multicast on its link local network. A router then will generate a router advertisement, which it will send back using unicast, which will contain enough information for the new node to configure itself.

The information supplied in the router adverts has basically two components:

- The *link local* address of the router, so that the node can send it messages.
- A *prefix* to be used as the network component of a *routable* address, to be used by the node to generate a unique local routable address for the node.

That's why you need a fixed link local address for the router to be used in router adverts. This is in addition to the fixed routable address, so that external clients can talk to the 6LoWPAN side of the gateway.

### radvd: Router Advertisement Dæmon for IPv6

The Linux dæmon to act as a router advertisement dæmon for UNIX-like systems is `radvd`. The version in the RPi repositories is unfortunately out of date, so you need to get a current version from GitHub and build it:

```
git clone https://github.com/linux-wpan/radvd.git -b 6LoWPAN
cd radvd
./autogen.sh
./configure --prefix=/usr/local --sysconfdir=/etc
→--mandir=/usr/share/man
make
sudo make install
```

You may need to install `bison` from the repositories if it can't find `flex`.

Once built and installed, `radvd` uses the configuration file `/etc/radvd.conf` with the following contents:

```
interface lowpan0
{
    AdvSendAdvert on;
    # UnicastOnly on;
    AdvCurHopLimit 255;
    AdvSourceLLAddress on;

    prefix fd28::/64
    {
        # AdvOnLink off;
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
    };

    abro fe80::1
    {
        AdvVersionLow 10;
        AdvVersionHigh 2;
        AdvValidLifeTime 2;
    };
};
```

This is adapted from Sebastian Meiling's page "Setup native 6LoWPAN router using Raspbian and RADVD" (<https://github.com/RIOT-Makers/wpan-raspbian/wiki/Setup-native-6LoWPAN-router-using-Raspbian-and-RADVD>). The prefix is the random prefix I used earlier, `fd28::/64`. The `abro` ("Authoritative Border Router Option") is the link local address of the router. You will need to set your own addresses—at a minimum, the routable prefix.

I've made a couple changes to Sebastian's configuration: I've set `AdvOnLink` to `on`; whereas he has it as `off`. Setting the advert to `on` means:

- The router itself won't get an address. That's okay; you want it to have a fixed address not assigned by `radvd`.
- For each node that has its IPv6 address set by `radvd`, entries will be made in the routing table to route `fd28::/64` addresses through the 6LoWPAN device. Addresses with this prefix are "on this link".
- Most important, general addresses (`::/0`) will route using the `lowpan0` NIC through the link local gateway address `fe80::1` to the external world.

I've also removed the `UnicastOnly` on setting. The reasons are:

- Router adverts contain a timeout, defaulting to 30 minutes.
- Unless updated, hosts will remove the routing entry on expiration of the timeout.
- Hosts don't usually request new router adverts, only once on startup; they expect the router to multicast new adverts every few minutes.

The `UnicastOnly` on setting stops `radvd` from sending out these adverts, so you need to remove it to allow the routing tables on hosts to be renewed.

## Router Configuration

All the work on the 6LoWPAN side is now done. On the Ethernet side, I also want to have an IPv6 network, and as I am using unique local addresses, this network will be my private network, probably with many link segments. To change it to be internet-global, I would just need to change the unique local addresses to unique global addresses.

Initially I had problems routing IPv6 packets on my private network. My home router (a Linksys EA6900) didn't seem to want to route packets from my "external" host through to the gateway. I fixed that by using a cross-over cable directly from my "external" host to the

gateway, and then after all, the home router decided to cooperate. Then with radvd also delivering adverts on the Ethernet side to my “external” host, I could ping from the 6LoWPAN network to the Ethernet network and vice versa.

In summary, the steps to go through on the 6LoWPAN side are:

- Configure /etc/radvd.conf.
- Bring up the 6LoWPAN device.
- Set link local and routable addresses on the 6LoWPAN device.
- Start up radvd.

The radvd configuration file is described above. The startup script for the rest should be run as root and is:

```
#!/bin/bash

# set the MAC address
ip link set dev wpan0 address 02:0:0:0:0:0:1

iwpan dev wpan0 set pan_id 0xbeef
ip link add link wpan0 name lowpan0 type lowpan
ifconfig wpan0 up
ifconfig lowpan0 up

# set the gateway address on the 6LoWPAN side
ip addr add fd28::1/64 dev lowpan0

# start the router advert daemon
radvd -m stderr
```

On the Ethernet side, I also had configured /etc/radvd.conf to deliver adverts with the fd44:: prefix, but I didn’t get around to simplifying the Ethernet MAC addresses.

