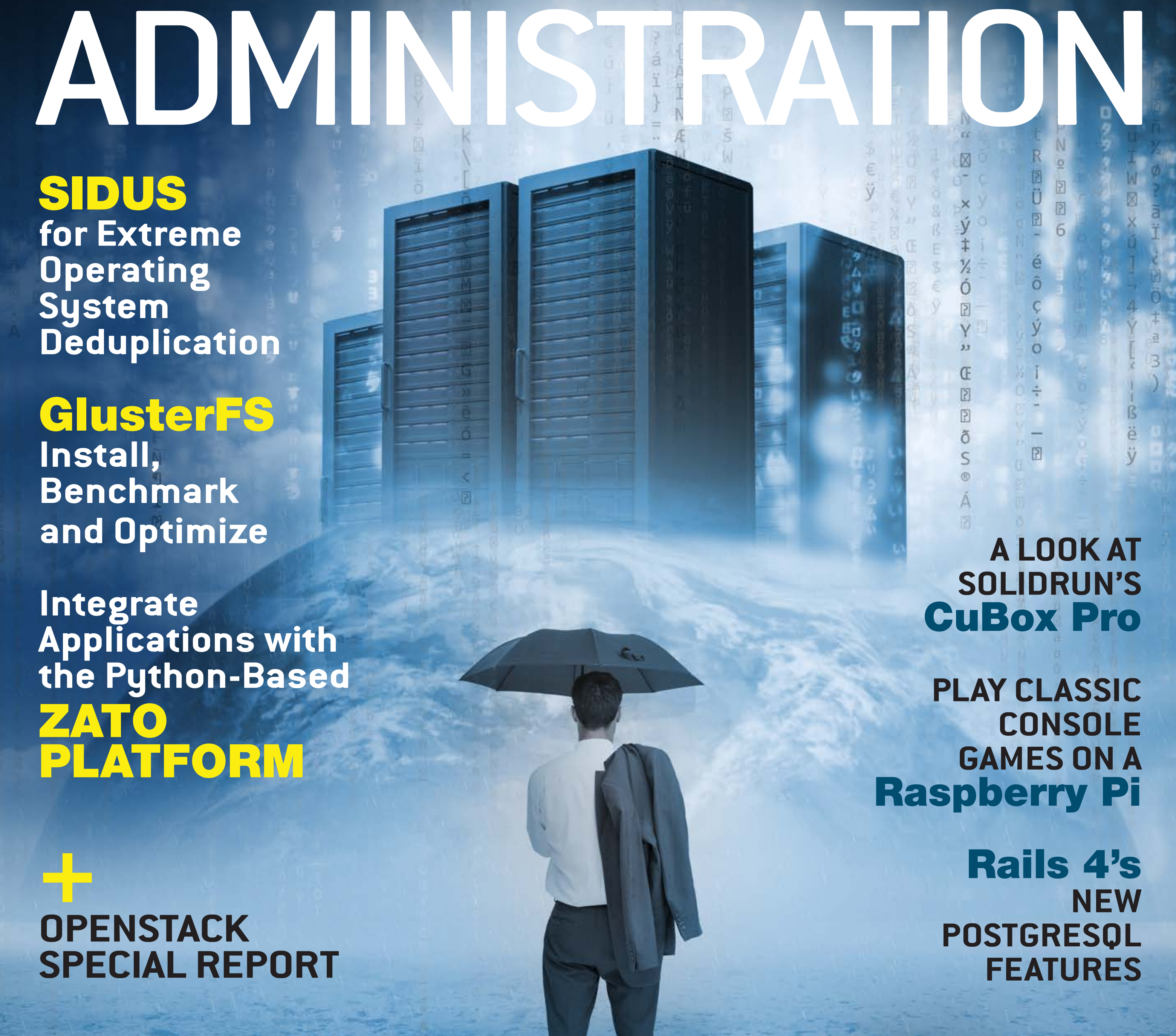
LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

SPONSORED BY New Relic®

NOVEMBER 2013 | ISSUE 235 | www.linuxjournal.com

SYSTEM ADMINISTRATION



SIDUS for Extreme Operating System Deduplication

GlusterFS Install, Benchmark and Optimize

Integrate Applications with the Python-Based **ZATO PLATFORM**

+ OPENSTACK SPECIAL REPORT

A LOOK AT SOLIDRUN'S **CuBox Pro**

PLAY CLASSIC CONSOLE GAMES ON A **Raspberry Pi**

Rails 4's NEW POSTGRESQL FEATURES

Attend the Largest Dedicated Android Conference in the Universe!

AnDevCon SAN FRANCISCO

November 12-15, 2013

Registration Now Open!

Get the best real-world Android developer training anywhere!

- Choose from more than 75 classes and tutorials
- Network with speakers and other Android developers
- Check out more than 40 exhibiting companies

"AnDevCon is a great opportunity to take your Android skills to the next level, get exposed to technologies you haven't touched yet, and network with some of the best Android developers in the world."
—Joe Mitchell, Software Engineer, Quicken Loans

"It's a blast learning and exchanging ideas with phenomenal speakers and cutting-edge experts who have the experience."
—Brad Holmes, Software Developer, uShip

Register Early and Save at www.AnDevCon.com

A BZ Media Event   Follow us: twitter.com/AnDevCon

SILICON MECHANICS

SERVERS • STORAGE • HPC

Are you considering software-defined storage?

zStax
ZFS Unified Storage

zStax StorCore ZFS Unified Storage from Silicon Mechanics is truly software defined storage.

From modest data storage needs to a multi-tiered production storage environment, the **zStax StorCore** ZFS unified storage appliances have the right mix of performance, capacity, and reliability to fit your needs.

zStax StorCore 64

zStax StorCore 104

November Case Study Feature

Expanding Maritime Remote Condition Monitoring Firm Finds a New Storage Paradigm

Talk with an expert today: 866-352-1173
www.siliconmechanics.com/zstax

CONTENTS

NOVEMBER 2013
ISSUE 235

SYSTEM ADMINISTRATION

FEATURES

72 Scale Out with GlusterFS
GlusterFS: install, benchmark and optimize.
Alex Davies and Alessandro Orsaria

84 Zato—Agile ESB, SOA, REST and Cloud Integrations in Python
Integrate applications in an elegant and agile manner.
Dariusz Suchojad

100 SIDUS—the Solution for Extreme Deduplication of an Operating System
Use SIDUS in a variety of environments to provide unprecedented flexibility for both users and administrators.
Emmanuel Quemener and Marianne Corvellec

Cover Image © Can Stock Photo Inc. / 4774344sean

OPENSTACK: SPECIAL REPORT

112 Introduction to OpenStack
A look at OpenStack—the fastest-growing Open Source community in the world.
Tom Fitfield

COLUMNS

36 Reuven M. Lerner's At the Forge
Rails and PostgreSQL

48 Dave Taylor's Work the Shell
Debugging Web Sites

52 Kyle Rankin's Hack and /
Super Pi Brothers

58 Shawn Powers' The Open-Source Classroom
Little Computers, Big Projects

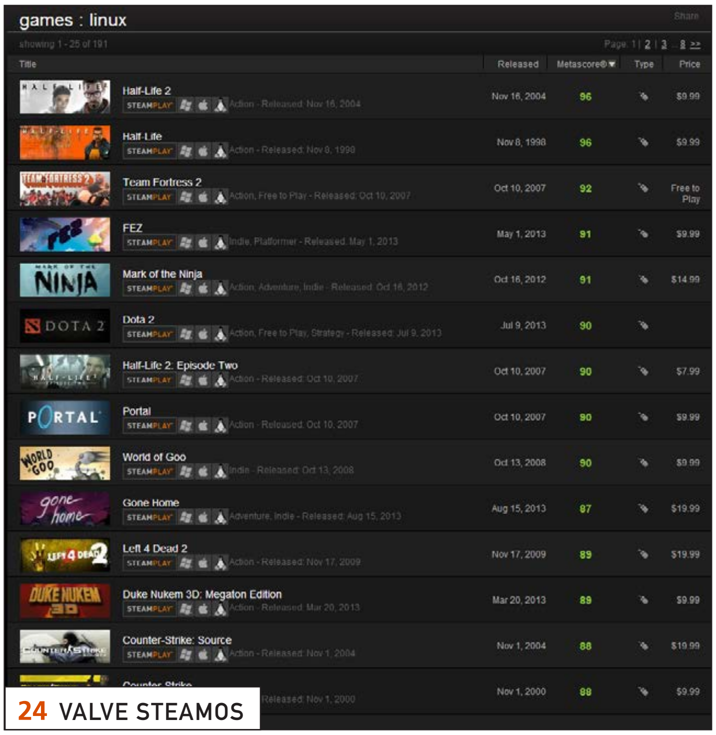
122 Doc Searls' EOF
Life on the Forked Road

KNOWLEDGE HUB

70 Webcasts and White Papers

IN EVERY ISSUE

8 Current_Issue.tar.gz
10 Letters
20 UPFRONT
32 Editors' Choice
66 New Products
125 Advertisers Index



games : linux

showing 1 - 25 of 191 — Page: 1 | 2 | 3 ... 8 >> — Share

Title	Released	Metascore®	Type	Price
Half-Life 2 STEAMPLAY Action - Released: Nov 16, 2004	Nov 16, 2004	96		$9.99
Half-Life STEAMPLAY Action - Released: Nov 8, 1998	Nov 8, 1998	96		$9.99
Team Fortress 2 STEAMPLAY Action, Free to Play - Released: Oct 10, 2007	Oct 10, 2007	92		Free to Play
FEZ STEAMPLAY Indie, Platformer - Released: May 1, 2013	May 1, 2013	91		$9.99
Mark of the Ninja STEAMPLAY Action, Adventure, Indie - Released: Oct 16, 2012	Oct 16, 2012	91		$14.99
Dota 2 STEAMPLAY Action, Free to Play, Strategy - Released: Jul 9, 2013	Jul 9, 2013	90		
Half-Life 2: Episode Two STEAMPLAY Action - Released: Oct 10, 2007	Oct 10, 2007	90		$7.99
Portal STEAMPLAY Action - Released: Oct 10, 2007	Oct 10, 2007	90		$9.99
World of Goo STEAMPLAY Indie - Released: Oct 13, 2008	Oct 13, 2008	90		$9.99
Gone Home STEAMPLAY Adventure, Indie - Released: Aug 15, 2013	Aug 15, 2013	87		$19.99
Left 4 Dead 2 STEAMPLAY Action - Released: Nov 17, 2009	Nov 17, 2009	89		$19.99
Duke Nukem 3D: Megaton Edition STEAMPLAY Action - Released: Mar 20, 2013	Mar 20, 2013	89		$9.99
Counter-Strike: Source STEAMPLAY Action - Released: Nov 1, 2004	Nov 1, 2004	88		$19.99
Counter-Strike Released: Nov 1, 2000	Nov 1, 2000	88		$9.99

24 VALVE STEAMOS

58 SOLIDRUN'S CUBOX PRO

ON THE COVER
- SIDUS for Extreme Operating System Deduplication, p. 100
- GlusterFS: Install, Benchmark and Optimize, p. 72
- Integrate Applications with the Python-Based Zato Platform, p. 84
- OpenStack Special Report, p. 112
- A Look at SolidRun's CuBox Pro, p. 58
- Play Classic Console Games on a Raspberry Pi, p. 52
- Rails 4's New PostgreSQL Features, p. 36

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

LINUX JOURNAL

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

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

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

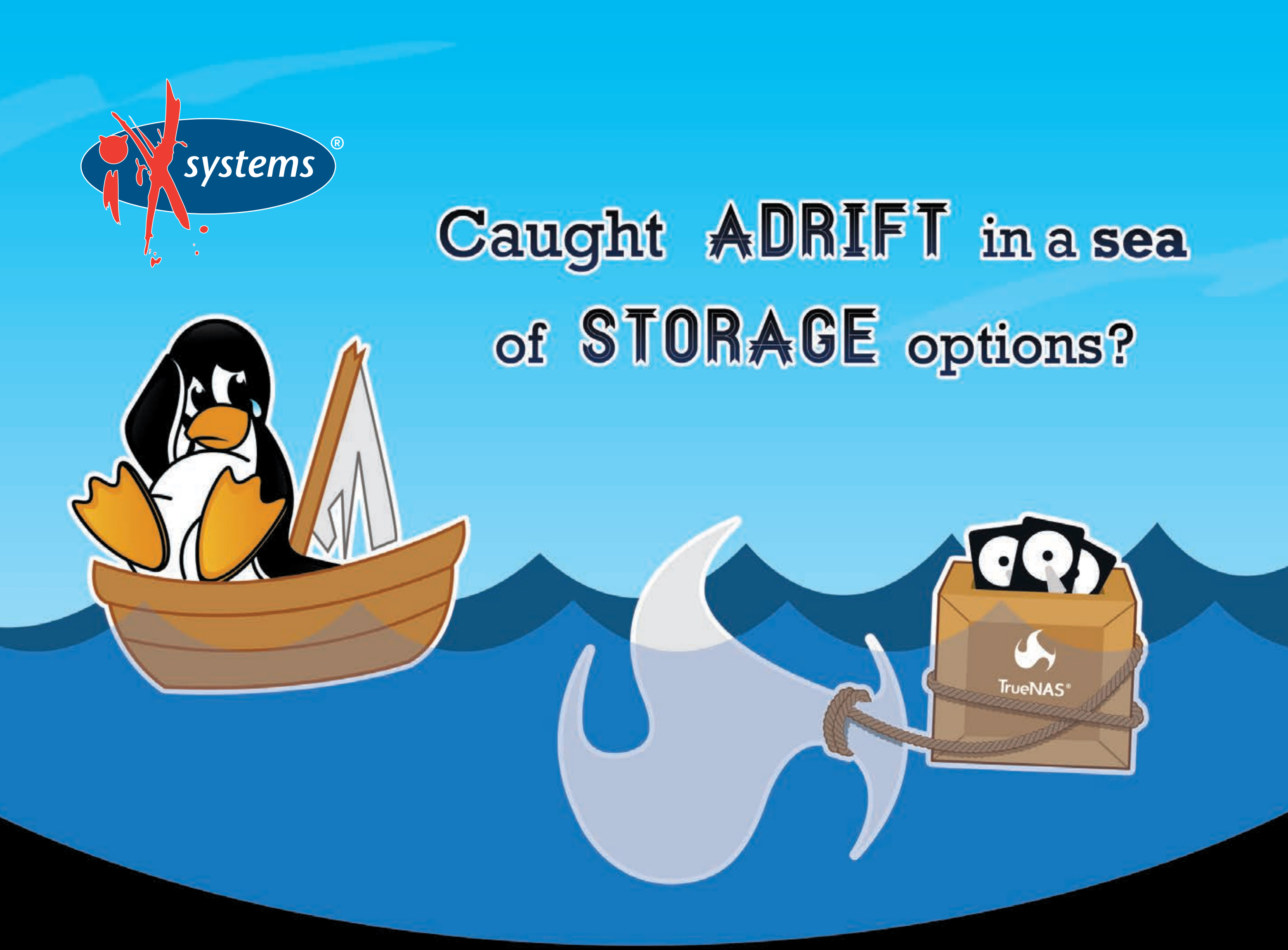
***Linux Journal* is published by, and is a registered trade name of, Belltown Media, Inc.**
PO Box 980985, Houston, TX 77098 USA

Editorial Advisory Panel
Brad Abram Baillio • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faltys • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruizenga • David A. Lane
Steve Marquez • Dave McAllister • Carson McDonald • Craig Oda
Jeffrey D. Parent • Charnell Pugsley • Thomas Quinlan • Mike Roberts
Kristin Shoemaker • Chris D. Stark • Patrick Swartz • James Walker

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

LINUX is a registered trademark of Linus Torvalds.



