LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

TIPS FOR USING THE HIGH-SECURITY **Qubes Desktop**

NOVEMBER 2017 | ISSUE 283
http://www.linuxjournal.com

Control a Heterogeneous Server Farm with SSH Agent

A LOOK AT ANSIBLE'S ROLES FEATURE

+

Launching External Processes in Python

Produce Readable Shell Scripts and Solve Equations

Deploy Instant Clusters in the Cloud

WATCH: ISSUE OVERVIEW

Deploying Kubernetes with Security and Compliance in Mind

Author:
Petros Koutoupis

Sponsor:
Twistlock

An Architect's Guide: Linux in the Age of Containers

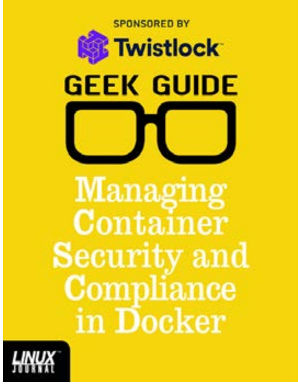
Author:
Sol Lederman

Sponsor:
SUSE

SQL Server on Linux

Author:
Reuven M. Lerner

Sponsor:
SUSE



Managing Container Security and Compliance in Docker

Author:
Petros Koutoupis

Sponsor:
Twistlock

Harnessing the Power of the Cloud with SUSE

Author:
Petros Koutoupis

Sponsor:
SUSE

DevOps for the Rest of Us

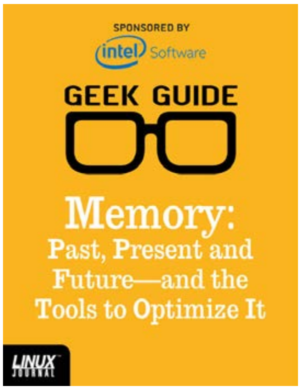
Author:
John S. Tonello

Sponsor:
Puppet

An Architect's Guide: Linux for Enterprise IT

Author:
Sol Lederman

Sponsor:
SUSE


SPONSORED BY
intel Software
GEEK GUIDE
Memory: Past, Present and Future—and the Tools to Optimize It
LINUX JOURNAL

Memory: Past, Present and Future—and the Tools to Optimize It

Author:
Petros Koutoupis

Sponsor:
Intel

CONTENTS

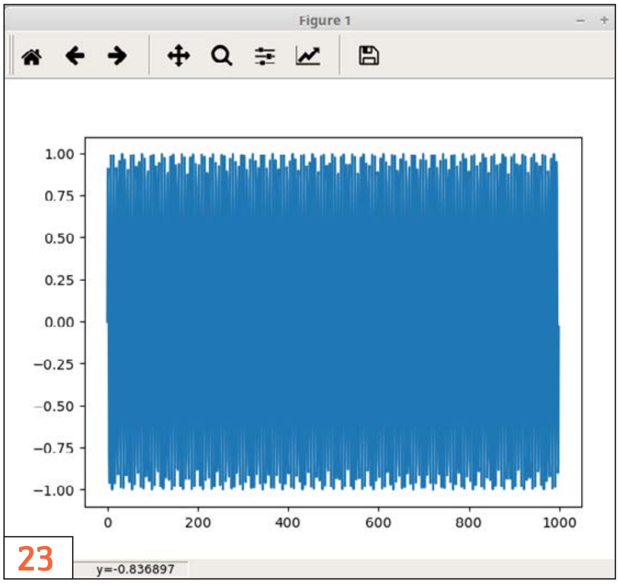
NOVEMBER 2017
ISSUE 283

FEATURES

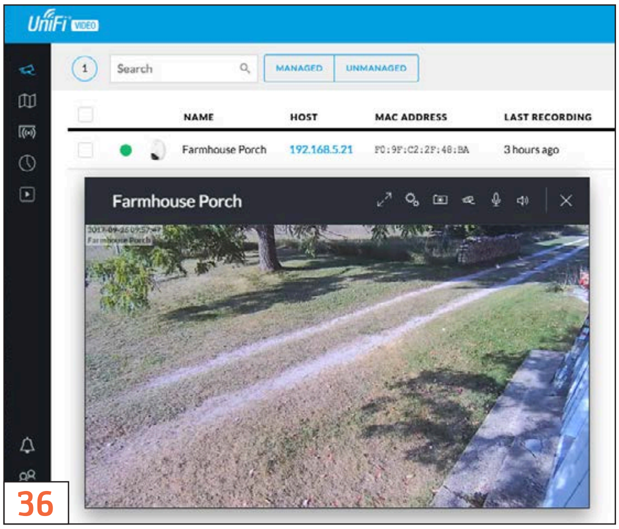
Cover Image: © Can Stock Photo / scanrail

COLUMNS

IN EVERY ISSUE



23



NAME	HOST	MAC ADDRESS	LAST RECORDING
Farmhouse Porch	192.168.5.21	F0:9F:C2:2F:48:BA	3 hours ago

36

ON THE COVER

- Control a Heterogeneous Server Farm with SSH Agent, p. 78
- Deploy Instant Clusters in the Cloud, p. 100
- Tips for Using the High-Security Qubes Desktop, p. 54
- A Look at Ansible's Roles Feature, p. 58
- Launching External Processes in Python, p. 40
- Produce Readable Shell Scripts and Solve Equations, p. 48

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., PO Box 980985, Houston, TX 77098 USA.
Subscription rate is $29.50/year. Subscriptions start with the next issue.

LINUX JOURNAL

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

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

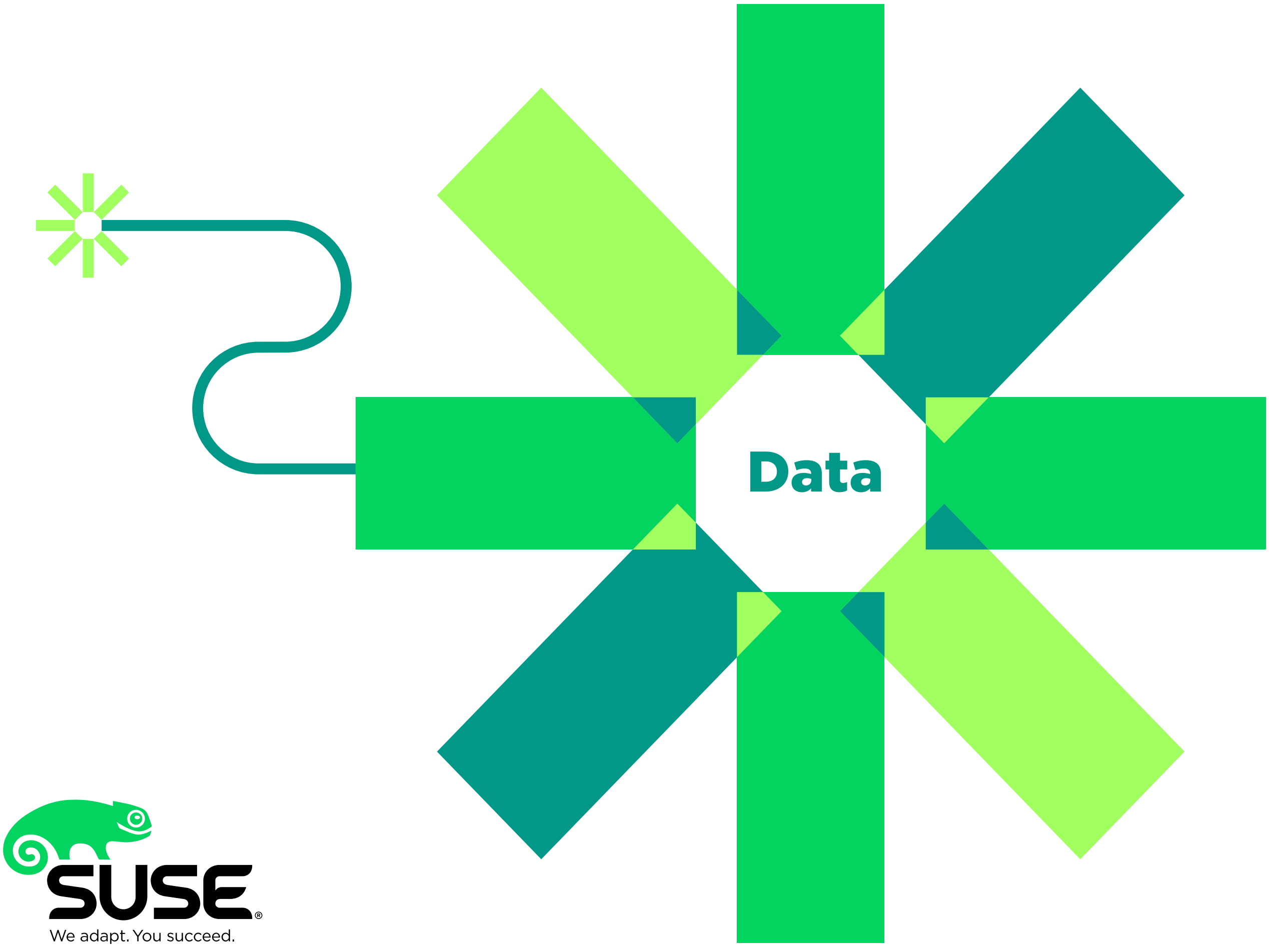
President	Carlie Fairchild publisher@linuxjournal.com
Publisher	Mark Irgang mark@linuxjournal.com
Associate Publisher	John Grogan john@linuxjournal.com
Director of Digital Experience	Katherine Druckman webmistress@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

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

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
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

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


Data

Arrogance, the Biggest Linux Security Problem

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. Or, swing by the #linuxjournal IRC channel on Freenode.net.

Linux is no longer an obscure platform avoided by those with malicious intent. It used to be that people with Windows 95 were the only ones getting viruses or experiencing security vulnerabilities, but that's before Linux migrated to the cloud. Now, basically everything runs on Linux, both inside and outside the office data center. That means network and OS security is more important than ever before, and now, Linux users need to be especially aware! The good news is, most Linux users know security is important and realize it's a topic that needs to be approached proactively. So this month, we look at some great ways to make our world a little more secure.

Our feature article is by Charles Fisher, and he explains how to use simple, but secure methods for maintaining multiple machines on your network using strong SSH keys and Parallel Distributed Shell. Although configuration management and system orchestration tools are powerful, sometimes it's important to strip back all the extraneous functionality and just execute remote commands on multiple computers over a secure connection. Charles describes how to configure your clients and perform tasks securely over the network.

VIDEO: Shawn Powers runs through the latest issue.

I previously mentioned the cloud and how Linux is a vital part in almost all cloud-based computing. In light of that, Nathan R. Vance and William F. Polik discuss how to go a step further and not only use cloud computer instances, but also to create an entire cluster of computers in the cloud. When scalability is instant, it means your cluster can grow and shrink as you need it, saving tons of money in hardware investment and resource management. Plus, the authors include information on a hybrid cluster, where the head node is on local hardware and the compute instances are spun up in the cloud only as needed.

Kyle Rankin wrote a Lightning Hacks article this month to provide a handful of really great ideas in a really short amount of time. This time, Kyle's focus is on Qubes tips and how he does some pretty nifty things with it on his system. As with most of Kyle's work, these tips can be adapted if you don't happen to be a Qubes user, but regardless of the system you use, it's always worthwhile to read Kyle's column!

I finish off my series on Ansible this month, which is another tool you may or may not be using. Hopefully after following along on this four-part series, you've at least given Ansible a try in your own network. Even the smallest implementation can save hours of work, and the time required to get started is minimal.

We also have great information for coders and developing developers. Reuven M. Lerner continues teaching about Python this month, with particular focus on launching external processes from inside a Python script. Some tools in the Linux shell are invaluable, and often it's nice to utilize them inside the script of another language, like Python.

Dave Taylor follows Reuven with his shell scripting column, which teaches all about those convenient shell tools! This month, he explores using mathematical evaluation tools in order to make a guessing game. Along the way, you'll learn to make clear, readable code so your guessing game can be the foundation for more complicated and usable code. As usual, Dave teaches valuable skills while readers get to play command-line games.

We also have new products, reviews, tech tips and all the other goodies you've come to expect month after month from *Linux Journal*. So whether you're trying to secure your existing infrastructure or just starting to build that infrastructure and want to do it in a wise and secure fashion, this issue is here to serve. Enjoy, and we'll see you again next month! ■

RETURN TO CONTENTS

◀ PREVIOUS
Current_Issue.tar.gz

NEXT ▶
UpFront

Cheap Android Device for Testing Apps

Some time ago Shawn Powers mentioned having found some device that replaced the Galaxy Player 4. I haven't found the article (again) yet, but I would like to find a cheap Android device to use in playing with Android apps. Can you give me a pointer to the issue where you discussed that, or a pointer to what is on the market that might work?

—Bob Rader

Shawn Powers replies: *The article was in issue 251, March 2015. I haven't looked at it in a while, and although conceptually, I'm sure it's still valid, the phones listed are way outdated. I'm currently using a Sony Xperia Z5 Compact, which is also a bit outdated, but it still works fast enough for playing audio and such. I actually had Cricket Wireless service on the phone for a few months, but I let that expire and just use it with Wi-Fi now. I'm extremely happy with the device after rooting it. I went with the Z5 because it was small, but if size isn't an issue, any Android phone from a generation or two back would be more than adequate for such things. Just do research in advance to make sure it's easy to root, because that is essential for most nerdy endeavors!*

BYOC Part III, Replacing btools with ClusterShell

In the very interesting articles about building your own cluster (see the "BYOC" series by Nathan R. Vance, Michael L. Poublon and William F. Polik in the May, June and July 2017 issues), the authors described writing some tools to reach all nodes from the cluster, the Btools scripts. I would suggest giving a try to ClusterShell, which is a Python tool and library to connect, through ssh, a big set of nodes in parallel. It also provides remote copy and so on:

```
bexec <command> -> clush -a <command>

bpush <file> <destfile>-> clush -a --copy <file> [--dest <destfile>]
```

See http://cea-hpc.github.io/clustershell.

—Aurélien Cedeyn

Nathan Vance, Michael Poublon and William Polik reply: *Thanks for your interest in cluster computing. There are excellent packages for administering nodes on a production cluster, such as ClusterShell (clush) or Cluster Command & Control (C3). The purpose of btools is to demonstrate a minimal, self-contained set of tools for cluster management. Any Linux users can then understand what is happening under the hood when they "Build Your Own Cluster (BYOC)"!*

A General Request

I wrote about Linux in the past (a long time ago). I would just strongly encourage the Linux community to try to debug old problems rather than make new software and jump on new distributions. It has been an observation of mine that although Linux does work on almost anything, the turnover is too fast for most humans to keep up with.

—Sujan Swearingen

Shawn Powers replies: *I understand and largely agree with your sentiment. As someone who has "re-invented the wheel" a few times, however, I can attest that sometimes figuring out someone else's work takes more time than starting from*

LINUX JOURNAL

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 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://ww.linuxjournal.com/advertising**. Contact us directly for further information: ads@linuxjournal.com or +1 713-344-1956 ext. 2.

scratch. That's not necessarily the fault of the original developer. As busy geeks, we tend to take the avenue of least resistance, and unfortunately, that often means leaving a trail of destruction and abandonment in our wake. Yours is a good reminder though that open source was built on the shoulders of those who came before. If we can build on the success of someone else, it will make an even better product in the end.

Purism

I'd like to inform you about this project: https://puri.sm/shop/librem-5.

I value it as a awesome project for Linux, Linux users and freedom. I'd like to see an article on it, and I really hope its crowdfunding will have success. Thanks for reading this.

—John

Shawn Powers replies: *I'm pretty sure that* LJ *columnist Kyle Rankin is chairman of the advisory board for that company. I know Kyle uses Purism laptops, and he has at least mentioned them from time to time in articles (search for "Purism" on* http://www.linuxjournal.com *to find them). Nevertheless, you're correct; Purism is awesome and deserves attention!*

Comments on Charles Fisher's inotify Article

Having needed to solve a similar problem in the past, I was very interested to read Charles Fisher's article ("Linux Filesystem Events with inotify") in the August 2017 issue. However, I noticed one problem with it that needs clarification. Twice in the text he states that inotify can't monitor remote, network-mounted filesystems, such as NFS. This is not entirely true. inotify can report on filesystem changes on an NFS-mounted filesystem, but only for the file activity on the system on which those changes are being made, just like a local filesystem. It will not report on remotely generated events (for example, a file being manipulated on the NFS share by another server).

—Todd Campbell

Charles Fisher responds: *Assuming that the remote NFS server is* *running Linux, this seems reasonable (although I have not tested it). There have been several Linux NFS implementations, starting with the user-space server in the 1990s and proceeding through all the protocol versions as they were introduced and implemented in the kernel. I'd be confident that inotify would function properly in most of them, but not certain.*

Obviously, any other file sharing protocol will have even bigger problems (for example, SMB), and if the remote server is not Linux, inotify is not a workable tool. I might also point out that one of my mentions of this was a direct quote from a systemd manual page, and it might be a hard sell to get that changed.

Banana Backup Article

I much enjoyed Kyle Rankin's "Banana Backups" article on using Banana Pi and a 2.5" drive for backups in the September 2017 issue. It's a really good idea. Another suggestion for the software is storeBackup. I have been using it on both my personal Bodhi Linux laptop where I do my daily work as well as on the little home server I run on an old Toshiba Satellite laptop.

The little server runs ownCloud, SubSonic, caliber-server, MySQL and a few other smaller services. ownCloud has a fair amount of data, including all my contacts, calendar, legal documents photos and music. So it's a lot of stuff that I don't want to lose.

storeBackup is a Perl application that is very simple to set up with a single configuration file, great documentation (downloadable or online), and it's simple to install. It has been a rock-solid performer for me. It uses hard links and compression to reduce disk storage size.

I won't say it is better than any other backup software, but if one likes Linux and Perl and open source, it is a solid solution in my experience and worth mentioning.

Thanks for an excellent article in an always excellent magazine!

—Wes Wieland

It's Time for a Linux Live CD That Works for Netflix

I just spent the past month watching Amazon Prime Video using a Linux Mint 18.1 Live CD (no HDD exposed to the internet).

I only had to install npapi to get around the DRM requirement. This setup worked for three of the four weeks during the one-month free trial, but then it stopped working for some unknown reason.

Perhaps, it's time for a Linux version of its Live CD that is specifically configured to work for Netflix or Amazon Prime Video without any configuration—it would be very helpful to us users.

Is there an article here for discussion?

—Stephen

Shawn Powers replies: *DRM is the bane of our open-source hearts, isn't it? I get so frustrated trying to make such things work on my system, that I have resorted to using apps and/or devices like Roku in order to watch streaming media. Well, honestly, I usually use Plex to stream media, but that requires an entire DVR system, which sort of takes away the whole point of Netflix, Hulu, Amazon Video and so on. I don't have a great answer, other than perhaps resort to a tablet with an Android app, as those are almost always going to be updated and will work well. I wish I was more help!*

Linux Live CD with VPN Included

Greetings, I surf the internet with a Mint CD without using a hard drive. I now require the use of a VPN.

What are my options? (Also there doesn't appear to be *any* company selling a VPN for this scenario.)

When the above was posted on a Mint help chat session, these were the responses:

- Time to learn "iso" making.

- Mint suggests a VPN service on its web page.

- Persistence install may help.

- In principle, it is possible to create/modify the cd image to include this, but it takes some effort; alternatively, you can create a USB Flash drive version with some persistent memory and install OpenVPN on that.

Now, I didn't really want to use a different Linux distribution just to get the included VPN application.

Also, my concern with using a USB Flash drive is that it could be easily hacked, since it is not read-only, correct?

Anyway, this could make for a good article, right?

I look forward to hearing back from you with some good insight on a solution to my dilemma!
—Stephen

Shawn Powers replies: *Stephen, like your Netflix question, I don't have a perfect answer. Rather than a live CD or a USB drive, perhaps a distribution like Qubes would work for you? I realize that's switching distributions, which you wanted to avoid, but if security is a concern, it might be worth considering. Apart from that, I think the USB drive with persistence is the best option.*

Firewall Help?

I am looking for a gigabit firewall to put between my ISP provider's modem and my internal network. Although the ISP modem contains a firewall, it is not so secure as others or as a custom firewall.

Maybe a project for later with these goals:

- Maintaining the throughput of 1 gigabit.

- Allowing video signals from IP cameras to pass.
- Dynamically blocking intrusion on different ports.
- Blocking defective UDP and TCP packets, and IPs from outside with inside IPs.
- Blocking traffic if not generated from inside network.
- Allowing VPN network and server.

I was looking for commercial firewalls with a reasonable price, but I found only licensed firewalls that did not fulfill my expectations for a private firewall at home.
—Patrick Op de Beeck

Shawn Powers replies: *Although certainly possible with Linux, I have to admit for instances like this, I usually go to pfSense. It's BSD-based, but it has a very powerful interface and an even more powerful system beneath. It's open source and free to install anywhere. The company does sell hardware through a partner, but the firewall can be installed on any system. If you do something on your own, maybe pitch the idea to us at ljeditor@linuxjournal.com; perhaps your experience will make a good article!*

WRITE *LJ* A LETTER

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

PHOTOS

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

RETURN TO CONTENTS

KEEP
CALM
SHAREPOINT
ON

PREVIOUS
Letters

NEXT
Editors' Choice

diff -u

What's New in Kernel Development

Salvatore Mesoraca recently posted a patch to make it harder for hostile users to trick regular users into putting sensitive data into files and pipes controlled by the attacker. The problem was that attackers could create a file, or a FIFO (first in/first out) pipe, that had a filename expected by one of the regular pieces of software on the system. Regular users then innocently would use the regular piece of software, thinking they safely could input a password or whatever, but instead of creating the safe and private file on the filesystem, the regular software mistakenly would open the attacker's file or FIFO instead and send the sensitive data right into the attacker's hands.

The solution, as Salvatore saw it, was to tighten the system's controls over directory permissions. Salvatore's patch would affect directories that had been set world-writeable, with the sticky bit set, such that the regular user, and the regular software, would not have permission to edit the hostile files and FIFOs created by the attacker. Instead, the software simply would fail to create the file it needed. This might result in users being unable to use the software until the hostile file had been identified and removed, but at least the attack would have been foiled.

Kees Cook was thrilled to see this particular security hole plugged. He had a couple coding suggestions and also suggested documenting an example attack that Salvatore's patch would stop.

But, **Alexander Peslyak** replied, "I doubt there are (m)any examples of attacks and blog posts on this, because most systems didn't have similar (sym)link restrictions until recently and those attacks are simpler." He pointed out that symbolic links also were susceptible to this sort of attack, and he advocated making /tmp, /dev/shm and other potential target directories entirely unwriteable except via known library interfaces.

By way of advocating for Salvatore's patch, Alexander also added, "policy enforcement like this implemented in a kernel module helped me find weaknesses in an old Postfix privsep implementation, which were promptly patched (that was many years ago). Having this generally available and easy to enable could result in more findings like this by more people."

One of the things that's good about Linux is that security fixes are treated like holy Silmarils. They even supersede **ABI** preservation in the importance given them by **Linus Torvalds** and others. Now that Linux systems are being targeted more seriously by government hackers around the world, it's all the more important to fix problems as they appear and to make no exceptions. I personally find it mind-boggling that anyone is still advocating putting "official" back doors into security software, and that **Microsoft** still relies so much on anti-virus software rather than closing the holes that allow attackers to get in in the first place.

We don't usually get to see Linus taking the lead on a particular kernel feature or fix. Usually he leaves that to the other contributors, while he gives the final thumbs-up on whatever makes it through the legions of reviewers and testers. The most notable time when Linus really led a full project—aside from the kernel itself—was when he created **Git** entirely from scratch, after **BitKeeper** failed, and no other revision control system was able to meet his requirements.

But recently, Linus did take the reins on a lesser project. **Masahiro**

Yamada raised the question of whether it would be okay to have the kernel build system require third-party tools like **Flex** and **Bison**. The current situation was to include files directly in the source tree that already had been processed by those tools. That way, Linux could avoid messy problems like versioning conflicts in the toolchain. But Masahiro said that if it were possible to overcome those problems, it would be useful to rely on real source files in the kernel source tree, rather than these processed untouchable blobs of generated C code.

Specifically, Masahiro had noticed that **kbuild** recently had added rules for regenerating those files, so it already was possible to do. But, he asked if it was therefore acceptable to do this all the time, or only under certain key circumstances. And in fact, he asked if it was possible to do away with the processed files altogether and simply have the build system regenerate them as part of its default behavior. As he put it, "the advantage is we do not need to version-control generated files, i.e. shipped files will be deleted."

Linus took a look at the situation, and said, "Yeah, I think we probably should do that."

He did a test run and found that one of the files would be regenerated improperly because **gperf** changed its behavior in version 3.1. He said, "I'm not sure how to detect that automatically....gperf doesn't seem to generate any version markers."

He worked around that particular problem by hand, only to run into a similar problem elsewhere.

Linus said, "one of the advantages of the pre-shipped files is that we can avoid those kind of crazy version issues with the tools. But if we can solve the versioning thing easily, I certainly don't mind getting rid of the pre-generated files. Having to have flex/bison/gperf isn't a huge onus on the kernel build system."

However, he also added the caveat, "the traditional way to handle this is autoconf, etc. Since I think autoconf is evil crap, I refuse to have anything what-so-ever to do with it."

He also felt that gperf was seriously misdesigned, because "it would have been trivial for them to add some kind of marker define so that you could test for this directly rather than depend on some kind of autoconf 'try to build and see if it fails' crap."

His solution was simply to ditch gperf altogether and make something else that could do the same work. And he added, "I assume that flex/bison are stable enough that we don't have the same kind of annoying stupid version issues with it."

He posted a patch to get rid of the gperf dependency in the build system and included many warnings that the code might be completely broken, but just happened to work for him. He added, "Honestly, the code is better and more legible without gperf."

A couple weeks later, Masahiro noticed that Linus' patch had made it into the kernel's git tree (surprise!), and he tested it out. He reported that with `CONFIG_MODVERSIONS` enabled, he saw a lot of error messages.

Linus looked it over, but wasn't able to see the problem. He hadn't used Modversions when he'd tested his patch, but said he'd take a look and see if he could identify the problem.

In fact, after doing a `git clean -dqfx`, he noticed a ton of warnings that had slipped past his normal build test, but that clearly were visible from a pristine build that regenerated various versioning data.

Digging deeper, he was able to identify a one-line change in a generated file that should not have happened. Apparently, it occurred throughout the rest of the kernel build as well. He remarked:

> What is special about that one particular function vs. the other ones in that file? I have absolutely no idea. So the really odd thing here is how things clearly still *work*. The parser works fine for everything else. And looking at the gprof-removal patch it's not at all obvious how everything could work fine except for some random thing. Strange.

But after a little more digging, he reported, "Found it. Stupid special case for 'typeof()' that used is_reserved_word() in ways I hadn't realized."

He put a fix into the Git tree.

So apparently, the kernel build system now will depend on tools like Flex and Bison, and will regenerate its C files from those sources at build time, rather than storing the regenerated files in the source tree itself at development time.