The resulting IPv6 addresses on the gateway are:

```
eth0      Link encap: ...
          inet6 addr: fd44::4adf:10a9:5c79:7954/64 Scope:Global
          inet6 addr: fe80::4adf:10a9:5c79:7954/64 Scope:Link

lowpan0   Link encap: ...
          inet6 addr: fd28::1/64 Scope:Global
          inet6 addr: fe80::1/64 Scope:Link
```

## Sensor Configuration

The RPi acting as sensor doesn't have to do much; `radvd` does most of it. The startup script is just:

```
#!/bin/bash

ip link set dev wpan0 address 02:0:0:0:0:0:0:2
iwpan dev wpan0 set pan_id 0xbeef
ip link add link wpan0 name lowpan0 type lowpan
ifconfig wpan0 up
ifconfig lowpan0 up
```

But, courtesy of `radvd`, the device now has an IPv6-routable address: `fd28::2`, as shown by `ifconfig`:

```
lowpan0   Link encap: ...
          inet6 addr: fd28::2/64 Scope:Global
          inet6 addr: fe80::2/64 Scope:Link
```

The routing table on the sensor RPi looks like:

```
$ route -A inet6
Kernel IPv6 routing table
Destination     Next Hop     Flag     Met Ref Use If
fd28::/64        ::         UAr     256  0      0 lowpan0
fe80::/64        ::         U       256  0      0 lowpan0
```

```
 ::/0          fe80::1      UGDAe 1024 0      0 lowpan0
ff00::/8      ::           U        256 1      18 lowpan0
```

As you can see, addresses with the prefix `fd28::/64` are on the link through the `lowpan0` device. The address `::/0` is the default route address, so all other packets are routed through the `lowpan0` NIC via the Next Hop address `fe80::1`.

## Testing Routing

You can test this from each RPi by pinging the other RPi. That just tests local routing though. To test this properly, you need to be able to talk through the Ethernet/Wi-Fi NIC on the RPi router to another IPv6 device.

I've got the RPi router talking to an "external" host through a crossover cable for simplicity, with `radvd` delivering router adverts to it.

So then from the desktop, I can ping my RPi sensor:

```
$ping6 fd28::2
PING fd28::2(fd28::2) 56 data bytes
64 bytes from fd28::2: icmp_seq=1 ttl=254 time=14.0 ms
64 bytes from fd28::2: icmp_seq=2 ttl=254 time=16.4 ms
64 bytes from fd28::2: icmp_seq=3 ttl=254 time=17.9 ms
```

If you get successful pings, you know it works.

With that in place, the server code from my previous article (in the November 2016 issue) can be modified to use routable addresses rather than link local addresses. This basically means you don't have to specify the "scope id" (the NIC) anymore:

```
#!/usr/bin/python3

import socket
from subprocess import PIPE, Popen

HOST = ''          # Symbolic name meaning all available interfaces
PORT = 2016        # Arbitrary non-privileged port
```

```
def get_cpu_temperature():
    process = Popen(['vcgencmd', 'measure_temp'], stdout=PIPE)
    output, _error = process.communicate()
    return output

def main():
    s6 = socket.socket(socket.AF_INET6, socket.SOCK_STREAM, 0)
    s6.bind((HOST, PORT, 0, 0))
    s6.listen(1)

    while True:
        conn, addr = s6.accept()
        conn.send(get_cpu_temperature())
        conn.close()

if __name__ == '__main__':
    main()
```

The client gets modified similarly, omitting the scope id:

```
#!/usr/bin/python3

import socket
import time

ADDR = 'fd28::2'
PORT = 2016

def main():
    while True:
        s6 = socket.socket(socket.AF_INET6, socket.SOCK_STREAM, 0)
        s6.connect((ADDR, PORT, 0, 0))
        data = s6.recv(1024)
        print(data.decode('utf-8'), end='')

        # get it again after 10 seconds
```

```
time.sleep(10)

if __name__ == '__main__':
    main()
```

The output from that on the client is:

```
temp=38.5'C
temp=38.5'C
temp=39.0'C
...
```

## Conclusion

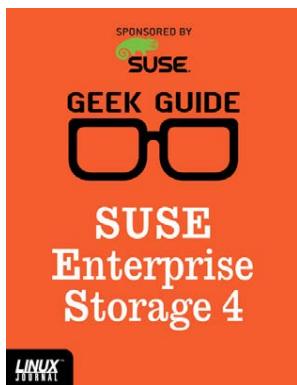
This article has shown that 6LoWPAN devices can communicate to other IPv6 systems on the routable internet. It has been mainly a journey about configuring IPv6 systems and setting up the IPv6 equivalent of DHCP. However, the story for low power wireless isn't over yet. The IoT at the application layer is standardizing on the CoAP and MQTT protocols, and in my next article, I'll take a look at the CoAP application protocol.■

---

**Jan Newmarch** has been using Linux since kernel 0.96. He has written many books and papers about software engineering, network programming, user interfaces and artificial intelligence, and he is currently digging into the IoT. He is in charge of ICT degrees at Box Hill Institute and Adjunct Professor at the University of Canberra.