TrueNAS® has the tools to keep you afloat.

TrueNAS® Unified Storage features the Intel® Xeon® Processor 5600 series and supports high availability, remote replication, deduplication, encryption, compression, and snapshots. It has the tools to deal with any storage challenge you may face.

Key Features:

- Dual Intel® Xeon® Processors 5600 Series
- Support for CIFS, NFS, iSCSI, and more
- Active Directory, LDAP, and NIS integration
- Multi-Petabyte Scalability

intel® inside™ Xeon®

Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX | www.iXsystems.com**

SHAWN POWERS

We Love Sysadmins

Here at *Linux Journal*, we love system administrators. Partially, that's because many of us *are* system administrators, but more than that, we all realize just how important sysadmins are to any organization—and how overlooked they usually are. This year, we decided to make *two* system administration-focused issues, because that's how much we care. (Also because system administration is one of the more popular topics, and a large percentage of our readers do some sort of sysadmin work. But, we do care. Really.)

Reuven M. Lerner starts off the issue with some awesome information about PostgreSQL and Ruby on Rails. With the latest version of Ruby on Rails, PostgreSQL has been integrated more closely than ever before. In a nerdy version of the "your chocolate is in my peanut butter" situation, Ruby on Rails is now far more yummy. Dave Taylor, on the other hand, whips out his scripting Kung-fu to update all the old URLs that broke when updating his content management system. CMSes are wonderful when it comes to delivering content, but moving between systems means dealing with the differences in how they create paths. Rather than repost every article he's ever written, Dave modifies the URLs with scripts. And, he show us how he does it.

Kyle Rankin and I both look at the lighter side of system administration this month. We didn't plan it, but perhaps it's a sign that every sysadmin needs a break from time to time. I covered my adventures with a CuBox computer, which is more powerful than a Raspberry Pi and about the size of a large ice cube. Whether you need a power-efficient server or a sleek XBMC unit with built-in IR, the CuBox is cool enough that I had to share. Kyle takes another stab at the Raspberry Pi this month, and he describes how to turn the little beasties into gaming machines. No, not those fancy new first-person-shooter type games, but *real* games, like the Nintendo Entertainment System. Get your nostalgia primed, and check out Kyle's experience with RetroPie. (And say goodbye to your weekend!)

Alex Davies and Alessandro Orsaria bring us back into the server room with their article on GlusterFS. As cloud computing drives bigger and bigger data centers, sometimes in our own server rooms, GlusterFS is one popular and powerful way to provide a distributed filesystem. Although many of us remember expanding storage by notching out the corner of a floppy disk, these days, petabytes of data can reside seamlessly across multiple servers. Alex and Alessandro explain how. Dariusz Suchojad follows them with his article on Zato, which is a platform for integrating Python into the cloud.

Next up this issue is SIDUS, which stands for Single-Instance Distributing Universal System. Emmanuel Quemener and Marianne Corvellec introduce us to this unique system that drastically reduces maintenance and installation time for new computer workstations. Not quite LTSP and not quite a fat client installation, SIDUS allows for the reuse of an operating system over the network. It's a fascinating concept and well worth the read. Then we finish the issue off with a special report on OpenStack. Every system administrator out there has at least heard of OpenStack, and in this report, Tom Fitfield provides a crash course in the technology. Whether you're an old hand or a newbie, Tom's information is invaluable to anyone interested in cloud computing.

We also have our regular assortment of product announcements, reviews, tips and vendor information in this issue. Heck, we even hand-pick our ads to find information that will be interesting and useful to our readers. Whether you're a system administrator or just an avid fan of Linux, this issue should entertain and inform. As for me, I'm planning to eat dessert first and give RetroPie a try. If my boss complains, I'll just blame Kyle. ■

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

letters

LINUX JOURNAL
Since 1994: The Original Magazine of the Linux Community
SEPTEMBER 2013 | ISSUE 233 | www.linuxjournal.com
A LOOK AT 20 YEARS OF WEB DEVELOPMENT
HOW-TOs
COMMAND-LINE TRICKS AND TIPS
TECHNIQUES FOR HARD DRIVE CACHING
USE VCSH TO MANAGE YOUR CONFIGS
REVIEWED: ACER C7 CHROMEBOOK
gcalcli THE CLOUD ON THE COMMAND LINE
SciDAVis FOR YOUR DATA ANALYSIS TASKS

Nitrous.IO

Thanks for a letter by jschiavon in the September 2013 issue that mentions his use of Nitrous.IO. It's absolutely brilliant. Thanks jschiavon!
—Amit Saha

Article on the Command Line

Regarding Janos Gyerik's article "Time-Saving Command-Line Tricks" in the September 2013 issue: what a let down—simple, not especially useful tips that largely missed the point. For instance, the author assumes vim as the editor of default. A good choice, but if you know vi, then `set -o vi` in bash is far superior than using the default. Then editing a config file: `set  aw` in vim, then from within vi run `:!command` to start the service. It's much better than starting a new window. Did I write the file in the other window? Who cares? `set  aw` does it for you.

I'm afraid the rest of the article was just not especially useful either. How about the whole writing a script with vi and `set  aw`? No need for the save, exit, run, fail, re-open editor cycle. Pah! UNIX is God's gift, and the command line is fantastic, but this was no exposition of that.
—Graham Nicholls

Janos Gyerik replies: *I'm glad that you already master this stuff. I wrote the article because too often I see folks not using half of these moves. It's not fun to watch them crawl when they could fly. As for vim, I didn't assume anything. If you look closely, the focus there is on the screen utility and how it lets you juggle multiple different activities.*

Graham Nicholls replies again: *You've entirely missed my point. These tips are okay, but nothing special, and in some cases, they are just plain wrong. In fact, having* *assumed the use of vi(m), which you have (your mention that less works like vi, for instance),* `set -o vi` *at the bash prompt is infinitely more powerful than using the default Ctrl-r and so on. You make the point that less is somewhat vim-like, but you miss that the shell can be also. I think you're selling people short by offering something that is less than optimal. I'm sure screen is very useful, but having a decent tabbed terminal emulator (konsole, for example in X; mremote or poderosa in the Windows environment) renders it somewhat irrelevant to me.*

As a UNIX user of >25 years, yes, I have mastered this stuff, but I also teach for a large international training organisation (in addition to my day job). In that role, I urge my students that if they expect to use UNIX (read Linux) every day, learning vi is a good idea. Then I show them the power of `set -o vi` *(at the bash prompt), which leverages the power of vi in command-line editing. Finally, I show them how using vi with autowrite switched on makes writing (for instance) scripts or config files very easy.*

Suggesting that people have multiple screens open and write the file in vim on one and then restart the dæmon in another is just plain bad. Rather, show them how in vi they can run a shell escape, which with autowrite switched on, will write the file first, then call the shell-escaped command. No more "did I save the file in the other session?" nonsense. No more "how do I get back to

Server in a Box

Compact SIB

- **Fanless x86 1GHz CPU**
- **Up to 512MB on-board RAM**
- **Up to 1GB Internal Flash Disk or Up to 64GB Compact Flash**
- **10/100 Base-T-Ethernet**
- **Compact Flash Socket**
- **Micro SD Flash Socket**
- **3 USB 2.0 Host Ports**
- **2 RS232 Ports**
- **15-pin D-type female VGA connector**
- **On Board AC 97 Audio line in and mic out**
- **PS/2 Keyboard & Mouse Port**
- **On Board Mini-PCI connector**
- **Dimensions: 4.53" x 4.53" x 1.38"**
- **Internal 2.5" Hard Drive (Optional)**
- **Internal Wireless 802.11g w/ 4dbi antenna (Optional)**
- **Wireless Antenna remote magnet mount 7dbi and remote 15dbi directional (Optional)**

2.6 KERNEL

The EMAC Compact Server-in-a-Box is an energy efficient small footprint server. Like all EMAC SIBs, the standard Compact SIB has no moving parts and features a rugged enclosure design. Qty 1 price starts at $223.

www.emacinc.com/linux_journal_nov

Since 1985 OVER 28 YEARS OF SINGLE BOARD SOLUTIONS

EMAC, inc. EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • www.emacinc.com

the session where I was editing the file" either. In my experience, and opinion, your method will lead to errors and misunderstandings. Restart the dæmon without saving the file = confusion.

Another advantage that vim has in this instance is that you get a history of shell escapes that you can edit, using vi, from within the file—now that's how to write a config file. Using screen by comparison is like using a slide rule for computing advanced calculus. It's perfectly possible, but why would you?

***Janos Gyerik replies again:** To you, maybe, the tips are nothing special. Believe it or not, I see many seasoned Linux users not using these moves. Using Ctrl-r is a great improvement over pushing the up arrow 20 times. Using Ctrl-a is a great improvement over using the left arrow to get the cursor all the way to the start. And using screen on a remote server is a great improvement over opening two terminal tabs to the same remote server. You gotta give me that.*

I did not assume that readers use vim. The text talks about editing a configuration file, and then in the example command, I used vim. Replace it with your favorite editor, and the text still makes sense. I mention vim a second time later to highlight the similarities with the shortcuts in the less command. That's just a small remark—not knowing vim is no disadvantage for the reader. Vim was not the message of the article. Replace it with pico, nano or emacs, and the message still will be the same.

You say that `set -o vi` *is "infinitely more powerful" than the tricks I highlight. Sure, and C is infinitely more powerful than bash, isn't it? But sometimes a bigger gun is neither the most practical nor the most ergonomic. Vim is my favorite editor, and I love to be efficient on the command line, but* `set -o vi` *is not practical for me. I think* `set -o vi` *is too advanced for the vast majority of Linux users of all levels.*

Regarding screen, note that I emphasized using it specifically when working on remote servers. Search for the term "remote" through the article, and you will see. If you take a look at the last paragraph about screen, I explicitly discouraged using it locally, and recommended tabbed *terminal emulators instead.*

You talk a lot about editing config files and restarting dæmons, but that's missing the point. Those were example activities you might do on a remote server. The point is doing many things at the same time. Read a man page in one window. Tail a log file in another. Compile source code in a third. Run a debugger in a fourth, and so on. You could replace the editing config and restarting dæmon examples with these or other activities, and the article still makes sense. What you do in the examples was not the point. The point was that you can do many things.

DDOS—Sendmail in Script

To Dave Taylor, regarding his article "Web Administration Scripts—Redux" in the September 2013 issue: just wondering, you say that `-s` for subjects is not supported by the sendmail client, but you can, of course, create a text-template-email where you can support all the (data-part smtp) headers you want, including a subject. When your instance of the template is ready, just feed it through a pipe to sendmail, and you've got everything you want, right?

Or, it is always possible I'm not understanding what you're doing, of course, in which case I'd ask you to discard this.

I appreciate your monthly article, Dave. Keep it up.
—Lieven

Dave Taylor replies: *You're absolutely correct! Indeed, it'd be easy to have something like "sendmails" that simply accepted two arguments, $1 as the recipient and $2 as the subject:*

```
(
echo "To: $1"
echo "Subject: $2"
echo ""
cat - ) | sendmail
```

Then, you'd invoke it as:

```
sendmails lievendp@gmail.com "this is a sample sendmail wrapper"
```

Is it elegant? Nah. But it'd work.

Thanks for your e-mail and for reading my column!

Oculus Rift First Impressions and Official Linux Support

You should check out this editorial and video: **http://www.gamingonlinux.com/articles/oculus-rift-first-impressions-and-official-linux-support.2133**.
—Liam Dawe

Dangit, Liam. I've never wanted to try a piece of hardware so bad!
—Shawn Powers

Reading Suggestions for Jeff Regan

In the September 2013 issue's Letters, reader Jeff Regan laments reading *LJ* on a tablet. He is right; tablets are terrible for reading. I highly recommend the B&N Nook Simpletouch or other E Ink device for reading. I have rooted mine (Android 2.1, which is enough to use Anki, read PDFs and do light Web browsing), and it has redefined the way I read. I cannot recommend it highly enough.
—Dotan Cohen

My last magazine-worthy tablet was an original Kindle DX, and although it made large format files readable, its speed (or lack thereof) made it frustrating. I'm guessing newer devices have better refresh rates and cleaner displays. I'd still miss the color though. Color E Ink, please come quickly!—Shawn Powers

Coarse Language

The language in the title of Doc Searls' article summary on Linux is entirely inappropriate [see Doc's "Linux vs. Bullshit" in the September 2013 issue]. Each of our acts moves standards and behaviour up or down. This moves things down.

Not good.
—Bryan Batten

***Doc Searls replies:** My main source for the column was* On Bullshit, *an academic work by Harry Frankfurt, a professor emeritus of philosophy at Princeton. It is, as far as I know, the only deep investigation of a vernacular English word that carries a high degree of meaning and for which there is no synonym. That it's also vulgar does not give us cause to avoid it. Could be that understanding it better might help us create a true synonym that isn't vulgar.*

*I believe Linux is also vernacular (though hardly vulgar) and visited this possibility in a column titled "The New Vernacular" in 2001 (***http://www.linuxjournal.com/article/4553***).*

Renewed Subscription

After many years of reading the paper version of *LJ*, I sadly had to let my subscription lapse when you went all electronic. It simply didn't fit with where the magazine went—for example, in the car, by the pool, cricket pitch and various others places whilst waiting for numerous kids to finish whatever they were doing. Although I had access to computers, none were portable enough for this use, and I couldn't justify the cost of another one. Yesterday, my new Nook arrived for £79 delivered—bargain.

But you never forgot me, sending me regular e-mail messages and inviting me back, so last night, I downloaded the free version and found all my favourite authors still there to entertain and inform us. I renewed immediately, of course. Thank you, and all the best for the future.
—Roger Greenwood

Roger, welcome back! Our original switch to digital was at a rough time in technology's evolution. There were tablets, but they weren't mainstream enough (or affordable enough). I'm glad to hear the Nook is working well for you.—Shawn Powers

September Letter "Electronic vs. Paper"

First, I am glad I renewed my subscription to *LJ*. I was a longish-time paper magazine subscriber. I do have some sympathy with those who would prefer a paper copy, but the copies on my hard drive do take up less room.