It's hard to tell when Linus will dive into a particular problem and work it out himself. I suspect he just found this to be a curious diversion that happened to interest him at the time. In the case of Git development, he resisted working on it for years, allowing kernel development to depend on a non-free piece of software, tolerating many flame wars along the way, allowing many competing projects to vie for his attention, and ultimately choosing none of them. Although in that instance, when he finally did decide to write Git, he actually put the entire kernel project on hold for several weeks while he devoted himself entirely to the new project.—Zack Brown

THEY SAID IT

You must lose a fly to catch a trout.
—George Herbert

I am looking for a lot of men who have an infinite capacity to not know what can't be done.
—Henry Ford

Success isn't permanent, and failure isn't fatal.
—Mike Ditka

The only people who can change the world are people who want to. And not everybody does.
—Huge Macleod

There are some things you learn best in calm, and some in storm.
—Willa Cather

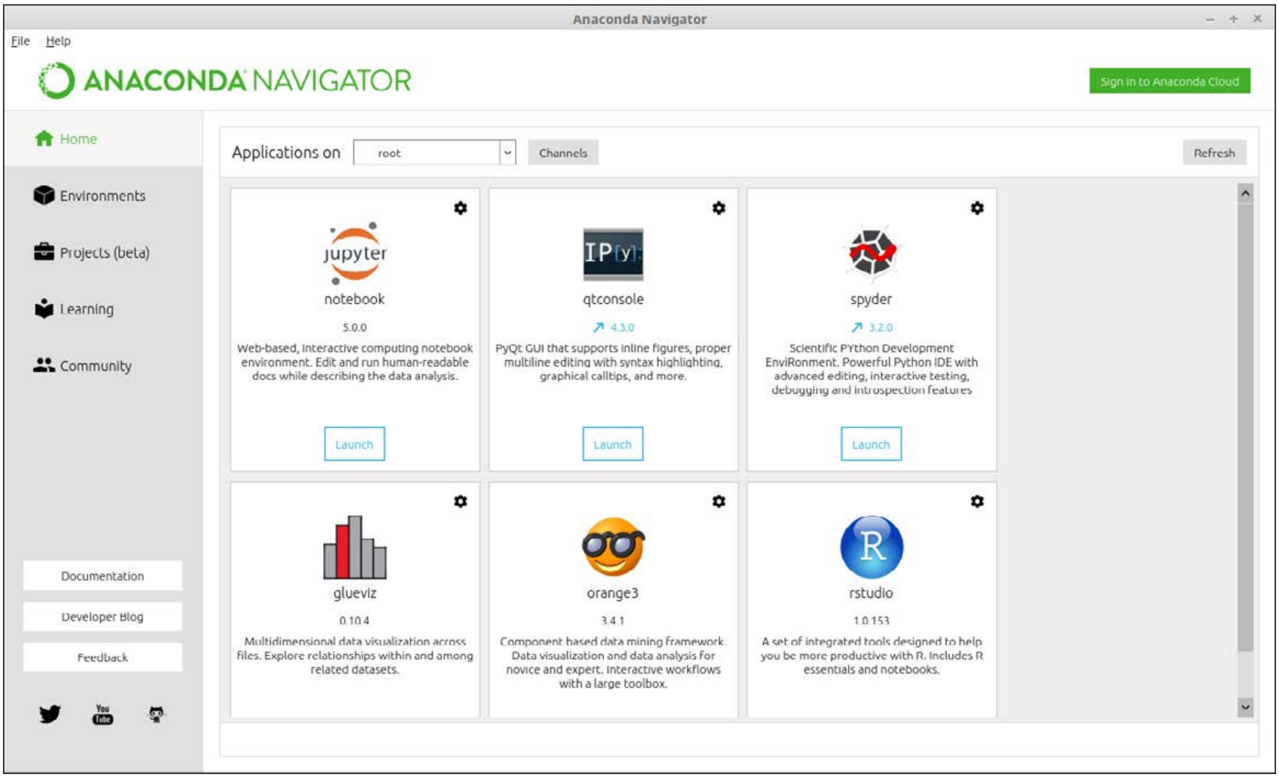
Spyders for Science

If you want to do science with Anaconda, one of the first things to consider is the spyder package, which is included in the basic Anaconda installation. Spyder is short for Scientific PYthon Development EnviRonment. Think of it as an IDE for scientific programming within Python.

You probably will want to have the latest version available, because it's under fairly constant development. You can be sure your entire Anaconda installation is up to date with the command:

```
conda update anaconda
```

There are two ways to launch spyder. If you're using the Anaconda Navigator, you simply can click the spyder icon.



notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole
4.3.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

spyder
3.2.0
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

glueviz
0.10.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

rstudio
1.0.153
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 1. The Anaconda Navigator provides a graphical interface for interacting with your installation of Anaconda.

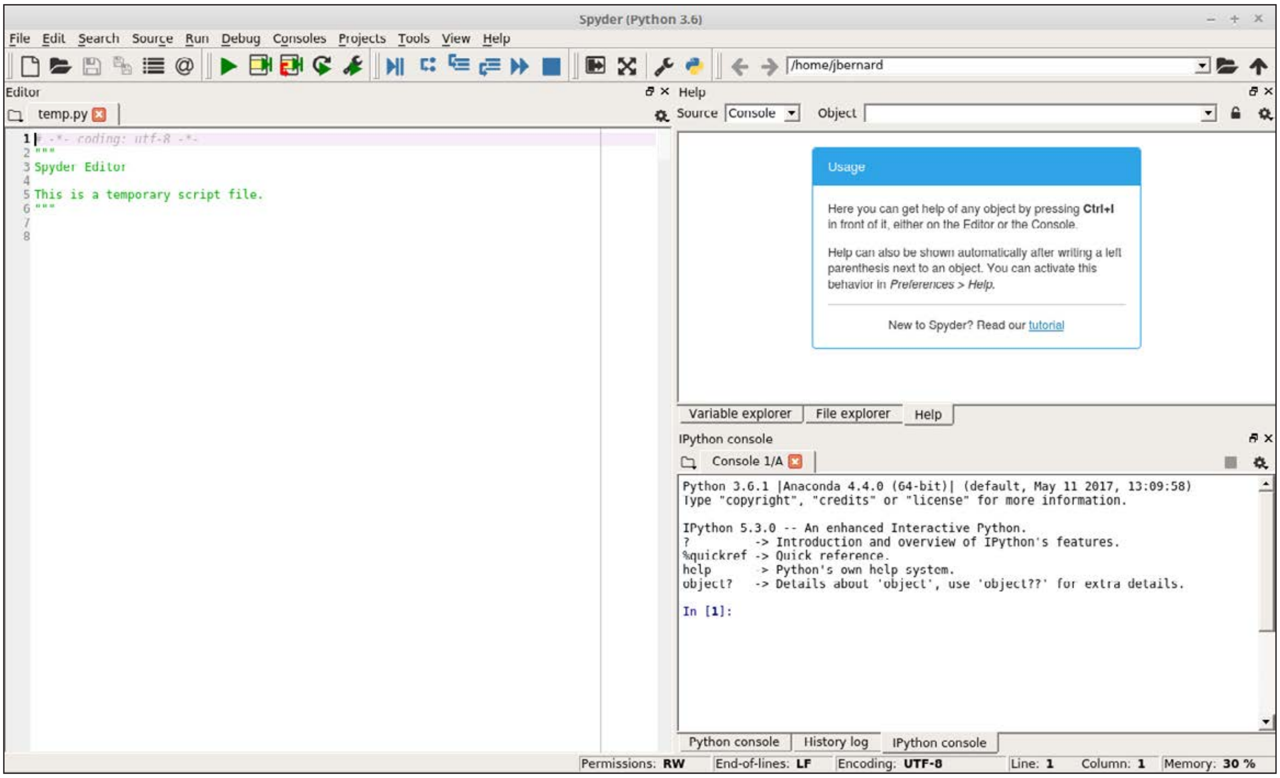
If you have a terminal window open, you can launch spyder simply typing `spyder` and pressing enter. You may get a pop up window saying that spyder is not the latest version. This is just because the version within Anaconda is a few revisions behind.

Once you have spyder started, you should see an open editor window on the left-hand side and a Python console window on the lower right-hand side.

The upper right-hand side is used for a help browser, a variable explorer and a file explorer. Like most IDEs, you can change which panes are visible and their layout within the window.

You can begin working with spyder immediately in the console window. The new default in spyder is to provide an IPython console that you can use to interact with the Python engine directly. It works, essentially, the same way that it works on the command line. The big difference is that spyder can inspect the contents of the Python engine and can do things like display variables and their contents within the variable explorer.

Although this is fine for smaller code snippets, you'll likely end up



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

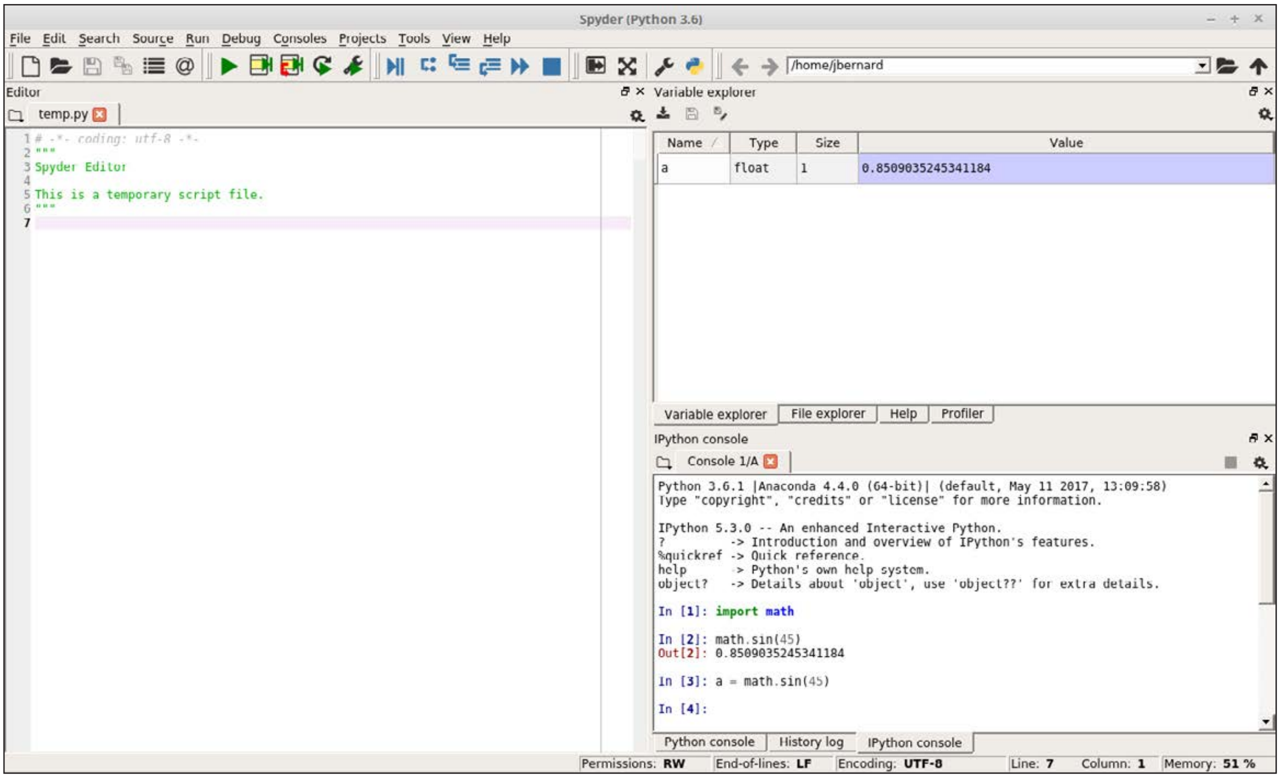
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:09:58)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Permissions: **RW** | End-of-lines: **LF** | Encoding: **UTF-8** | Line: **1** | Column: **1** | Memory: **30 %**

Figure 2. Starting up spyder gives you an empty editor window to start your first project.

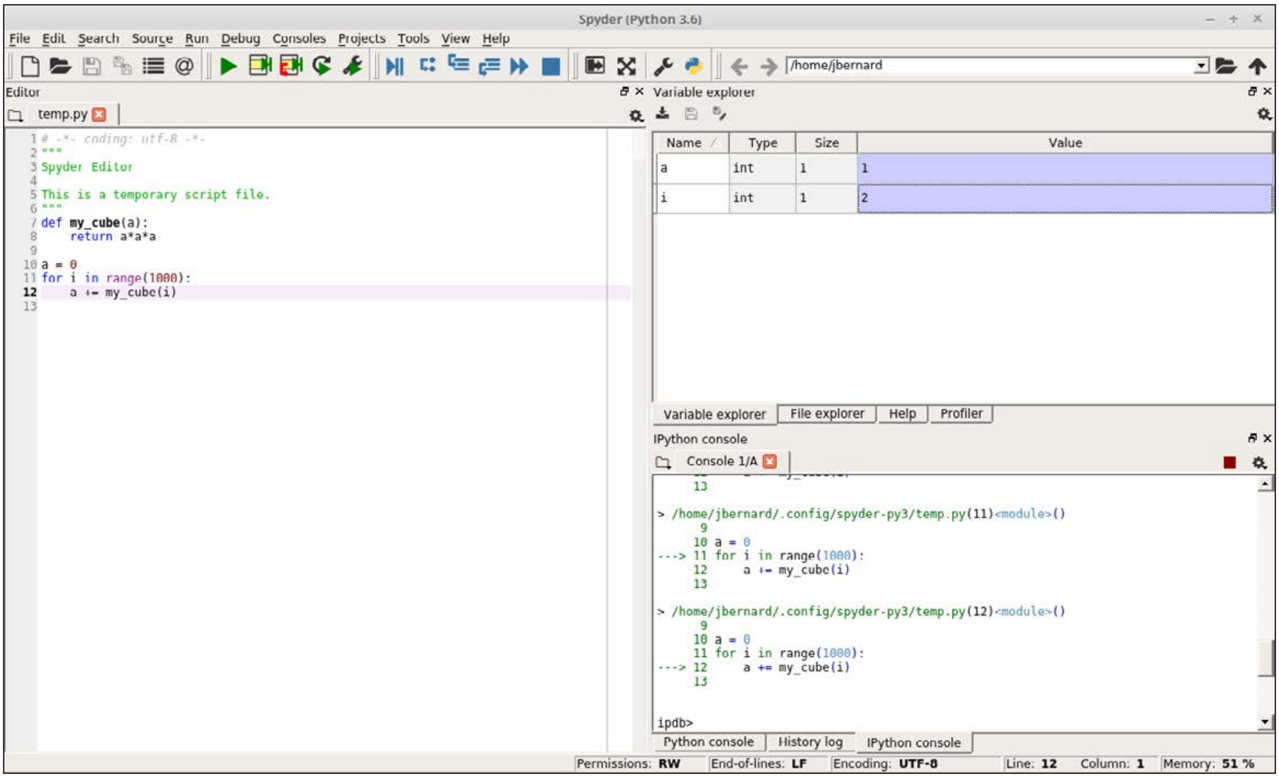

Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/jbernard
Editor
temp.py
Variable explorer
Variable explorer File explorer Help Profiler
IPython console
Console 1/A
Python console History log IPython console
Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 7 Column: 1 Memory: 51 %

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

```

Name	Type	Size	Value
a	float	1	0.8509035245341184

```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:09:58)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import math

In [2]: math.sin(45)
Out[2]: 0.8509035245341184

In [3]: a = math.sin(45)

In [4]:
```

Figure 3. You can interact directly with the IPython console.


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/jbernard
Editor
temp.py
Variable explorer
Variable explorer File explorer Help Profiler
IPython console
Console 1/A
Python console History log IPython console
Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 12 Column: 1 Memory: 51 %

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
def my_cube(a):
    return a*a*a

a = 0
for i in range(1000):
    a += my_cube(i)

```

Name	Type	Size	Value
a	int	1	1
i	int	1	2

```
     13

> /home/jbernard/.config/spyder-py3/temp.py(11)<module>()
      9
     10 a = 0
---> 11 for i in range(1000):
     12     a += my_cube(i)
     13

> /home/jbernard/.config/spyder-py3/temp.py(12)<module>()
      9
     10 a = 0
     11 for i in range(1000):
---> 12     a += my_cube(i)
     13

ipdb>
```

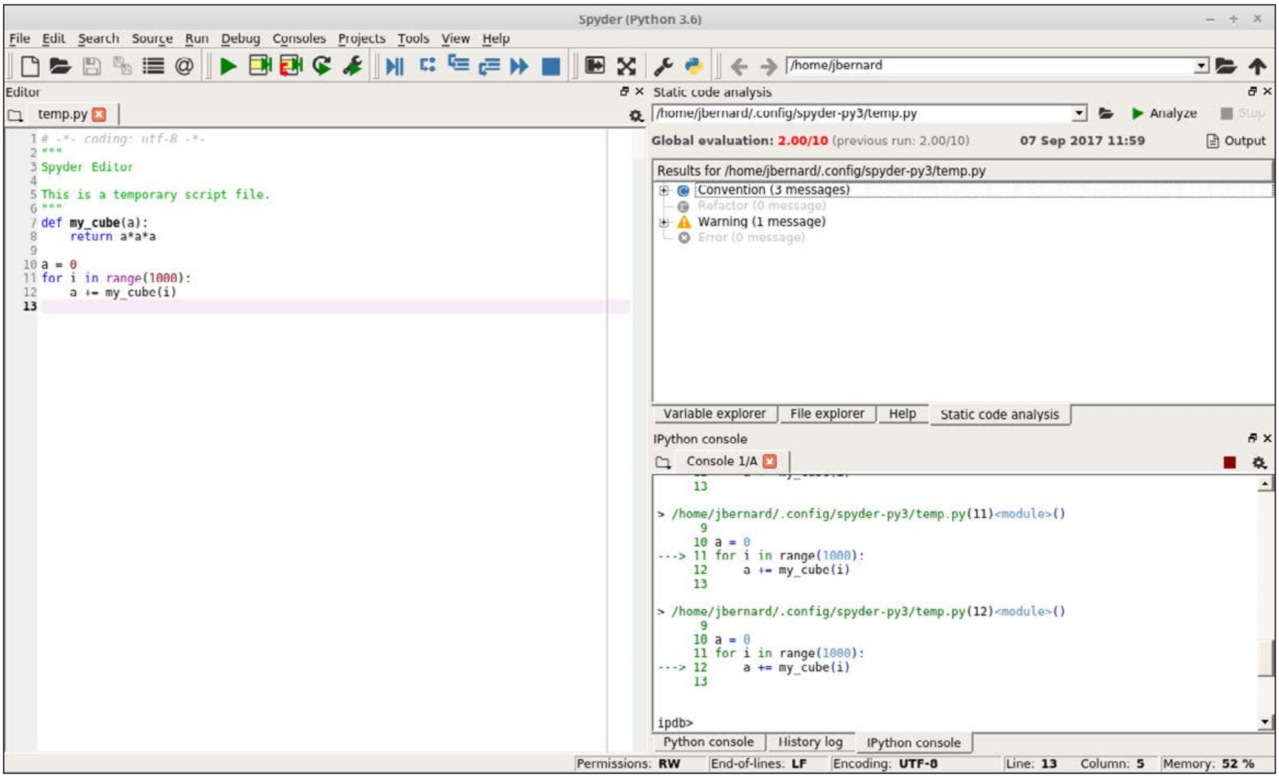
Figure 4. Spyder includes a front end, allowing you to interact with ipdb, the IPython debugger.

working on much larger chunks of code. In that case, you can use the editor to write functions and larger pieces. In order to execute this Python code, you can click the green arrow icon, click the menu item Run→Run or press the F5 key. Again, the results are available from within the variable explorer. If instead you click the blue arrow icon (or click on the menu item Debug→Debug), your code will be run within the IPython debugger, which lets you step through your code one line at a time.

You can gain more control over the debugging by adding breakpoints to your code. To do so, double-click the left-hand gutter in the editor pane. You should see a dot added for each breakpoint you insert.

Several tools are available for working on code and algorithm quality. You'll probably want to start with a static code analysis. Run it by clicking the "Source→Run static code analysis" menu item or by pressing F8. This will run the analysis and will provide the results in a new pane that will pop up in the top right-hand pane.

The results are categorized into convention breaks, refactoring suggestions, syntax warnings and actual errors in your code. This will



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
def my_cube(a):
    return a*a*a

a = 0
for i in range(1000):
    a += my_cube(i)

```

/home/jbernard/.config/spyder-py3/temp.py

Global evaluation: 2.00/10 (previous run: 2.00/10) 07 Sep 2017 11:59

Results for /home/jbernard/.config/spyder-py3/temp.py

- Convention (3 messages)
- Refactor (0 message)
- Warning (1 message)
- Error (0 message)

```
    13

> /home/jbernard/.config/spyder-py3/temp.py(11)<module>()
      9
     10 a = 0
---> 11 for i in range(1000):
     12     a += my_cube(i)
     13

> /home/jbernard/.config/spyder-py3/temp.py(12)<module>()
      9
     10 a = 0
     11 for i in range(1000):
---> 12     a += my_cube(i)
     13

ipdb>
```

Permissions: RW | End-of-lines: LF | Encoding: UTF-8 | Line: 13 Column: 5 | Memory: 52 %

Figure 5. You can run a static code analysis to check for syntactic errors.

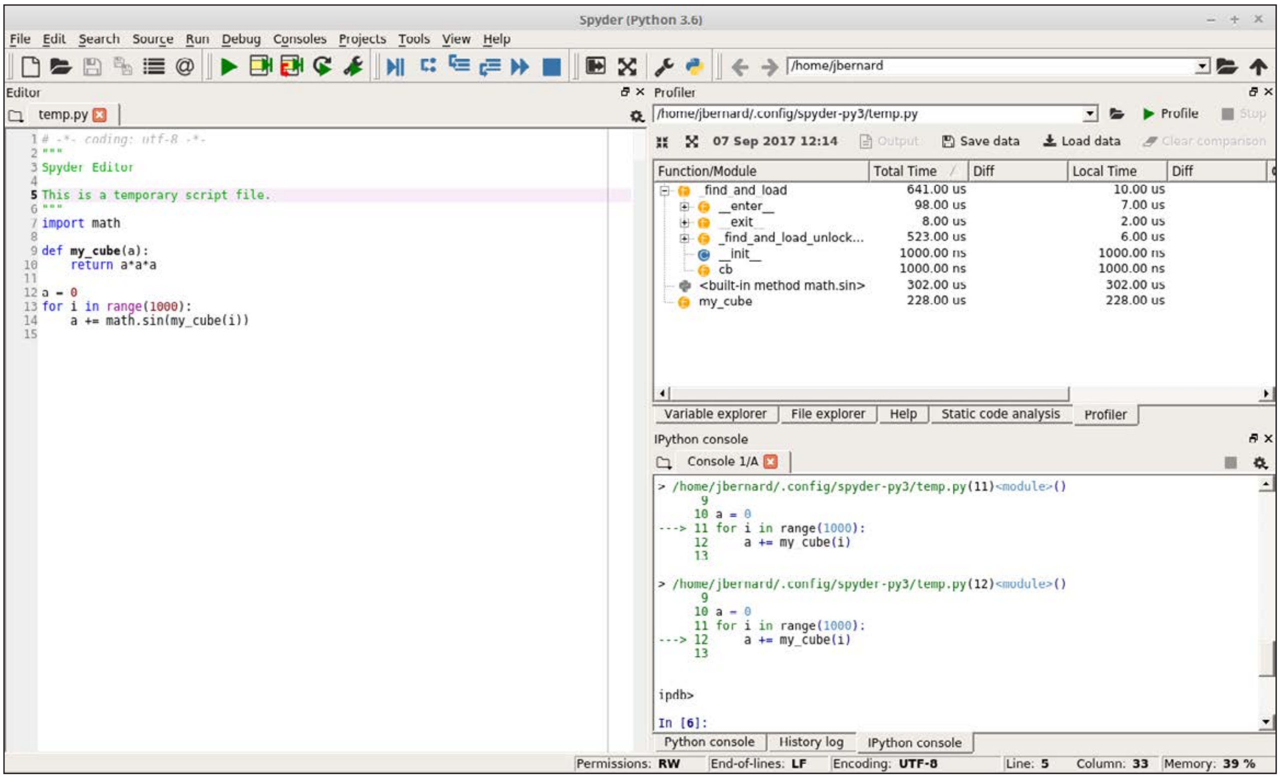
catch the most obvious errors.

Once you have code that actually works, the next step is to check that code's performance. Spyder includes a front end that gives you access to the profiler included in the standard Python library. Start it by clicking the Run→Profile menu item or by pressing F10. Once it finishes, a new pane will appear in the same upper left-hand location.

Unfortunately, the default profiler goes down only to the function level, and that may not be fine enough in detail. If that's the case, you can dive into one of the great features of spyder, its plugin architecture. Several plugins are already included within the Anaconda repository. Use the following command to install the line profiler plugin:

```
conda install -c spyder-ide spyder-line-profiler
```

Then you can add the function decorator @profile to any functions that you want to explore, and then start the line profiler by either clicking the



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import math

def my_cube(a):
    return a*a*a

a = 0
for i in range(1000):
    a += math.sin(my_cube(i))

```

Function/Module	Total Time	Diff	Local Time	Diff
_find_and_load	641.00 us		10.00 us	
__enter__	98.00 us		7.00 us	
__exit__	8.00 us		2.00 us	
_find_and_load_unlock...	523.00 us		6.00 us	
__init__	1000.00 ns		1000.00 ns	
cb	1000.00 ns		1000.00 ns	
<built-in method math.sin>	302.00 us		302.00 us	
my_cube	228.00 us		228.00 us	

```
> /home/jbernard/.config/spyder-py3/temp.py(11)<module>()
      9
     10 a = 0
---> 11 for i in range(1000):
     12     a += my_cube(i)
     13

> /home/jbernard/.config/spyder-py3/temp.py(12)<module>()
      9
     10 a = 0
     11 for i in range(1000):
---> 12     a += my_cube(i)
     13


ipdb>

In [6]:
```

Permissions: RW | End-of-lines: LF | Encoding: UTF-8 | Line: 5 | Column: 33 | Memory: 39 %

Figure 6. When you run the profiler, you'll get a display of how much time is being used in each function.

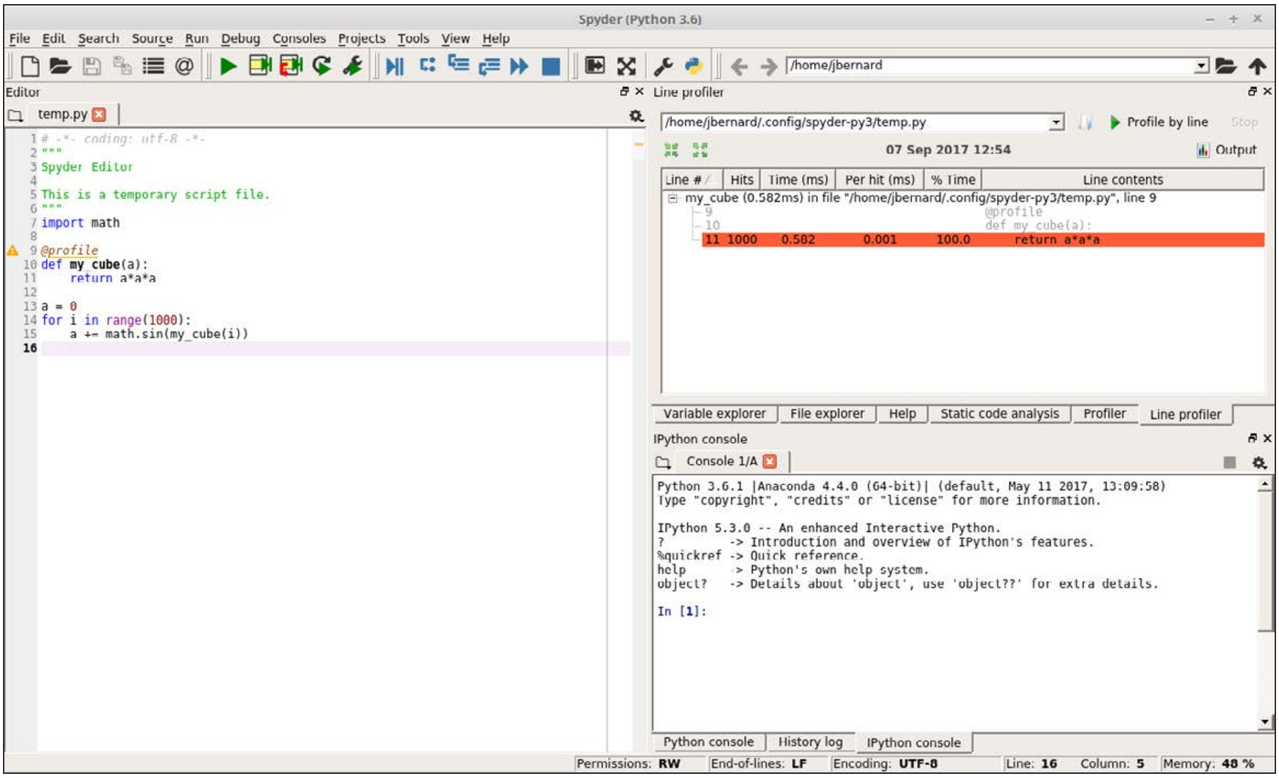
"Run→Profile line by line" menu item or by pressing Shift-F10. You'll then get the results in a new output pane.