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

**RETURN TO CONTENTS**



## GEEK GUIDE

# SUSE Enterprise Storage 4

By Ted Schmidt

### Introduction

I wrote a previous Geek Guide, titled *Ceph: Open-Source SDS*, that briefly introduced a Ceph-based, data storage management system called SUSE Enterprise Storage. Based on the response from readers of that ebook and given recent advancements in the maturity of SUSE Enterprise Storage (SES), it seemed logical to explore some of the features of SES more closely.

In this ebook, I review the characteristics of software-defined storage, along with its business benefits. Then, I explore the features of SUSE Enterprise Storage and how those capabilities can really help your organization leverage the benefits of open-source software-defined storage (SDS).

### A Quick Review of Software-Defined Storage

If you read the predecessor to this Geek Guide, you'll remember that software-defined storage is a technique for virtualizing storage capabilities on commodity hardware in an effort to reduce costs and improve efficiency. By separating storage management software capabilities from hardware, an enterprise can remove its dependency on proprietary

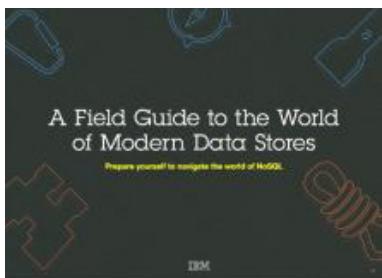
software and any associated limitations. It also frees the organization to leverage the lower cost model of commodity hardware. This ends up being a key benefit of any SDS solution: reducing capital expenditure (CAPEX).

More on the benefits later—first, let's do a quick review of the defining characteristics of an SDS solution. I've already mentioned that to be SDS (and this is a point on which both Gartner and IDC agree), a solution must be hardware-agnostic—that is, it must be able to use any brand of hardware. Otherwise, it just perpetuates the cost issue that comes from dependence on proprietary software.

In addition to hardware agnosticism, an SDS solution also has to provide the ability to establish policies for managing not only storage, but data services as well. It must provide tagging of metadata for both storage and data services, as well as disaggregation of storage and data services. A true SDS solution also will provide automated management of storage and a self-service graphical UI as well. As you'll see later, SUSE Enterprise Storage provides a GUI that is informative and intuitive.

To continue reading, download the complete eBook for FREE at <http://geekguide.linuxjournal.com>.

# FREE DOWNLOADS



## A Field Guide to the World of Modern Data Stores

There are many types of databases and data analysis tools to choose from when building your application. Should you use a relational database? How about a key-value store? Maybe a document database? Is a graph database the right fit? What about polyglot persistence and the need for advanced analytics?

If you feel a bit overwhelmed, don't worry. This guide lays out the various database options and analytic solutions available to meet your app's unique needs.

You'll see how data can move across databases and development languages, so you can work in your favorite environment without the friction and productivity loss of the past.

Sponsor: IBM

> <https://geekguide.linuxjournal.com/content/field-guide-world-modern-data-stores>



## Why NoSQL? Your database options in the new non-relational world

The continual increase in web, mobile and IoT applications, alongside emerging trends shifting online consumer behavior and new classes of data, is causing developers to reevaluate how their data is stored and managed. Today's applications require a database that is capable of providing a scalable, flexible solution to efficiently and safely manage the massive flow of data to and from a global user base.

Developers and IT alike are finding it difficult, and sometimes even impossible, to quickly incorporate all of this data into the relational model while dynamically scaling to maintain the performance levels users demand. This is causing many to look at NoSQL databases for the flexibility they offer, and is a big reason why the global NoSQL market is forecasted to nearly double and reach USD3.4 billion in 2020.

Sponsor: IBM

> <https://geekguide.linuxjournal.com/content/why-nosql-your-database-options-new-non-relational-world>



## RunKeeper Case Study

Boston-based fitness start-up RunKeeper was struggling with its database, and could not keep pace with the company's expansion. With new users joining every day, this limitation threatened to halt the company's operations. With a database of 30 million users and growing fast, scaling up also became an issue.