Anyway on to business—I am a member of the Radio Society of Great Britain (RSGB), which runs an excellent Web site full of useful news and radio-amateur-centric information, which practice, I believe, *Linux Journal* has copied successfully. However, the RSGB also brings out in the autumn (for Christmas) a "Year Book" that repeats much of the technical specifications found on the Web site plus relevant articles on things radio, such as the advantages of building receiver type X over receiver type Y and so on.

I just wanted to add this to any thoughts *LJ* may have in this direction. I certainly look at my copy a number of times in the year, as it is just so much more convenient and quicker than using the desktop. I know you do an archive CD, but a book makes a better present.
—Roy Read

*Gift giving—that's an interesting notion. I agree it's difficult to wrap 1s and 0s. Although I can't promise anything, thank you for the idea. It's worth considering for sure.
—Shawn Powers*

How Would the World Be According to Free Software?

I'm Rafael Cervera, Communication Manager of **http://www.portalprogramas.com**. We have just published "A world inspired by the philosophy of free software", an infographics article about Software Freedom Day, with thoughts about what would happen in a world ruled by the free software philosophy, such as:

- Affordable health care for all due to no patents for medicines.
- Real open government, so Wikileaks wouldn't be needed.

Also, you'll be able to find some fun facts and advantages of free software that will amuse your readers for sure, such as Coca Cola's secret recipe would be public.

You can see it and get more information at **http://www.en.portalprogramas.com**.

You are free to share this work as you wish; it is under a Creative Common license.

I remain open to any feedback. Thank you very much for your attention.
—Rafael Cervera

Thank you for the link and for the information. The geek in me instantly thought of the Star Trek *world, where monetary gain is no longer the driving force of humanity. Is it possible outside fiction? I don't know, but again, I appreciate the link. Very cool.—Shawn Powers*

Reuven M. Lerner's Column on the Historical Web

To Reuven M. Lerner: I really enjoy reading your column in *Linux Journal*. It is one of the columns that I can say, "he knows what he is talking about."

However, in the September 2013 issue, when you look back over Web development history, you didn't mention any Microsoft technologies. I am not affiliated with the company, and I know it's not the magazine's main target, but for a historical column, I think Microsoft deserves to be mentioned. With its Web-develop NET frameworks, it evolved and diverged from the company's vision and become more and more open source. It's still investing on the cloud with Azure, and I think it actually moved the bar up for every technology provider.

Regardless, I enjoy your column. I think it would have been a lot

nicer if you were more objective like a historian.
—Can

Reuven M. Lerner replies: *Thanks for your message. I often tell people that there are open-source people who hate Microsoft, and then I explain that I'm not one of them. Rather, I say I'm an open-source person who more or less ignores Microsoft—that I know it's a major (indeed, dominant) force in the computer industry, but that it doesn't directly affect my day-to-day life, and that I'm rather ignorant about its technologies.*

There are oodles of smart people working at Microsoft, and they have clearly done a lot to advance Web technologies. But I must admit that I know nothing, or almost nothing, about those technologies.

So for me to have written about Microsoft technologies would have undoubtedly demonstrated my ignorance, and/or required research that I don't have time to do. That said, I completely accept what you're saying, and I probably should have noted that the column was based on my personal experience, and the technologies I personally used—meaning that I undoubtedly left out a lot of influential and important ones.

It's always great to hear from people who read my columns, even when you're being critical. Thanks again for writing, and always feel free to let me know what you think, or if you have any suggestions for future topics.

Dave Taylor's DDOS

I'm wondering if Dave Taylor has tried ModSecurity to fight his DDOS [see Dave's column in the August and September 2013 issues]. I'm evaluating it for my own use, and it seems like it could help. If the attacker is placing something unusual in the HTTP headers, like a special user-agent string, ModSecurity can be used to drop the connection before the request is processed.

There is a reasonably good e-book about it at **http://www.packtpub.com**.
—Paul Gassoway

Dave Taylor replies: *Thanks for your suggestion, Paul. As it happens, I ended up just moving to a new hosting company that has a team focused on helping out with these sort of problems after all!*

Work the Shell Idea

As a blogger who deals with climate and weather, sometimes I like to know right away if a major sudden event is happening, such as a tsunami. The National Weather Service has a tsunami alert system (@NWS_WCATWC) on Twitter. Whenever there is an earthquake, and there is no tsunami expected, they tweet something that includes "Tsunami NOT expected", but when there is a tsunami expected, they tweet a different string (along with info on the event).

I would like to see a script that regularly checks Twitter (as a chron job) for reliable sources (such as the NWS specialized feeds and so on) for certain key phrases, and if there is an event indicated, then to search Twitter for an appropriate hashtag (such as #tsunami and #[place it is happening]). The script then should send me an alert of some kind, so I can look at the list of tweets to figure out what is going on.

This is a little like the DDOS script, because if you just search for #tsunami on Twitter, there would be mostly junk most of the time, but when there is an official alert, the #tsunami hash tag would include a lot of actually useful information.

A new script by Dave that uses the current Twitter API would be useful, as that changes now and then, and a script that includes parsing a select Twitter feed, responding to that feed by then parsing other feeds, would be intelligent(ish) and a great demonstration of how to use a remarkable 20th-century technology (bash, Linux) to solve a problem in a very 21st-century way.

—Greg Laden

Thanks for the Unicode Article

To Reuven M. Lerner: I am a systems administrator, and to be honest, I normally skip over your articles in *Linux Journal*. However, I've got to give you some credit. The article on Unicode in the June 2013 issue was spectacular.

You clearly and concisely explained Unicode, and for the first time, I understand how UTF-8 works and why it has become so popular.

Thanks for the great article!

—Carl

Reuven M. Lerner replies: *Thanks for the warm feedback; I'm delighted to know that you enjoyed the article!*

Photo of the Month

Enjoy.

—Can Erten

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

PHOTO OF THE MONTH
Remember, send your Linux-related photos to ljeditor@linuxjournal.com!

LINUX JOURNAL At Your Service

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

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

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

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

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

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

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

The **stable kernel series** recently got bit by an over-eagerness to accept patches. Typically, the rule had been that **Greg Kroah-Hartman** wouldn't accept patches that had not gotten into **Linus Torvalds**' development kernel first. People sometimes complained that this was absurd, since the patches in question always were intended for the stable tree, and so it seemed to make no sense to wait for them to be accepted into a completely different, development-centric tree. But the rule existed to ensure that the good and useful stabilization patches wouldn't go only into the ephemeral stable releases, but also would be incorporated into the long-term future of the kernel (that is, Linus' tree).

Nevertheless, sometimes people complained. And, Greg did try to take patches as soon as they crossed the threshold into Linus' git repository. But recently he did this with a **genetlink** patch that seemed fine at first, but it ended up having a bad interaction with other code that wasn't caught until after the 3.0.92 stable kernel came out.

Greg viewed this as a bit of a wake-up call to slow down on accepting patches until they were not only in Linus's tree, but also had a chance to be tested for the duration of one -rc cycle. That would provide a much better chance of shaking out any bugs and keep the stable kernels as stable as one might reasonably expect.

Linus pointed out that tying the waiting period to an -rc cycle was fairly arbitrary. He said waiting a week or so probably would be fine. He also added that for particularly critical or well-discussed patches, a delay probably would be unnecessary.

Greg posted a more formal description of his new patch-acceptance policy, which also specified that if an official maintainer signed off on the patch, or if Greg saw a sufficient amount of discussion about it, or for that matter, if the patch was just obvious, then the delay for acceptance could be skipped.

Greg's initial proposal also included his idea of waiting for a full -rc cycle. **Nicholas A. Bellinger** had no problem with that, and **Willy**

Tarreau said that even waiting for a full dot release would be fine. But Greg vetoed that, saying three months would be too long.

Josh Boyer offered to help out in some way, and Greg said it would be useful to know when things were broken. He said, "Right now, that doesn't seem to happen much, so either not much is breaking, or I'm just not told about it, I don't know which." So Josh volunteered to send any such information he collected as part of his work on **Fedora** to Greg. Greg said this would be great. Greg also suggested that patches going into distribution kernels also should be sent to him as a matter of course, "as those obviously are things that are needed for a valid reason that everyone should be able to benefit from."

At some point in the discussion, Linus came back and again recommended that Greg just wait a week to accept patches, rather than wait for an -rc release. Linus pointed out that some people did test the bleeding-edge git kernels, and there at least should be some encouragement for them to keep doing so.

This led to a discussion of whether people actually should run the git kernels in between -rc releases. **Borislav Petkov** suggested that at any point in time the raw git tree would be fairly arbitrary and possibly broken. But, **Tony Luck** said that if the git tree never got any testing, all the arbitrariness and breakage simply would cascade forward into the -rc release.

It does seem as though Greg will be more careful about when patches go into the stable tree, and hopefully, some new people will start testing the raw git kernels.

The **device tree** (DT) bindings have begun a major overhaul. **Grant Likely** announced that maintainership would be changing to a group model to keep up with the pace of change in that area. He asked for volunteers to join in, and he said that the entire schema for specifying DT bindings will be revised.

DT bindings are generalized, machine-parsable descriptions of certain aspects of hardware. They make it possible for the kernel code to be somewhat more generic and avoid the need to hard-code a lot of hardware-specific details. It makes for a much cleaner and more readable kernel.

Grant said that **Tomasz Figa** was going to be leading the work on revising the DT schema. This, he said, hopefully would result in something that could be validated programmatically, while at the same time being human-readable, in order to double as documentation.

The development process itself also will change. At the time of Grant's announcement, the process had become too frantic, and there was no distinction between DT bindings that were clearly stable and those that were in flux and still under development.

Tomasz suggested having a clear process for migrating any given DT binding from a staging area into a stable area. Once stable, he said, they should have the sanctity of an application binary interface (ABI), which typically must never be allowed to break backward compatibility.

David Gibson also was very interested in this project. He'd made an earlier attempt at schema validation, and he wanted to see if Tomasz could fare better. He also pointed out that there was a perception problem among users, and that they tended to think that a given DT binding—as being distributed in the kernel—was valid for that kernel and no others. But David said this was not the case, and that some effort should be made to inform users that DT bindings are intended to be generalized descriptions of hardware and not tied to any particular kernel version. But, he wasn't sure how that could be done.

The discussion took many paths. At some point, someone suggested using **XML** with a corresponding document type definition as the DT schema. There was some support for this, but Tomasz didn't like mixing XML with the existing DT schema format.

The overarching discussion was open-ended, but clearly there are big changes on the way, regarding the DT-binding specifications.

—ZACK BROWN

They Said It

It is wonderful how quickly you get used to things, even the most astonishing.
—Edith Nesbitt

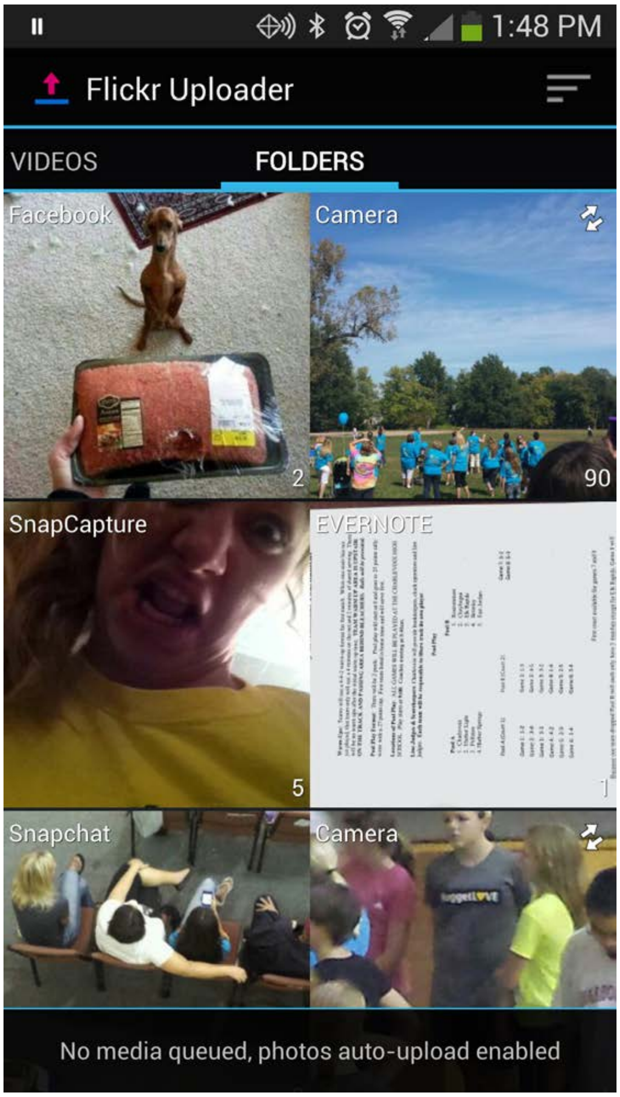
There is nothing like the occasional outburst of profanity to calm jangled nerves.
—Kirby Larson,* Hattie Big Sky, *2006

The art of being wise is the art of knowing what to overlook.
—William James

Laughter is inner jogging.
—Norman Cousins

Facing it, always facing it, that's the way to get through. Face it.
—Joseph Conrad

Android Candy: Flickr Uploader



No media queued, photos auto-upload enabled

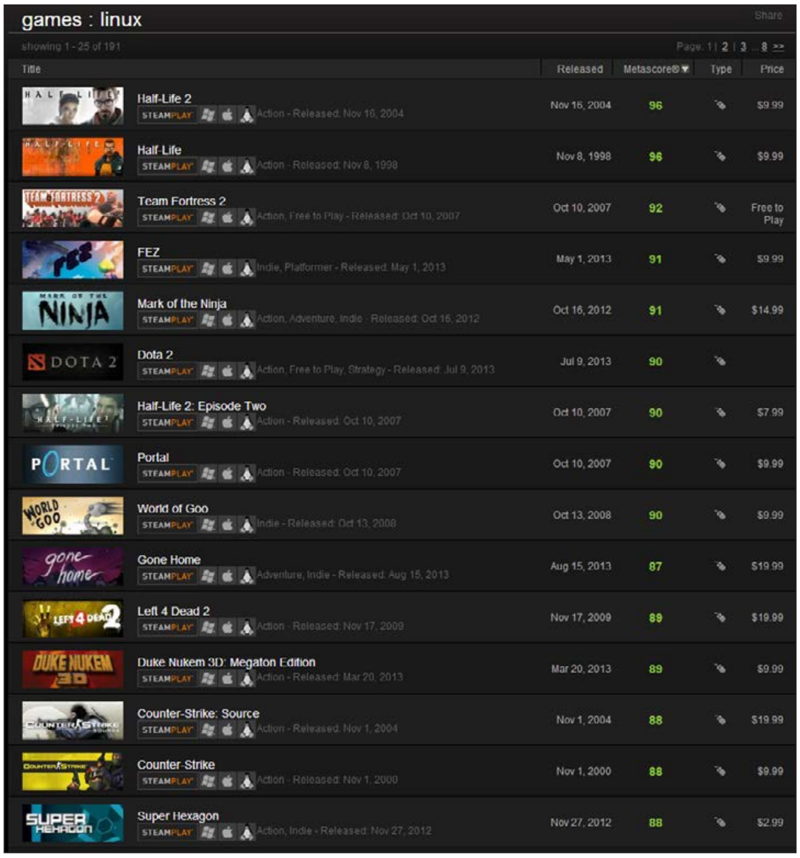
As luck would have it, shortly after I purchased a Flickr Pro subscription, Yahoo decided to eliminate the pay-for option with Flickr and give everyone a free Terabyte of space to store photos. Although I'm still not convinced Flickr is the best way to store photos, I have found it to be very flexible, so for now, my family members dump all their photos to Flickr from all their devices. (And then I back them up with **https://pypi.python.org/pypi/flickrbackup** because I'm paranoid.)