You can look at how much time is spent on each line, both per hit and the total for the complete program run. This way, you can focus on the most costly parts of your code to get better performance.

Along with optimizing time, the other parameter you'll want to look at optimizing is memory usage. This is becoming much more important as more and more research is focusing on big data problems. In those cases, you'll want to use the following command to install the memory profiler plugin for spyder:

```
conda install -c spyder-ide spyder-memory-profiler
```

Once the plugin is installed, you can add the decorator @profile just as with the line profiler. Start the memory profiler by clicking "Run→Profile memory line by line" or by pressing Ctrl-Shift-F10. Another pane will appear in the top right-hand side where you can see how memory usage



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import math

@profile
def my_cube(a):
    return a*a*a

a = 0
for i in range(1000):
    a += math.sin(my_cube(i))

```

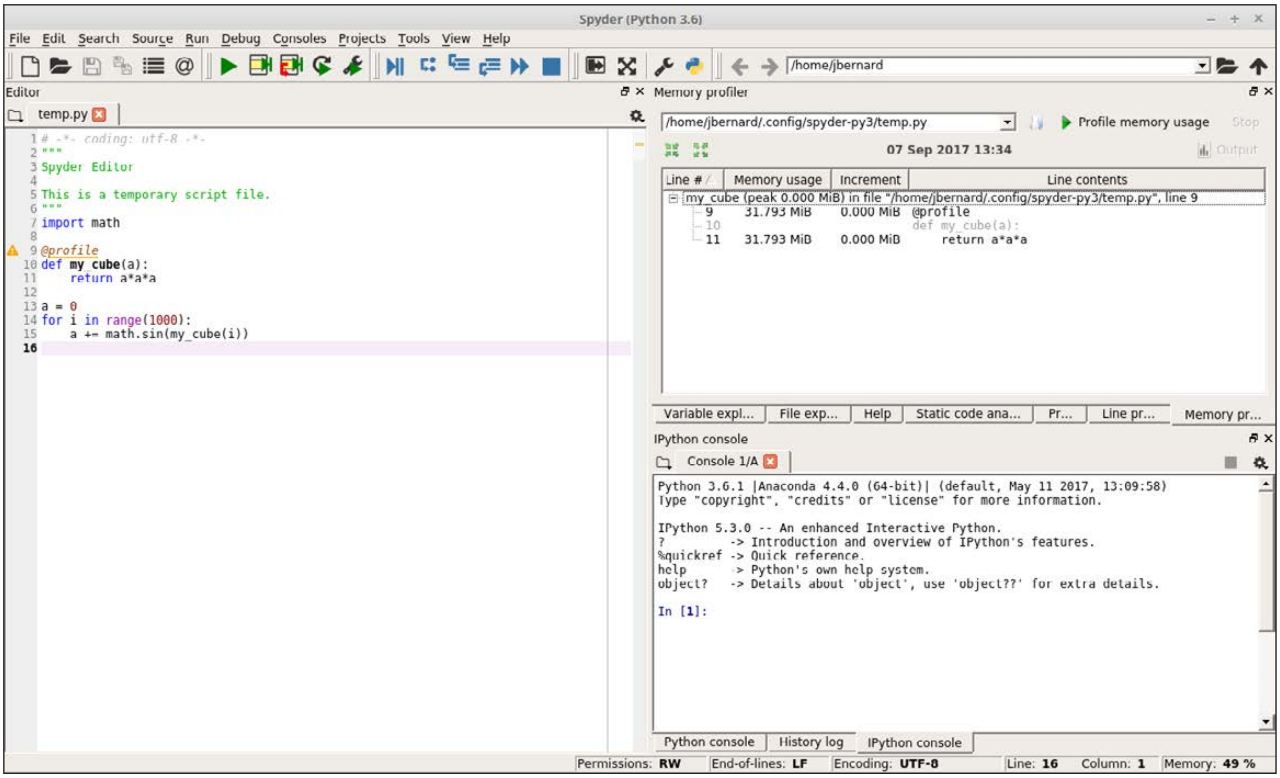
/home/jbernard/.config/spyder-py3/temp.py

07 Sep 2017 12:54

```
Line #  Hits  Time (ms)  Per hit (ms)  % Time  Line contents
my_cube (0.582ms) in file "/home/jbernard/.config/spyder-py3/temp.py", line 9
  9                                            @profile
  10                                           def my_cube(a):
  11    1000  0.582      0.001         100.0       return a*a*a
```

```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:09:58)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Figure 7. You can use the line profiler to see how efficient functions are in more detail.


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/jbernard
Editor
temp.py
Memory profiler
Profile memory usage
Stop
Output
Variable expl... File exp... Help Static code ana... Pr... Line pr... Memory pr...
IPython console
Console 1/A
Python console History log IPython console

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import math

@profile
def my_cube(a):
    return a*a*a

a = 0
for i in range(1000):
    a += math.sin(my_cube(i))
```

/home/jbernard/.config/spyder-py3/temp.py

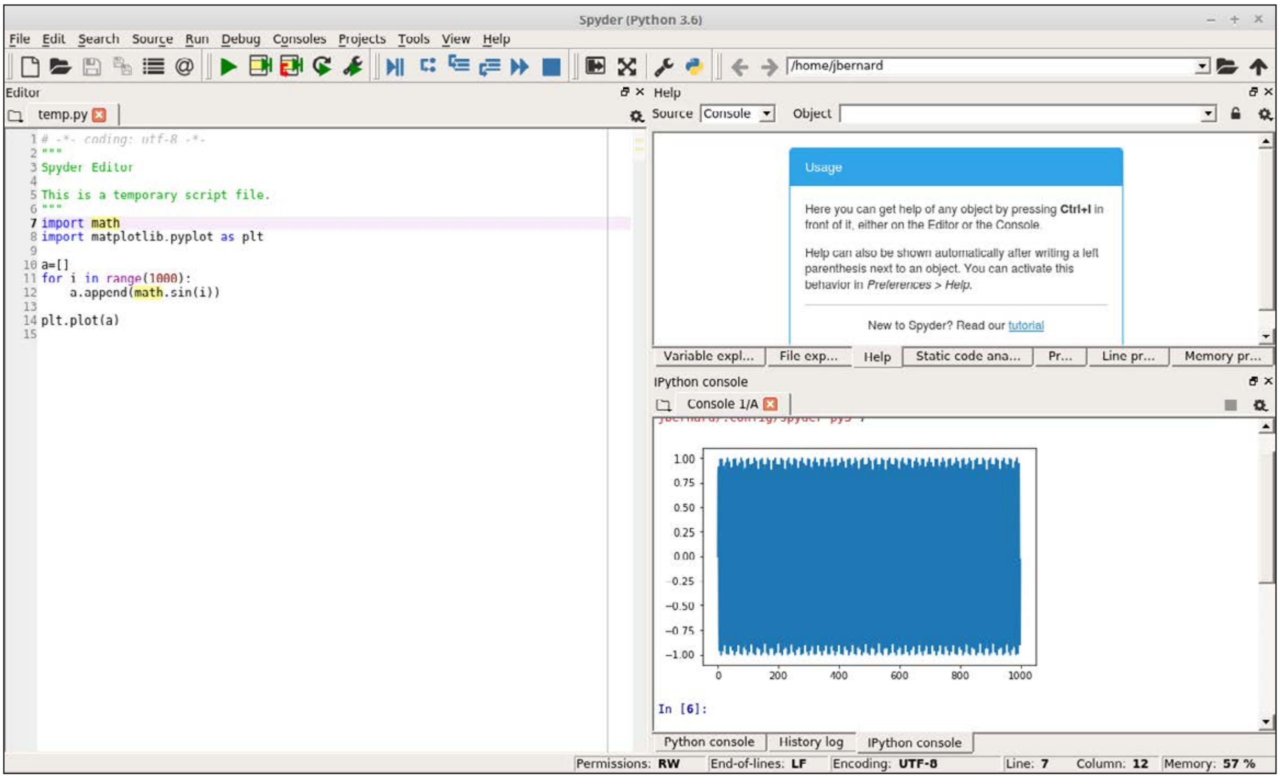
07 Sep 2017 13:34

```
Line #   Memory usage   Increment   Line contents
⊟ my_cube (peak 0.000 MiB) in file "/home/jbernard/.config/spyder-py3/temp.py", line 9
  9      31.793 MiB     0.000 MiB   @profile
  10                                def my_cube(a):
  11     31.793 MiB     0.000 MiB       return a*a*a
```

```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:09:58)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 16 Column: 1 Memory: 49 %

Figure 8. There is a memory profiler plugin for spyder that allows you to figure out how to optimize memory usage.


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/jbernard
Editor
temp.py
Help
Source
Console
Object
Variable expl... File exp... Help Static code ana... Pr... Line pr... Memory pr...
IPython console
Console 1/A
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
200
400
600
800
1000
Python console History log IPython console

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import math
import matplotlib.pyplot as plt

a=[]
for i in range(1000):
    a.append(math.sin(i))

plt.plot(a)
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
In [6]:
```

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 7 Column: 12 Memory: 57 %

Figure 9. By default, spyder allows you to generate plots within the IPython console.

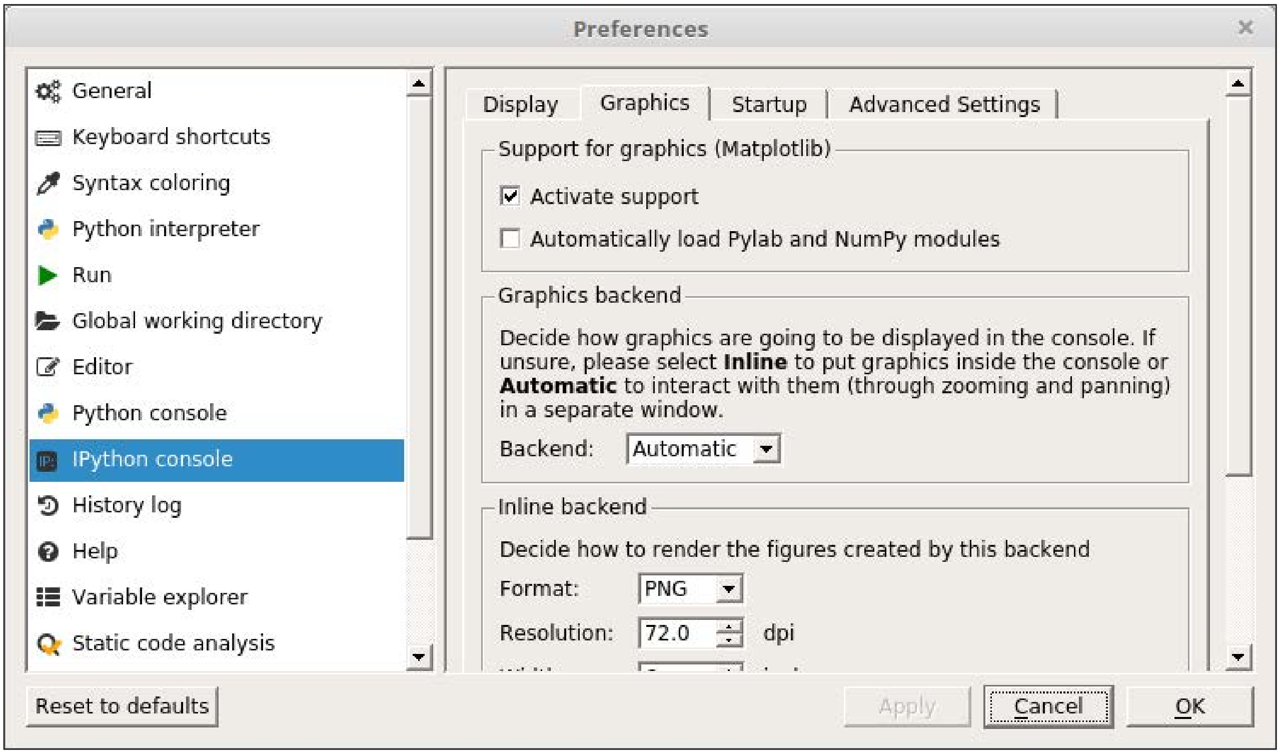
changes after each line of code. This will allow you to figure out which lines of code are being wasteful and where to focus your brain power for improving performance.

For scientific computing, the last item I want to look at is the ability to visualize data. Humans often can make intuitive leaps by being able to see how data looks. The default setting for spyder is that graphs are drawn inline within the IPython console.

This is fine for a quick glance at the data, but it isn't the easiest to look at. If you click Tools→Preferences, you'll see a new window where you can change this behavior and have plots show up in a different window instead.

If you rerun the code, you'll now get the plot in a new window. This allows for the ability to play with the plot display and even save off the final image. If you change the settings around plotting in the preferences, you may need to restart the IPython engine to pick up the new preferences.

And, that should be enough to get started using spyder in your computational science problems. In my next article, I'll look at the version



Support for graphics (Matplotlib)

☑ Activate support

□ Automatically load Pylab and NumPy modules

Graphics backend

Decide how graphics are going to be displayed in the console. If unsure, please select **Inline** to put graphics inside the console or **Automatic** to interact with them (through zooming and panning) in a separate window.

Backend: Automatic

Inline backend

Decide how to render the figures created by this backend

Format: PNG

Resolution: 72.0 dpi

Figure 10. You can change the preferences on how plots are generated and displayed.

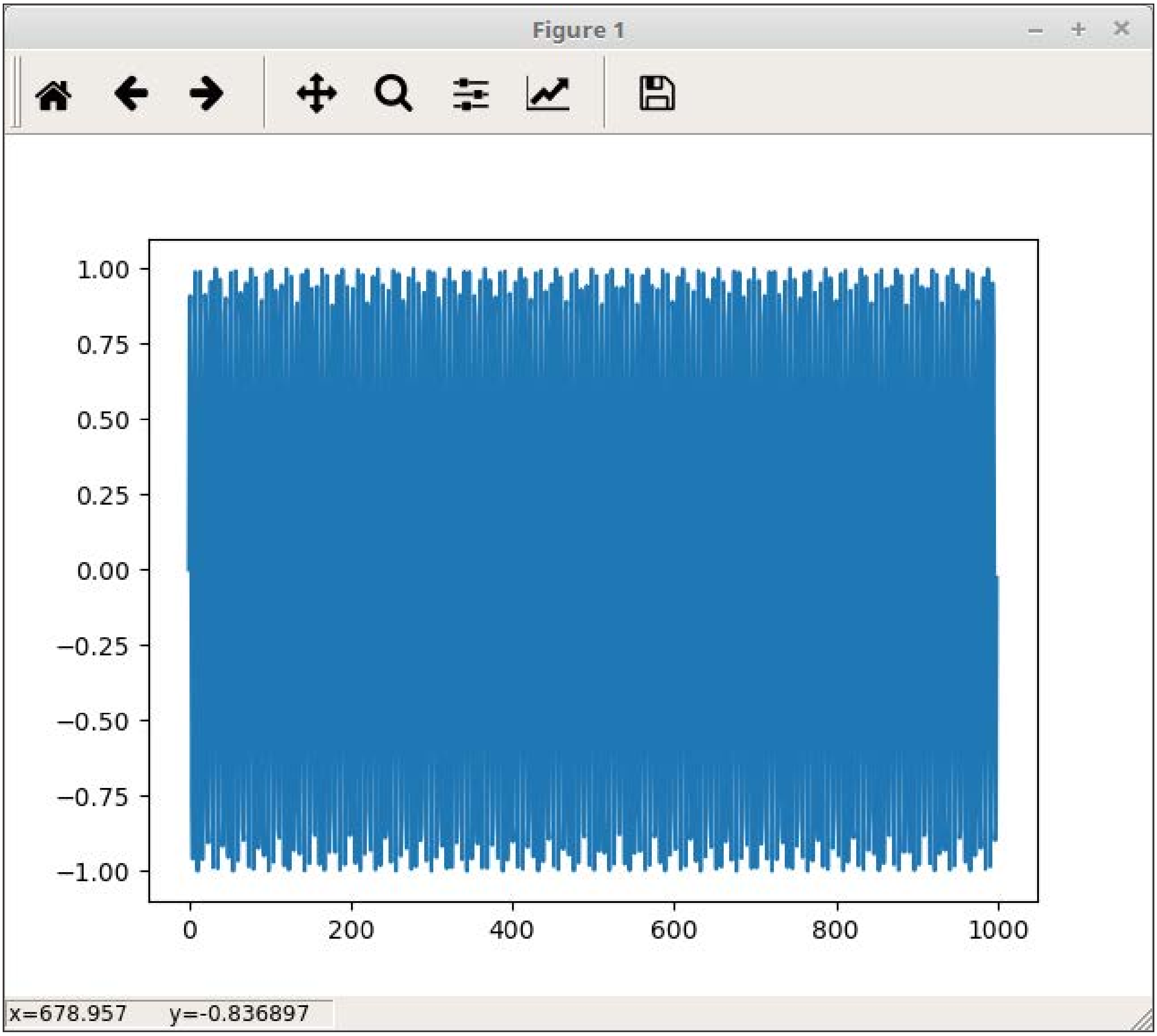


Figure 11. Generating plots in their own window allows for more interaction.

of Jupyter that is included with Anaconda and see how it can be used effectively. Both tools are good, but they fit within slightly different ecological niches.—Joey Bernard

Where Wire Won't Work

You'll probably get tired of hearing about my farm, but it has been a great opportunity for me technology-wise to learn about new products. I've never had property with acreage, and so the idea of remote outbuildings is new. If you look at Figure 1, you'll see my farmhouse is about 200 feet away from the barn. We don't raise animals, but we're remodeling the barn into a tech-friendly location for music, parties, worship and movie nights. That requires bandwidth, if nothing else, so Spotify can be streamed for entertainment. The metal roof extends down the sides of the building and makes it impossible to get a Wi-Fi signal from inside the barn. So I tried to find an affordable wireless bridge that was reliable and wouldn't saturate the 2.4GHz spectrum on my farm. Thankfully, the EnGenius ENH500 outdoor wireless bridge (available in pairs on Amazon) did the trick.

The ENH500 is 5GHz-only, and once configured, it acts like a physical wire connecting the two ends. The devices obviously don't show up

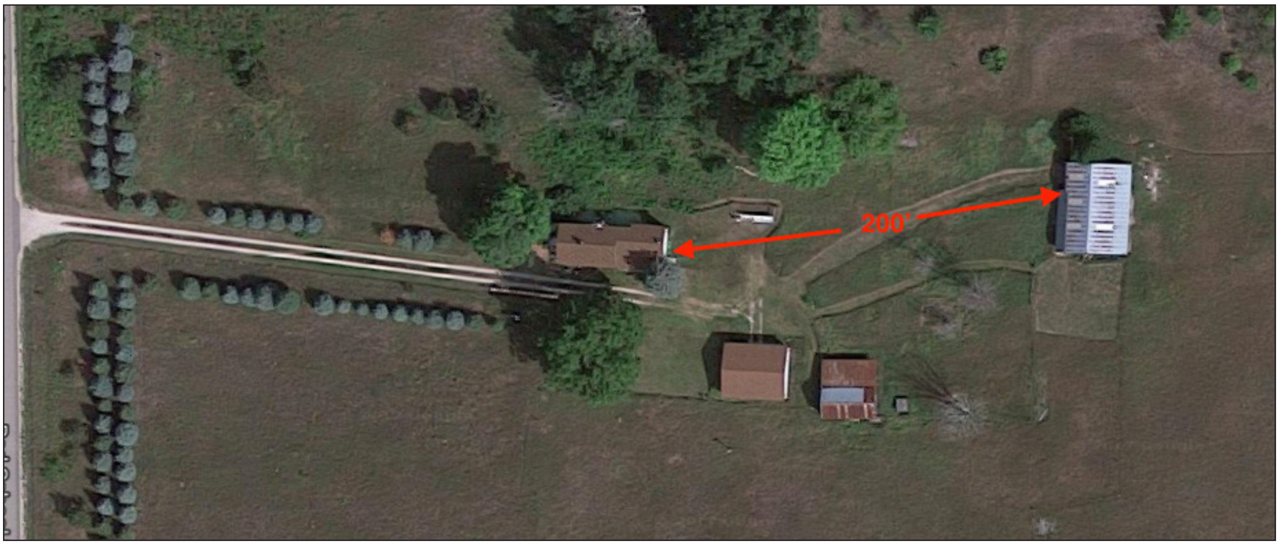


Figure 1. The Farm

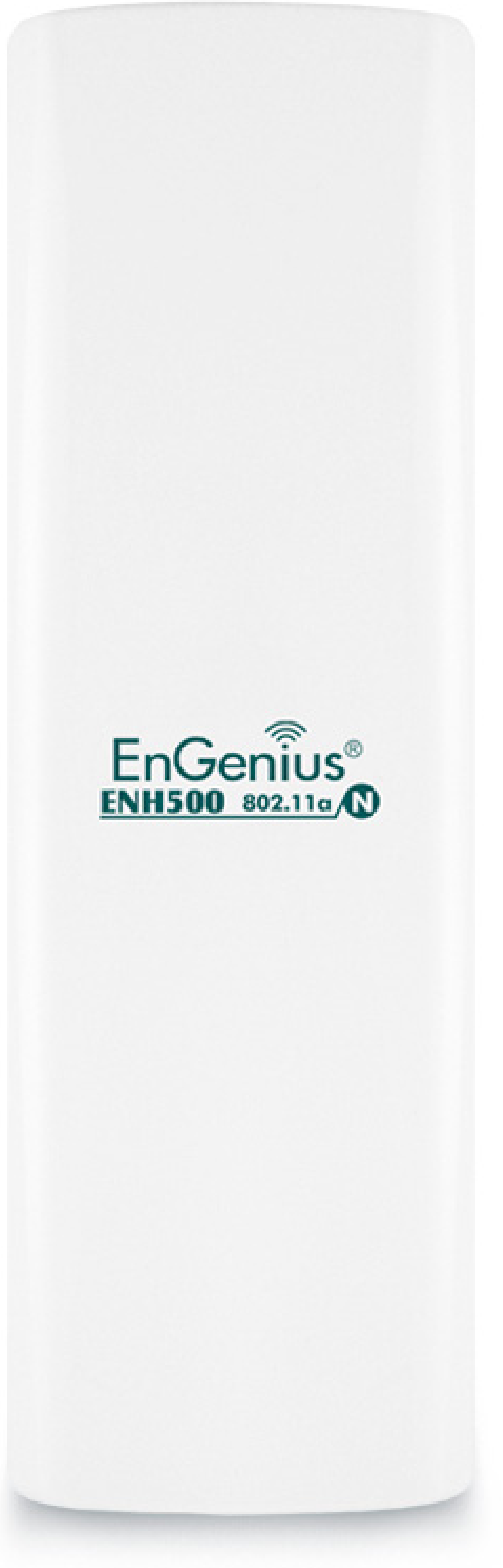


Figure 2. The EnGenius ENH500 Outdoor Wireless Bridge

in my UniFi WiFi management software, but they also don't interfere with communication to the remote APs in the barn. I set them up in WPS bridge mode, which involves entering the MAC addresses of each device into the other, and pointed them in the general direction of each other. The connection is rock-solid, throughput is amazing, and since they're using the uncluttered 5GHz frequency range, there aren't any issues with my other devices. I do wish the "signal strength" meter worked in WPS Bridge mode, but for some reason, the layer-2 bridge doesn't allow the fancy strength lights to work. You have to check the dB strength via the web interface to make sure you have a solid connection.

Honestly, once they're set up, the EnGenius bridge devices just work. I don't notice any difference from having a wired connection, and it was much easier (and cheaper!) than running a fiber-optic cable out to the barn. If you have an outbuilding or need to extend network coverage across an area that would be challenging to hard-wire, the EnGenius bridges work as advertised!
—Shawn Powers

The Last Mile

You may have noticed that I've mentioned in my last few articles that my family recently bought a farm. It's beautiful. There are rolling hills, scenic landscapes and this time of year, the autumn leaves are stunning. When we considered buying the place, my first concern was the availability of broadband internet. Yes, I eventually checked out the house's foundation and such, but really, if I couldn't get broadband, it was a showstopper. Thankfully, the farmhouse is serviced by CenturyLink DSL. And so, we bought the farm. Unfortunately, I didn't realize DSL speeds in rural areas could be as slow as 2.5mbps down and less than 0.5mbps up. Not only is it impossible to stream HD video, it's almost impossible to browse the internet at all!

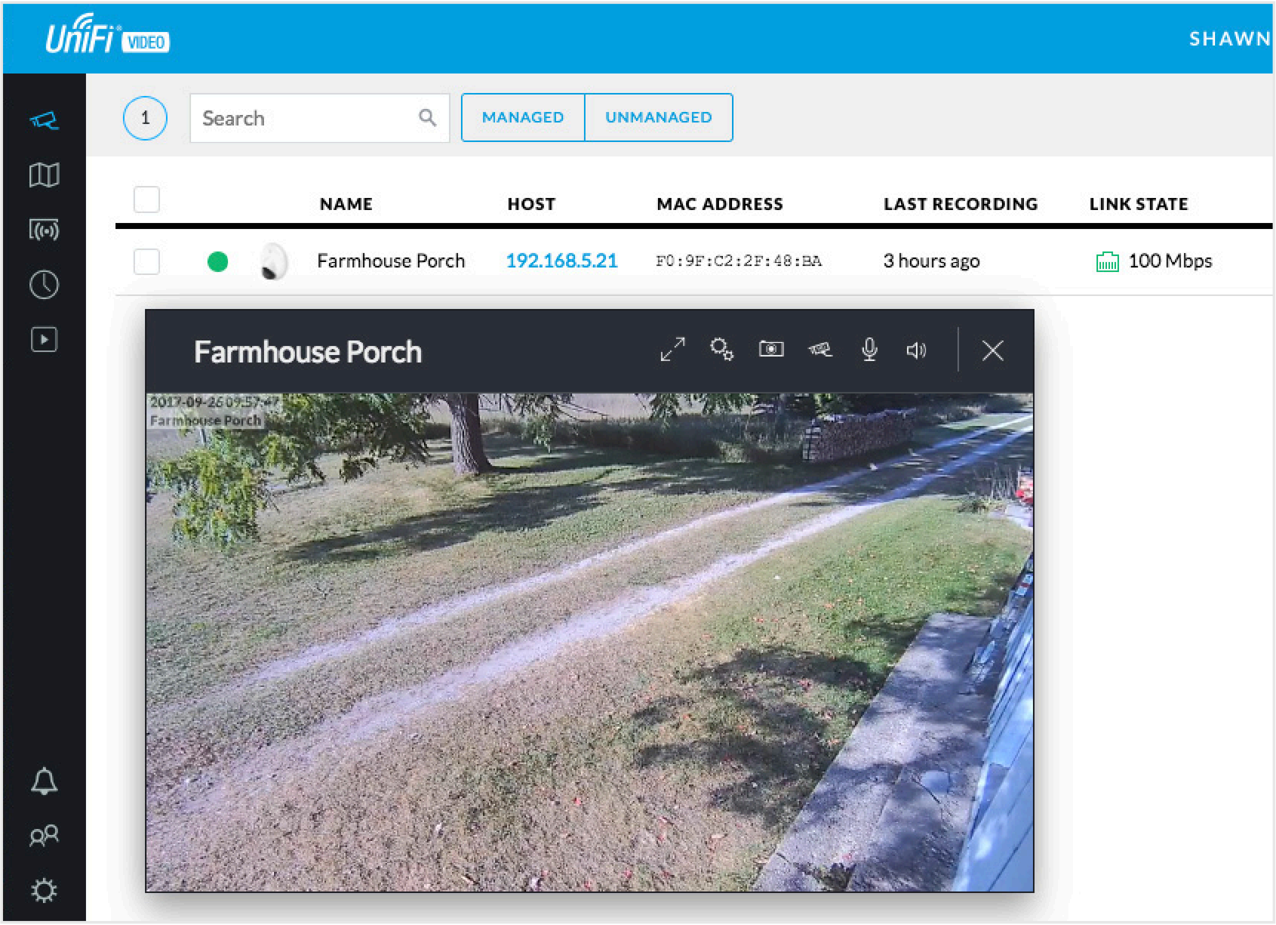
So for the past few months, I've been doing everything I can to cache videos, proxy web traffic and delay network maintenance until the wee hours of the night to preserve what precious throughput is available. Then I stumbled across https://unlimitedlteadvanced.com. The URL sounds like a phishing site. The site looks like a pre-baked phishing site. And the product offering seems too good to be true, and it comes with a hefty up-front fee. There is little in the way of reviews online, but I still pursued the idea for a couple reasons:

1. Did I mention the DSL speeds?

2. Unlimited LTE Advanced resells T-Mobile cellular data. Historically, T-Mobile has had zero coverage in my area. Oddly enough, however, there's a tiny little tower about a mile from my farm that T-Mobile leases space on. In fact, I literally can *see* the tower from my farmhouse kitchen window. And although T-Mobile does QOS management on crowded towers, the odds of my little tower being crowded is slim to none.

I decided to try it. For the Sim-only plan with a Netgear LB-1120 cellular modem (awesome product, by the way) so I could use my own router, it was close to $300 to start up. There is a 14-day trial window (shipping time counts against the 14 days!) that allows you to cancel, but you're out the activation fee, which is the majority of the initial cost. And unless you have T-Mobile, there's no way to know what your bandwidth actually might look like until you get the hardware.

In the end, I got the equipment and sim card in two different shipments. They came from two different companies, neither named "Unlimited LTE Advanced". But when I plugged everything in, I got five bars of service, and my bandwidth is around 18mbps up and 7mbps down. The service is $76 a month, which might seem expensive, but crappy DSL was $50 a month. And so far, I haven't noticed any issues with data caps or throttling, even when streaming 1080p HD video. If you need broadband service, and you are in a T-Mobile LTE coverage area, I can actually recommend Unlimited LTE Advanced for internet service. But boy does it seem sketchy during the sign-up period!—Shawn Powers



□			NAME	HOST	MAC ADDRESS	LAST RECORDING	LINK STATE
□	●		Farmhouse Porch	192.168.5.21	F0:9F:C2:2F:48:BA	3 hours ago	100 Mbps

Security Cameras, Free Software

It's no secret that I love Ubiquiti hardware. Its Wi-Fi access points are amazing, and its management software installs perfectly on my Linux servers completely free. Since we recently purchased a farm that we visit only on the weekends, I decided to give the Ubiquiti security camera system a try. I'm happy to say, it works just as well as the Wi-Fi and networking systems. In fact, it has a web-based management system that installs on my Linux servers as well. It's completely free, and it has all the major features you'd expect from an NVR.

There are a few oddities with the Ubiquiti UniFi Video system, but all are tolerable. First, Ubiquiti wants you to purchase its NVR hardware to manage the cameras. It's a small, energy-efficient computer that works perfectly fine. But it's just that, a computer running Linux. If you dig a bit on the website, you can find the software and install it on your own computer! It's the exact same hardware, and it even gives you free access to Ubiquiti's "cloud access", which allows the software to log in to the cloud server and gives you remote access via the web (http://video.ubnt.com) or mobile apps without the need to forward any ports into your network.

The cameras are incredibly high-quality, and they provide 1080p video with sound. The dome camera is POE+, but the soda-can style camera is sadly 24v passive POE only. If you buy them singly, they come with power injectors, but if you buy a five-pack of the cameras, know that they don't come with any POE injectors, regardless of the camera style you get. If you use UniFi switches that support 24v passive POE, it's no problem, but otherwise, you need to figure a way to power the cameras.

The UniFi software (again, free, but not open source) allows you to record motion and be notified if any motion has been recorded. You can access the recording or the live feed remotely (as in the picture here). You can set the software to delete old footage after a certain time frame or tell it to start erasing old video once a certain amount of free space has been reached on the computer. It supports a large number of cameras, and so far, I've been extremely impressed by the quality of the hardware and software. Not surprising, but still, it's great to see Ubiquiti carrying its product quality into the video-surveillance world as well.—Shawn Powers

PREVIOUS
UpFront

NEXT
Reuven M. Lerner's
At the Forge

No Snooze for You!

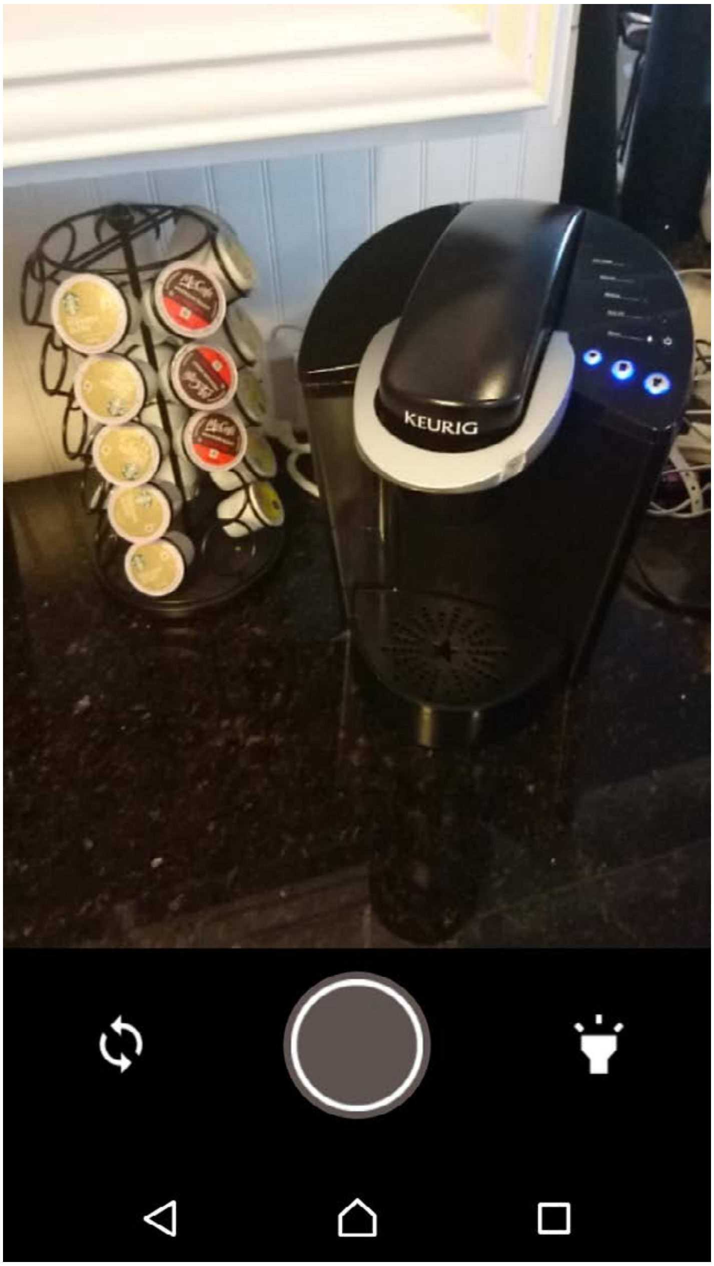
LINUX JOURNAL EDITORS' CHOICE

I realized a while back that I've started setting my alarm an hour early so I can snooze over and over before waking up. Intellectually, I know that's silly, but there's just something fulfilling about hitting snooze and snuggling back into bed. But since I end up losing an hour of *good* sleep, I decided I needed a change. Change is hard, and so I opted for an app to help me. Meet Alarmy.

Alarmy is an Android app that calls itself "The World's Most Annoying Alarm Clock App", and after using it, I think I agree. Like other apps designed to make you actually wake up before snoozing, Alarmy supports things like math problems that must be solved or violently shaking your phone in order to turn off the alarm. But Alarmy doesn't stop there. The most popular (and heinous) mode has you register a place in your home that must be visited and photographed in order to stop the alarm.

When you're fully awake, it's easy to figure out what sort of place makes the most sense to register. The Alarmy folks recommend the bathroom sink, but I personally recommend the coffee pot. If I trot out to the kitchen, and take a photo of the coffee pot, you can bet I'm also going to make a cup of coffee while I'm there. It's just too tempting. I'll probably visit the bathroom while my Keurig brews a cup, but by that point, I'm awake and craving coffee, so I'm unlikely to go back to bed.

Is it cruel? Yes. Does it work? Absolutely. Alarmy is free and offers ad-elimination for a small fee. Ads usually don't bother me in an app like



this, but if I like an app, I usually pay for it anyway to support the developer—in this case, the sadistic, cruel developer! Check out Alarmy at the Google Play Store: https://play.google.com/store/apps/details?id=droom.sleepIfUCan.

We have a sense of humor here at *Linux Journal*, so thanks to Alarmy's annoyingly wonderful way of forcing users to wake up, it gets this month's Editors' Choice award (and also a bit of loathing, but we don't have an award for that).—Shawn Powers

RETURN TO CONTENTS

Launching External Processes in Python

Think it's complex to connect your Python program to the UNIX shell? Think again!

REUVEN M. LERNER

Reuven M. Lerner, a longtime Web developer, offers training and consulting services in Python, Git, PostgreSQL and data science. He has written two programming ebooks (*Practice Makes Python* and *Practice Makes Regexp*) and publishes a free weekly newsletter for programmers, at http://lerner.co.il/newsletter. Reuven tweets at @reuvenmlerner and lives in Modi'in, Israel, with his wife and three children.

◀ PREVIOUS
Editors' Choice

NEXT
Dave Taylor's
Work the Shell ▶

IN MY PAST FEW ARTICLES, I've been looking into concurrency in Python via threads. The good news with threads is that they are relatively easy to work with and let you share data among threads without too much trouble. The bad news is that if you're not careful, you can end up with serious problems—because data isn't shared, and Python data structures aren't thread-safe. But perhaps a bigger problem is that Python's global interpreter lock (GIL) guarantees that only one thread runs at a time.

In many cases, this isn't really a problem. In particular, if you're writing programs that work with the filesystem or network, you probably won't feel the pain of Python threads too badly. That's because while only one thread runs at a time, a thread gives up control of the CPU whenever it uses I/O. This is because disks and networks are many times slower than CPUs; while you're waiting for the filesystem to give you the data you've requested, another thread can be running.

That said, there definitely are times when Python's threads show their limitations. In particular, if you're writing code that is CPU-bound—that is, in which the CPU is the bottleneck—you'll find that threads are limited. After all, if you have a nice 48-core machine with which to play, doesn't it seem silly to have only one of those cores actually doing something?

There is, of course, a solution to these problems—one that many traditional UNIX users consider to be superior under many circumstances: *processes*. Rather than run a function in a new thread, run it in a new process!

So in this article, I take an initial look at working with processes in Python to do a very common task: invoking external commands. In so doing, I also cover how working with processes is structured, leading to my next article's topic: the "multiprocessing" module.

Process Basics

For Linux users, nothing is more basic and everyday than a process. When I fire up Emacs, I start a process. When I start the Apache HTTP server, I start a process, which then starts multiple, additional processes. When I invoke `ls` on the command line, I'm starting a process. And when I tell my computer to shut down, it does so by killing each of those processes.

Think of a process as a data structure that represents a computer at a particular moment in time. A process has code that is running (including code that has yet to run); it has data on which the program works; it has access to memory to store and retrieve additional data, and it can talk to external devices, from filesystems and networks to keyboards and screens.

A single Linux machine can run many, many processes at once. For the split second during which a process runs, it has the illusion of having complete control over the computer. It's thanks to the fact that modern computers are so fast that you can run so many processes and yet have them all appear to be running concurrently. True, modern computers have multiple CPUs (aka "cores"), which lets you divide the work among those cores.

There are all sorts of ways to start processes in Python. In modern versions of the language, you can use the "subprocess" module to start up a process and even retrieve the result. For example, you can invoke the `ls` program in a new process and then view the results:

```
>>> subprocess.check_output('ls')
```

From this function, you get a string containing the output from the `ls` command. It's a big ugly one to see, especially if you're used to seeing things printed nicely. In such a case, you don't want to view the string that was returned, but rather to print it. The thing is, that doesn't seem to work, at least not in Python 3:

```
>>> print(subprocess.check_output('ls'))
```

The problem is that, by default, `subprocess.check_output` returns a "bytestring", similar to a Python 2 string, in that it contains a sequence of bytes, rather than a sequence of Unicode characters. The issue here is that when you print a bytestring, Python doesn't actually go to a new line when it sees `\n`.

You can get around this problem by telling Python to interpret newline characters liberally and to return a string instead of a bytestring:

```
>>> print(subprocess.check_output('ls', universal_newlines=True))
```

This seems to work quite nicely. But what if you want to print only a subset of the files in the current directory? It seems natural to want to say, for example, `ls -l`. Let's try that:

```
>>> print(subprocess.check_output('ls -l', universal_newlines=True))
```

When you do that, you get:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/Cellar/python3/3.6.2/Frameworks/
➥Python.framework/Versions/3.6/lib/python3.6/subprocess.py",
 ➥line 336, in check_output
    **kwargs).stdout
  File "/usr/local/Cellar/python3/3.6.2/Frameworks/
➥Python.framework/Versions/3.6/lib/python3.6/subprocess.py",
 ➥line 403, in run
    with Popen(*popenargs, **kwargs) as process:
  File "/usr/local/Cellar/python3/3.6.2/Frameworks/
➥Python.framework/Versions/3.6/lib/python3.6/subprocess.py",
 ➥line 707, in __init__
    restore_signals, start_new_session)
  File "/usr/local/Cellar/python3/3.6.2/Frameworks/
➥Python.framework/Versions/3.6/lib/python3.6/subprocess.py",
 ➥line 1333, in _execute_child
    raise child_exception_type(errno_num, err_msg)
FileNotFoundError: [Errno 2] No such file or directory: 'ls -l'
```

What's wrong here? Very simply, Python is trying to run an external process, giving it the Linux command `ls -l`. You might think that this is normal and reasonable, since running `ls -l` is something you likely do all the time in your day-to-day lives. But remember that `ls` is the command, and `-l` is a flag to that command. You can understand the difference, and the shell typically separates them for you. But if you simply hand that command name to Linux, it's going to get confused and complain.

So instead of passing a single string, you'll need to pass a list of strings, in which each represents a "word" of the command. For example:

```
>>> print(subprocess.check_output(['ls', '-l'], universal_newlines=True))
```

This works just fine. You can add other arguments, including the names of files:

```
>>> print(subprocess.check_output(['ls', '-l', 'urls.txt'],
>>> universal_newlines=True))
```

What if you want to get a long listing of all ".txt" files? Just try this:

```
>>> print(subprocess.check_output(['ls', '-l', *.txt'],
 ➥universal_newlines=True))

>>> print(subprocess.check_output(['ls', '-l', '*.txt'],
>>> universal_newlines=True))
ls: cannot access '*.txt': No such file or directory
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/Cellar/python3/3.6.2/Frameworks/
➥Python.framework/Versions/3.6/lib/python3.6/subprocess.py",
 ➥line 336, in check_output
    **kwargs).stdout
  File "/usr/local/Cellar/python3/3.6.2/Frameworks/
➥Python.framework/Versions/3.6/lib/python3.6/subprocess.py",
 ➥line 418, in run
    output=stdout, stderr=stderr)
subprocess.CalledProcessError: Command '['ls', '-l', '*.txt']'
 ➥returned non-zero exit status 2.
```

It complains that "*.txt" isn't a legitimate file. That's because while you might think that Linux always knows that * represents all of the files in a directory, that's not the case—it is the shell that performs the interpretation of such characters as "*", dividing things up and then passing them along to the underlying operating system.

So, how can you list all of the files with a "*.txt" suffix? You can invoke the same call once again, but tell Python to pass the parameters through the UNIX shell:

```
>>> print(subprocess.check_output(['ls', '-l', '*.txt'],
                                  shell=True,
                                  universal_newlines=True))
```

Aha! It now seems to work just fine.

So, what happened here? This started a new process (a "subprocess", if you will), and in that process, executed a UNIX program. The program returned some text, that Python captured, and then printed it out.

The Python documentation makes it clear that having `shell=True` in your call to `subprocess.check_output` (and other functions) is a potential security risk. If you're getting input from an unknown or untrusted user, that person can insert arbitrary commands into the system on which `check_output` is running. Be sure to consider the security implications of `shell=True` before using it.

More Generally

`subprocess.check_output` is a specific function, one that's designed to run a program and retrieve its output. If you want a bit more flexibility, you can run other functions from "subprocess".

For example, let's say you want to take the output from `ls` and put it into a file. On the UNIX command line, you could say:

```
ls -l > file-list.txt
```

In Python, this is a bit more complex, but not terribly so if you use `subprocess.run` This function is new (as of Python 3.5), but it makes life a bit easier.

You can try this:

```
>>> subprocess.run(['/bin/ls', '-l'], universal_newlines=True)
```

As you can see, `subprocess.run` takes many similar arguments to `subprocess.check_output`. But what's different is that it doesn't return a string, even when `universal_newlines` is set to `True`. Instead, it returns an instance of `subprocess.CompletedProcess`, which contains all sorts of information about the process that ran.

You can grab this, and then see what the `CompletedProcess` contains:

```
>>> cp = subprocess.run(['/bin/ls', '-l'], universal_newlines=True)
>>> vars(cp)
```

You'll get back:

```
{'args': ['/bin/ls', '-l'], 'returncode': 0, 'stderr': None,
'stdout': None}
```

Hmm, that's likely not quite what you wanted. The `args` is fine, and `returncode` is accurately showing 0, meaning that everything ended just fine. But what happened to the output? The answer is that when it comes to `subprocess.run`, you need to indicate where the output should go.

The way to indicate that you want to get something back is to pass `subprocess.PIPE` as the value of the `stdout` keyword argument:

```
>>> cp = subprocess.run(['/bin/ls', '-l'], stdout=subprocess.PIPE,
>>> universal_newlines=True)
>>> vars(cp)
```

You'll now get the following:

```
{'args': ['/bin/ls', '-l'],
 'returncode': 0,
 'stderr': None,
 'stdout': 'total 344\ndrwxr-xr-x  1454 reuven  staff  49436
   ➥Sep 17 09:29 Archive\ndrwxr-xr-x    37 reuven  staff   1
```

I'm not even going to show you the rest, because it's so long, but the `stdout` value is precisely right.

You also can assign `stderr` to `subprocess.PIPE` in order to receive it. Note that in the case of both stdout and stderr, you can assign not just `subprocess.PIPE`, which lets you grab and work with the program's output, but also an open (writable) file object. This means you can invoke an external process and put its output into an arbitrary file. I'd argue that most of the time, the reason you would be executing an external process in Python is that you want to do something to the text, but this will work.

You might be wondering whether you can not only write to stderr and stdout, but also read from stdin. And the answer is definitely. Just provide a file object, and `subprocess.run` will do the rest. For example:

```
>>> cp = subprocess.run(['/bin/cat', '-n'], stdin=open('/etc/passwd'),
                        stdout=subprocess.PIPE, universal_newlines=True)
```

In this case, you run `/bin/cat` with the `-n` option, numbering the lines of a file. What's the input file? /etc/passwd. And where does the output go? To your `subprocess.PIPE` object, which is a kind of communication channel to external processes.

For me, the most interesting thing is the `CompletedProcess` object (`cp`), from which you can grab different pieces of information about the completed process. Note that `subprocess.run` will return only after the external program has finished running, at which point the `cp` variable will be set. And from there, you can grab `stdout`, which is normally a bytestring, but which is an actual (Unicode) string if you set `universal_newlines` to `True`.

Conclusion

You've now seen how you can use the "subprocess" module to communicate with external processes. But let's face it. This doesn't exactly solve the initial problem: breaking a problem up and using different processes to handle it. Rather, this shows, at some level, how Python works with processes and the basic ways in which it communicates with them, using bytestrings and pipes. That's because processes are separate and cannot simply share variables with the main thread, which is what you're doing when using threads.

In my next article, I'll discuss how you can break problems apart in a thread-like fashion using the "multiprocessing" module. That has the advantage of opening new processes for each task you want to accomplish while giving you a thread-like interface to do so.■

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

RETURN TO CONTENTS

A Number-Guessing Game

Guess a number—Dave writes a simple guessing game as a demonstration of how to produce clear, readable shell scripts and solve mathematical equations.

PREVIOUS
Reuven M. Lerner's
At the Forge

NEXT
Kyle Rankin's
Hack and /

DAVE TAYLOR

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

THERE ARE SOME BASIC COMPUTER ALGORITHMS THAT SUGGEST GAMES, weird as that may sound. One example leaps right to mind: search as a strategy for a process of elimination for a guessing game. The game *Mastermind* is based on that, with its colored pegs and oft-confusing feedback mechanism for you to (hopefully) zero in on the secret sequence.

Let's go simpler than that though. Let's implement a number-guessing game as a way to learn about binary search. The concept is easy. If you have to guess a number between 1 and "n", each guess should focus on dividing the remaining pool of possible values in half.

Your first guess might be 50. If it's too low, you just chopped out 51 of the 100 possible values (1..49 and the value 50 itself, since it wasn't a match). If it's too high, same again, but now you know it must be in the sequence 1..49.

There's math behind this actually, and it revolves around logarithms. Yes, I can't believe we're talking about logs, but the formula to calculate the worst-case number of guesses is log2(LISTSIZE).

For example, with a list of 100 entries: log2(100) = 6.644. Since you can't have a fractional guess, of course, that means the computer never should guess more than seven times to identify any randomly chosen number 1..100. And, who knows, it might guess it a lot faster than that. Imagine if 50 was the randomly chosen value, for example. That should be guessed in, well, one guess.

To calculate this value in Linux, the `bc` binary calculator is the tool for the job. Unfortunately, it doesn't know how to calculate base-2 logarithms, but math to the rescue! log2(N) is equal to log(N) / log(2). You knew that, right?

That's a formula you can feed to `bc`, although as one of the oldest Linux programs, `bc` is famously un-user-friendly. In fact, here's how I use this formula along with the usual `bc` rigmarole to get a solution for "size":

```
echo "scale=4;(l($LISTSIZE)/l(2))" | bc -l
```

`bc` is weird in that if you specify a scale of zero, it won't calculate any values after the decimal point for any interim calculations either. The result: log(2) = 0, and the equation fails with a divide-by-zero error. D'oh.

To turn this formula into a usable calculation for a script, you need to keep in mind that any value > 0.000, even 0.0001, must round upward for maximum/worst-case guess. That's a ceiling function, as they say in mathematics, but `bc` doesn't have that either. Instead, here's a hack: simply add 0.999 to the resultant value and chop off everything after the decimal point.

It works. Try it.

Here's my resultant formula, all ready for a shell script:

```
steps="$(echo "scale=4;(l($LISTSIZE)/l(2)+0.99)" | bc -l \
       | cut -d. -f1)"
```

That's actually probably the hardest part of this program. The other piece is to choose a value randomly between 1..n for some value of "n", but that's a breeze:

```
value=$(( ( $RANDOM % $LISTSIZE ) + 1 ))
```

The main loop consists of prompting the user for a value, then indicating whether it's a match (well done!), too high or too low. Then looping and prompting them again. It's pretty simple, actually, and you hopefully should be able to code it all by yourself without ever reading further into this column.

Still here?

Okay, let's proceed.

You'll recall that `echo -n` omits the carriage return at the end of a line, so the sequence of:

```
echo -n "Enter something: "
read userinput
```

is quite a common one in interactive shell scripts. This will be no different, prompting like this:

```
echo -n "Your guess: "
read playerguess
```

You'll notice that one of the other things I'm demonstrating in this particular shell script is the use of long, mnemonic variable names. Too many scripts have "i" and "j" and "k" as variables without ever explaining what they do or what they represent. That's just bad coding.

There's a fun, classic way to create an infinite loop in a shell script, and that's what I'll do with this number-guessing game too:

```
while [ /bin/true ]
do
   statements
done
```

Within the loop, end conditions must be checked, and when met, the exit command easily can be used to quit the script. Just want to quit the loop? That's what "break" can do for you in this sort of situation. Simply specify how many levels of loop you want to jump out of as part of the break statement if one isn't enough.

That's now enough that you can figure out what I'm doing, I expect:

```
pick a number
while looping
   ask for guess
   if guess = number
      you got it.
   if guess < number
      too low, guess higher
   else guess > number
      too high, guess lower
loop
```

What does that look like as a shell script? Hey, I thought you'd never ask:

```
while [ /bin/true ] ; do
  /bin/echo -n "Your guess: "
  read playerguess
  if [ $playerguess -eq $value ] ; then
    echo -n "Got it! Nice. That took you $guess guesses."
    steps="$(echo "scale=4;(l($max)/l(2)+0.99)" | bc -l \
           | cut -d. -f1)"
    echo "I can solve it in less than $steps steps."
    exit 0
  elif [ $playerguess -lt $value ] ; then
    echo "Nope. Too low."
  else
    echo "Nah. Too high."
  fi
  guess=$(( $guess + 1 ))          # another guess...
done
```

You can see that I've added a guess counter, ingeniously called `guess`, and that the player guess is, well, `playerguess`. This makes the code nice and readable, although the mathematical equation tucked into the middle is a bit gnarly looking by comparison. It definitely could do with a comment to make it more clear.

The game gets a bit more interesting if the user can specify a list size when invoking the program, which easily can be done by having `LISTSIZE` (okay, I call it "max") as a variable instead of hard-coded:

```
max=100         # maxvalue

if [ $# -gt 0 ] ; then
  max=$1
fi
```

I probably should check that the value is indeed an integer, but I'll leave that task as an exercise for you, the reader.

A few tweaks to the prompts and output make it a bit friendlier and more grammatically correct. Here's a test:

```
$ guess-number.sh 50

Guess my number between 1 and 50. Ready? Go!
Guess #1 is: 25
Nah. Too high.
Guess #2 is: 20
Nah. Too high.
Guess #3 is: 10
```

```
Nah. Too high.
Guess #4 is: 5
Nope. Too low.
Guess #5 is: 6
Nope. Too low.
Guess #6 is: 7
Got it! Nice. That took you 6 guesses.
For the record, I would have solved it no more than 6 steps.
```

I deliberately did a pretty inefficient search. In fact, each time, you should divide the remaining values by two and make that your guess, so when you learned 25 was too high, the next guess should have been 12.5 (or 12) and so on. Still, this did no worse than the worst-case scenario of six guesses.

If you really want to do something interesting, note how by using this strategy, you could guess from a list of `MAXINT` values (32,767) and still have no more than 16 guesses, worst case, to figure out a randomly chosen number in the range 1..MAXINT.

Ah, the power of math and the joy of binary search algorithms.■

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

RETURN TO CONTENTS

Lightning Hacks: Qubes Tips

Learn a few tips to get the most out of your Qubes desktop.

◀ PREVIOUS
Dave Taylor's
Work the Shell

NEXT
Shawn Powers'
The Open-Source
Classroom ▶

KYLE RANKIN

Kyle Rankin is VP of engineering operations at Final, Inc., the author of many books including *Linux Hardening in Hostile Networks*, *DevOps Troubleshooting* and *The Official Ubuntu Server Book*, and a columnist for *Linux Journal*. Follow him @kylerankin.

EVERY SO OFTEN I WRITE A LIGHTNING HACKS PIECE IN THIS SPACE. The idea behind it is to gather up some tips that wouldn't be enough to fill out a full article but that are useful nonetheless. The idea is based on lightning talks you'll see at conferences—ten-minute talks so the speaker can present various ideas that wouldn't take up a full 50-minute slot on their own.

I've been using the high-security Qubes operating system for quite some time now, and I wrote a multipart series for *Linux Journal* in the past. While I've been using it, I've gathered a few useful tips for it, and in this article, I cover a few tips specifically tailored for Qubes. Even though these tips are for Qubes and assume a desktop full of VMs, you could adapt the overall ideas to other desktop environments.