RunKeeper's initial database PostgreSQL failed to provide the required speed and scale. Partnering with IBM, RunKeeper transformed using IBM Cloudant's Dedicated Cluster as its new data layer.

"We were impressed by the wealth of experience that the IBM team was able to draw on to adapt the solution to meet our business needs," says Joe Bondi, CTO and Co-founder of RunKeeper.

Sponsor: IBM

> <https://geekguide.linuxjournal.com/content/run-keeper-case-study>

## FREE DOWNLOADS



### The 2016 State of DBaaS Report: How managed services are transforming database administration

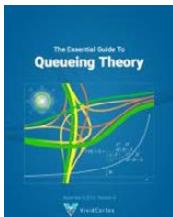
If you didn't have to manage your database, what would you do with your free time? All those hours you previously spent micromanaging your data layer—ensuring it keeps your application running 24/7 and is able to scale up or down based on demand—would suddenly reappear in your day. You could spend more time building your applications, from adding key features to improving the experience of your users, and you would even get some hours back in your personal life.

The 2016 State of DBaaS Report, commissioned by IBM, assessed the business and technical impact of database-as-a-service (DBaaS), as identified by 680 executive and technical enterprise users, and found that developers are saving a substantial amount of time after adopting DBaaS. All of those surveyed were using a managed, NoSQL database service across a variety of industries, including insurance, healthcare, gaming, retail and finance.

Sponsor: IBM

> <https://geekguide.linuxjournal.com/content/2016-state-dbaas-report-how-managed-services-are-transforming-database-administration>

---



### The Essential Guide To Queueing Theory

Whether you're an entrepreneur, engineer, or manager, learning about queueing theory is a great way to be more effective. Queueing theory is fundamental to getting good return on your efforts. That's because the results your systems and teams produce are heavily influenced by how much waiting takes place, and waiting is waste. Minimizing this waste is extremely important. It's one of the biggest levers you will find for improving the cost and performance of your teams and systems.

Author: Baron Schwartz

Sponsor: VividCortex

> <https://geekguide.linuxjournal.com/content/essential-guide-queueing-theory>

---



### Sampling a Stream of Events With a Probabilistic Sketch

Stream processing is a hot topic today. As modern Big Data processing systems have evolved, stream processing has become recognized as a first-class citizen in the toolbox. That's because when you take away the how of Big Data and look at the underlying goals and end results, deriving real-time insights from huge, high-velocity, high-variety streams of data is a fundamental, core use case. This explains the explosive popularity of systems such as Apache Kafka, Apache Spark, Apache Samza, Apache Storm, and Apache Apex—to name just a few!

Author: Baron Schwartz

Sponsor: VividCortex

> <https://geekguide.linuxjournal.com/content/sampling-stream-events-probabilistic-sketch>

# Progress on Privacy

There are now four ways we can protect our privacy online: encryption, agreements, fiduciaries and laws. Three of those are new.



**DOC SEARLS**

---

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

---

## PREVIOUS

- ◀ Feature: Low Power Wireless:  
Routing to the Internet
- 

The internet didn't come with privacy, any more than the planet did. But at least the planet had nature, which provided raw materials for the privacy technologies we call clothing and shelter. On the net, we use human nature to make our own raw materials. Those include code, protocols, standards, frameworks and best practices, such as those behind free and open-source software.

So far, our best privacy tech is encryption. But I won't dwell on that one, because I assume all *Linux Journal* readers are experts at that. Instead, I want to visit three others, all of which are new.

The first is agreements.

The most popular informal agreements in the physical world are called *secrets*. These aren't especially enforceable, but they are backed by *norms*,

which are powerful constraints operating in a social context. For example, we trust that people, other than the intended recipient, won't open a sealed envelope, even if they can. The seal (such as the one shown in Figure 1) signals secrecy and has been in use for hundreds of years.

More formal are the legal agreements we call *terms*. We encounter these every time we click "agree" to something that looks like what is shown in Figure 2.

Did you read that? Go back and try reading it again.

These are "contracts of adhesion", defined (by the *Legal Dictionary*) as "a standardized contract offered to consumers on a 'take it or leave it' basis without giving the consumer an opportunity to bargain for terms that are more favorable" (<http://legaldictionary.net/adhesion-contract>). After industry won the industrial revolution, large companies needed to create legal agreements for dealing with up to millions of customers. Contracts of adhesion were the only way. Alas, this also sidelined freedom of contract (<http://www.lawteacher.net/free-law-essays/contract-law/the-doctrine-of-freedom-of-contract.php>), "which allows parties to provide for the terms and conditions that will govern the relationship" (says LawTeacher.net).