On my phone, a Galaxy S4, I find it cumbersome to upload photos directly to Flickr, so I bought Flickr Uploader from the Google Play store, and now all my photos are uploaded to Flickr instantly when I take them. Options are available to limit uploading to Wi-Fi only, or to upload only certain kinds of photos (camera photos but not screenshots, for example). The application is $4.99, but the developer has created "coupons" to get the application for a huge discount or even free (**https://github.com/rafali/flickr-uploader/wiki/Coupons**).

I find the application to be smooth, stable and incredibly convenient. It's important to realize, however, that if you're the sort of person to take personal or compromising photos, double- and triple-check your settings to make sure you don't upload publicly viewable shots automatically! If you're a Flickr user, check out Flickr Uploader today: **https://play.google.com/store/apps/details?id=com.rafali.flickruploader**.—**SHAWN POWERS**

Valve—It Really *Does* Love Linux

I've teased about Steam, speculated about Steam and even bragged about Steam finally coming to Linux. Heck, check out the screenshot for just a partial list of games already running natively under our beloved OS. Little did I know that the folks at Valve not only planned to support Linux, but they're also putting a big part of their future behind it as well!



games : linux

Share

showing 1 - 25 of 191

Page: 1 | 2 | 3 ... 8 >>

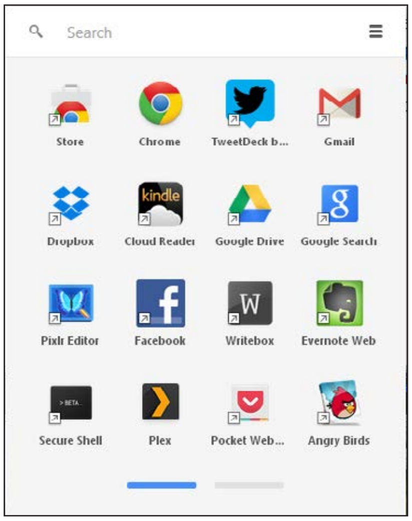
Title	Released	Metascore®	Type	Price
Half-Life 2 STEAMPLAY Action - Released: Nov 16, 2004	Nov 16, 2004	96		$9.99
Half-Life STEAMPLAY Action - Released: Nov 8, 1998	Nov 8, 1998	96		$9.99
Team Fortress 2 STEAMPLAY Action, Free to Play - Released: Oct 10, 2007	Oct 10, 2007	92		Free to Play
FEZ STEAMPLAY Indie, Platformer - Released: May 1, 2013	May 1, 2013	91		$9.99
Mark of the Ninja STEAMPLAY Action, Adventure, Indie - Released: Oct 16, 2012	Oct 16, 2012	91		$14.99
Dota 2 STEAMPLAY Action, Free to Play, Strategy - Released: Jul 9, 2013	Jul 9, 2013	90		
Half-Life 2: Episode Two STEAMPLAY Action - Released: Oct 10, 2007	Oct 10, 2007	90		$7.99
Portal STEAMPLAY Action - Released: Oct 10, 2007	Oct 10, 2007	90		$9.99
World of Goo STEAMPLAY Indie - Released: Oct 13, 2008	Oct 13, 2008	90		$9.99
Gone Home STEAMPLAY Adventure, Indie - Released: Aug 15, 2013	Aug 15, 2013	87		$19.99
Left 4 Dead 2 STEAMPLAY Action - Released: Nov 17, 2009	Nov 17, 2009	89		$19.99
Duke Nukem 3D: Megaton Edition STEAMPLAY Action - Released: Mar 20, 2013	Mar 20, 2013	89		$9.99
Counter-Strike: Source STEAMPLAY Action - Released: Nov 1, 2004	Nov 1, 2004	88		$19.99
Counter-Strike STEAMPLAY Action - Released: Nov 1, 2000	Nov 1, 2000	88		$9.99
Super Hexagon STEAMPLAY Action, Indie - Released: Nov 27, 2012	Nov 27, 2012	88		$2.99

Valve recently announced SteamOS (**http://store.steampowered.com/livingroom/SteamOS**), which is a Linux-based operating system designed from the ground up to play games. Steam games. The operating system will be free, and it also will be the OS running on Valve's next big thing, SteamMachines (**http://store.steampowered.com/livingroom/SteamMachines**). Although the hardware won't be available until sometime in 2014, the OS should be out before then.

Will Valve be the company to bridge the gap between computer gaming and console gaming? Will PC games translate to a television screen smoothly? Will the SteamOS's "game streaming" technology effectively bring the entire Steam library to Linux? I've learned enough not to make any predictions, but I can tell you it's an exciting prospect, and it's even more exciting because it's all running on Linux!

—SHAWN POWERS

Non-Linux FOSS: Chrome Desktop Applications



Hopefully by the time you're reading this, Chrome Desktop Applications will be available for Linux. In the meantime, this is a Windows treat. The ability to make a "single-purpose" browser has been around Chrome/Chromium for a long time, but with the new breed of Chrome Applications, the browser is a base for a standalone, off-line application. According to Google, the new Chrome apps will have the following features (from **http://snar.co/chromeapps**):

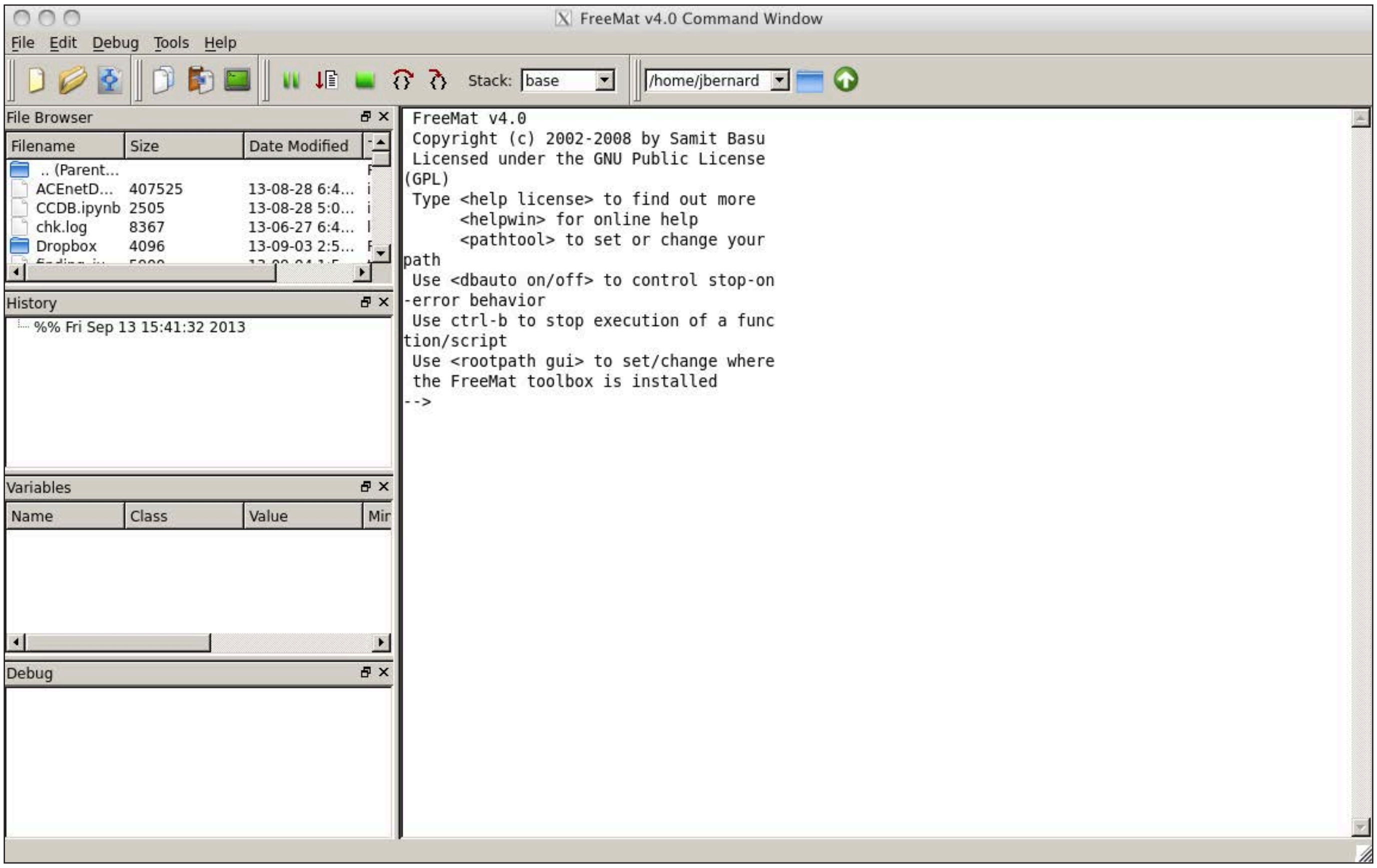
- Work off-line: keep working or playing, even when you don't have an Internet connection.
- More app, less Chrome: no tabs, buttons or text boxes.
- Connect to the cloud: access and save the documents locally and in the cloud.
- Desktop notifications: you can get reminders, updates and even take action, right from the notification center.
- Local device support: interact with your USB, Bluetooth and other devices.
- Automatic updates: apps update silently (unless permissions change).
- Chrome App Launcher: appears on the taskbar when you install your first new Chrome App.

Chrome dabbles with off-line abilities with many of its current Web applications, but with the new Chrome Apps, this should go to an entire new level. If the hype is correct, these should be local applications, not just Web apps with off-line hooks. Also, although Chrome Apps are Windows-only today, Google promises Linux support in the future.—**SHAWN POWERS**

FreeMat—Yet Another MATLAB Replacement

Many programs exist that try to serve as a replacement for MATLAB. They all differ in their capabilities—some extending beyond what is available in MATLAB, and others giving subsets of functions that focus on some problem area. In this article, let's look at another available option: FreeMat.

The main Web site for FreeMat is hosted on SourceForge. Installation for most Linux distributions should be as easy as using your friendly neighborhood package manager. FreeMat also is available as source code, as well as installation packages for Windows and Mac OS X. Once it's installed, you can go ahead and start it up. This



Filename	Size	Date Modified
.. (Parent...		
ACEnetD...	407525	13-08-28 6:4...
CCDB.ipynb	2505	13-08-28 5:0...
chk.log	8367	13-06-27 6:4...
Dropbox	4096	13-09-03 2:5...

```
%% Fri Sep 13 15:41:32 2013
```

```
 FreeMat v4.0
 Copyright (c) 2002-2008 by Samit Basu
 Licensed under the GNU Public License
(GPL)
 Type <help license> to find out more
      <helpwin> for online help
      <pathtool> to set or change your
path
 Use <dbauto on/off> to control stop-on
-error behavior
 Use ctrl-b to stop execution of a func
tion/script
 Use <rootpath gui> to set/change where
 the FreeMat toolbox is installed
-->
```

Figure 1. When you first start up FreeMat, you are given a fresh console.

brings up the main window with a working console that should be displaying the license information for FreeMat (Figure 1).

As with most programs like FreeMat, you can do arithmetic right away. All of the arithmetic operations are overloaded to do "the right thing" automatically, based on the data types of the operands. The core data types include integers (8, 16 and 32 bits, both signed and unsigned), floating-point numbers (32 and 64 bits) and complex numbers (64 and 128 bits). Along with those, there is a core data structure, implemented as an N-dimensional array. The default value for N is 6, but is arbitrary. Also, FreeMat supports heterogeneous arrays, where different elements are actually different data types.

Commands in FreeMat are line-based. This means that a command is finished and executed when you press the Enter key. As soon as you do, the command is run, and the output is displayed immediately within your console. You should notice that results that are not saved to a variable are saved automatically to a temporary variable named "ans". You can use this temporary variable to access and reuse the results from previous commands. For example, if you want to find the volume of a cube of side length 3, you could so with:

```
--> 3 * 3
ans =
 9
--> ans * 3
ans =
 27
```

Of course, a much faster way would be to do something like this:

```
--> 3 * 3 * 3
ans =
 27
```

Or, you could do this:

```
--> 3^3
ans =
 27
```

If you don't want to clutter up your console with the output from intermediate calculations, you can tell FreeMat to hide that output by adding a semicolon to the end of the line. So, instead of this:

```
--> sin(10)
ans =
   -0.5440
```

you would get this:

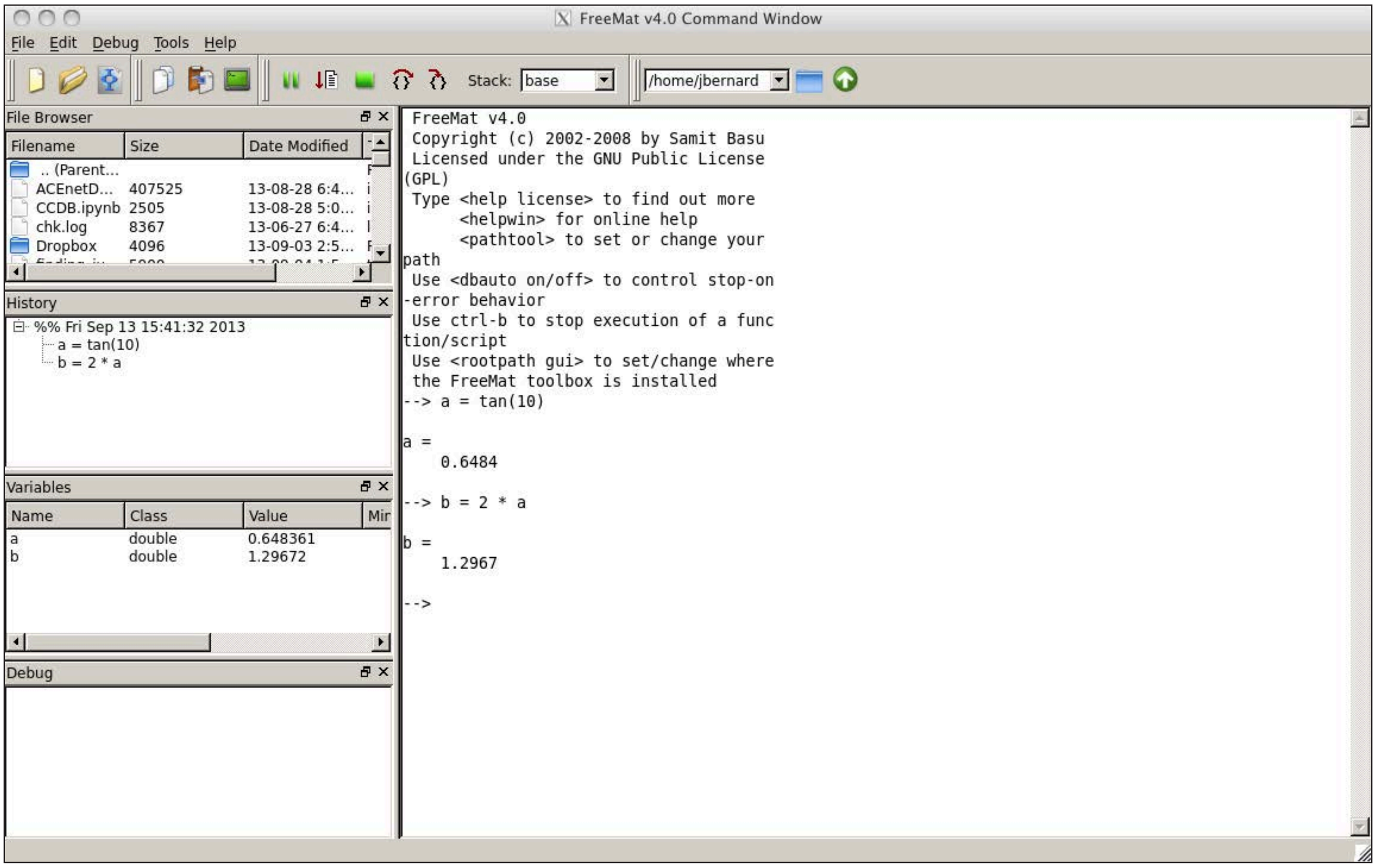
```
--> sin(10);
-->
```

Assignment in FreeMat is done with the equals operator (=). Variable names are untyped, so you can reuse the same name for different data types in different parts of your worksheet. For example, you can store the value of tan(10) in the variable a, and the value of 2 times a in the variable b, with:

```
--> a = tan(10)
a =
    0.6484
--> b = 2 * a
b =
    1.2967
```

Notice that the variables active in your current session are all listed in the variable window on the left-hand side (Figure 2).

The arithmetic operators are



Filename	Size	Date Modified
.. (Parent...		
ACEnetD...	407525	13-08-28 6:4...
CCDB.ipynb	2505	13-08-28 5:0...
chk.log	8367	13-06-27 6:4...
Dropbox	4096	13-09-03 2:5...

```
%% Fri Sep 13 15:41:32 2013
   a = tan(10)
   b = 2 * a
```

Name	Class	Value
a	double	0.648361
b	double	1.29672

```
 FreeMat v4.0
 Copyright (c) 2002-2008 by Samit Basu
 Licensed under the GNU Public License
(GPL)
 Type <help license> to find out more
      <helpwin> for online help
      <pathtool> to set or change your
path
 Use <dbauto on/off> to control stop-on
-error behavior
 Use ctrl-b to stop execution of a func
tion/script
 Use <rootpath gui> to set/change where
 the FreeMat toolbox is installed
--> a = tan(10)

a =
    0.6484

--> b = 2 * a

b =
    1.2967

-->
```

Figure 2. All of the current variables, along with their values and data types, are listed in the variable window.

overloaded when it comes to interacting with arrays too. For example, let's say you want to take all of the elements of an array and double them. In a lower-level language, like C or Fortran, you would need to write a fair bit of code to handle looping through each element and multiplying it by 2. In FreeMat, this is as simple as:

```
--> a = [1,2,3,4]
a =
 1 2 3 4
--> a * 2
ans =
 2 4 6 8
```

If you are used to something like NumPY in the Python programming language, this should seem very familiar. Indexing arrays is done with brackets. FreeMat uses 1-based indexes, so if you wanted to get the second value, you would use:

```
--> a(2)
```

You also can set the value at a particular index using the same notation. So you would change the value of the third element with:

```
--> a(3) = 5
```

These arrays all need to have the same data type. If you have need of a heterogeneous list of elements, this is called a cell array. Cell arrays are defined using curly braces. So, you could set up a name and phone-number matrix using:

```
--> phone = { 'Name1', 5551223; 'name2', 5555678 }
```

There are two new pieces of syntax introduced here. The first is how you define a string. In FreeMat, strings are denoted with a set of single quotation marks. So this cell array has two strings in it. The second new syntax item is the use of a semicolon in the definition of your array. This tells FreeMat that you're moving to a new row. In essence, you're now creating a matrix instead of an array.

Plotting in FreeMat is similar to plotting in R or matplotlib. Graphics functions are broken down into high-level and low-level functions. The most basic high-level function is plot(). This function is overloaded to try to do the right thing based on the input. A basic example would be plotting a sine function:

```
--> t = -63:64;
--> signal = sin(2*pi*t/32);
--> plot(t, signal)
```

This will pop up a new window to contain the results of the plot function (Figure 3). You then can use low-level functions to alter these plots. For example, you can set the title of your graph with the function:

```
--> title('This is my plot')
```

There are low-level functions to alter almost all of the elements of your plots.

One process that FreeMat tries to excel at is making the importation of external code easy on the end user. The import function is used to import a function from an external library and make it available to be used just like any other FreeMat function. The signature of the import function is:

```
import(libraryname,symbol,function,return,arguments)
```

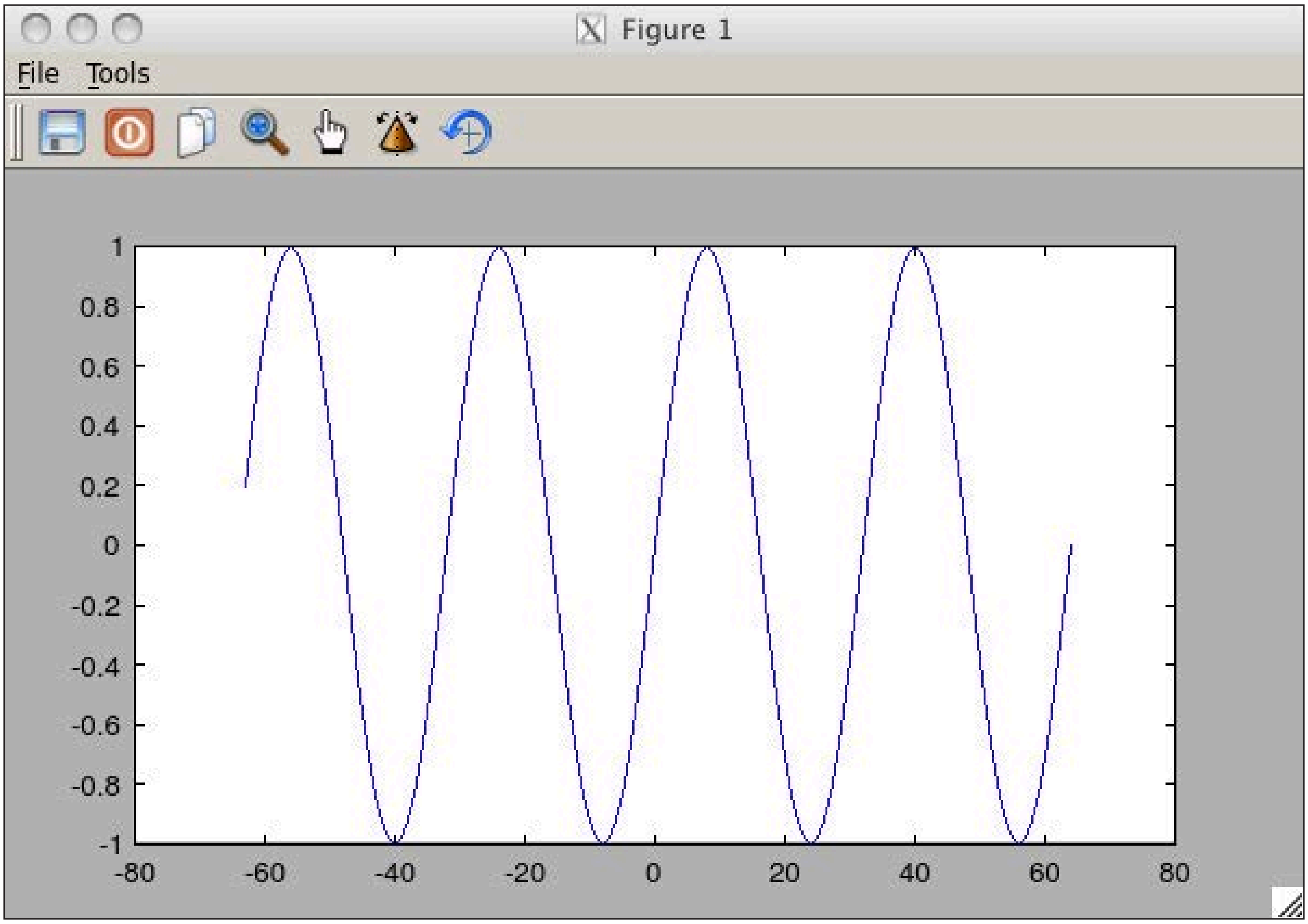


Figure 3. Plots are generated and displayed in their own window.

This way, if you have some piece of code that is already written and tuned in C to get the maximum performance, you simply can import it and use it from within FreeMat with very little fuss.

Since FreeMat is as capable as MATLAB or Octave, you can assume that I've barely covered everything offered by FreeMat here. Hopefully, you have seen enough to spark your interest and will make some time to go and explore even more.

Resources

- Main Web Site: **http://freemat.sourceforge.net**
- FreeMat Tutorials: **https://code.google.com/p/freemat/wiki/Tutorials**
- FreeMat Wiki: **https://code.google.com/p/freemat/w/list**

—JOEY BERNARD

Fight the Good Fight with SmokePing

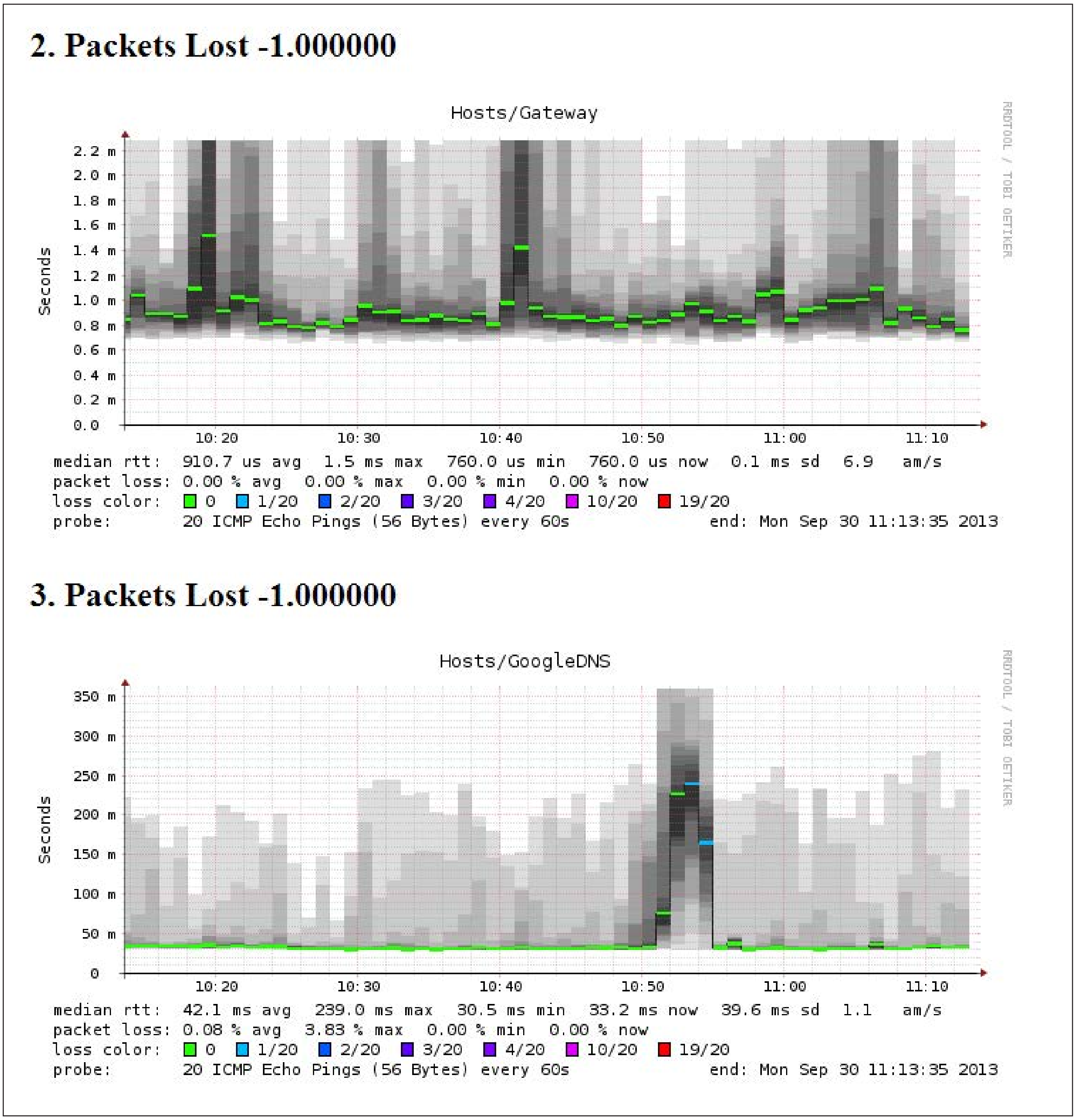
My Internet connection is unstable. I do realize ISPs generally claim some downtime is expected, and service is not guaranteed, and countless other excuses are common for intermittent service. I currently pay $120/month for business-class service, however, and I expect to get reliable Internet access on a regular basis. The most frustrating part is that the folks at my ISP don't believe I'm having intermittent problems, because every time they look, it seems fine.

Enter SmokePing.

With past ISP problems, I've been able to run a continuous ping to an outside IP address and show the tech-support representative that I have packet loss. Unfortunately, a running ping command doesn't give a history of *when* the packets are lost. With SmokePing, not only is there a record of when packets are lost, but there's also a graphical representation of how many packets were lost, and from several IP addresses to boot.