Clock In, Clock Out

Generally speaking, it's a good idea to separate your personal and work environments completely on different machines. It's better for security, because if your personal machine gets hacked, you don't risk infecting your work environment and vice versa. Of course, if for some reason you don't have the luxury of two machines, or if you want to set up a travel laptop that's configured both with your work and personal settings (like I've mentioned in prior articles), you'll want some way to switch between work and personal modes.

Because Qubes does everything through many different VMs, this means writing a simple pair of scripts, clock_in and clock_out, that are stored in the dom0 VM. Both scripts define a list of personal and work VMs, and they will shut down or start up VMs depending on whether you are clocking in or clocking out. Here's an example clock_in script:

```
#!/bin/bash

PERSONAL_VMS="fb personal personal-web vault finance
 ➥writing sys-whonix"
WORK_VMS="work work-web stage prod1 prod2 vault-work"

for i in $PERSONAL_VMS; do qvm-shutdown $i; done
for i in $WORK_VMS; do qvm-start $i; done
```

Compare this to my clock_out script, and you'll see that the list of VMs is different:

```
#!/bin/bash

PERSONAL_VMS="fb personal personal-web vault"
WORK_VMS="work work-web stage prod1 prod2 vault-work stage-gpg
 ➥prod-gpg sys-vpn-stage sys-vpn-prod1 sys-vpn-prod2"

for i in $PERSONAL_VMS; do qvm-shutdown $i; done
for i in $WORK_VMS; do qvm-start $i; done
```

The reason the list is different is that in both cases I want to be comprehensive in the VMs I shut down, but need only particular VMs to start up when I clock in or out. By creating separate lists, I can make sure all the VMs that might be running are all shut down, and I start only the VMs I need.

VM Selector for URLs

One great thing about Qubes is that if you get a questionable file or URL in one VM, you can open it in a less-trusted or disposable VM. Typically when it comes to URLs though, this means going through Qubes's more-secure copy-and-paste method, which takes twice the number of keystrokes as a normal copy and paste. I realized that I commonly wanted to open a URL from a more-trusted VM in a certain list of less-trusted ones, so I created the following script called vm_picker that pops up a simple GUI selector I can use to choose the VM with which I want to open a URL:

```
#!/bin/bash

VM=$(zenity --list --title 'Open in which VM?' --column='VM Name' \
  untrusted \
  dispVM \
  personal-web \
)

if [ "$VM" == "dispVM" ]; then
  qvm-open-in-dvm $@
else
  qvm-open-in-vm $VM $@
fi
```

In this script, I've defined three different VMs, my completely untrusted one I use for normal web browsing, a disposable VM for particularly risky VMs and my personal-web VM that I use for more trusted authenticated sessions. The script uses zenity, which is a handy command-line tool you can use to display basic GUI elements—in this case, a list. Once you select the VM, zenity assigns it to the VM variable, and if it's a disposable VM, I use a special Qubes command for disposable VMs. If it's any other VM, I use a separate one.

Save the script in any VMs from which you want to open URLs, and then go into that VM's Preferred Applications program (you may have to update the list of visible shortcuts for that VM to see this program). Set this script as your preferred web browsing application and then all of your right-click "Open URL" dialogs in terminals or other web-aware programs will use your VM picker.

Obviously, you'll want to customize the script to present your own VMs and in your preferred order. I find I have a different list and order depending on which VM I call it from, so each VM has a slightly different version of the script. I also set up a custom version for the KeePassX program that runs in my vault, because it allows you to specify a URL assigned to a particular user name and password, and you can tell it to open the URL with a keybinding. Buried in the KeePassX settings is a setting that allows you to define a custom URL handler, so I set it to my VM-picker script.

Conclusion

So if you use Qubes, I hope you've found these tips to be handy. If you don't use Qubes, you still could adapt these ideas to your desktop. For instance, you simply could change the clock_in and clock_out scripts to shut down and launch specific programs (or programs with specific modes set). Instead of the vm_picker script choosing specific VMs, you could change it so that it allows you to pick between your different web browsers so you can open some URLs in Firefox and others in Chrome. You even could inspect the URL ahead of time and automatically launch a particular browser without a prompt at all.■

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

RETURN TO CONTENTS

Ansible, Part IV: Putting It All Together

Roles are the most complicated and yet simplest aspect of Ansible to learn.

◀ PREVIOUS
Kyle Rankin's
Hack and /

NEXT
New Products ▶

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. Or, swing by the #linuxjournal IRC channel on Freenode.net.

I'VE MENTIONED BEFORE THAT ANSIBLE'S AD-HOC MODE OFTEN IS OVERLOOKED AS JUST A WAY TO LEARN HOW TO USE ANSIBLE. I couldn't disagree with that mentality any more fervently than I already do. Ad-hoc mode is actually what I tend to use most often on a day-to-day basis. That said, using playbooks and roles are very powerful ways to utilize Ansible's abilities. In fact, when most people think of Ansible, they tend to think of the roles feature, because it's the way most Ansible code is shared. So first, it's important to understand the relationship between ad-hoc mode, playbooks and roles.

Ad-hoc Mode

This is a bit of a review, but it's easy to forget once you start creating playbooks. Ad-hoc mode is simply a one-liner that uses an Ansible module to accomplish a given task on a set of computers. Something like:

```
ansible cadlab -b -m yum -a "name=vim state=latest"
```

will install vim on every computer in the cadlab group. The `-b` signals to elevate privilege ("become" root), the `-m` means to use the `yum` module, and the `-a` says what actions to take. In this case, it's installing the latest version of vim.

Usually when I use ad-hoc mode to install packages, I'll follow up with something like this:

```
ansible cadlab -b -m service -a "name=httpd state=started
 ➥enabled=yes"
```

That one-liner will make sure that the httpd service is running and set to start on boot automatically (the latter is what "enabled" means). Like I said at the beginning, I most often use Ansible's ad-hoc mode on a day-to-day basis. When a new rollout or upgrade needs to happen though, that's when it makes sense to create a playbook, which is a text file that contains a bunch of Ansible commands.

Playbook Mode

I described playbooks in my last article. They are YAML- (Yet Another Markup Language) formatted text files that contain a list of things for Ansible to accomplish. For example, to install Apache on a lab full of computers, you'd create a file something like this:

```
---

- hosts: cadlab
  tasks:
  - name: install apache2 on CentOS
    yum: name=httpd state=latest
    notify: start httpd
    ignore_errors: yes

  - name: install apache2 on Ubuntu
    apt: update_cache=yes name=apache2 state=latest
    notify: start apache2
    ignore_errors: yes

  handlers:
  - name: start httpd
    service: name=httpd enable=yes state=started

  - name: start apache2
    service: name=apache2 enable=yes state=started
```

Mind you, this isn't the most elegant playbook. It contains a single play that tries to install httpd with yum and apache2 with apt. If the lab is a mix of CentOS and Ubuntu machines, one or the other of the installation methods will fail. That's why the `ignore_errors` command is in each task. Otherwise, Ansible would quit when it encountered an error. Again, this method works, but it's not pretty. It would be much better to create conditional statements that would allow for a graceful exit on incompatible platforms. In fact, playbooks that are more complex and do more things tend to evolve into a "role" in Ansible.

Roles

Roles aren't really a mode of operation. Actually, roles are an integral part of playbooks. Just like a playbook can have tasks, variables and handlers, they can also have roles. Quite simply, roles are just a way to organize the various components referenced in playbooks. It starts with a folder layout:

```
roles/
  webserver/
    tasks/
      main.yml
    handlers/
```

```
      main.yml
    vars/
      main.yml
    templates/
      index.html.j2
      httpd.conf.j2
    files/
      ntp.conf
```

Ansible looks for a roles folder in the current directory, but also in a system-wide location like /etc/ansible/roles, so you can store your roles to keep them organized and out of your home folder. The advantage of using roles is that your playbooks can look as simple as this:

```
---

- hosts: cadlab
  roles:
    - webserver
```

And then the "webserver" role will be applied to the group "cadlab" without needing to type any more information inside your playbook. When a role is specified, Ansible looks for a folder matching the name "webserver" inside your roles folder (in the current directory or the system-wide directory). It then will execute the tasks inside webserver/tasks/main.yml. Any handlers mentioned in that playbook will be searched for automatically in webserver/handlers/main.yml. Also, any time files are referenced by a template module or file/copy module, the path doesn't need to be specified. Ansible automatically will look inside webserver/files/ or /webserver/templates/ for the files.

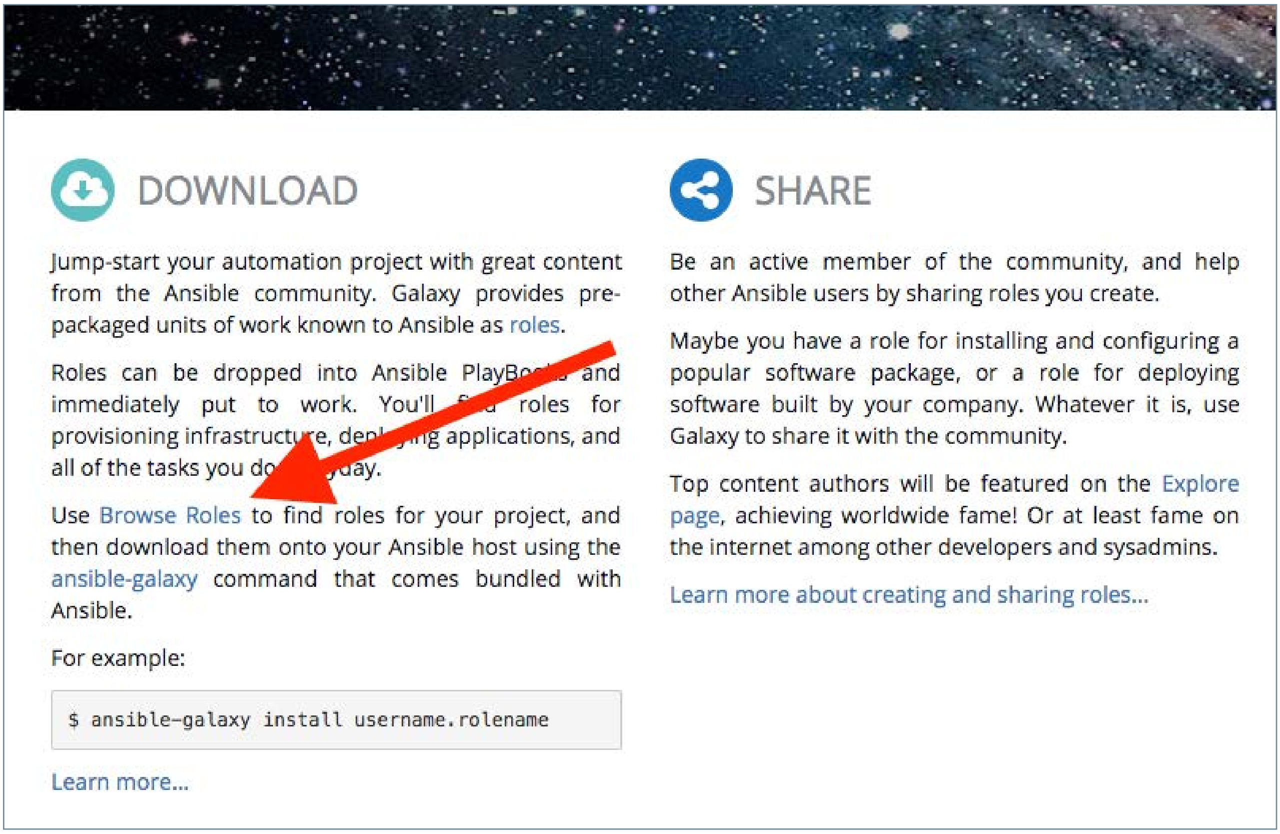
Basically, using roles will save you lots of path declarations and include statements. That might seem like a simple thing, but the organization creates a standard that not only makes it easy to figure out what a role does, but also makes it easy to share your code with others. If you always know any files must be stored in roles/rolename/files/, it means you can share a "role" with others and they'll know exactly what to do with it—namely, just plop it in their own roles folder and start using it.

Sharing Roles: Ansible Galaxy

One of the best aspects of current DevOps tools like Chef, Puppet and Ansible is that there is a community of people willing to share their hard work. On a small scale, roles are a great way to share with your coworkers, especially if you have roles that are customized specifically for your environment. Since many of environments are similar, roles can be shared with an even wider audience—and that's where Ansible Galaxy comes into play.

I'll be honest, part of the draw for me with Ansible is the sci-fi theme in the naming convention. I know I'm a bit silly in that regard, but just naming something Ansible or Ansible Galaxy gets my attention. This might be one of those "built by nerds, for nerds" sort of things. I'm completely okay with that. If you head over to https://galaxy.ansible.com, you'll find the online repository for shared roles—and there are a ton.

For simply downloading and using other people's roles, you don't need



DOWNLOAD

Jump-start your automation project with great content from the Ansible community. Galaxy provides pre-packaged units of work known to Ansible as roles.

Roles can be dropped into Ansible PlayBooks and immediately put to work. You'll find roles for provisioning infrastructure, deploying applications, and all of the tasks you do everyday.

Use Browse Roles to find roles for your project, and then download them onto your Ansible host using the ansible-galaxy command that comes bundled with Ansible.

For example:

```
$ ansible-galaxy install username.rolename
```

Learn more...

SHARE

Be an active member of the community, and help other Ansible users by sharing roles you create.

Maybe you have a role for installing and configuring a popular software package, or a role for deploying software built by your company. Whatever it is, use Galaxy to share it with the community.

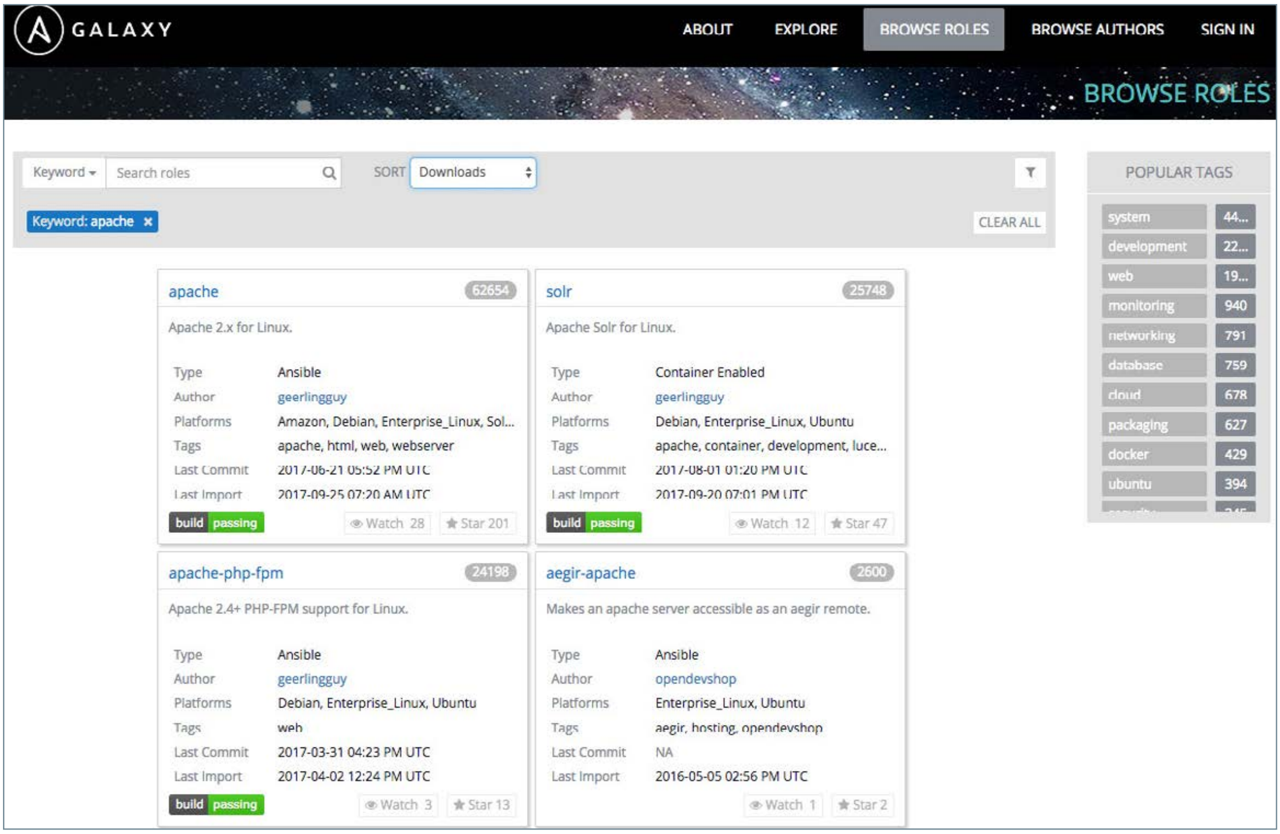
Top content authors will be featured on the Explore page, achieving worldwide fame! Or at least fame on the internet among other developers and sysadmins.

Learn more about creating and sharing roles...

Figure 1. Click that link to browse and search for roles.

any sort of account on Ansible Galaxy. You can search on the website by going to https://galaxy.ansible.com and clicking "Browse Roles" on the left side of the page (Figure 1). There are more than 13,000 roles currently uploaded to Ansible Galaxy, so I highly recommend taking advantage of the search feature! In Figure 2, you'll see I've searched for "apache" and sorted by "downloads" in order to find the most popular roles.

Many of the standard roles you'll find that are very popular are written by Jeff Geerling, whose user name is geerlingguy. He's an Ansible developer who has written at least one Ansible book that I've read and possibly others. He shares his roles, and I encourage you to check them out—not only for using them, but also for seeing how he codes around issues like conditionally choosing the correct module for a given distribution and things like that. You can click on the role name and see all the code involved. You might notice that if you want to examine the code, you need to click on the GitHub link. That's one of the genius moves of Ansible Galaxy—all roles are stored on a user's GitHub page as opposed to an Ansible Galaxy server. Since most developers keep their



SORT Downloads

Keyword: apache ×

apache 62654

Apache 2.x for Linux.

Type Ansible
Author geerlingguy
Platforms Amazon, Debian, Enterprise_Linux, Sol...
Tags apache, html, web, webserver
Last Commit 2017-06-21 05:52 PM UTC
Last Import 2017-09-25 07:20 AM UTC

build passing · Watch 28 · Star 201

solr 25748

Apache Solr for Linux.

Type Container Enabled
Author geerlingguy
Platforms Debian, Enterprise_Linux, Ubuntu
Tags apache, container, development, luce...
Last Commit 2017-08-01 01:20 PM UTC
Last Import 2017-09-20 07:01 PM UTC

build passing · Watch 12 · Star 47

apache-php-fpm 24198

Apache 2.4+ PHP-FPM support for Linux.

Type Ansible
Author geerlingguy
Platforms Debian, Enterprise_Linux, Ubuntu
Tags web
Last Commit 2017-03-31 04:23 PM UTC
Last Import 2017-04-02 12:24 PM UTC

build passing · Watch 3 · Star 13

aegir-apache 2600

Makes an apache server accessible as an aegir remote.

Type Ansible
Author opendevshop
Platforms Enterprise_Linux, Ubuntu
Tags aegir, hosting, opendevshop
Last Commit NA
Last Import 2016-05-05 02:56 PM UTC

Watch 1 · Star 2

Figure 2. Jeff Geerling's roles are always worth checking out.

code on GitHub, they don't need to remember to upload to Ansible Galaxy as well.

Incidentally, if you ever desire to share your own Ansible roles, you'll need to use a GitHub user name to upload them, because again, roles are all stored on GitHub. But that's getting ahead of things; first you need to learn how to use roles in your environment.

Using ansible-galaxy to Install Roles

It's certainly possible to download an entire repository and then unzip the contents into your roles folder. Since they're just text files and structured folders, there's not really anything wrong with doing it that way. It's just far less convenient than using the tools built in to Ansible.

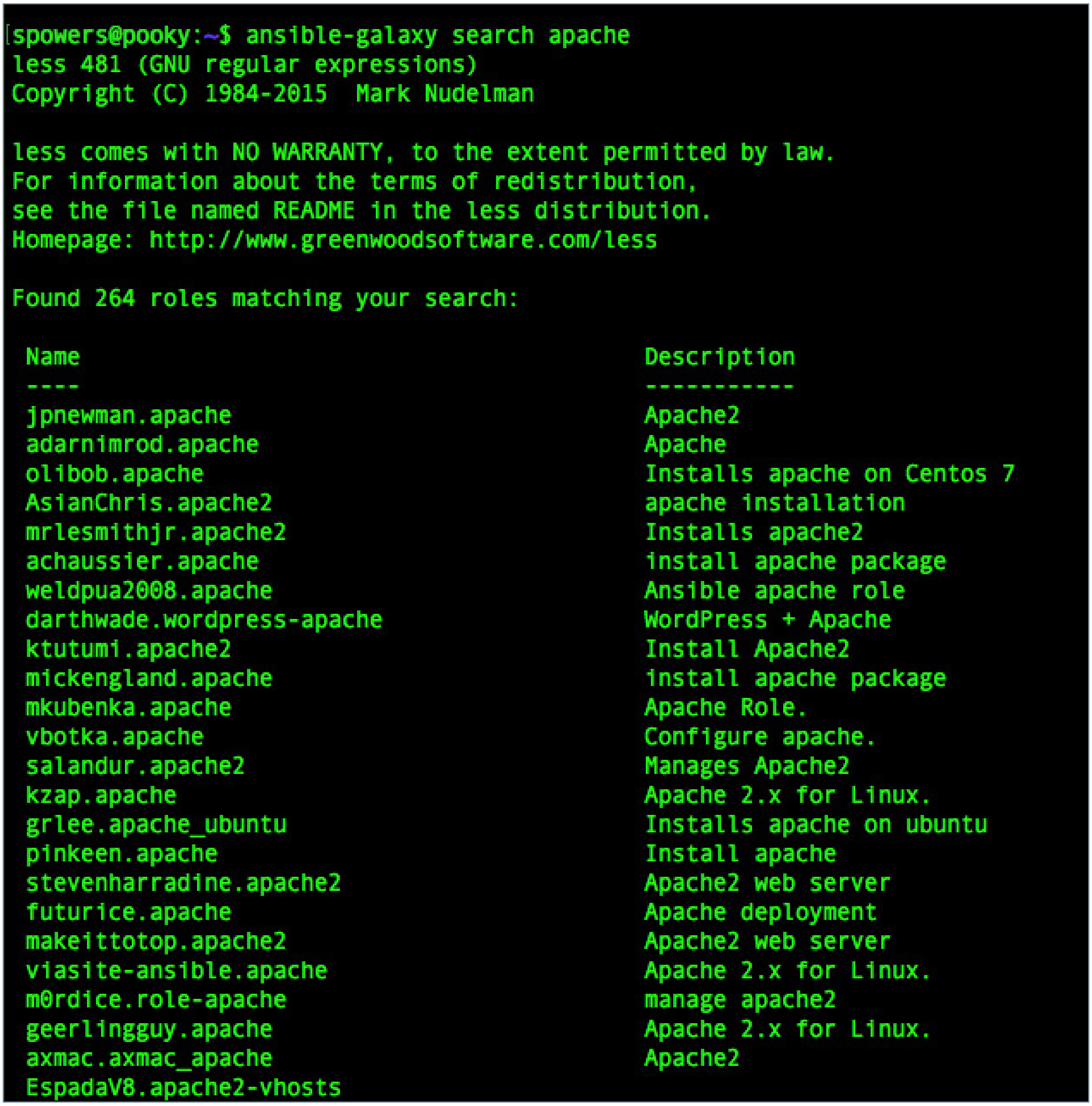
There is a search mechanism on the Ansible command line for searching the Ansible Galaxy site, but in order to find a role I want to use, I generally go to the website and find it, then use the command-line tools to download and install it. Here's an example of Jeff Geerling's "apache" role. In order to use Ansible to download a role, you need to do this:

```
sudo ansible-galaxy install geerlingguy.apache
```

Notice two things. First, you need to execute this command with root privilege. That's because the `ansible-galaxy` command will install roles in your system-wide roles folder, which isn't writable (by default) by your regular user account. Second, take note of the format of roles named on Ansible Galaxy. The format is username.rolename, so in this case, geerlingguy.apache, which is also how you reference the role inside your playbooks.

If you want to see roles listed with the correct format, you can use `ansible-galaxy`'s search command, but like I said, I find it less than useful because it doesn't sort by popularity. In fact, I can't figure out what it sorts by at all. The only time I use the command-line search feature is if I also use `grep` to narrow down roles by a single person. Anyway, Figure 3 shows what the results of `ansible-galaxy` search look like. Notice the username.rolename format.

Once you install a role, it is immediately available for you to use in your own playbooks, because it's installed in the system-wide roles

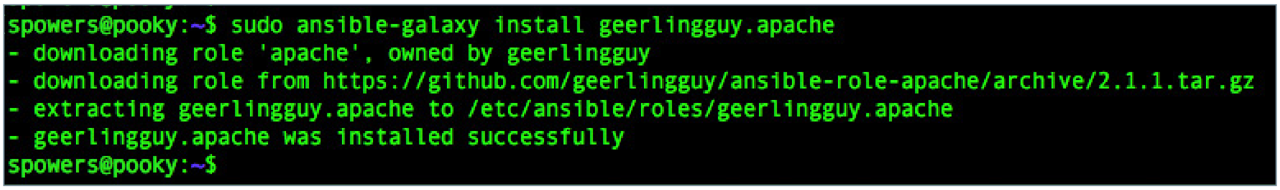


```
spowers@pooky:~$ ansible-galaxy search apache
less 481 (GNU regular expressions)
Copyright (C) 1984-2015  Mark Nudelman

less comes with NO WARRANTY, to the extent permitted by law.
For information about the terms of redistribution,
see the file named README in the less distribution.
Homepage: http://www.greenwoodsoftware.com/less

Found 264 roles matching your search:

 Name                                       Description
 ----                                       -----------
 jpnewman.apache                            Apache2
 adarnimrod.apache                          Apache
 olibob.apache                              Installs apache on Centos 7
 AsianChris.apache2                         apache installation
 mrlesmithjr.apache2                        Installs apache2
 achaussier.apache                          install apache package
 weldpua2008.apache                         Ansible apache role
 darthwade.wordpress-apache                 WordPress + Apache
 ktutumi.apache2                            Install Apache2
 mickengland.apache                         install apache package
 mkubenka.apache                            Apache Role.
 vbotka.apache                              Configure apache.
 salandur.apache2                           Manages Apache2
 kzap.apache                                Apache 2.x for Linux.
 grlee.apache_ubuntu                        Installs apache on ubuntu
 pinkeen.apache                             Install apache
 stevenharradine.apache2                    Apache2 web server
 futurice.apache                            Apache deployment
 makeittotop.apache2                        Apache2 web server
 viasite-ansible.apache                     Apache 2.x for Linux.
 m0rdice.role-apache                        manage apache2
 geerlingguy.apache                         Apache 2.x for Linux.
 axmac.axmac_apache                         Apache2
 EspadaV8.apache2-vhosts
```

Figure 3. I love the command line, but these search results are frustrating.

folder. In my case, that's /etc/ansible/roles (Figure 4). So now, if I create a playbook like this:

```
---
- hosts: cadlab
  roles:
    - geerlingguy.apache
```



```
spowers@pooky:~$ sudo ansible-galaxy install geerlingguy.apache
- downloading role 'apache', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-apache/archive/2.1.1.tar.gz
- extracting geerlingguy.apache to /etc/ansible/roles/geerlingguy.apache
- geerlingguy.apache was installed successfully
spowers@pooky:~$
```

Figure 4. Easy Peasy, Lemon Squeezy

Apache will be installed on all my cadlab computers, regardless of what distribution they're using. If you want to see how the role (which is just a bunch of tasks, handlers and so forth) works, just pick through the folder structure inside /etc/ansible/roles/geerlingguy.apache/. It's all right there for you to use or modify.