But now we have the internet, a natural heterarchy (<http://www.linuxjournal.com/content/opening-minds-spheres-among-us>) defined by protocols that start by assuming that every entity on it is

Figure 1. Seal Signaling Secrecy



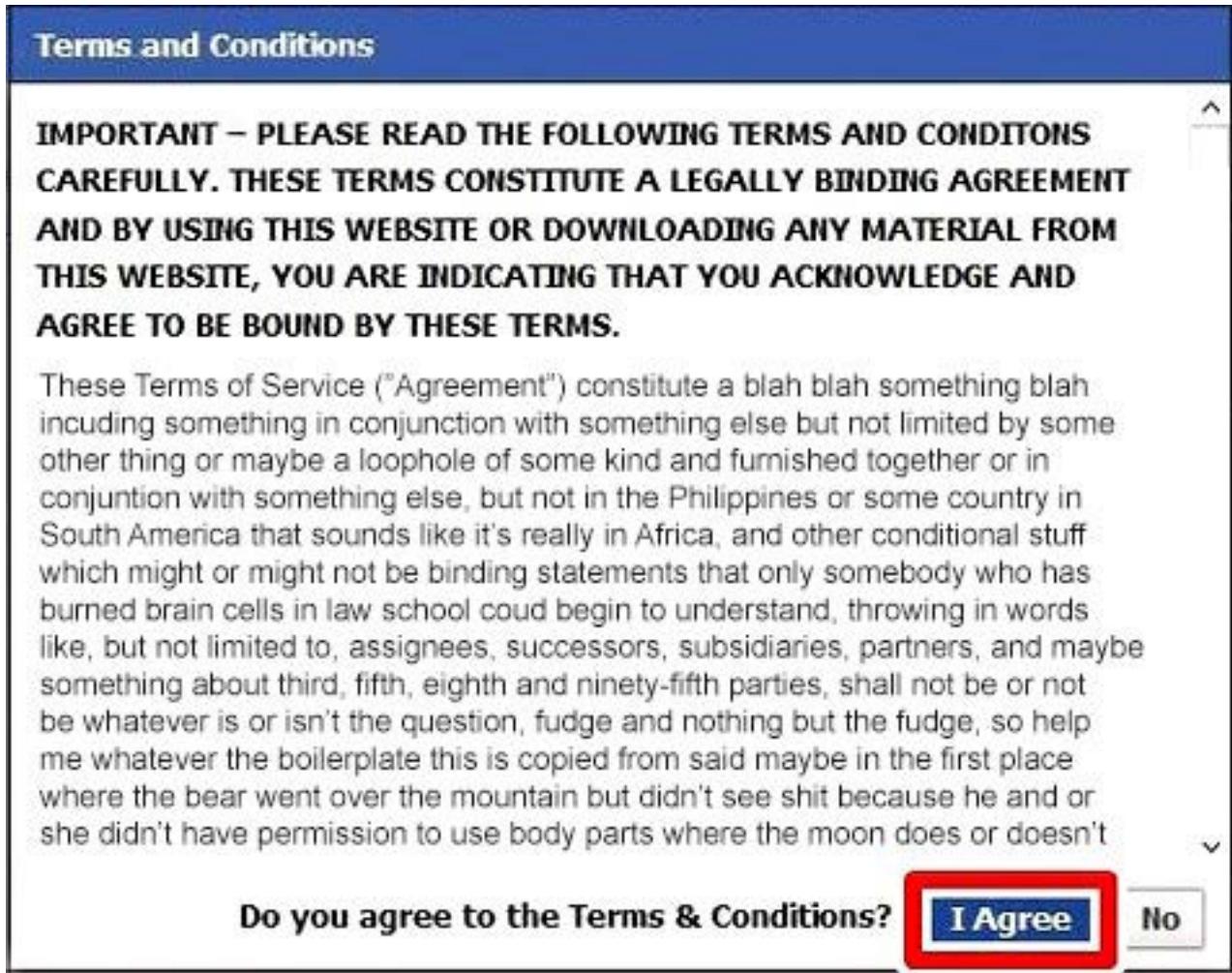
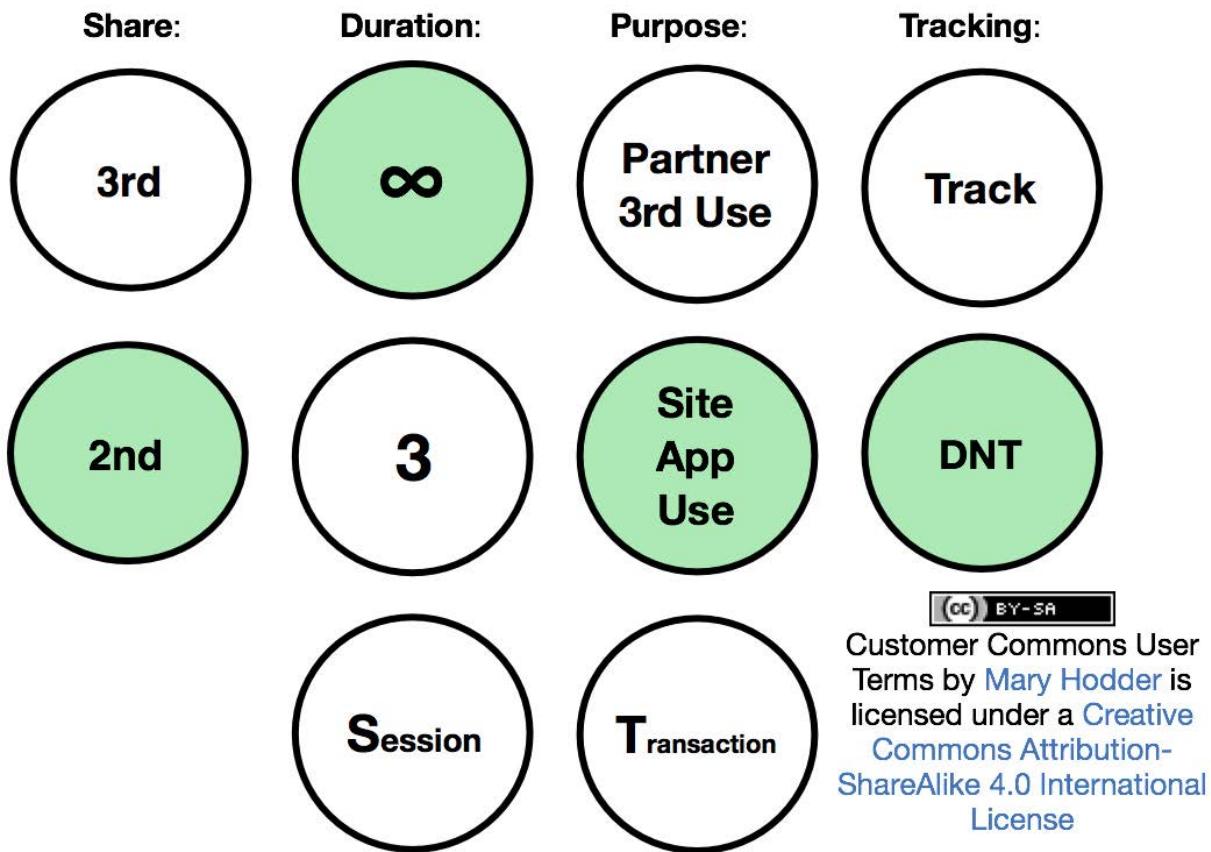