For my purposes, I keep track of pings to my local router, to the gateway provided by my ISP, and then a Google IP address and a foreign IP address. With that information, I usually not only can tell the ISP when the packets drop, but also whether it's an issue between me and its gateway or routing somewhere past my subnet. (For what it's worth, the problem is almost *always* between my router and my ISP's gateway, because there's some problem with its line coming to my office.)

If you need to prove packet loss, or if you just like to keep track of potential problems between hosts, SmokePing is an awesome uptime tracker that comes with colorful graphs and lots of useful information. For its incredible usefulness and straightforward approach to monitoring,

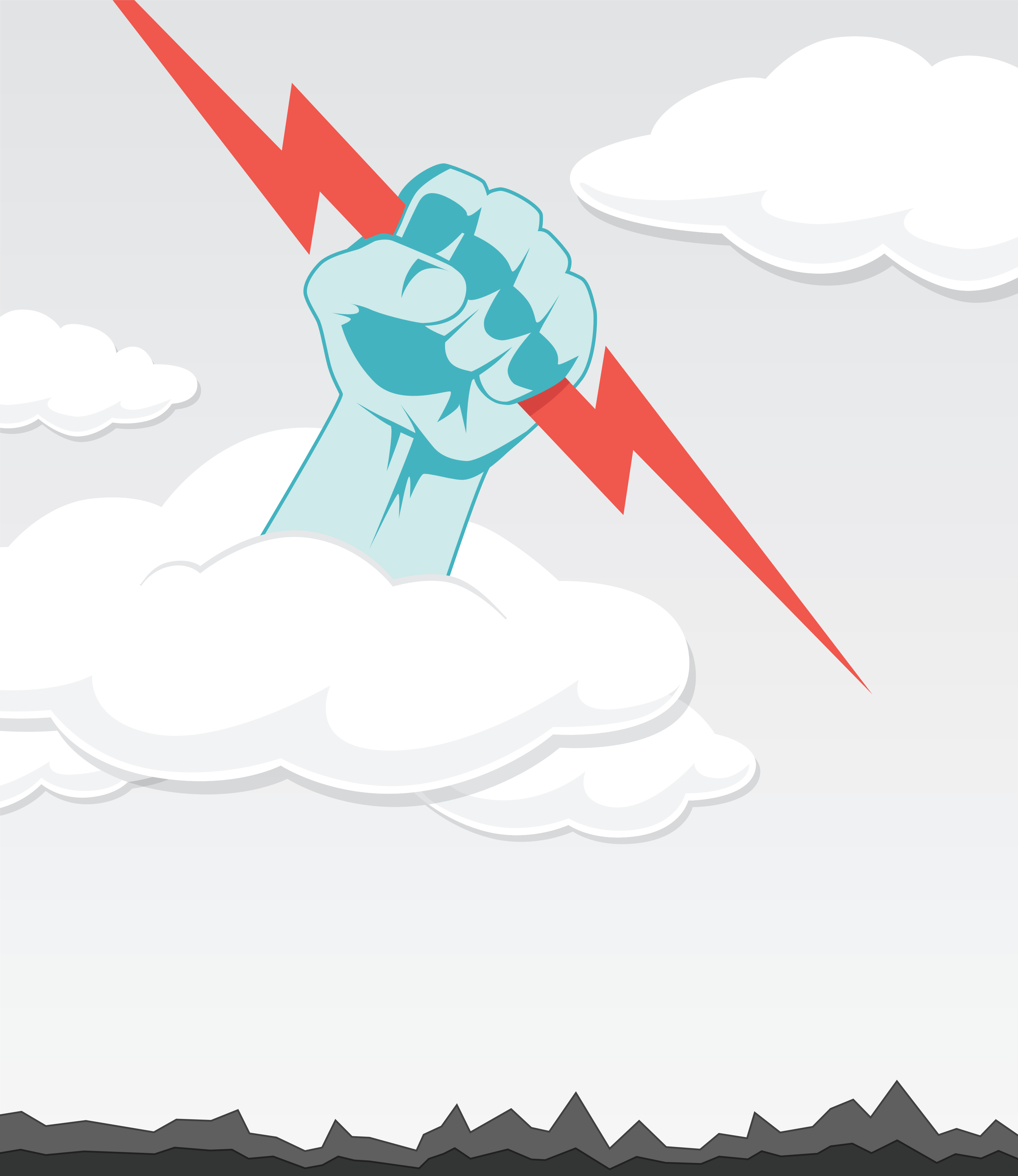


```
median rtt:  910.7 us avg   1.5 ms max   760.0 us min   760.0 us now   0.1 ms sd   6.9   am/s
packet loss: 0.00 % avg   0.00 % max   0.00 % min   0.00 % now
loss color:  🟩 0  🟦 1/20  🟦 2/20  🟪 3/20  🟪 4/20  🟪 10/20  🟥 19/20
probe:       20 ICMP Echo Pings (56 Bytes) every 60s             end: Mon Sep 30 11:13:35 2013
```

```
median rtt:  42.1 ms avg   239.0 ms max   30.5 ms min   33.2 ms now   39.6 ms sd   1.1   am/s
packet loss: 0.08 % avg   3.83 % max   0.00 % min   0.00 % now
loss color:  🟩 0  🟦 1/20  🟦 2/20  🟪 3/20  🟪 4/20  🟪 10/20  🟥 19/20
probe:       20 ICMP Echo Pings (56 Bytes) every 60s             end: Mon Sep 30 11:13:35 2013
```

Notice packet loss to the Google DNS server, but none to my gateway. So, the problem isn't with my house connection.

SmokePing gets this month's Editors' Choice Award. Check it out at **http://oss.oetiker.ch/smokeping**.—**SHAWN POWERS**



THE PINNACLE OF APPLICATION PERFORMANCE MONITORING

Make your cloud even more powerful.

Spot bottlenecks and smite bugs before they slow down app performance!

New Relic gives you powerful code-level visibility into your application performance so you can spot problems and fix them. Fast. Now you can identify and solve problems before they affect your users.

New Relic®

REUVEN M.
LERNER

Rails and PostgreSQL

The latest version of Ruby on Rails now offers closer integration with PostgreSQL.

Regular readers of this column won't be surprised to hear that I love both Ruby on Rails and PostgreSQL. Rails has been my primary server-side Web development framework for about eight years, and it has managed to provide solutions for a large number of consulting and personal projects. As for PostgreSQL, I've been using it for about 15 years, and I continue to be amazed by the functionality it has gained in that time. PostgreSQL is no longer just a relational database. It's also a platform supporting the storage and retrieval of many types of data, built on a rock-solid, ACID-compliant, transactional core.

When I started to develop using Ruby on Rails, most of the other developers (including the core Rails developers at 37Signals) were using MySQL. As a result, Rails didn't offer any support for PostgreSQL-specific features. Indeed, one of my favorite Rails features always has been database migrations, which allow developers to change a database schema incrementally. The downside of such platform independence is that special features often are ignored, and indeed, in order to serve the lowest common denominator, many of PostgreSQL's features were ignored or relegated to third-party gems.

During the past few years, PostgreSQL has grown in popularity, both overall and within the Rails community. This is partly due to the large (and constantly growing) feature set that PostgreSQL provides. However, I'm guessing that it also has to do with the fact that Oracle now owns MySQL, along with the growth of the popular Heroku hosting service. Whether Heroku is an appropriate choice for your application is a decision that should be made on a case-by-case basis. However, the fact that Heroku offers a free tier for tiny data sets, and that it uses PostgreSQL by default, has made it a popular option for people learning Rails, for small applications and for many people who want to outsource their hosting.

As as result of PostgreSQL's growing popularity, the latest (4.x) version of Ruby on Rails includes extensive, built-in support for many PostgreSQL features. In this article, I introduce a number of these features, both from the perspective of a Rails developer and from that of a PostgreSQL administrator and DBA. Even if you aren't a Rails or PostgreSQL user, I hope these examples will give you a chance to think about how much you can and should expect from your database, as opposed to handling it from within your application.

UUIDs as Primary Keys

One of the first things database developers learn is the need for a primary key, a field that is guaranteed to be unique and indexed, and that can be used to identify a complete record. That's why many countries have ID numbers; using that number, government agencies, banks and health-care systems quickly can pull up your information. The usual standard for primary keys is an integer, which can be defined in PostgreSQL using the SERIAL pseudo-type:

```
CREATE TABLE People (
    id      SERIAL PRIMARY KEY,
    name    TEXT,
    email   TEXT
);
```

When you use the SERIAL type in PostgreSQL, that actually creates a "sequence" object, on which you can invoke the "nextval" function. That function is guaranteed to give you the next number in the sequence. Although you can define it to have a step of more than one, or to wrap around when it's done, the most usual case is to use a sequence to increment an ID counter. When you ask PostgreSQL to show you how this table is defined, you can see how the "id" field's definition has been expanded:

```
\d people

                          Table "public.people"

+--------+---------+------------------------------------------+
| Column |  Type   |                  Modifiers               |
+--------+---------+------------------------------------------+
| id     | integer | not null default
                     ➥nextval('people_id_seq'::regclass) |
| name   | text    |        |
| email  | text    |        |
+--------+---------+------------------------------------------+
Indexes:
    "people_pkey" PRIMARY KEY, btree (id)
```

So, you can see that there's nothing special about the "id" column, except that it has a default value. If you don't specify the value of "id" in your INSERT statement, PostgreSQL will invoke nextval on a sequence. In this way, you can be sure that the "id" column always will have a unique value.

But, what if you don't want to use integers? I've always been partial to them, but it is common and popular to use UUIDs (universally unique IDs). One of the advantages of UUIDs is that they are (more or less) guaranteed to be unique across computers, allowing you to merge records from multiple servers. If you were to do this with an integer primary key, you might well have multiple records with an ID of 5 or 10. But with UUIDs, this is far less likely.

In theory, PostgreSQL always has supported the use of UUIDs as primary keys. After all, you can just use a text field and have your application generate and insert the UUIDs. But that puts the onus on the application, which isn't really appropriate. A better solution is to use PostgreSQL's uuid-ossp extension, which has shipped with the last few versions of the database. In a modern version of PostgreSQL, you can issue the SQL command:

```
CREATE EXTENSION "uuid-ossp";
```

Note that you must use double quotes here, because there is a - character in the identifier. Double quotes tell PostgreSQL to keep an identifier precisely as you have written it (don't confuse this with single quotes, which are used for text strings).

Also note that extensions are installed only in the database where you issued the `CREATE EXTENSION` command. Thus, if you add an extension to the "foo_development" database, it won't be in the "foo_production" database automatically. To ensure that an extension is present in all databases, add it to "template1", from which all new databases are copied.

Once you have installed the extension successfully (which the database will confirm by echoing your command, `CREATE EXTENSION`), you can use it. Like many PostgreSQL extensions, uuid-ossp defines a new data type and functions that know how to use it. For example, you now can invoke the `uuid_generate_v1()` function, getting back data of type "uuid":

```
select uuid_generate_v1();
+--------------------------------------+
|           uuid_generate_v1           |
+--------------------------------------+
| 6167603c-276b-11e3-b71f-28cfe91f81e7 |
+--------------------------------------+
(1 row)
```

If you want to use a UUID as your primary key, you can redefine the table as follows:

```
CREATE TABLE People (
    id UUID NOT NULL PRIMARY KEY DEFAULT uuid_generate_v1(),
    name TEXT,
    email TEXT
);
```

As you can see, here you have replaced the SERIAL type with a UUID type (defined by the extension) and have instructed PostgreSQL to invoke the UUID-generating function when no UUID value is provided. If you insert a row into this table, you'll see that the UUID is indeed generated:

```
INSERT INTO People (name, email)
VALUES ('Reuven', 'reuven@lerner.co.il');

SELECT * FROM People;
```

```
+--------------------------------------+--------+---------------------+
|                  id                  |  name  |        email        |
+--------------------------------------+--------+---------------------+
| 9fc82492-276b-11e3-a814-28cfe91f81e7 | Reuven | reuven@lerner.co.il |
+--------------------------------------+--------+---------------------+
```

Now, all if this is great if you're working directly at the database level. But Rails migrations are supposed to provide a layer of abstraction, allowing you to specify your database changes via Ruby method calls. Starting in Rails 4, that's possible. I can create a new Rails application with:

```
rails new pgfun -d postgresql
```

This will create a new "pgfun" Rails app, using PostgreSQL as the back-end database. I then create an appropriate database user at the command line (giving that user superuser privileges on the database in question):

```
createuser -U postgres -s pgfun
```

I then create the development database:

```
createdb -U pgfun pgfun_development
```

Now you're now ready to create your first migration. I use the built-in

Rails scaffold mechanism here, which will create a migration (as well as controllers, models and views) for me:

```
rails g scaffold person name:text email:text
```

Notice that I haven't specified a primary key column. That's because Rails normally assumes there will be a numeric column named "id", which will contain the primary key. However, you're going to change that by opening up the migration file that was created in db/migrations. By default, the migration looks like this:

```
class CreatePeople <
ActiveRecord::Migration
  def change
    create_table :people do |t|
      t.text :name
      t.text :email

      t.timestamps
    end
  end
end
```

By passing an additional parameter to `create_table` (before the block, in the first line), you can indicate that you want the primary key to be a UUID:

```
class CreatePeople < ActiveRecord::Migration
  def change
    create_table :people, id: :uuid do |t|
      t.text :name
      t.text :email

      t.timestamps
    end
  end
end
```

With that in place, your primary key still will be called "id", but it will be of type UUID. You can run the migration to be sure with:

```
bundle exec rake db:migrate
```

Sure enough, the table has been defined as you might like:

```
\d people

                  Table "public.people"

+------------+--------------------------+------------------------+

|   Column   |           Type           |        Modifiers       |

+------------+--------------------------+------------------------+

| id         | uuid                     | not null default

                                         ➥uuid_generate_v4() |

| name       | text                     |                  |

| email      | text                     |                  |

| created_at | timestamp without time zone |               |

| updated_at | timestamp without time zone |               |


+------------+--------------------------+------------------------+

Indexes:
    "people_pkey" PRIMARY KEY, btree (id)
```

If you look carefully, however, you'll see there's a difference between the default that the Rails migration generated and the one generated by hand earlier. The difference is in the function that is being used to generate a UUID—in the manual version, you generated a "version 1" UUID, based on the MAC address of the computer that created it. Rails, by contrast, uses the "version 4" UUID algorithm, which is completely random. The advantage of the v4 UUID is that the number is more random, thus reducing the chance that someone can guess it. However, because the data is random, it'll be slower for PostgreSQL to index it. If you want to tell Rails to use the v1 function, add a line to the migration:

```
class CreatePeople < ActiveRecord::Migration
  def change
    create_table :people, id: false do |t|
      t.primary_key :id, :uuid, default: 'uuid_generate_v1()'
      t.text :name
      t.text :email

      t.timestamps
    end
  end
end
```

Note that if you want to run the modified migration, it's probably easiest and best just to drop and re-create the "people" and "schema_migrations" tables. Rails remembers which migrations already have been applied, and it won't re-run one, even if you have modified the file:

```
\d people
                          Table "public.people"
+------------+----------------------------+-------------------------+
|   Column   |            Type            |               Modifiers   |
+------------+----------------------------+-------------------------+
| id         | uuid                       | not null default
                                              ➥uuid_generate_v1() |
| name       | text                       |                 |
| email      | text                       |                 |
| created_at | timestamp without time zone |                 |
| updated_at | timestamp without time zone |                 |
+------------+----------------------------+-------------------------+
Indexes:
    "people_pkey" PRIMARY KEY, btree (id)
```

With this default in place, your "people" table now will use UUIDs.