Creating Your Own Roles

There's really no magic here, since you easily can create a roles folder and then create your own roles manually inside it, but `ansible-galaxy` does give you a shortcut by creating a skeleton role for you. Make sure you have a roles folder, then just type:

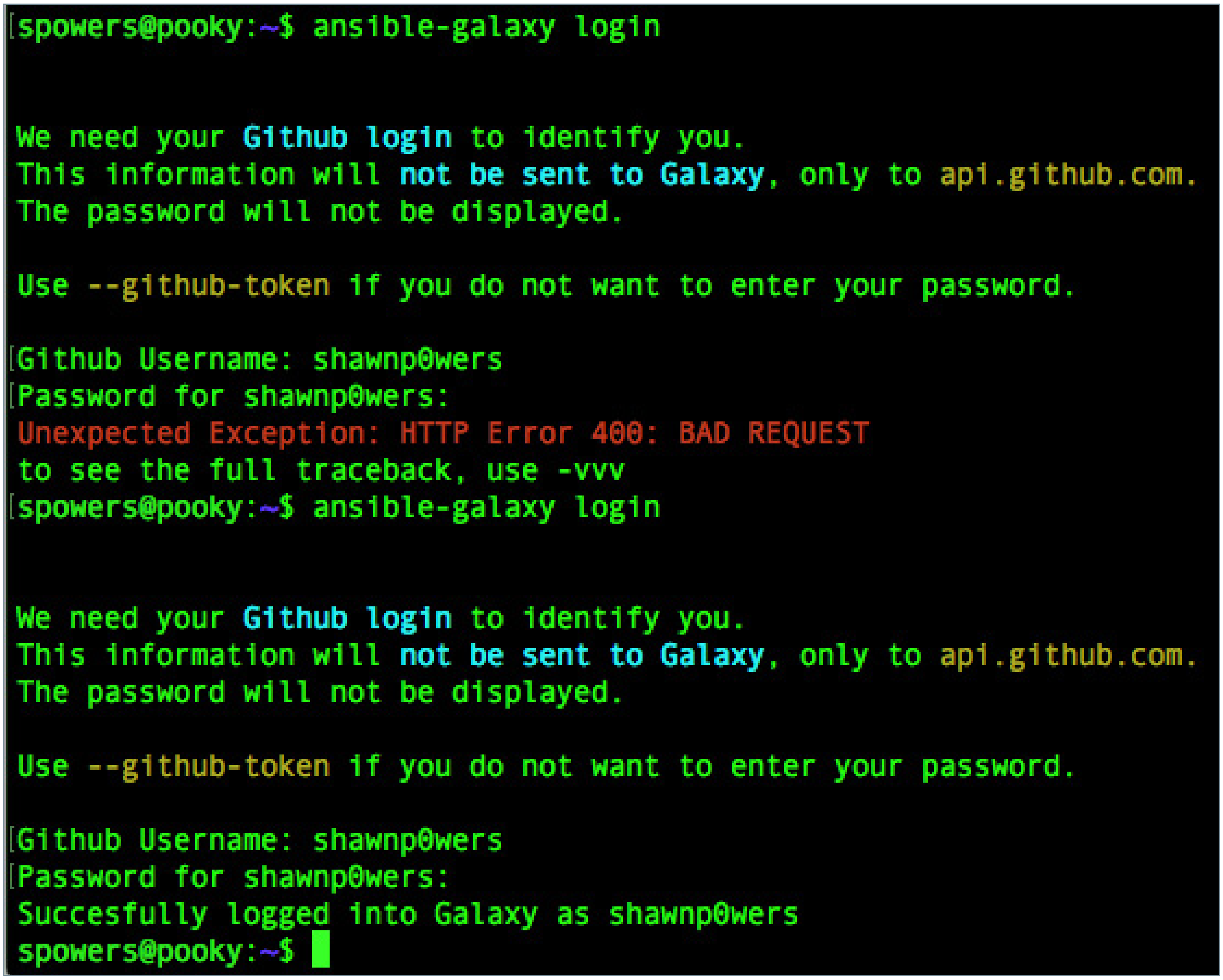
```
ansible-galaxy init roles/rolename
```

and you'll end up with a nicely created folder structure for your new role. It doesn't do anything magical, but as someone who has misspelled "Templates" before, I can tell you it will save you a lot of frustration if you have clumsy fingers like me.

Sharing Your Roles

If you get to the point where you want to share you roles on Ansible Galaxy, it's fairly easy to do. Make sure you have your role on GitHub (using git is beyond the scope of this article, but using git and GitHub is a great way to keep track of your code anyway). Once you have your roles on GitHub, you can use `ansible-galaxy` to "import" them into the publicly searchable Ansible Galaxy site. You first need to authenticate:

```
ansible-galaxy login
```



```
[spowers@pooky:~$ ansible-galaxy login


 We need your Github login to identify you.
 This information will not be sent to Galaxy, only to api.github.com.
 The password will not be displayed.

 Use --github-token if you do not want to enter your password.

[Github Username: shawnp0wers
[Password for shawnp0wers:
 Unexpected Exception: HTTP Error 400: BAD REQUEST
 to see the full traceback, use -vvv
[spowers@pooky:~$ ansible-galaxy login


 We need your Github login to identify you.
 This information will not be sent to Galaxy, only to api.github.com.
 The password will not be displayed.

 Use --github-token if you do not want to enter your password.

[Github Username: shawnp0wers
[Password for shawnp0wers:
 Succesfully logged into Galaxy as shawnp0wers
 spowers@pooky:~$ █
```

Figure 5. It drove me nuts trying to figure out why I couldn't authenticate.

Before you try to log in with the command-line tool, be sure you've visited the Ansible Galaxy website and logged in with your GitHub account. You can see in Figure 5 that at first I was unable to log in. Then I logged in on the website, and after that, I was able to log in with the command-line tool successfully.

Once you're logged in, you can add your role by typing:

```
ansible-galaxy import githubusername githubreponame
```

The process takes a while, so you can add the `-no-wait` option if you want, and the role will be imported in the background. I really don't recommend doing this until you have created roles worth sharing. Keep in mind, there are more than 13,000 roles on Ansible Galaxy, so there are many "re-inventions of the wheel" happening.

From Here?

Well, it's taken me four articles, but I think if you've been following along, you should be to the point where you can take it from here. Playbooks and roles are usually where people focus their attention in Ansible, but I also encourage you to take advantage of ad-hoc mode for day-to-day maintenance tasks. Ansible in some ways is just another DevOps configuration management tool, but for me, it feels the most like the traditional problem-solving solution that I used Bash scripts to accomplish for decades. Perhaps I just like Ansible because it thinks the same way I do. Regardless of your motivation, I encourage you to learn Ansible enough so you can determine whether it fits into your workflow as well as it fits into mine.■

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

RETURN TO CONTENTS

```
 *
 * Usage: $(obj).insertAtCursor(text);
 *   obj = a textarea or textfield
 *   text = a string to insert
 */
$.fn.extend({
  insertAtCaret: function(myValue){
    return $(this).each(function(){
      // If target element is hidden, don't do it.
      if ($(this).is(':hidden') || $(this).parents(':hidde
        return;
      }
      if (document.selection) {
        this.fo
        sel = d
        sel.tex
        this.fo
      }
      else if (
        var sta
        var end
        var scr
        this.va
        this.fo
        // High
```

PREVIOUS
Shawn Powers'
The Open-Source
Classroom

NEXT
Feature: Rapid,
Secure Patching:
Tools and Methods

DivvyCloud Platform for VMware Cloud on AWS

DivvyCloud's unique niche in the IT ecosystem is helping organizations automate and manage their multi-cloud infrastructure at scale. The latest innovation from the company is the DivvyCloud Platform for VMware Cloud on AWS, a solution enabling consistent policy enforcement and automation of cloud best practices to customers of VMware Cloud on Amazon Web Services (AWS). VMware Cloud on AWS unites VMware's enterprise-class Software-Defined Data Center (SDDC) software together with the elastic, bare-metal infrastructure from AWS, which results in a consistent operating model and application mobility for the private and public cloud. DivvyCloud maintains that its software is unique in the marketplace, due to its ability to track real-time changes across clouds and take customer-defined, autonomous action to fix problems and ensure policy compliance. Standard automation bots proactively address myriad security, cost and compliance challenges commonly faced by any organization adopting or expanding cloud-based infrastructure. The visibility and policy automation afforded by the DivvyCloud solution remediate security, cost and compliance issues, adds the firm.
https://divvycloud.com

NETGEAR 48-Port Gigabit Smart Managed Plus Switch (GS750E)

More than ever, small to mid-sized businesses demand and rely on their networks to carry out mission-critical business activities. As always, however, budgets and expertise constrain these companies from using complex managed switches to run their networks. Extending a hand to assist is NETGEAR, Inc., whose new NETGEAR 48-port Gigabit Smart Managed Plus Switch (GS750E) provides an easy, reliable and affordable connectivity solution for expanding networks for workstations/servers, Network Attached Storage (NAS) and PCs. NETGEAR's "industry-first" GS750E 48-port switch is designed to meet current and future needs of any IP network, enabling network optimization and eliminating bottlenecks and featuring a leading speed/affordability ratio. The device, with its convenient web-based management, further helps companies in need of network intelligence to separate and prioritize voice and video traffic from data to support applications, such as VoIP phones and IP cameras, on its Ethernet infrastructure. The fanless GS750E supports VLAN, QoS, LAG and IGMP management capabilities and includes a full set of configurable advanced L2 features, such as traffic prioritization and link aggregation.
https://www.netgear.com

Zentera Systems, Inc.'s CoIP Security Enclave

On the heels of being crowned "Cool Vendor in Cloud Security" by Gartner, Zentera Systems, Inc., announced an upgrade to its flagship CoIP Security Enclave solution. The solution enables enterprises to specify their micro-segmentation policies, which the Enclave software automatically converts into application-aware segmentation rules that protect application workloads in unified virtual overlay networks called "enclaves". Those workloads can be running anywhere, including on-premises and across any cloud, hybrid and multicloud environments. This new release extends the flagship application with CoIP Smart Discovery capability, which self-scrutinizes workload behavior to uncover existing application compute flows and behavior. Based on this intel, enterprise IT security teams then can complete micro-segmentation definitions and find any potential gaps in their segmentation implementation quickly. Such intelligent automation saves teams considerable time and effort, especially in a hybrid environment where numerous applications and workloads are combined. The Smart Discovery functionality is fully integrated with CoIP Application Interlock, an existing security capability that allows companies to specify which authorized applications in a specific CoIP enclave are allowed to access the enclave's network. All other applications are locked out, greatly enhancing enclave security. With CoIP Secure Enclave, says Zentera, a hybrid or cloud environment is no different from on-premises, and enterprises maintain complete control over connectivity and security to implement one unified security policy across all environments.
https://zentera.net

IBM's LinuxONE Enterprise System

Plentiful new capabilities are to be found in the newly unveiled next generation of the IBM LinuxONE Enterprise System, which Big Blue describes as "the industry's most advanced enterprise Linux platform". The new system provides capabilities that will boost the security of popular open-source-based container technologies like Docker and Kubernetes significantly, thanks to IBM Secure Service Container technology running on LinuxONE. Applications running in a container solution take on the security capabilities of Secure Service Container without any change to the software. These new features remove the burden of building security into applications, allowing developers to spend their time innovating instead. IBM LinuxONE Secure Service Containers provide applications significant protection against external and insider threats, including automatic encryption of data in-flight and at-rest, and tamper-resistance during installation and runtime. Also new to the platform is the LinuxONE Emperor II, based on IBM Z mainframe technology and featuring the industry's fastest microprocessor, running at 5.2GHz, and a highly engineered, scalable system structure. LinuxONE Emperor II can support the following: a 17TB MongoDB Enterprise instance in a single system with up to 10x better read/write latency than an x86-based implementation; certified infrastructure for Docker EE with integrated management and scale tested up to two million containers; vertical scale to 170 cores and industry-leading performance for Java workloads; integrated pause-less garbage collection and up to 2.6x better performance than x86 alternatives.
https://www.ibm.com/linuxone

Murat Yener and Onur Dundar's *Expert Android Studio* (Wrox Press)

Join the discussion @ p2p.wrox.com
Wrox Programmer to Programmer™
wrox
Expert
Android Studio
Murat Yener, Onur Dundar

Unleashing the potential of Android Studio is what developers can accomplish with Wrox Press' new book *Expert Android Studio*, states the tech publisher. This new resource from self-professed Android geeks Murat Yener and Onur Dundar, both based at Intel, plugs the holes in one's Android programing skills on the provided tools including Android Studio, NDK, Gradle and Plugins for IntelliJ Idea Platform. Filled with best practices, advanced techniques and tips on Android tools, development cycle, continuous integration, release management, testing and performance, this book provides professional guidance to experienced developers who want to go beyond the ordinary with the Android platform's developer tools. Readers of *Expert Android Studio* will master topics like the basics of working in Android Studio and Gradle, the application architecture of the latest Android platform, the Native Development Kit and its integration with Android Studio, the development lifecycle and both Gradle and custom plugins.
http://www.wrox.com/WileyCDA

vCISO Services, LLC's CISO as a Service

The bad guys know about and are exploiting the growing rift between the security officer "haves" and "have nots". They target smaller businesses because they know that their information security programs are not as strong as the big companies' are. To level the playing field, vCISO Services, LLC, has begun offering CISO as a Service (CaaS) products, cost-effective packages of part-time, seasoned executive information-security expertise to organizations that lack the capacity to staff a full-time Chief Information Security Officer (CISO). These packages include blocks of ongoing virtual CISO time or on a targeted project basis. Because vCISO Services operates nearly 100% virtually, it saves businesses money by not passing on travel and other related face-to-face costs. The company typically reserves the on-site activities for when it matters most, such as interacting with auditors or presenting to the board of directors. The firm specializes in the executive components of information security management, such as risk assessments, policy and standard creation, vendor reviews, regulatory gap analysis and general interim and ongoing CISO activities.
https://vcisoservices.com

Attala Systems' Composable Storage Infrastructure

Upon its exit from stealth mode, Attala Systems revealed news of its new high-performance Composable Storage Infrastructure product. Attala Systems' NVMe over Fabric (NVMe-oF) solution product utilizes Intel FPGAs and NVMe Flash storage to provide "breakthrough performance" for its target customers, including cloud-service providers, e-commerce sites, managed-service providers, telco providers, financial services and real-time digital enterprises. The Attala Composable Storage Infrastructure, according to its creator, marks the start of a significant change in how storage is used for cloud and real-time analytics. Attala's FPGA-based fabric delivers advances in predictable storage latency, IOPS, agility and cost efficiency. With the complement of Attala's SPARA automation and management software, the result is a storage system with high and predictable performance and extremely simple management. The product utilizes a scale-out fabric running on standard 25, 40 or 50Gb/sec Ethernet to interconnect a data center's servers and data nodes. By eliminating legacy storage management layers, the Composable Storage Infrastructure product provides more than ten million IOPS per scale-out node and latencies as low as 16 microseconds at a per-gigabyte price lower than competitive solutions. The result is an adaptable storage infrastructure that is essentially an elastic block storage (EBS) solution "on steroids", adds Attala.

https://www.attalasystems.com

VMware Workstation

Welcome to VMware Workstation 14 Player

Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.

Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.

Download a Virtual Appliance
Download a virtual appliance from the marketplace. You can then open it in Player.

Help
View online help.

While the two versions of VMware Workstation serve multiple end user types, both enable multiple operating systems to run as virtual machines on a single Linux or Windows PC. The updated VMware Workstation 14 Pro gives IT professionals and developers indispensable tools when designing, testing and operating data centers and networks. Key new features in version 14 include support for Virtual Based Security, a new Network Latency Simulator, an improved open virtual format and open virtual appliance support. Data-center administrators also can leverage advanced host power management to connect to VMware vSphere and VMware vCenter quickly to manage virtual machines and perform power operations to ESXi hosts remotely. Expanded operating support includes Ubuntu 17.04, Fedora 26 and Windows 10 Fall Creators Update. The streamlined offering is the VMware Workstation 14 Player product line that leverages the same hypervisor technology, complete with many of the same capabilities, such as the broadest OS support, high performance and the power to run restricted VMs that comply with corporate policy. The commercial solution, adds VMware, is ideal for businesses seeking to run a single virtual machine on a corporate or BYO PC. A free edition is available for personal, non-commercial use.
https://www.vmware.com

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

RETURN TO CONTENTS

Rapid, Secure Patching: Tools and Methods

Generate enterprise-grade SSH keys and load them into an agent for control of all kinds of Linux hosts. Script the agent with the Parallel Distributed Shell (pdsh) to effect rapid changes over your server farm.

CHARLES FISHER

◀ PREVIOUS
New Products

NEXT
Feature: CLIC:
CLuster In the Cloud ▶

It is with some measure of disbelief that the computer science community has greeted the recent EternalBlue-related exploits that have torn through massive numbers of vulnerable systems (https://en.wikipedia.org/wiki/EternalBlue). The SMB exploits have kept coming (the most recent being SMBLoris presented at the last DEF CON, which impacts multiple SMB protocol versions, and for which Microsoft will issue no corrective patch: http://securityaffairs.co/wordpress/61530/hacking/smbloris-smbv1-flaw.html). Attacks with these tools incapacitated critical infrastructure to the point that patients were even turned away from the British National Health Service (http://www.telegraph.co.uk/news/2017/05/13/nhs-cyber-attack-everything-need-know-biggest-ransomware-offensive).

It is with considerable sadness that, during this SMB catastrophe, we also have come to understand that the famous Samba server presented an exploitable attack surface on the public internet in sufficient numbers for a worm to propagate successfully. I previously have discussed SMB security in *Linux Journal*, and I am no longer of the opinion that SMB server processes should run on Linux (http://www.linuxjournal.com/content/smbclient-security-windows-printing-and-file-transfer).

In any case, systems administrators of all architectures must be able to down vulnerable network servers and patch them quickly. There is often a need for speed and competence when working with a large collection of Linux servers. Whether this is due to security situations or other concerns is immaterial—the hour of greatest need is not the time to begin to build administration tools. Note that in the event of an active intrusion by hostile parties, forensic analysis may be a legal requirement, and no steps should be taken on the compromised server without a careful plan and documentation (https://staff.washington.edu/dittrich/misc/forensics). Especially in this new era of the black hats, computer professionals must step up their game and be able to secure vulnerable systems quickly.

Secure SSH Keypairs

Tight control of a heterogeneous UNIX environment must begin with best-practice use of SSH authentication keys. I'm going to open this section with a simple requirement. SSH private keys must be one of three types:

Ed25519, ECDSA using the E-521 curve or RSA keys of 3072 bits. Any key that does not meet those requirements should be retired (in particular, DSA keys must be removed from service immediately).

The Ed25519 key format (https://ed25519.cr.yp.to) is associated with Daniel J. Bernstein, who has such a preeminent reputation in modern cryptography that the field is becoming a DJB monoculture (http://www.metzdowd.com/pipermail/cryptography/2016-March/028824.html). The Ed25519 format is deigned for speed, security and size economy. If all of your SSH servers are recent enough to support Ed25519, then use it, and consider nothing else.

Guidance on creating Ed25519 keys suggests 100 rounds for a work factor in the "-o" secure format (https://blog.g3rt.nl/upgrade-your-ssh-keys.html). Raising the number of rounds raises the strength of the encrypted key against brute-force attacks (should a file copy of the private key fall into hostile hands), at the cost of more work and time in decrypting the key when ssh-add is executed. Although there always is controversy and discussion with security advances (https://news.ycombinator.com/item?id=12563899), I will repeat the guidance here and suggest that the best format for a newly created SSH key is this:

If you cannot upgrade your old SSH clients and servers, your next best option is likely E-521, available in the ECDSA key format.

```
ssh-keygen -a 100 -t ed25519
```

Your systems might be too old to support Ed25519—Oracle/CentOS/Red Hat 7 have this problem (the 7.1 release introduced support). If you cannot upgrade your old SSH clients and servers, your next best option is likely E-521, available in the ECDSA key format.

The ECDSA curves came from the US government's National Institute of Standards (NIST). The best known and most implemented of all of the NIST curves are P-256, P-384 and E-521. All three curves are approved for secret communications by a variety of government entities, but a number of cryptographers have expressed growing suspicion that the P-256 and P-384 curves are tainted (http://safecurves.cr.yp.to/rigid.html). Well known cryptographer Bruce Schneier has remarked (https://en.wikipedia.org/wiki/Curve25519): "I no longer trust the constants. I believe the NSA has manipulated them through their relationships with industry." However, DJB has expressed limited praise of the E-521 curve (http://blog.cr.yp.to/20140323-ecdsa.html): "To be fair I should mention that there's one standard NIST curve using a nice prime, namely $2^{521} – 1$; but the sheer size of this prime makes it much slower than NIST P-256." All of the NIST curves have greater issues with "side channel" attacks than Ed25519—P-521 is certainly a step down, and many assert that none of the NIST curves are safe. In summary, there is a slight risk that a powerful adversary exists with an advantage over the P-256 and P-384 curves, so one is slightly inclined to avoid them. Note that even if your OpenSSH (source) release is capable of E-521, it may be disabled by your vendor due to patent concerns (https://lwn.net/Articles/573166), so E-521 is not an option in this case. If you cannot use DJB's $2^{255} – 19$ curve, this command will generate an E-521 key on a capable system:

```
ssh-keygen -o -a 100 -b 521 -t ecdsa
```

And, then there is the unfortunate circumstance with SSH servers that support neither ECDSA nor Ed25519. In this case, you must fall back to RSA with much larger key sizes. An absolute minimum is the modern default of 2048 bits, but 3072 is a wiser choice:

```
ssh-keygen -o -a 100 -b 3072 -t rsa
```

Then in the most lamentable case of all, when you must use old SSH clients that are not able to work with private keys created with the `-o` option, you can remove the password on id_rsa and create a naked key, then use OpenSSL to encrypt it with AES256 in the PKCS#8 format, as first documented by Martin Kleppmann (http://martin.kleppmann.com/2013/05/24/improving-security-of-ssh-private-keys.html). Provide a blank new password for the keygen utility below, then supply a new password when OpenSSL reprocesses the key:

```
$ cd ~/.ssh

$ cp id_rsa id_rsa-orig

$ ssh-keygen -p -t rsa
Enter file in which the key is (/home/cfisher/.ssh/id_rsa):
Enter old passphrase:
Key has comment 'cfisher@localhost.localdomain'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.

$ openssl pkcs8 -topk8 -v2 aes256 -in id_rsa -out id_rsa-strong
Enter Encryption Password:
Verifying - Enter Encryption Password:

mv id_rsa-strong id_rsa
chmod 600 id_rsa
```

After creating all of these keys on a newer system, you can compare the file sizes:

```
$ ll .ssh
total 32
-rw-------. 1 cfisher cfisher  801 Aug 10 21:30 id_ecdsa
-rw-r--r--. 1 cfisher cfisher  283 Aug 10 21:30 id_ecdsa.pub
-rw-------. 1 cfisher cfisher  464 Aug 10 20:49 id_ed25519
-rw-r--r--. 1 cfisher cfisher  111 Aug 10 20:49 id_ed25519.pub
-rw-------. 1 cfisher cfisher 2638 Aug 10 21:45 id_rsa
-rw-------. 1 cfisher cfisher 2675 Aug 10 21:42 id_rsa-orig
-rw-r--r--. 1 cfisher cfisher  583 Aug 10 21:42 id_rsa.pub
```

Although they are relatively enormous, all versions of OpenSSH that I have used have been compatible with the RSA private key in PKCS#8 format. The Ed25519 public key is now small enough to fit in 80 columns without word wrap, and it is as convenient as it is efficient and secure.

Note that PuTTY may have problems using various versions of these keys, and you may need to remove passwords for a successful import into the PuTTY agent.

These keys represent the most secure formats available for various OpenSSH revisions. They really aren't intended for PuTTY or other general interactive activity. Although one hopes that all users create strong keys for all situations, these are enterprise-class keys for major systems activities. It might be wise, however, to regenerate your system host keys to conform to these guidelines.

These key formats may soon change. Quantum computers are causing increasing concern for their ability to run Shor's Algorithm (https://en.wikipedia.org/wiki/Shor's_algorithm), which can be used to find prime factors to break these keys in reasonable time. The largest commercially available quantum computer, the D-Wave 2000Q (https://www.dwavesys.com/d-wave-two-system), effectively presents under 200 qubits for this activity (https://crypto.stackexchange.com/questions/40893/can-or-can-not-d-waves-quantum-computers-use-shors-and-grovers-algorithm-to-f), which is not (yet) powerful enough for a successful attack. NIST has announced a competition for a new quantum-resistant public key system with a deadline of November 2017 (https://yro.slashdot.org/story/16/12/21/2334220/nist-asks-public-for-help-with-quantum-proof-cryptography). In response, a team including DJB has released source code for NTRU Prime (https://ntruprime.cr.yp.to/index.html). It does appear that we will likely see a post-quantum public key format for OpenSSH (and potentially TLS 1.3) released within the next two years, so take steps to ease migration now.

Also, it's important for SSH servers to restrict their allowed ciphers, MACs and key exchange lest strong keys be wasted on broken crypto (3DES, MD5 and arcfour should be long-disabled). My previous guidance on the subject (http://www.linuxjournal.com/content/cipher-security-how-harden-tls-and-ssh) involved the following (three) lines in the SSH client and server configuration (note that formatting in the sshd_config file requires all parameters on the same line with no spaces in the options; line breaks have been added here for clarity):

```
Ciphers chacha20-poly1305@openssh.com,
        aes256-gcm@openssh.com,
        aes128-gcm@openssh.com,
        aes256-ctr,
        aes192-ctr,
        aes128-ctr

MACs    hmac-sha2-512-etm@openssh.com,
        hmac-sha2-256-etm@openssh.com,
        hmac-ripemd160-etm@openssh.com,
        umac-128-etm@openssh.com,
        hmac-sha2-512,
        hmac-sha2-256,
        hmac-ripemd160,
        umac-128@openssh.com

KexAlgorithms curve25519-sha256@libssh.org,
              diffie-hellman-group-exchange-sha256
```

Since the previous publication, RIPEMD160 is likely no longer safe and should be removed. Older systems, however, may support only SHA1, MD5 and RIPEMD160. Certainly remove MD5, but users of PuTTY likely will want to retain SHA1 when newer MACs are not an option. Older servers can present a challenge in finding a reasonable Cipher/MAC/KEX when working with modern systems.

At this point, you should have strong keys for secure clients and servers. Now let's put them to use.

Scripting the SSH Agent

Modern OpenSSH distributions contain the ssh-copy-id shell script for easy key distribution. Below is an example of installing a specific, named key in a remote account:

```
$ ssh-copy-id -i ~/.ssh/some_key.pub person@yourserver.com
ssh-copy-id: INFO: Source of key(s) to be installed:
   "/home/cfisher/.ssh/some_key.pub"
ssh-copy-id: INFO: attempting to log in with the new key(s),
   to filter out any that are already installed
ssh-copy-id: INFO: 1 key(s) remain to be installed --
   if you are prompted now it is to install the new keys
person@yourserver.com's password:

Number of key(s) added: 1

Now try logging into the machine, with:
   "ssh 'person@yourserver.com'"
and check to make sure that only the key(s) you wanted were added.
```

If you don’t have the ssh-copy-id script, you can install a key manually with the following command:

```
$ ssh person@yourserver.com 'cat >> ~/.ssh/authorized_keys' < \
      ~/.ssh/some_key.pub
```

If you have SELinux enabled, you might have to mark a newly created authorized_keys file with a security type; otherwise, the sshd server dæmon will be prevented from reading the key (the syslog may report this issue):

```
$ ssh person@yourserver.com 'chcon -t ssh_home_t
 ➥~/.ssh/authorized_keys'
```

Once your key is installed, test it in a one-time use with the `-i` option (note that you are entering a local key password, not a remote authentication password):

```
$ ssh -i ~/.ssh/some_key person@yourserver.com
Enter passphrase for key '/home/v-fishecj/.ssh/some_key':
Last login: Wed Aug 16 12:20:26 2017 from 10.58.17.14
yourserver $
```

General, interactive users likely will cache their keys with an agent. In the example below, the same password is used on all three types of keys that were created in the previous section:

```
$ eval $(ssh-agent)
Agent pid 4394

$ ssh-add
Enter passphrase for /home/cfisher/.ssh/id_rsa:
Identity added: ~cfisher/.ssh/id_rsa (~cfisher/.ssh/id_rsa)
Identity added: ~cfisher/.ssh/id_ecdsa (cfisher@init.com)
Identity added: ~cfisher/.ssh/id_ed25519 (cfisher@init.com)
```

The first command above launches a user agent process, which injects environment variables (named `SSH_AGENT_SOCK` and `SSH_AGENT_PID`) into the parent shell (via `eval`). The shell becomes aware of the agent and passes these variables to the programs that it runs from that point forward.