Figure 2. The Legal Agreements We Call *Terms*

both free to participate and a peer. In this world we can bring back freedom of contract by writing code that gives each of us ways to make and assert our own terms, and to apply leverage as well—in other words, to give *us* scale, as first parties. The second parties are companies we deal with, and which can agree to *our* terms. This isn't about turning the tables on companies, but rather setting a table that's flat, with both parties operating in a trusting way with each other.

To get this rolling, we now have Customer Commons (<http://customercommons.org>), which will do for personal terms what Creative Commons (<https://creativecommons.org>) does for copyright. Right now Customer Commons is co-baking standard

## MY TERMS: Icon format and structure



**NOTE: I'm the first party. My terms are: 2nd-∞-SU-DNT**

Figure 3. Customer Commons User Terms

terms with the Consent & Information Sharing Working Group at Kantara (<https://kantarainitiative.org/confluence/display/infosharing/User+Submitted+Terms+project+overview>). For example, see Figure 3 (<http://customercommons.org/2016/01/04/terms-what-are-they-and-why-should-you-care>).

In this example, the individual (as the first party) says personal data in a session is to be shared only with the second party (the website) for site use only, and to obey the Do Not Track request ([https://en.wikipedia.org/wiki/Do\\_Not\\_Track](https://en.wikipedia.org/wiki/Do_Not_Track)) expressed by the HTTP header in the individual's browser.

Another one is #NoStalking (<http://customercommons.org/2016/07/15/latest-draft-of-the-no-stalking-for-advertising-term-v-2>), shown in Figure

"Just give  
me ads  
not based  
on tracking  
me.."

Figure 4.  
#NoStalking

4, copied off a whiteboard last May at VRM Day (<https://www.flickr.com/photos/docsearls/sets/72157668230728355>).