Arrays

Arrays are another PostgreSQL feature that is now supported natively by Rails. PostgreSQL has supported arrays for a number of years, and although I personally find the syntax to be a bit difficult to deal with, there's no doubt that arrays can simplify some database designs. (I should note, however, that arrays should be a last resort, because they tend to result in a non-normalized database design, potentially leading to unnecessary duplication of data.) For example, if I want to create a "posts" table for my blog and then allow people to store one or more social tags, I could define it as:

```
create table posts (
    id        UUID NOT NULL PRIMARY KEY DEFAULT uuid_generate_v1(),
    headline TEXT,
    body      TEXT,
    tags      TEXT[]
);
```

Notice the data type associated with the "tags" column. By using square brackets after the TEXT type, I've indicated that the column can contain zero or more text strings. For example:

```
INSERT INTO Posts (headline, body, tags)
VALUES ('my headline', 'my body', '{general, testing}');
```

Notice that the array value is inserted as a string, the first and final characters of which are curly braces. Now you can get information from the array as follows using square brackets, remembering that unlike many languages, PostgreSQL indexes arrays starting with 1:

```
SELECT headline, body, tags[1], tags[2] FROM Posts;
+-------------+---------+---------+---------+
|  headline   |  body   |  tags   |  tags   |
+-------------+---------+---------+---------+
| my headline | my body | general | testing |
+-------------+---------+---------+---------+
(1 row)
```

Notice how you can retrieve each of the tag elements separately by using their index. If you try to use an index for which there is no value, you get a NULL instead. You also can use the ANY operator to find rows in which a particular tag value is assigned:

```
select headline, body, tags from posts where 'general' = ANY(tags);
+-------------+---------+-------------------+
|  headline   |  body   |       tags        |
+-------------+---------+-------------------+
| my headline | my body | {general,testing} |
+-------------+---------+-------------------+
```

Note that the ANY operator must be on the right-hand side of the comparison. Otherwise, you'll get a syntax error from PostgreSQL.

There was little or no support for PostgreSQL arrays in earlier versions of Ruby on Rails. But starting with Rails 4, there is support for such functionality. You not only can define a column to contain an array, but you also can use ActiveRecord to manipulate it. First and foremost, let's create a scaffold for the resource:

```
rails g scaffold post headline:text body:text tags:string
```

This generates the necessary files. Don't run the migration just yet, however; you first need to turn "tags" from a string into an array of strings and your ID into a UUID:

```
class CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts, id: false do |t|
      t.primary_key :id, :uuid, default: 'uuid_generate_v1()'
      t.text :headline
      t.text :body
      t.string :tags, array:true, default:[]

      t.timestamps
    end
  end
end
```

Now you will have a UUID for a primary key, but you also will define tags to be an array. After running the migration, this is what you'll see:

```
\d posts
                              Table "public.posts"

+------------+---------------------------+-------------------------+
|   Column   |            Type           |        Modifiers        |
+------------+---------------------------+-------------------------+
| id         | uuid                      | not null default
                                           ➥uuid_generate_v1() |
| headline   | text                      |              |
| body       | text                      |              |
| tags       | character varying(255)[]  | default '{}'::character
                                           ➥varying[]   |
| created_at | timestamp without time zone |            |
| updated_at | timestamp without time zone |            |
+------------+---------------------------+-------------------------+
Indexes:
    "posts_pkey" PRIMARY KEY, btree (id)
```

From a database perspective, things seem great; you can perform an INSERT:

```
INSERT INTO Posts (headline, body, tags, created_at, updated_at)
VALUES ('my headline', 'my body', '{general, testing}', now(), now());
```

Sure enough, you can see the post in the database. The magic, however, is that ActiveRecord allows you to treat the PostgreSQL array as if it were a Ruby array. For example, you can say:

```
Post.first.tags.each {|t| puts t}
```

This tells Rails to ask ActiveRecord for the first record in the Posts table and to call up its "tags" column, which is returned as a Ruby array of strings. You then can iterate over those strings, printing them (or otherwise manipulating them). Although this isn't very efficient or smart, you also can do the following:

```
Post.all.select {|p| p.tags.member?('foo')}
```

A more efficient way to do this would be to use the `ANY` operator that you saw earlier, passed to PostgreSQL in a string:

```
Post.where("'general' = ANY(tags)").first
```

Unfortunately, it doesn't seem that you can add elements to a PostgreSQL array using the standard Ruby << (append) operator. Rather, if you want to add one or more elements to an array via ActiveRecord, you must do so manually:

```
p.update_attributes(tags: ['general', 'testing', 'zzz'])
```

This is somewhat annoying, but not fatal, particularly not for the first version where this functionality is included.

Summary

Rails 4, although not breaking

Advertorial Feature

DDoS attacks: Does your hosting provider keep you safe?

The vast majority of companies underestimate their exposure to the risk of a Denial of Service attack (DDoS). As a result, few take measures to protect themselves from one. However, internet attacks are on the rise, occurring more frequently, with greater intensity, and targeting a wider scope than ever before. While in times past such attacks were primarily conducted by hackers in the pay of unscrupulous competitors or underground networks wreaking havoc, these days they have become very easy to unleash, whether as a challenge or simply for fun. So the question is no longer whether you need DDoS protection, but whether your hosting provider is able to effectively protect you when you become the target of such an attack.

Initially, the main objective of a cyber attack was to damage a corporation, brand, or government agency. To accomplish this, hackers employed highly advanced techniques, and hired out their skills at no small cost. These days, the needed techniques are easily accessible on the web via online forums or dedicated websites. More significantly, the resources needed to mount such an attack have become trivial. For $10, almost anyone can rent a number of botnets (remotely controlled «zombie» machines) and set them to run a ready-made script that exploits the weaknesses of a website, web application, portal or blog. The result? These days, most DDoS attacks are launched by teenagers, nicknamed «Script Kiddies». This has become a major nuisance for companies large and small across all industry sectors, and especially for those who are heavily dependent on maintaining an uninterrupted online presence.

DDoS protection: Included in all OVH.com hosting plans

The protective measures taken by many hosting providers have proven to be increasingly ineffective in stemming the exponential growth, intensity and sophistication of certain denial of service attacks. This trend has led OVH.com to take a close look at its data hosting operations and design a DDoS protection system that provides non-stop protection of its clients' physical and virtual IT infrastructures.
This set of tools is able to detect an attack, counter it, and greatly minimize its impact. The costs of implementing such protection are significant, and most hosting providers pass these costs on to their customers by offering DDoS protection as a prohibitively expensive paid option. OVH.com is taking a markedly different approach by making DDoS protection a standard service included in all of its hosting plans.

Why make these costly protective services available to all?

If an attack is launched against one OVH.com client, there can be secondary negative effects on neighboring clients within the same server bay. Depending on the intensity of the attack, such collateral damage could escalate to impact a sub-network, router, or even an entire datacenter. And then there are «side attacks», where, in order to reach a difficult target, hackers first attack neighboring virtual servers that are less protected. In summary, in order for DDoS protection to be truly effective, everyone needs to be protected. This is why OVH.com's business model consists of spreading the costs of protection across all of its servers, by providing it by default to all. OVH.com is the only hosting provider to include permanent DDoS protection in all of its hosting plans.

For more information:
www.ovh.com/us/anti-ddos

OVH.COM

compatibility with its predecessors nearly as much as Rails 3, does introduce a great deal of new functionality. For me, one of the most interesting areas of this functionality is a shift toward PostgreSQL, with ActiveRecord migrations and functionality abandoning some of its platform independence. In this article, I showed two of the features that are now available, namely UUIDs and arrays. However, there are additional features, such as native support for INET (that is, IP address) data types, for JSON (which PostgreSQL 9.3 supports even better than it did in the past), ranges and even for hstore, a NoSQL-like storage system built on top of PostgreSQL.

No technology, including PostgreSQL, is the right answer for everyone at all times. However, in my experience, PostgreSQL offers excellent performance, features and stability, with a fantastic community that answers questions and strives for correctness. The fact that Rails 4 has embraced many of these features is likely to expose even more people to PostgreSQL, which can only be good for Web and database developers who use open-source products.■

Web developer, trainer and consultant Reuven M. Lerner is finishing his PhD in Learning Sciences at Northwestern University. He lives in Modi'in, Israel, with his wife and three children. You can read more about him at http://lerner.co.il, or contact him at reuven@lerner.co.il.

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

Resources

The PostgreSQL home page is at **http://postgresql.org**. From that site, you can download the software, read documentation and subscribe to e-mail lists. The latest version, 9.3, includes a large number of goodies that should excite many developers.

Ruby on Rails, in version 4.0.0 at the time of this writing, is available at **http://rubyonrails.org**. You likely will want to use Ruby version 2.0, although version 1.9.3 is known to work with Rails 4. You also will want to include the "pg" Ruby gem, which connects to PostgreSQL. Excellent documentation on Rails is available in the "Guides" series at **http://guides.rubyonrails.org**.

A nice blog posting describing many of the updates to Rails 4 from a PostgreSQL perspective is at **http://blog.remarkablelabs.com/2012/12/a-love-affair-with-postgresql-rails-4-countdown-to-2013**.

Why Join USENIX?

We support members' professional and technical development through many ongoing activities, including:

- Open access to research presented at our events
- Workshops on hot topics
- Conferences presenting the latest in research and practice
- LISA: The USENIX Special Interest Group for Sysadmins
- *;login:*, the magazine of USENIX
- Student outreach

Your membership dollars go towards programs including:

- Open access policy: All conference papers and videos are immediately free to everyone upon publication
- Student program, including grants for conference attendance
- Good Works program

OPEN ACCESS

Helping our many communities share, develop, and adopt ground-breaking ideas in advanced technology

Join us at www.usenix.org

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

DAVE TAYLOR

Debugging Web Sites

Dave has to fight a fire with shell scripts when he migrates his ten-year-old Web site onto a new content management system!

I know, I'm in the middle of a series of columns about how to work with ImageMagick on the command line, but when other things arise, well, I imagine that a lot of you are somehow involved in the management of servers or systems, so you all understand firefighting.

Of course, this means you all also understand the negative feedback loop that is an intrinsic part of system administration and IT management. I mean, people don't call you and the CEO doesn't send a memo saying, "system worked all day, printer even printed. Thanks!"

Nope, it's when things go wrong that you hear about them, and that propensity to ignore the good and have to deal with the bad when it crops up is not only a characteristic of being in corporate IT, it's just as true if you're running your own system—which is how it jumped out of the pond and bit me this month.

It all started ten years ago with my Ask Dave Taylor site. You've probably bumped into it (**http://www.AskDaveTaylor.com**), as it's been around for more than a decade and served helpful tutorial information for tens of millions of visitors in that period.

Ten years ago, the choice of Movable Type as my blogging platform made total sense and was a smart alternative to the raw, unfinished WordPress platform with its never-ending parade of hacks and problems. As every corporate IT person knows, however, sometimes you get locked in to the wrong platform and are then stuck, with the work required to migrate becoming greater and greater each month nothing happens.

For the site's tenth anniversary, therefore, it was time. I had to bite the bullet and migrate all 3,800 articles and 56,000 comments from Movable Type to WordPress, because yes, WordPress won and is clearly the industry standard for content management systems today.

The task was daunting, not just because of the size of the import (it required the consulting team rewriting the standard import tool to work with that many articles and comments), but because the naming scheme changed. On Movable Type, I'd always had it set to convert the article's name into a URL like this:

Name: Getting Started with Pinterest
URL: /getting_started_with_pinterest.html

That was easy and straightforward, but on WordPress, URLs have dashes, not underscores, and, more important, they don't end with .html because they're generated dynamically as needed. This means the default URL for the new WordPress site would look like this:

New URL: /getting-started-with-pinterest/

URLs can be mapped upon import so that the default dashes become underscores, but it was the suffix that posed a problem, and post-import there were 3,800 URLs that were broken because every single link to xx_xx.html failed.

Ah! A 301 redirect! Yes, but thousands of redirects slow down the server for everyone, so a rewrite rule is better. Within Apache, you can specify "if you see a URL of the form xx_xx.html, rewrite it to 'xx_xx' and try again", a darn handy capability.

But life is never that easy, because although this rewrite will work for 95% of the URLs on the old site, there were some that just ended up with a different URL because I'd monkeyed with things somewhere along the way. Yeah, there's always something.

For example, the old site URL /schedule_facebook_photo_upload_fan_page.html is now on the server with the URL /schedule-a-facebook-photo-upload-to-my-fan-page/.

That's helpful, right? (Sigh.)

These all can be handled with a 301 redirect, but the question is, out of almost 4,000 article URLs on the old site, which ones don't actually successfully map with the rewrite rule (.html to a trailing slash) to a page on the new server?

Finally Some Scripting

To identify these rewrite fails, I had to create a script—and fast. After all, while the internal linkages still might work, the thousands of external links from sites like *Popular Science*, the *Wall Street Journal*, *Wired* and elsewhere were not broken. Yikes—not good at all.

I started out on the command line with one that I knew failed. Here's what happened when I used curl to grab a bad URL on the new site:

```
$ curl
http://www.askdavetaylor.com/
➥schedule-facebook-photo-upload-to-my-fan-page.html
| head -5

% Total  % Received % Xferd  Average Speed  Time   Time   Time Current
                             Dload  Upload  Total Spent Left Speed
0     0  0    0     0     0      0     0 --:--:-- --:--:-- --:--:--
0<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="UTF-8" />
<title>Nothing found for
Schedule-A-Facebook-Photo-Upload-To-My-Fan-Page</title>
100 31806   0 31806  0    0  110k  0 --:--:-- --:--:-- --:--:-- 110k
curl: (23) Failed writing body (0 != 754)
```

Ugh, what a mess this is, and it's not surprising because I forgot to add the `-silent` flag to `curl` when I invoked it.

Still, there's enough displayed here to provide a big clue. It's a 404 error page, as expected, and the <title> indicates just that:

```
<title>Nothing found for ...
```

So that's an easy pattern to search for:

```
curl -silent URL | grep '<title>Nothing found for'
```

That does the trick. If the output is non-zero, the link failed and generated a 404 error, but if the link worked, it'll be the proper title of the article, and the words "Nothing found for" will appear.