When launched, the ssh-agent has no credentials and is unable to facilitate SSH activity. It must be primed by adding keys, which is done with `ssh-add`. When called with no arguments, all of the default keys will be read. It also can be called to add a custom key:

```
$ ssh-add  ~/.ssh/some_key
Enter passphrase for /home/cfisher/.ssh/some_key:
Identity added: /home/cfisher/.ssh/some_key
 ➥(cfisher@localhost.localdomain)
```

Note that the agent will not retain the password on the key. `ssh-add` uses any and all passwords that you enter while it runs to decrypt keys that it finds, but the passwords are cleared from memory when `ssh-add` terminates (they are not sent to `ssh-agent`). This allows you to upgrade to new key formats with minimal inconvenience, while keeping the keys reasonably safe.

The current cached keys can be listed with `ssh-add -l` (from, which you can deduce that "some_key" is an Ed25519):

```
$ ssh-add -l
3072 SHA256:cpVFMZ17oO5n/Jfpv2qDNSNcV6ffOVYPV8vVaSm3DDo
     /home/cfisher/.ssh/id_rsa (RSA)
521 SHA256:1L9/CglR7cstr54a600zDrBbcxMj/a3RtcsdjuU61VU
     cfisher@localhost.localdomain (ECDSA)
256 SHA256:Vd21LEM4lixY4rIg3/Ht/w8aoMT+tRzFUR0R32SZIJc
     cfisher@localhost.localdomain (ED25519)
256 SHA256:YsKtUA9Mglas7kqC4RmzO6jd2jxVNCc1OE+usR4bkcc
     cfisher@localhost.localdomain (ED25519)
```

While a “primed” agent is running, the SSH clients may use (trusting) remote servers fluidly, with no further prompts for credentials:

```
$ sftp person@yourserver.com
Connected to yourserver.com.
sftp> quit

$ scp /etc/passwd person@yourserver.com:/tmp
passwd                              100% 2269    65.8KB/s   00:00

$ ssh person@yourserver.com
   (motd for yourserver.com)
$ ls -l /tmp/passwd
-rw-r--r--  1 root  wheel  2269 Aug 16 09:07 /tmp/passwd
$ rm /tmp/passwd
$ exit
Connection to yourserver.com closed.
```

The OpenSSH agent can be locked, preventing any further use of the credentials that it holds (this might be appropriate when suspending a laptop):

```
$ ssh-add -x
Enter lock password:
Again:
Agent locked.
```

```
$ ssh yourserver.com
Enter passphrase for key '/home/cfisher/.ssh/id_rsa': ^C
```

It will provide credentials again when it is unlocked:

```
$ ssh-add -X
Enter lock password:
Agent unlocked.
```

You also can set `ssh-agent` to expire keys after a time limit with the `-t` option, which may be useful for long-lived agents that must clear keys after a set daily shift.

General shell users may cache many types of keys with a number of differing agent implementations. In addition to the standard OpenSSH agent, users may rely upon PuTTY's pageant.exe, GNOME keyring or KDE Kwallet, among others (the use of the PUTTY agent could likely fill an article on its own).

However, the goal here is to create "enterprise" keys for critical server controls. You likely do not want long-lived agents in order to limit the risk of exposure. When scripting with "enterprise" keys, you will run an agent only for the duration of the activity, then kill it at completion.

There are special options for accessing the root account with OpenSSH—the `PermitRootLogin` parameter can be added to the sshd_config file (usually found in /etc/ssh). It can be set to a simple `yes` or `no`, `forced-commands-only`, which will allow only explicitly authorized programs to be executed, or the equivalent options `prohibit-password` or `without-password`, both of which will allow access to the keys generated here.

Many hold that root should not be allowed any access. Michael W. Lucas addresses the question in *SSH Mastery* (https://www.michaelwlucas.com/tools/ssh):

> Sometimes, it seems that you need to allow users to SSH in to the system as root. This is a colossally bad idea in almost all environments. When users must log in as a regular user and then

> change to root, the system logs record the user account, providing accountability. Logging in as root destroys that audit trail....It is possible to override the security precautions and make sshd permit a login directly as root. It's such a bad idea that I'd consider myself guilty of malpractice if I told you how to do it. Logging in as root via SSH almost always means you're solving the wrong problem. Step back and look for other ways to accomplish your goal.

When root action is required quickly on more than a few servers, the above advice can impose painful delays. Lucas' direct criticism can be addressed by allowing only a limited set of "bastion" servers to issue root commands over SSH. Administrators should be forced to log in to the bastions with unprivileged accounts to establish accountability.

However, one problem with remotely "changing to root" is the statistical use of the Viterbi algorithm (https://people.eecs.berkeley.edu/~dawnsong/papers/ssh-timing.pdf). Short passwords, the `su -` command and remote SSH calls that use passwords to establish a trinary network configuration are all uniquely vulnerable to timing attacks on a user's keyboard movement. Those with the highest security concerns will need to compensate.

For the rest of us, I recommend that `PermitRootLogin without-password` be set for all target machines.

Finally, you can easily terminate `ssh-agent` interactively with the `-k` option:

```
$ eval $(ssh-agent -k)
Agent pid 4394 killed
```

With these tools and the intended use of them in mind, here is a complete script that runs an agent for the duration of a set of commands over a list of servers for a common named user (which is not necessarily root):

```
# cat artano

#!/bin/sh

if [[ $# -lt 1 ]]; then echo "$0 - requires commands"; exit; fi
```

```
R="-R5865:127.0.0.1:5865" # set to "-2" if you don't want
 ➥port forwarding

eval $(ssh-agent -s)

function cleanup { eval $(ssh-agent -s -k); }

trap cleanup EXIT

function remsh { typeset F="/tmp/${1}" h="$1" p="$2";
 ➥shift 2; echo "#$h"
 if [[ "$ARTANO" == "PARALLEL" ]]
 then ssh "$R" -p "$p" "$h" "$@" < /dev/null >>"${F}.out"
  ➥2>>"${F}.err" &
 else ssh "$R" -p "$p" "$h" "$@"
 fi }    # HOST                                          PORT CMD

if ssh-add ~/.ssh/master_key
then remsh yourserver.com                                  22 "$@"
     remsh container.yourserver.com                      2200 "$@"
     remsh anotherserver.com                               22 "$@"
     # Add more hosts here.
else echo Bad password - killing agent. Try again.
fi

wait

#######################################################################
# Examples:           # Artano is an epithet of a famous mythical being
# artano 'mount /patchdir'      # you will need an fstab entry for this
# artano 'umount /patchdir'
# artano 'yum update -y 2>&1'
# artano 'rpm -Fvh /patchdir/\*.rpm'
#######################################################################
```

This script runs all commands in sequence on a collection of hosts by default. If the `ARTANO` environment variable is set to `PARALLEL`, it instead will launch them all as background processes simultaneously and append their `STDOUT` and `STDERR` to files in /tmp (this should be no problem when dealing with fewer than a hundred hosts on a reasonable server). The `PARALLEL` setting is useful not only for pushing changes faster, but also for collecting audit results.

Below is an example using the `yum update` agent. The source of this particular invocation had to traverse a firewall and relied on a proxy setting in the /etc/yum.conf file, which used the port-forwarding option (`-R`) above:

```
# ./artano 'yum update -y 2>&1'
Agent pid 3458
Enter passphrase for /root/.ssh/master_key:
Identity added: /root/.ssh/master_key (/root/.ssh/master_key)
#yourserver.com
Loaded plugins: langpacks, ulninfo
No packages marked for update
#container.yourserver.com
Loaded plugins: langpacks, ulninfo
No packages marked for update
#anotherserver.com
Loaded plugins: langpacks, ulninfo
No packages marked for update
Agent pid 3458 killed
```

The script can be used for more general maintenance functions. Linux installations running the XFS filesystem should “defrag” periodically. Although this normally would be done with cron, it can be a centralized activity, stored in a separate script that includes only on the appropriate hosts:

```
# artano-fsr 'xfs_fsr -g 2>&1'
Agent pid 7897
Enter passphrase for /root/.ssh/master_key:
Identity added: /root/.ssh/master_key (/root/.ssh/master_key)
```

```
#yourserver.com
#container.yourserver.com
#anotherserver.com
Agent pid 7897 killed
```

An easy method to collect the contents of all authorized_keys files for all users is the following `artano` script (this is useful for system auditing and is coded to remove file duplicates):

```
artano 'awk -F: {print\$6\"/.ssh/authorized_keys\"} \
     /etc/passwd | sort -u | xargs grep . 2> /dev/null'
```

It is convenient to configure NFS mounts for file distribution to remote nodes. Bear in mind that NFS is clear text, and sensitive content should not traverse untrusted networks while unencrypted. After configuring an NFS server on host 1.2.3.4, I add the following line to the /etc/fstab file on all the clients and create the /patchdir directory. After the change, the `artano` script can be used to mass-mount the directory if the network configuration is correct:

```
# tail -1 /etc/fstab
1.2.3.4:/var/cache/yum/x86_64/7Server/ol7_latest/packages
 ➥/patchdir nfs4 noauto,proto=tcp,port=2049 0 0
```

Assuming that the NFS server is mounted, RPMs can be upgraded from images stored upon it (note that Oracle Spacewalk or Red Hat Satellite might be a more capable patch method):

```
# ./artano 'rpm -Fvh /patchdir/\*.rpm'
Agent pid 3203
Enter passphrase for /root/.ssh/master_key:
Identity added: /root/.ssh/master_key (/root/.ssh/master_key)
#yourserver.com
Preparing...                          ########################
Updating / installing...
xmlsec1-1.2.20-7.el7_4                ########################
```

```
xmlsec1-openssl-1.2.20-7.el7_4        ########################
Cleaning up / removing...
xmlsec1-openssl-1.2.20-5.el7          ########################
xmlsec1-1.2.20-5.el7                  ########################
#container.yourserver.com
Preparing...                          ########################
Updating / installing...
xmlsec1-1.2.20-7.el7_4                ########################
xmlsec1-openssl-1.2.20-7.el7_4        ########################
Cleaning up / removing...
xmlsec1-openssl-1.2.20-5.el7          ########################
xmlsec1-1.2.20-5.el7                  ########################
#anotherserver.com
Preparing...                          ########################
Updating / installing...
xmlsec1-1.2.20-7.el7_4                ########################
xmlsec1-openssl-1.2.20-7.el7_4        ########################
Cleaning up / removing...
xmlsec1-openssl-1.2.20-5.el7          ########################
xmlsec1-1.2.20-5.el7                  ########################
Agent pid 3203 killed
```

I am assuming that my audience is already experienced with package tools for their preferred platforms. However, to avoid criticism that I've included little actual discussion of patch tools, the following is a quick reference of RPM manipulation commands, which is the most common package format on enterprise systems:

- `rpm -Uvh` *`package`*`.i686.rpm` — install or upgrade a package file.
- `rpm -Fvh` *`package`*`.i686.rpm` — upgrade a package file, if an older version is installed.
- `rpm -e` *`package`* — remove an installed package.
- `rpm -q` *`package`* — list installed package name and version.

- `rpm -q --changelog` *`package`* — print full changelog for installed package (including CVEs).

- `rpm -qa` — list all installed packages on the system.

- `rpm -ql` *`package`* — list all files in an installed package.

- `rpm -qpl` *`package`*`.i686.rpm` — list files included in a package file.

- `rpm -qi` *`package`* — print detailed description of installed package.

- `rpm -qpi` *`package`* — print detailed description of package file.

- `rpm -qf` *`/path/to/file`* — list package that installed a particular file.

- `rpm --rebuild` *`package`*`.src.rpm` — unpack and build a binary RPM under /usr/src/redhat.

- `rpm2cpio` *`package`*`.src.rpm | cpio -icduv` — unpack all package files in the current directory.

Another important consideration for scripting the SSH agent is limiting the capability of an authorized key. There is a specific syntax for such limitations (https://man.openbsd.org/sshd#AUTHORIZED_KEYS_FILE_FORMAT). Of particular interest is the `from=""` clause, which will restrict logins on a key to a limited set of hosts. It is likely wise to declare a set of "bastion" servers that will record non-root logins that escalate into controlled users who make use of the enterprise keys.

An example entry might be the following (note that I've broken this line, which is not allowed syntax but done here for clarity):

```
from="*.c2.security.yourcompany.com,4.3.2.1" ssh-ed25519
 ➥AAAAC3NzaC1lZDI1NTE5AAAAIJSSazJz6A5x6fTcDFIji1X+
➥svesidBonQvuDKsxo1Mx
```

A number of other useful restraints can be placed upon `authorized_keys` entries. The `command=""` will restrict a key to a single program or script and will set the `SSH_ORIGINAL_COMMAND` environment variable to the client's attempted call—scripts can set alarms if the variable does not contain approved contents. The `restrict` option also is worth consideration, as it disables a large set of SSH features that can be both superfluous and dangerous.

Although it is possible to set server identification keys in the known_hosts file to a `@revoked` status, this cannot be done with the contents of authorized_keys. However, a system-wide file for forbidden keys can be set in the sshd_config with `RevokedKeys`. This file overrides any user's `authorized_keys`. If set, this file must exist and be readable by the sshd server process; otherwise, no keys will be accepted at all (so use care if you configure it on a machine where there are obstacles to physical access). When this option is set, use the `artano` script to append forbidden keys to the file quickly when they should be disallowed from the network. A clear and convenient file location would be /etc/ssh/revoked_keys.

It is also possible to establish a local Certificate Authority (CA) for OpenSSH that will allow keys to be registered with an authority with expiration dates (https://ef.gy/hardening-ssh). These CAs can become quite elaborate in their control over an enterprise (https://code.facebook.com/posts/365787980419535/scalable-and-secure-access-with-ssh). Although the maintenance of an SSH CA is beyond the scope of this article, keys issued by such CAs should be strong by adhering to the requirements for Ed25519/E-521/RSA-3072.

pdsh

Many higher-level tools for the control of collections of servers exist that are much more sophisticated than the script I've presented here. The most famous is likely Puppet (https://puppet.com), which is a Ruby-based configuration management system for enterprise control. Puppet has a somewhat short list of supported operating systems. If you are looking for low-level control of Android, Tomato, Linux smart terminals or other "exotic" POSIX, Puppet is likely not the appropriate tool. Another popular Ruby-based tool is Chef (https://www.chef.io), which is known for its complexity. Both Puppet and Chef require Ruby installations on both clients and servers, and they both will catalog any SSH keys that they find, so this key strength discussion is completely applicable to them.

There are several similar Python-based tools, including Ansible (https://www.ansible.com), Bcfg2 (http://bcfg2.org), Fabric (http://www.fabfile.org) and SaltStack (https://saltstack.com). Of these, only Ansible can run "agentless" over a bare SSH connection; the rest will require agents that run on target nodes (and this likely includes a Python runtime).

Another popular configuration management tool is CFEngine (https://cfengine.com), which is coded in C and claims very high performance. Rudder (http://www.rudder-project.org/site) has evolved from portions of CFEngine and has a small but growing user community.

Most of the previously mentioned packages are licensed commercially and some are closed source.

The closest low-level tool to the activities presented here is the Parallel Distributed Shell (pdsh), which can be found in the EPEL repository (https://fedoraproject.org/wiki/EPEL). The pdsh utilities grew out of an IBM-developed package named dsh designed for the control of compute clusters. Install the following packages from the repository to use pdsh:

```
# rpm -qa | grep pdsh
pdsh-2.31-1.el7.x86_64
pdsh-rcmd-ssh-2.31-1.el7.x86_64
```

An SSH agent must be running while using pdsh with encrypted keys, and there is no obvious way to control the destination port on a per-host basis as was done with the `artano` script. Below is an example using pdsh to run a command on three remote servers:

```
# eval $(ssh-agent)
Agent pid 17106

# ssh-add  ~/.ssh/master_key
Enter passphrase for /root/.ssh/master_key:
Identity added: /root/.ssh/master_key (/root/.ssh/master_key)
```

```
# pdsh -w hosta.com,hostb.com,hostc.com uptime
hosta: 13:24:49 up 13 days,  2:13, 6 users, load avg: 0.00, 0.01, 0.05
hostb: 13:24:49 up  7 days, 21:15, 5 users, load avg: 0.05, 0.04, 0.05
hostc: 13:24:49 up  9 days,  3:26, 3 users, load avg: 0.00, 0.01, 0.05

# eval $(ssh-agent -k)
Agent pid 17106 killed
```

The `-w` option above defines a host list. It allows for limited arithmetic expansion and can take the list of hosts from standard input if the argument is a dash (-). The `PDSH_SSH_ARGS` and `PDSH_SSH_ARGS_APPEND` environment variables can be used to pass custom options to the SSH call. By default, 32 sessions will be launched in parallel, and this "fanout/sliding window" will be maintained by launching new host invocations as existing connections complete and close. You can adjust the size of the "fanout" either with the `-f` option or the `FANOUT` environment variable. It's interesting to note that there are two file copy commands: `pdcp` and `rpdcp`, which are analogous to `scp`.

Even a low-level utility like pdsh lacks some flexibility that is available by scripting OpenSSH, so prepare to feel even greater constraints as more complicated tools are introduced.

Conclusion

Modern Linux touches us in many ways on diverse platforms. When the security of these systems is not maintained, others also may touch

NOTE: An exploit compromising Ed25519 was recently demonstrated that relies upon custom hardware changes to derive a usable portion of a secret key (https://research.kudelskisecurity.com/2017/10/04/defeating-eddsa-with-faults). Physical hardware security is a basic requirement for encryption integrity, and many common algorithms are further vulnerable to cache timing or other side channel attacks that can be performed by the unprivileged processes of other users. Use caution when granting any access to systems that process sensitive data.

our platforms and turn them against us. It is important to realize the maintenance obligations when you add any Linux platform to your environment. This obligation always exists, and there are consequences when it is not met.

In a security emergency, simple, open and well understood tools are best. As tool complexity increases, platform portability certainly declines, the number of competent administrators also falls, and this likely impacts speed of execution. This may be a reasonable trade in many other aspects, but in a security context, it demands a much more careful analysis. Emergency measures must be documented and understood by a wider audience than is required for normal operations, and using more general tools facilitates that discussion.

I hope the techniques presented here will prompt that discussion for those who have not yet faced it.■

Charles Fisher has an electrical engineering degree from the University of Iowa and works as a systems and database administrator for a Fortune 500 mining and manufacturing corporation. He has previously published both journal articles and technical manuals on Linux for *UnixWorld* and other McGraw-Hill publications.

DISCLAIMER: The views and opinions expressed in this article are those of the author and do not necessarily reflect those of *Linux Journal*.

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

RETURN TO CONTENTS

SUPERMICRO®
MARKETPLACE

Powered by Silicon Mechanics

Broad Selection

Zero Defects

3 YEAR WARRANTY

3-Year Warranty

Your Source for Supermicro Platform Technology

Twin, TwinPro & BigTwin

High-density, high-value servers

Configure Now

FatTwin

Highest performance per watt

Configure Now

SuperStorage

Flexible and efficient storage

Configure Now

1U Servers

Entry & enterprise1U form factor servers

Configure Now

2U Servers

Flexible 2U form factor servers

Configure Now

3U+ Servers

All 3U and larger form factor servers

Configure Now

Ultra Servers

Unrivaled performance, flexibility & scalability

Configure Now

MP Servers

Servers based on the Intel® Xeon® E7 Product Family

Configure Now

SuperWorkstations

Server-grade performance at your desk

Configure Now

Talk to a Supermicro Expert! 866.352.1173

CLIC

CLuster In the Cloud

Cloud computing is a powerful tool. Learn how cluster computing concepts can be applied to deploy an instant cluster in the cloud.

NATHAN R. VANCE and WILLIAM F. POLIK

◀ PREVIOUS
Feature: Rapid, Secure Patching: Tools and Methods

NEXT
EOF ▶

High-performance computing currently typically uses clusters, but its future is in the cloud. System administrators now find themselves in the middle of a transition period, maintaining old clusters while porting software to cloud environments.

There are many reasons to migrate to the compute cloud. For starters, there is zero upfront cost and no physical hardware to maintain. The cloud is exceptionally reliable, often with better than 99.95% uptime. The cloud provides an extremely convenient environment, as you can create, use, shut down, restart, snapshot and delete cloud instances from the comfort of your web browser. The cloud is also relatively secure—if you trust the cloud provider, of course.

One of the most attractive features of the cloud is dynamic scalability, meaning that one can increase or decrease the amount of resources used in real time. Coupled with the cloud's utility model of pricing, this makes it cost efficient to spin up a cloud instance to perform a task and then shut it down when that task completes.

Early in the learning curve of utilizing the compute cloud, one often creates a snapshot of a cloud instance configured with appropriate software. A typical workflow experience might then be:

1. Launch a new cloud instance from the snapshot.

2. `ssh` in.

3. Run the software.

4. Accidentally close the laptop lid, breaking the SSH session.

5. Reconnect and restart the job using `nohup`.

6. `scp` the output back to the laptop.

7. At 3:00 am, realize the instance is still running and frantically shut it down.

Life would be much easier if this process were streamlined and automated. Ideally, one simply could submit a job to a software package that automatically launches a cloud instance, executes the job, saves the results and deletes the cloud instance. This vision suggests a project that automatically creates and uses cloud computing resources.

Several open-source projects have a similar trajectory. Unfortunately, the stable ones require one to containerize or hadoopize all application software. This is a fine standard to impose, but if the software already works perfectly well in a cluster, why rework it on a per-application basis

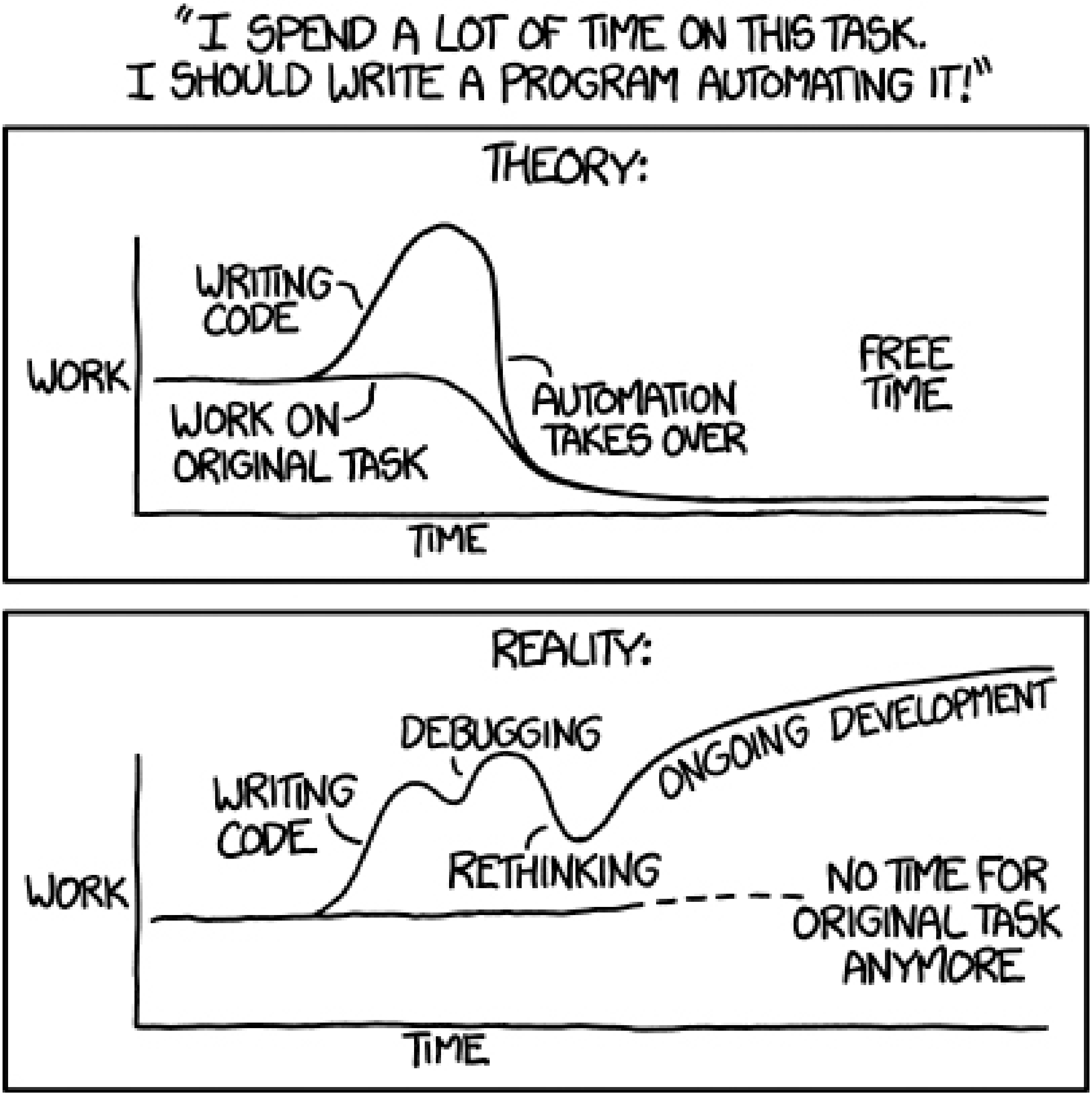


Figure 1. https://xkcd.com/1319

for a new environment? A more general solution would be to replicate a cluster environment in the cloud and run the cluster software unmodified. Of course, that cluster should be one that resizes itself, and the software to automate that task had to be written—and everyone knows how automating tasks tends to work (Figure 1).

Design Goals

The central goal of CLuster In the Cloud (CLIC) is to replicate a physical cluster as a virtual cluster in the cloud, so that any application software that's designed for clusters doesn't know the difference. Just like a physical cluster, there are a head node and some compute nodes. However, because of the dynamic scalability of the cloud, the compute nodes are created only when they are needed and are deleted once they no longer are needed.

To ensure the longevity of this project, it must be versatile in as many dimensions as possible. Ideally these include:

- Linux distro.
- Cloud node architecture.
- Location of head node (in cloud or physical).
- Cloud service provider.
- Job scheduler.

The final goal is to make CLIC easy to install. Cloud computers, being virtual machines, are by nature disposable. It would be undesirable to have to configure the head node manually every time it gets deleted or if one wants to try something different.

Building a CLuster In the Cloud

In order to understand what building a virtual cluster in the cloud entails, it is worth describing the process. For this purpose, we're using the Google Compute Engine (GCE), as it provides both graphical and API controls. GCE handles a number of the required elements of building a cluster out of the box:

- OS installation.
- Firewall.
- dhcp.
- Hostname resolution.
- ntp.
- Monitoring (via the console).

This leaves the setup task deceptively simple:

- SSH keys.
- NFS.
- Job scheduler (SLURM).

To create a cluster manually, spin up two CentOS instances on GCE, one designated as the head node and the other as the compute node. The first task is to set up passwordless SSH. However, the standard procedure of generating an SSH key pair on the head node and copying the public key to the compute node does not work.

No documentation confirms this, but through experimentation we discovered that there are two layers of SSH key enforcement in GCE: the host OS and GCE itself. In order to install new keys, instead of copying them to the cloud instances, GCE needs to be made aware of them, either via the graphical console or the API. GCE then propagates the keys to the cloud instances.