While Do Not Track, tracking protection and ad blocking have prophylactic effects on privacy threats to individuals, #NoStalking works as a peace offering to publishers in the midst of the “war” over ad blocking (<http://blogs.harvard.edu/doc/the-adblock-war>). As I explain in “Why #NoStalking is a good deal for publishers” (<http://blogs.harvard.edu/vrm/2016/05/11/why-a-nostalking-deal-is-good-for-publishers>), “it’s a good one for both sides. Individuals proffering the #NoStalking term get guilt-free use of the goods they come to the publisher for, and the publisher gets to stay in business—and improve that business by running advertising that is actually valued by its recipients.”

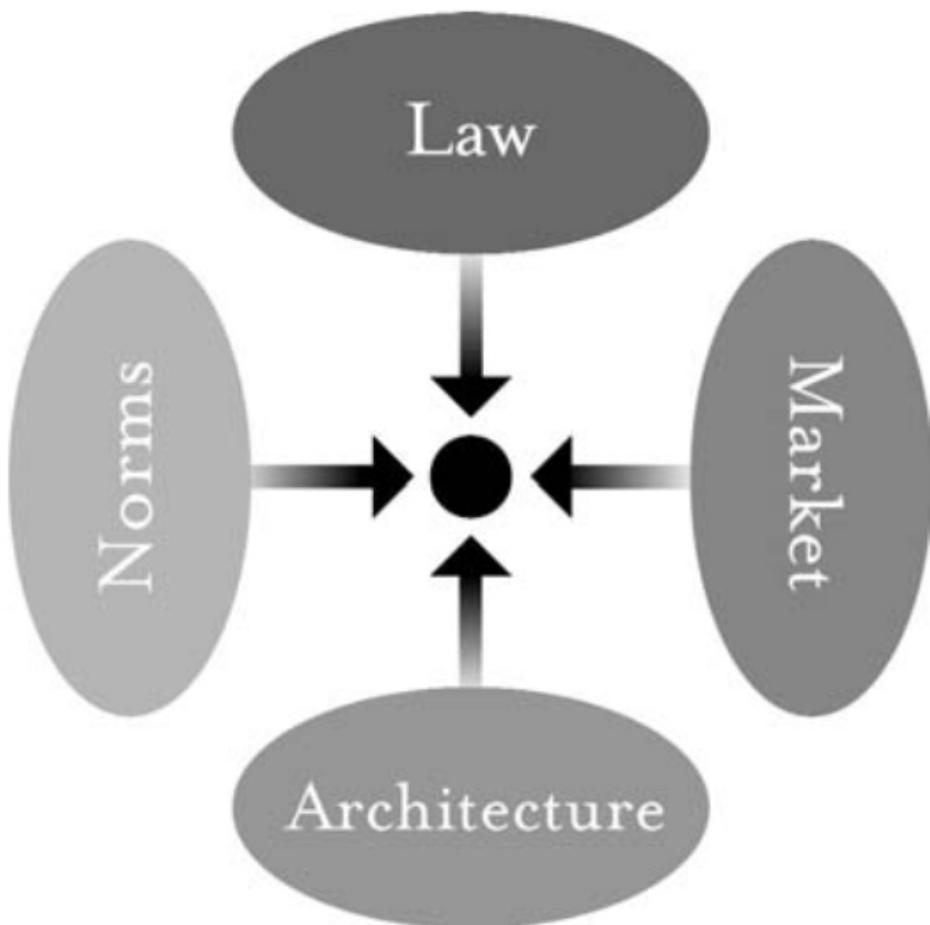
By valued I mean *not based* on tracking. As Don Marti (<http://zgp.org/~dmarti>) explains in “Targeted Advertising Considered Harmful” (<http://zgp.org/targeted-advertising-considered-harmful>), “The more targetable that an ad medium is, the less it’s worth.” That’s because non-targeted (that is, non-tracking-based) ads support the value of the publication they sponsor while also being supported by it. As Don puts it, non-targeted ads carry an economic *signal* ([https://en.wikipedia.org/wiki/Signalling\\_%28economics%29](https://en.wikipedia.org/wiki/Signalling_%28economics%29)), which “is proportional to the value of the content, not just the ad itself”. So, when your browser tells a

publisher you want #NoStalking from them, and the publisher agrees, you know that the ads you'll see are ones that value the content you came to the site for, rather than ones based on robotic surveillance of your life online, and likely to have little or nothing to do with the value of publication itself (serving, as it does, only as a sluice of convenience for advertising messages). It also says both the publication and the advertiser value your privacy.