The next task is to NFS-mount the home directory. Of course, this mounts over users' .ssh directories, breaking passwordless SSH. The solution should have been to add the head node's keys to the head node's .ssh, but this messes up GCE's redundant security. The underlying mechanism for GCE is undocumented and probably subject to change, but to satisfy it, one has to keep NFS from mounting over ~/.ssh. The following script uses `mount --bind` to access the original ~/.ssh:

```
mount -t nfs4 -o rw $HEAD_IP:/home /home
mount --bind / /bind-root
for user in `ls /home`; do
    mount --bind /bind-root/home/$user/.ssh /home/$user/.ssh
done
```

The final component is the job scheduler, SLURM. It is built from source using SSL authentication from the instructions at https://slurm.schedmd.com. At this point, one has a functional two-node cluster in the cloud! The CLIC installer automates and generalizes this process.

CLIC Structure

In designing CLIC, the first structural decision is how to handle job scheduling. One possibility is to make CLIC be the job scheduler. The proposed process of operation is so simple that any dedicated job scheduler would be overpowered:

1. User submits job.

2. Node is created.

3. Job is run on node.

4. Node is deleted.

Notice that the scheduler doesn't actually do any scheduling. Because an unlimited number of compute nodes are potentially available, one simply can spin up additional compute nodes in a virtual cluster instead of deferring jobs to run on the next available node in a physical cluster. Under this cloud computing paradigm, the scheduler is just a glorified remote execution platform, coming into play only for steps 1 and 3. Why should a heavy-powered suite be required for such a simple task?

It turns out that the job scheduler is required to replicate the cluster environment. A design goal of CLIC is that the cluster software doesn't know the difference between a physical cluster and a CLuster In the Cloud. The scheduler—whether it schedules or not—plays a vital role in this regard by providing a known interface for job submission. Therefore, short of re-implementing an incredibly complex interface, the job scheduler is necessary. We chose SLURM for this purpose because its popularity in cluster computing is increasing, and it can handle changes in node availability.

Now that jobs are submitted with SLURM and slated to be run on GCE, the job of CLIC is to bridge the gap. There are a few ways to go about doing this. A project called SLURM Elastic Computing started to integrate similar functionality into SLURM itself, although this particular project seems to have stalled. CLIC certainly could do something like that and integrate directly into SLURM. However, this would prevent swapping out job schedulers in the future, making it unversatile. Alternatively, it's possible to integrate tightly with the cloud provider, but that often requires the containerization of application software, which goes against the initial design goal of being able to run unmodified. Therefore, CLIC is a standalone dæmon that monitors the SLURM queue for resources needed and manipulates GCE accordingly.

Next, the structure of the cluster must be addressed. Since it is a cluster, it needs a head node and compute nodes. The head node is a running instance with the CLIC dæmon installed on it. Because the number of compute nodes can vary, they collectively take the form of a single image from which CLIC can create them as needed. The compute nodes have mostly the same software as the head node, so the image can simply be taken of an already configured head node.

CLIC Dæmon

The CLIC dæmon is the central point of the project. It implements the algorithm that monitors the job queue and manipulates the cloud API to expand and contract the cluster.

The main loop in CLIC collects data on the length of the SLURM queue, the number of nodes sitting idle and the number of nodes that are currently booting. A naive algorithm would be to create as many nodes as jobs that are in the queue, minus the number of idle nodes and the number of nodes currently booting. Alternatively, if the queue length is zero, delete the idle nodes.

This algorithm is naive because GCE instances take 1–2 minutes to allocate resources and boot. The state of the queue when the boot process starts may not be the same as when it ends, often resulting in CLIC overshooting the required adjustment.

Instead of spinning up the same number of nodes as queued jobs, a better approach is to create nodes for half of the queued jobs, rounded up. Similarly, we delete half of the idle nodes, rounded up:

```
nodesToCreate = ceil(queueLength / 2) - nodesIdle - nodesBooting
nodesToDelete = ceil(nodesIdle / 2) when queueLength = 0
```

Using this method, instead of resizing the virtual cluster to the instantaneous demand of two minutes ago, it approaches that demand geometrically. The benefit is that the size of the cluster converges rapidly to computational power demanded when they differ substantially, but slows down as they approach, thus keeping it from overshooting.

Variability: Heterogeneous Architectures

Up to this point in CLIC's description, we assumed that all compute nodes are single CPU machines with 3.75GB of RAM and 10GB hard disks (this is the default instance configuration in GCE). Those specs actually describe a smartphone quite well, but they are inadequate for most cluster applications!

To allow for variability in compute instance architectures, we added fields in CLIC's configuration file for CPUs, memory and disk size, each of which may take multiple values. CLIC then can create nodes for each combination of values, and SLURM can run jobs on the appropriately sized machines.

A minor complication is that SLURM was developed under the assumption that the cluster would remain static. It does clever things that are useful in that context, such as packing, where it runs multiple jobs with small resource demands on larger machines if smaller machines aren't available. However, packing prevents the deletion of large nodes in the virtual cluster.

To illustrate the problem, if the cluster has a single 16-CPU compute node when a 1-CPU job is submitted, then SLURM will run that job on the 16-CPU node, leaving 15 CPUs unutilized. Perhaps additional jobs might be submitted so that a greater portion of that machine is utilized, but there likely will be significant periods of time that it is wastefully sitting mostly empty.

Instead of packing small jobs onto large machines, in a cloud environment, it is better to partition out architectures so that jobs are run only on same-sized machines. This is more economical and allows nodes with unused cores to be deleted.

To implement this, CLIC gives SLURM a partition for each architecture, and CLIC keeps track of each partition separately when creating and deleting nodes. Since there is potentially a large number of partitions, each referred to by a unique name, such as "1cpu10diskstandard", CLIC generates a job submission plugin written in Lua that places jobs in the best partitions given their requirements.

Variability: Hybrid Clusters

So far in this description of CLIC, the head node has been in the cloud with all the compute nodes. It's nice to have it in the same aethereal location as the compute nodes, because it can communicate with them over a private low-latency LAN. However, there are several reasons why a physical head node may be more desirable:

- If the head node runs 24/7, there is no opportunity for cost savings from dynamic scalability.
- Storing results of calculations on a cloud head node can be expensive, and if you don't trust big data companies, unsettling.
- Transferring large amounts of data from the cloud all at once is time-consuming.
- I already have a web server. Why do I need another?

For these reasons, CLIC also allows for a local head node. Some issues associated with a local head node are trivial to fix. The firewall for GCE has to be modified to allow SLURM traffic through. Additionally, the

GCE API has to be installed and authenticated. The installation process differs because an image can't be generated from the head node.

A few things are more complex. First is compute node hostname resolution. When the head node was in the cloud, GCE provided hostname resolution for the compute nodes in the form of a DNS. One could ignore the IP addresses of the nodes completely, referring to them only by name. This becomes a problem when the head node is no longer able to resolve the names of the compute nodes because it can't access GCE's DNS, and the compute nodes can't resolve the hostname of the head node because GCE's DNS isn't aware of it.

To resolve this issue, /etc/hosts on both the head and compute nodes must contain address/name pairings for the cluster. Every time CLIC boots a compute node, it uses the GCE API to obtain the compute node's IP address and adds it to /etc/hosts on the head node. Then it `ssh`'s in to the compute node and adds the head node's address/name pairing there.

Because the head and compute nodes no longer are generated from the same image, the UIDs and GIDs of users don't necessarily match between the head and compute nodes. This is an issue, because NFS identifies users and groups by ID, not by name, resulting in filesystem permission problems. The solution is to change the UIDs and GIDs for users on the compute nodes. CLIC provides a script for this process that is run on compute nodes when they boot up.

Another benefit when the head and compute nodes were in the cloud was that all intra-cluster communication occurred over a private LAN, so unencrypted NFS is just fine. When the head node is separated from the compute nodes by a wide expanse of the dangerous internet, unencrypted anything is really bad. The solution to this issue is to direct all NFS traffic through an SSH tunnel.

CLIC Installation

CLIC can be added on top of any standard system to enable the creation of a virtual cluster. The cluster can be entirely in the cloud, or the head node can be a local computer and the compute nodes can be in the cloud.

When installing a pure cloud cluster, one needs to generate only a single image, which can serve as both the head and compute nodes.

The installation of a hybrid cluster isn't as convenient as the pure cloud cluster, because it's not possible to create a physical head node from a cloud image. After configuring the head and compute node separately, the CLIC installation software syncs the physical head node with the compute node to create a compute node image.

The CLIC installation code is available at https://github.com/nathanrvance/clic.

Installing CLIC requires only a single command that installs cloud APIs, the CLIC dæmon, SLURM and other cluster software. The user is prompted for local sudo access and cloud API access when needed.

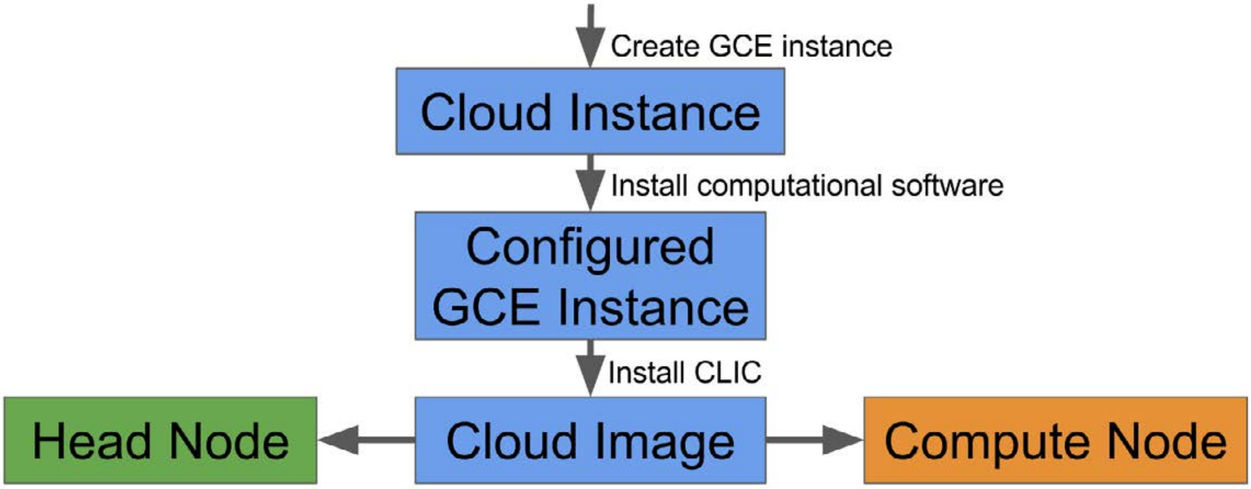


Figure 2. Pure cloud cluster installation: create a single cloud image with all necessary software, and use this image for the head and compute nodes.

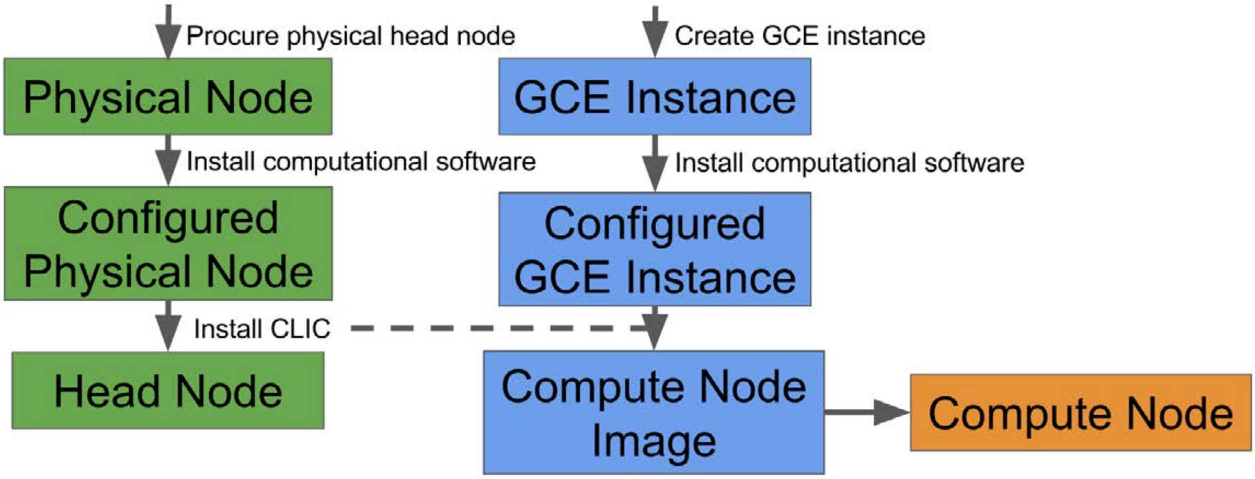


Figure 3. Hybrid cluster installation: the physical head node and cloud compute node are configured separately, and CLIC syncs them.

Performance Testing

At this point, it's possible to run a stress test on CLIC. A simple test is to submit a random number of `sleep X` jobs to SLURM every ten minutes, where X is a random value between 500 and 1000 seconds. CLIC automatically spins up needed nodes to accommodate the demand. No jobs waited longer than 12 minutes.

CLIC also shut down nodes that no longer were needed. The total node runtime was 54 hours, including startup and shutdown times. The

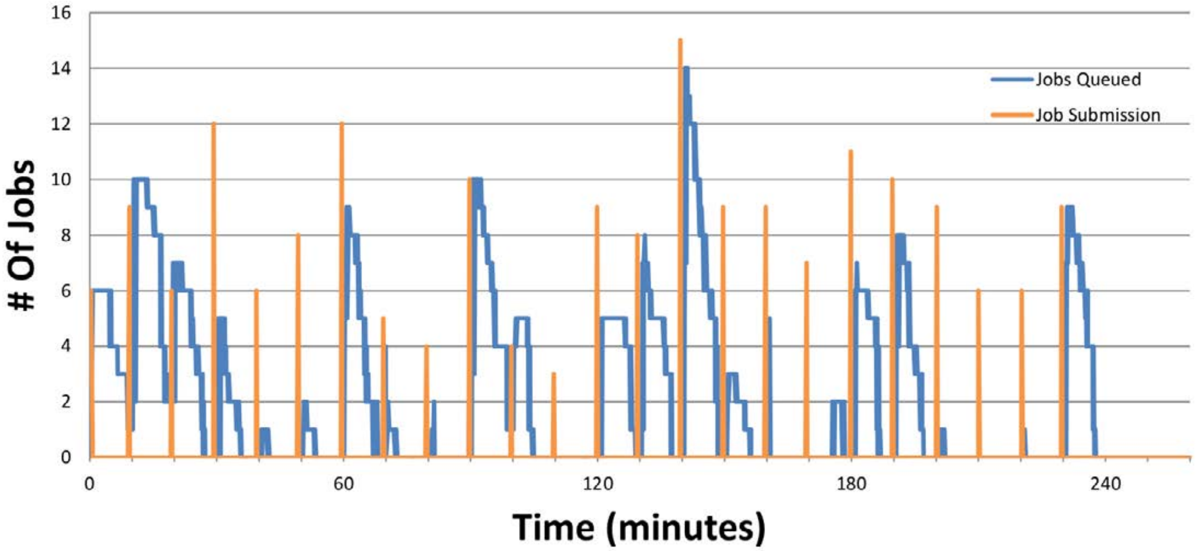


Figure 4. Job Submission Rate and Queue Length

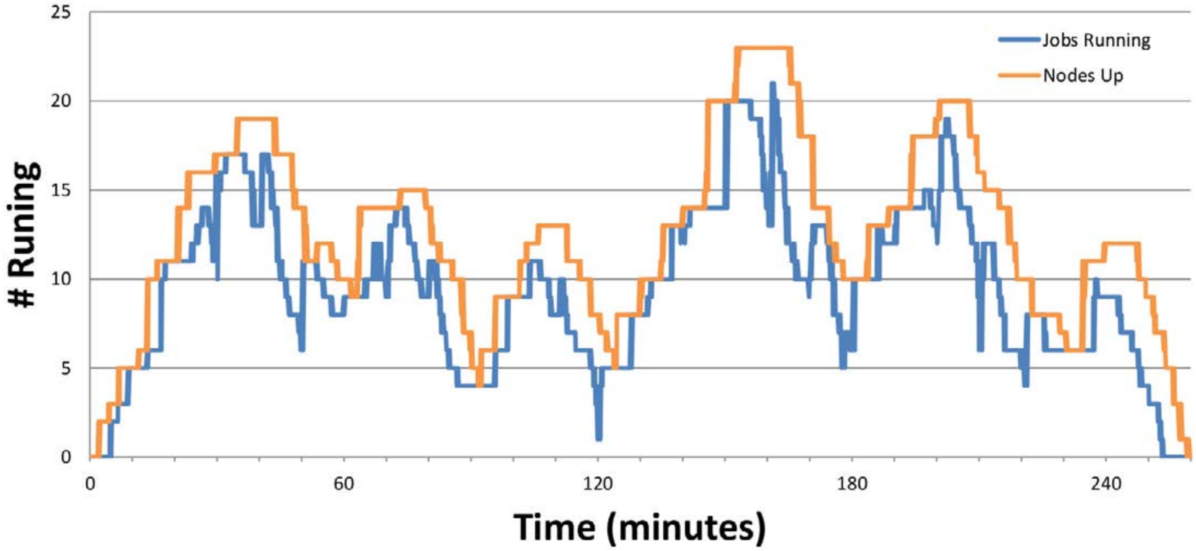


Figure 5. Jobs and Nodes Running

total job runtime was 41 hours. Therefore, 41 out of 54 hours paid for were spent running jobs, which is an efficiency of 76% for the 193 jobs submitted over four wall-clock hours.

The real test of CLIC is to run unmodified cluster software on it. We installed several computational chemistry engines, such as Gaussian and NWChem, and WebMO, which is a computational chemistry front end that utilizes a cluster environment to submit jobs and retrieve results. These packages were all installed using their standard installation procedures and without any modifications to the software itself. It all just worked. Mission accomplished.

Conclusions and Future Work

At this point, CLIC is able to run unmodified cluster software on a virtual, dynamic cluster in the cloud. It allows for pure clusters in the cloud or on-demand usage of cloud resources to augment physical hardware.

There is still development opportunity for CLIC. CLIC is currently dependent on GCE and SLURM. CLIC could support a more general and standardized cloud API so that it can support other cloud platforms. It also could generalize job scheduler calls so that users can access a variety of job schedulers. But regardless, CLIC serves as a useful and proven model for deploying a CLuster In the Cloud.■

Nathan Vance is a computer science major at Hope College in Holland, Michigan. He discovered Linux as a high-school junior and currently uses Arch Linux. In his free time, he enjoys running, skiing and writing software.

William Polik is a computational chemistry professor at Hope College in Holland, Michigan. His research involves high-accuracy quantum chemistry using computer clusters. He co-founded WebMO LLC, a software company that provides web and portable device interfaces to computational chemistry programs.

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

RETURN TO CONTENTS

New Hope for Digital Identity

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.

Centralized approaches only make the problem worse. But there's hope at the edge: with you and me.

◀ PREVIOUS
Feature: CLIC: CLuster In the Cloud

Identity is personal. You need to start there.

In the natural world where we live and breathe, personal identity can get complicated, but it's not broken. If an Inuit family from Qikiqtaaluk wants to name their kid Anuun or Issorartuyok, they do, and the world copes. If the same kid later wants to call himself Steve, he does. Again, the world copes. So does Steve.

Much of that coping is done by Steve *not* identifying himself unless he needs to, and then by not revealing more than what's required. In most cases Steve isn't accessing a service, but merely engaging with other people, and in ways so casual that in most cases no harm is done if the other person forgets Steve's name or how he introduced himself. In fact, most of what happens in the social realms of the natural world are free of identifiers, and that's a feature rather than a bug. Dunbar's number (https://en.wikipedia.org/wiki/Dunbar's_number) exists for a reason. So does the fact that human memory is better at forgetting details than at remembering them. This too is a feature. Most of what we know is tacit rather than explicit. As the scientist and philosopher Michael Polanyi (https://en.wikipedia.org/wiki/Michael_Polanyi) puts it (in perhaps his only quotable line), "We know more than we can tell." This is why we can easily recognize a person without being able to describe exactly how we do that, and without knowing his or her name or other specific "identifying" details about them.

Steve's identity can also be a claim that does not require proof, or even need to be accurate. For example, he may tell the barista at a coffee shop that his name is Clive to avoid confusion with the guy ahead of him who just said *his* name is Steve.

How we create and cope with identity in the natural world has lately come to be called *self-sovereign*, at least among digital identity obsessives such as myself. Self-sovereign identity starts by recognizing that the kind of naming we get from our parents, tribes and selves is at the root level of how identity works in the natural world, and needs to frame our approaches in the digital one as well.

Our main problem with identity in the digital world is that we understand it entirely in terms of organizations and their needs. These approaches are administrative rather than personal or social. They work for the convenience of organizations first. In administrative systems, identities are just records, usually kept in databases. Aside from your business card, every name imprinted on a rectangle in your wallet was issued to you by some administrative system: the government, the Department of Motor Vehicles, the school, the drug store chain. None are your identity. All are identifiers used by organizations to keep track of you.

For your inconvenience, every organization's identity system is also a separate and proprietary silo, even if it is built with open-source software and methods. Worse, an organization might have many different silo'd identity systems that know little or nothing about each other. Even an organization as unitary as a university might have completely different identity systems operating within HR, health care, parking, laundry, sports and IT—as well as within its scholastic realm, which also might have any number of different departmental administrative systems, each with its own record of students past and present.

While ways of "federating" identities between silos have been around since the last millennium, there is still no standard or open-source way for you to change, say, your surname or your mailing address with all the administrative systems you deal with, in one move. In fact, doing so is unthinkable as long as our understanding of identity remains framed inside the norms of silo'd administrative systems and thinking.

Administrative systems have been built into civilized life for as long as we've had governments, companies and churches, to name just three institutions. But every problem we ever had with any of those only got worse once we had ways to digitize what was wrong with them, and then to network the same problems. This is why our own ability to administrate the many different ways we are known to the world's identity systems only gets worse every time we click "accept" to some site's, service's or app's terms and conditions, and create yet another login, password and namespace to manage.

Unfortunately, the internet was first provisioned to the mass market over dial-up lines, and both ISPs and website developers made client-server the defaulted way to deal with people. By design, client-server is slave-master, because it puts nearly all power on the server side. The client has no more agency or identity than the server allows it.

True, a website works (or ought to work) by answering client requests for files. But we see how much respect that gets by looking at the history of Do Not Track (https://en.wikipedia.org/wiki/Do_Not_Track). Originally meant as a polite request by clients for servers to respect personal privacy, it was opposed so aggressively by the world's advertisers and commercial publishers that people took matters into their own hands by installing browser extensions for blocking ads and tracking. Then the W3C itself got corrupted by commercial interests, morphing Do Not Track into "tracking preference expressions" (http://www.w3.org/2011/tracking-protection). If individuals had full agency on the web in the first place, this never would have happened. But they didn't, and it did.

So we won't solve forever-standing identity problems with client-server, any more than we would have solved the need for personal computing with more generous mainframes.

So we won't solve forever-standing identity problems with client-server, any more than we would have solved the need for personal computing with more generous mainframes.

If we want fully human digital identity to work on the internet, we have to respect the deeply human need for self-determination. That requires means for individuals to assert self-sovereign identities, and for systems to require only verified claims when they need useful identity information. Anything else will be repeating mistakes of the past.

It should help to remember that most human interaction is not with big administrative systems. For example, around 99% of the world's businesses are small. (See "Small is the New Big": https://shift.newco.co/small-is-the-new-big-5e7f2c715fa1.) Even if every business of every size becomes digital and connected, they need to be able to operate without requiring outside (such as government or platform) administrative systems, for the simple reason that most of the ways people identify each other in the offline world is both minimally and on a need-to-know basis. It is only inside administrative systems that fixed identities and identifiers are required. And even they only really need to deal with verified claims.

So we need to recognize three things, in this order:

1. That everybody comes to the networked world with sovereign-source identities of their own, that they need to be able to make verifiable claims for various identity-related purposes; but that they don't need to do either at all times and in all circumstances.

2. That the world is still full of administrative systems, and that those systems can come into alignment once they recognize the self-sovereign nature of human beings. That means seeing human beings

as fully human and not just as "consumers" or "users" of products and services provided by organizations. And it means coming up, at last, with standard and trusted ways individual human beings can alter identity information with many different administrative systems, using standards-based tools of their own.

3. There are billions (the World Bank says 2.5: http://www.worldbank.org/en/topic/ict/brief/the-identity-target-in-the-post-2015-development-agenda-connections-note-19) of people in the world who lack any "official identification". Thus "official ID for all" is a goal of the United Nations, the World Bank and other large organizations trying to help masses of people who will be coming online during the next few years, especially refugees. Some of these people have good reasons not to be known, while others have good reasons to be known. It's complicated. Still, the commitment is there. The UN's Sustainable Development Goal 16.9 says "By 2030, provide legal identity for all, including birth registration" (https://sustainabledevelopment.un.org/sdg16).

What we need for all of these is an open-source and distributed approach that's NEA: Nobody owns it, Everybody can use it and Anybody can improve it. Within that scope, much is possible.

In "Rebooting the Web of Trust" (https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-fall2017/blob/master/topics-and-advance-readings/functional-identity-primer.md), Joe Andrieu says *"Identity is how we keep track of people and things and, in turn, how they keep track of us."* Among many other helpful things in that piece, Joe says this:

> Engineers, entrepreneurs, and financiers have asked "Why are we spending so much time with a definition of identity? Why not just build something and fix it if it is broken?" The vital, simple reason is **human dignity**.
>
> When we build interconnected systems without a core understanding of identity, we risk **inadvertently** compromising human dignity. We risk **accidentally** building systems that deny self-expression, place individuals in harm's way, and unintentionally oppress those most in need of self-determination.

> There are times when the needs of **security** outweigh the need for human dignity. Fine. It's the job of our **political** systems—local, national, and international—to minimize abuse and to establish boundaries and practices that respect basic human rights.
>
> But when engineers **unwittingly** compromise the ability of individuals to self-express their identity, when we expose personal information in unexpected ways, when our systems deny basic services because of a flawed understanding of identity, these are **avoidable tragedies**. What might seem a minor technicality in one conversation could lead to the loss of privacy, liberty, or even life for an individual whose identity is **unintentionally compromised**.
>
> That's why it pays to understand identity, so the systems we build intentionally enable human dignity instead of accidentally destroy it.

Phil Windley (http://www.windley.com), whom I have sourced often in these columns (see "Doing for User Space What We Did for Kernel Space": http://www.linuxjournal.com/content/doing-user-space-what-we-did-kernel-space and "The Actually Distributed Web": http://www.linuxjournal.com/content/actually-distributed-web, for example), has lately turned optimistic about developing decentralized identity approaches (http://www.windley.com/archives/2017/08/the_case_for_decentralized_identity.shtml). His own work chairing the Sovrin Foundation (https://sovrin.org) is toward what he calls "a global utility for identity" based on a distributed ledger such as blockchain. And, of course, open source. He writes:

> A universal *decentralized* identity platform offers the opportunity for services to be decentralized...I don't have to be a sharecropper for some large corporation. As an example, I can imagine a universal, decentralized identity system giving rise to apps that let anyone share rides in their car without the overhead of a Lyft or Uber because the identity system would let others vouch for the driver and the passenger.

That vouching is done by a verified claim. Not by calling on some centralized "identity provider".

Phil, Kaliya (Identity Woman: https://identitywoman.net) and I put on the Internet Identity Workshop (http://www.internetidentityworkshop.com) twice a year at the Computer History Museum in Silicon Valley. We had our 25th just last month. All three of our obsessions with identity go back to the last millennium. At no time since then have I felt more optimistic than I do now about the possibility that we might finally solve this thing. But we'll need help. I invite everyone here who wants to get in on a good thing soon to weigh in and help out.■

Send comments or feedback via http://www.linuxjournal.com/contact or to 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	http://drupalize.me	69
Peer 1 Hosting	http://go.peer1.com/linux	113
Silicon Mechanics	http://www.siliconmechanics.com	99
SPTech Con	http://www.sptechcon.com	17
SUSE	http://suse.com/storage	7