The third privacy protection comes through *fiduciaries*. This is both an old and a new idea. In their book *Net Worth: Shaping Markets When Customers Make the Rules* (Harvard Business Review Press, 1999, <https://www.amazon.com/Net-Worth-Shaping-Markets-Customers/dp/0875848893>), John Hagel and Marc Singer coined the term infomediary, for "a trusted third party" or "a kind of agent" that will "become the custodians and brokers of customer information". In "A Grand Bargain to Make Tech Companies Trustworthy" (*The Atlantic*, October 3, 2016, <http://www.theatlantic.com/technology/archive/2016/10/information-fiduciary/502346>), law professors Jack Balkin of Yale (<https://www.law.yale.edu/jack-m-balkin>) and Jonathan Zittrain of Harvard (<http://hls.harvard.edu/faculty/directory/10992/Zittrain>) advance the concept of an *information fiduciary*: "a person or business that deals not in money but in information". Like doctors, lawyers and accountants, fiduciaries "have to keep our secrets and they can't use the information they collect about us against our interests". This gives companies like Facebook and Google a job they didn't know they took on when they began to gather mountains of personal information about us. "The important question is whether these businesses, like older fiduciaries, have legal obligations to be trustworthy. The answer is that they should."

This is a legal and rhetorical hack of the first water. Brilliant.

It also nicely frames up advances in regulation, which is the fourth form of privacy protection. In Australia and the European Union, personal data protection is already baked into laws imposing strong privacy protection obligations on those collecting personal data about us. Of special interest is the General Data Protection Regulation ([https://en.wikipedia.org/wiki/General\\_Data\\_Protection\\_Regulation](https://en.wikipedia.org/wiki/General_Data_Protection_Regulation)), aka the GDPR, in the EU. Search on Google for <https://www.google.com/search?q=General+Data+Protection+Regulation>, and you'll have to look down past a pile of advertising toward "compliance"



**Figure 5.**  
Lawrence Lessig's  
Diagram of the  
Individual as  
the Target of  
Regulation

offices at big companies before you get to the link I just used (to the GDPR's Wikipedia article). That's because the sanctions imposed on violators ([https://en.wikipedia.org/wiki/General\\_Data\\_Protection\\_Regulation#Sanctions](https://en.wikipedia.org/wiki/General_Data_Protection_Regulation#Sanctions)) include "a fine up to 20,000,000 EUR, or in the case of an undertaking, up to 4% of the total worldwide annual turnover of the preceding financial year, whichever is higher" (Article 83, Paragraph 5 & 6). That doesn't hit until 2018, but it's a big ouch in the meantime. But, with the looming threat of GDPR enforcement, new terms coming from the individual (another great hack) can offer genuine relief, even if lawmakers didn't see them coming. (Note that I avoid the term "user". That's because "user" positions the individual as the subordinate party, always "using" something provided by others. When the individual is the first party, sites and services such as those addressed by the GDPR are the actual users of personal data, and of terms to which they agree before using that data.)

On page 121 of *Free Culture* (Penguin Press, 2004, <http://www.free-culture.cc/freeculture.pdf>), Lawrence Lessig introduced a diagram that has since attained the status of canon (Figure 5).

Below it he explains, “At the center of this picture is a regulated dot: the individual or group that is the target of regulation, or the holder of a right....The ovals represent four ways in which the individual or group might be regulated—either constrained or, alternatively, enabled.”

We’re talking about enablement here, and the assertion of rights. So think of those arrows pointing outward from the individual, influencing all four of those domains.

So how do these four approaches to privacy protection match up with those domains? *Encryption* is pure architecture. Balkin and Zittrain’s *fiduciary* hack is on norms and law. *New privacy rules* such as the GDPR are already law. And *terms proffered by individuals*, in a freedom-of-contract way, are laws of their own, supported by architecture in the form of code, and influencing both norms and the market as well.

The result will be privacy that’s as casual and uncontroversial online as it is in the offline world. But first we have to finish scaling up terms and the code and protocols required to make them work. Those four domains aren’t going to fix themselves. ■

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

**RETURN TO CONTENTS**

## ADVERTISER INDEX

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

ADVERTISER	URL	PAGE #
Drupalize.me	<a href="http://drupalize.me">http://drupalize.me</a>	47
Peer 1 Hosting	<a href="http://go.peer1.com/linux">http://go.peer1.com/linux</a>	13
Silicon Mechanics	<a href="http://www.siliconmechanics.com">http://www.siliconmechanics.com</a>	81
SUSE	<a href="http://suse.com/storage">http://suse.com/storage</a>	7

### ATTENTION ADVERTISERS

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