That's most of the needed logic for the script. The only other step is to simulate the rewrite rule so that all the links that do work aren't flagged as a problem. Easy:

```
newname="$(echo $name | sed 's/\.html/\//')"
```

This is a super-common sequence that I use in scripts, actually with a subshell invocation `$( )` echoing a variable's current value, just to push it through a `sed` substitution, in this case replacing .html with a trailing slash (which needs to be escaped with a leading backslash, hence the complexity of the pattern).

Wrap this in a for loop that steps through all possible *.html files, and here's what it looks like:

```
for name in *.html ; do
  newname="$(echo $name | sed 's/\.html/\//')"
  test=$($curl $base/$newname | grep "$pattern")
  if [ -n "$test" ]
  then
    echo "* URL $base/$name fails to resolve."
  fi
done
```

That's boring though, because while

That's boring though, because while I'm at it, I'd like to know how many URLs were tested and how many errors were encountered.

I'm at it, I'd like to know how many URLs were tested and how many errors were encountered. I mean, why not, right? Quantification = good.

It's easily added, as it turns out, with the addition of two new variables (both of which need to be set to zero at the top of the script):

```
for name in *.html ; do
  newname="$(echo $name | sed 's/\.html/\//')"
  test=$($curl $base/$newname | grep "$pattern")
  if [ -n "$test" ] ; then
    echo "* URL $base/$name fails to resolve."
    error=$(( $error + 1 ))
  fi
  count=$(( $count + 1 ))
done
```

Then at the very end of the script, after all the specific errors are reported, a status update:

```
echo ""; echo "Checked $count links, found $error problems."
```

Great. Let's run it:

```
$ bad-links.sh | tail -5

* URL http://www.askdavetaylor.com/whats_a_fast_way_to_add_a_
➥store_and_shopping_cart_to_my_site.html fails to resolve.

* URL http://www.askdavetaylor.com/whats_amazons_simple_
➥storage_solution_s3.html fails to resolve.

* URL http://www.askdavetaylor.com/whats_my_yahoo_
➥account_password_1.html fails to resolve.

* URL http://www.askdavetaylor.com/youtube_video_
➥missing_hd_resolution.html fails to resolve.

Checked 3658 links, found 98 problems.
```

Phew. Now I know the special cases and can apply custom 301 redirects to fix them. By the time you read this article, all will be well on the site (or better be).

Next month, back to ImageMagick, I promise—unless another fire erupts that I have to solve.■

Dave Taylor has been hacking shell scripts for more than 30 years. Really. He's the author of the popular *Wicked Cool Shell Scripts* and can be found on Twitter as @DaveTaylor and more generally at http://www.DaveTaylorOnline.com.

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

Super Pi Brothers

KYLE RANKIN

Why dust off all of your old consoles when you can play the same games inside a spare Raspberry Pi?

I don't game as much as I used to. Although I've certainly spent countless hours of my life in front of a Nintendo, SNES, or after that, playing a first-person shooter on my computer (Linux only, thank you), these days, my free time tends to go toward one of the many nongaming hobbies I've accumulated. Recently though, I found myself dusting off my Wii console just so I could play an NES and SNES game I re-purchased for it. The thing is, those games require using a somewhat strange controller, and I already have a modified SNES controller that can connect over USB. That was enough to encourage me to search for a better solution. Of course, I simply could connect three or four consoles and stack up games in my living room, but I've grown accustomed to ripping my CDs and DVDs and picking what I want to listen to or watch from a central media center. It would be nice if I didn't have to get up and find a cartridge every time I wanted to switch games. This, of course, means going with emulation, but although in the past I'd had success with a modified classic Xbox, I didn't have that hardware anymore. I figured someone must have gotten this set up on the Raspberry Pi, and sure enough, after a brief search and a few commands, I had a perfect retro-gaming arcade set up on a spare Raspberry Pi.

One nice thing about the Raspberry Pi project is the large number of people out there with identical hardware. For me, that meant instead of having to go through someone else's instructions, knowing I'd likely have to tweak it to suit my setup, I basically could follow someone else's guide verbatim. In my case, I found the RetroPie project, which wrapped up all of the commands you would need to install everything on a Raspberry Pi into a single large script. At the end, you have the RetroArch project fully installed and configured, which includes all of the major emulators you'd want and a centralized method to configure them, plus an EmulationStation graphical front end the Pi can boot directly into that makes it simple to navigate to the game you want, all from a gamepad.

Install RetroPie

Before you install RetroPie, you will want to make sure your Raspbian distribution (the default Linux distribution for a Raspberry Pi, and the one this project assumes you will use) is completely up to date, including any new firmware images. This just means a few common `apt` commands. Although you certainly could connect a keyboard to your Raspberry Pi for this step, I've found it more convenient to `ssh` in to the device so I could copy and paste commands:

```
$ sudo apt-get update
$ sudo apt-get -y upgrade
```

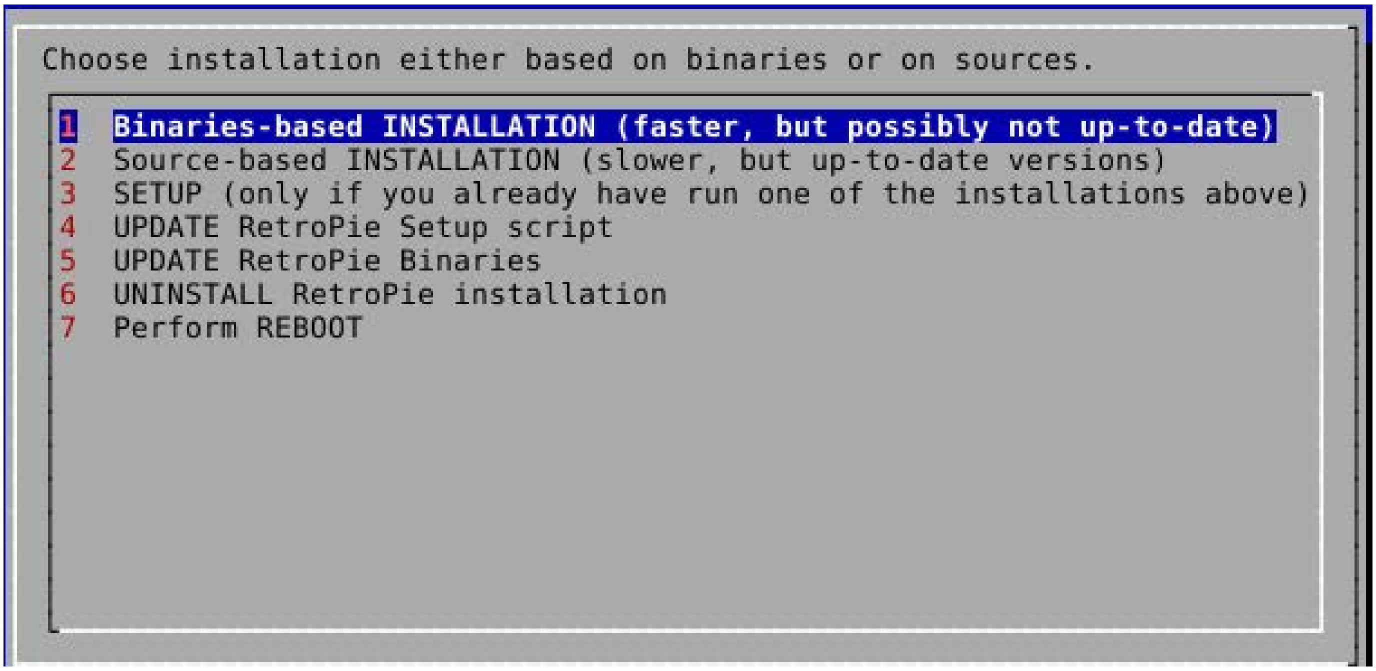
Now that the Raspberry Pi is up to date, make sure the git and dialog packages are installed, and then use git to download RetroPie:

```
$ sudo apt-get -y install git dialog
$ cd
$ git clone --depth=0
➥git://github.com/petrockblog/RetroPie-Setup.git
```

This will create a RetroPie-Setup directory containing the main setup script. Now you just need to go inside that directory and execute it:

```
$ cd RetroPie-Setup
$ chmod +x retropie_setup.sh
$ sudo ./retropie_setup.sh
```

This script presents you with an in-terminal menu (Figure 1) where you can choose to perform a binary installation or source installation, set up RetroPie, or perform a series of updates for the RetroPie setup script and binaries. Choose either the binary or source installation. The binary installation won't take as much time, but you may risk running older versions of some of the software. The source installation requires you to compile software, so it takes longer, but at the end, you will have the latest possible versions of everything. Personally, I opted for the binary install, knowing I always could re-run the



```
Choose installation either based on binaries or on sources.

1  Binaries-based INSTALLATION (faster, but possibly not up-to-date)
2  Source-based INSTALLATION (slower, but up-to-date versions)
3  SETUP (only if you already have run one of the installations above)
4  UPDATE RetroPie Setup script
5  UPDATE RetroPie Binaries
6  UNINSTALL RetroPie installation
7  Perform REBOOT
```

Figure 1. RetroPie Setup Menu

script and go with the source install if I found any problems.

This part of the process will take quite some time on a vanilla Raspbian image, as there are a lot of different packages to download and install. Once the installation completes, go back to the main RetroPie setup screen and select SETUP from the main menu. In this submenu, you can tweak settings, such as whether to start EmulationStation from boot (recommended) and whether to enable a splash screen. In my case, I enabled both settings as I intended my device to be a standalone emulation machine. Note that if you do allow EmulationStation to start up from boot, you still can always `ssh` in to the machine and run the original RetroPie configuration script to change the settings.

Adding ROMs

You also can add ROMs within the RetroPie setup screen. If you choose the Samba method in the menu, you then can locate a local Samba mountpoint on your network, and you can copy ROMs from that. With the USB stick method, RetroPie will generate a directory structure on a USB stick that you plug in to your Raspberry Pi that represents the different emulators it supports. After this point, you can take that USB stick to another computer and copy ROMs over to the appropriate directory, and the next time you plug it in to the Raspberry Pi, it automatically will sync the files over. Finally (and this is what I did), you just can use scp or rsync to copy over ROMs to the appropriate directory under ~/RetroPie/roms/. So for instance, NES games would be copied to ~/RetroPie/roms/nes/.

Once you are done with the configuration and exit out of the RetroPie setup script, you will want to reboot back into EmulationStation, but before you do, you should reconfigure the memory split on the Raspberry Pi so that it is set to 192 or 128, so run:

```
$ sudo raspi-config
```

and go to the Advanced Settings to change the memory split setting. Now you can reboot safely.

EmulationStation

Once you reboot, you should be greeted with the initial EmulationStation screen, which will prompt you to set up your joystick, gamepad or keyboard buttons so it can work with the EmulationStation menu. Note that this doesn't affect how your controllers work within games, just within the EmulationStation menu. After your controller or controllers are set up, you should be able to press right and left on your controller to switch between the different emulator menus. In my case, all of the buttons on my gamepad were going to be used within games, so I made a point to bind a key on a separate keyboard to the menu function so I could exit out of games when I was done without having to reboot the Raspberry Pi.

EmulationStation will show you only menus that represent emulators for which it has detected ROMs, so if you haven't copied ROMs for a particular emulator yet, you will want to do that and potentially restart your Raspberry Pi before you can see them. Also, by default, your controller will not be configured for any games, but if you press the right arrow enough times within EmulationStation, you will get to an input configuration screen that allows you to map keys on your controller to keys inside a game. The nice thing about this setup is that after you configure the keys, it will apply appropriately within each emulator.

That's it. From this point, you can browse through your collection of games and press whatever button you bound to Accept to start playing. At first I was concerned the Raspberry Pi wouldn't have the horsepower to play my games, but so far, it has been able to play any games I tried without a problem.■

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

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

Resources

The RetroPie Project: **http://blog.petrockblock.com/retropie**

RetroPie Installation Docs: **https://github.com/petrockblog/RetroPie-Setup**

Unleash the power of

Free Tools

to free you from everyday

IT management tasks!

Trusted by Businesses of **ALL SIZES**

| Active Directory, Exchange, SharePoint, and SQL Monitoring

| Desktop Management | Windows Monitoring

| Virtualization & Cloud Management

| Ping, Traffic, and SNMP Troubleshooting Utilities | Mobile Apps

http://www.manageengine.com/free-tools

ManageEngine
Free Tools

SHAWN POWERS

Little Computers, Big Projects

We all love Pi, but what if you could pack twice the power into half the size?

System administrators tend to be geeks. Our jobs are geeky, our hair is geeky, and even our hobbies are geeky. (My article from last month is proof. I can't even feed the birds without cron jobs and network streaming.) Although I've spent countless hours fiddling with Raspberry Pis during the past year, they're no longer the only kid on the block when it comes to tiny ARM-based computers. So in this article, I'm going to talk about my newest toy, the CuBox Pro from SolidRun (**http://www.solid-run.com/cubox**). The $159 device is more expensive than a Raspberry Pi, but it includes a case, an even tinier form factor and as much power as several RPi units. Plus, SolidRun is releasing the CuBox-i series, which is just as powerful and half the price.

I originally planned to review the CuBox like any other product, and if you prefer to see this as an extended review, that's okay. The folks at SolidRun have made such a neat product, however, that I'm really more interested in showing you the cool things it can do than just giving you a list of specs and opinions. That said, giving you the specs makes sense, so I'll start there and move on to the more fun stuff afterward.

What It's Got

The CuBox is a 2" plastic cube. Although that sounds boring, its cool look (Figure 1) is one of the things that makes it so attractive for computing projects. I love the Raspberry Pi as much as the next guy, but my wife never

Figure 1. The CuBox is almost too small to look real.

has liked the dangling little circuit board hanging off the back of the television. The CuBox looks great. Inside that simple case it has:

- System board based on Marvell Armada 510 SoC (Figure 2).
- 800MHz dual-issue ARM PJ4 processor, VFPv3, wmmx SIMD and 512KB L2 cache.
- 1080p Video Decode Engine.
- OpenGL|ES 2.0 graphic engine.

- SPDIF (optical audio).
- HDMI 1080p Output (with CEC function for TVs that support it).
- 1GB DDR3 at 800MHz (2GB in the Pro model I have).
- Gigabit Ethernet (bootable via TFTP).
- eSata 3Gbps (bootable).
- 2xUSB 2.0 (bootable).
- MicroSD (bootable).
- Micro-USB (console).
- Standard infra-red receiver for 38KHz-based IR controllers.

The little system comes with a DC power adapter and draws a whopping 3 watts of power while in use. 3 watts. That's not a typo. Regardless of the use case you find for your CuBox, power consumption shouldn't be an issue. In fact, 3 watts really lends itself to battery power and a solar panel for those computing projects off the grid. That might have to wait until next month for me, however, because I already have more indoor projects than I have CuBoxes. Check out how well they've crammed the ports onto the back plane of the little thing in Figure 3.

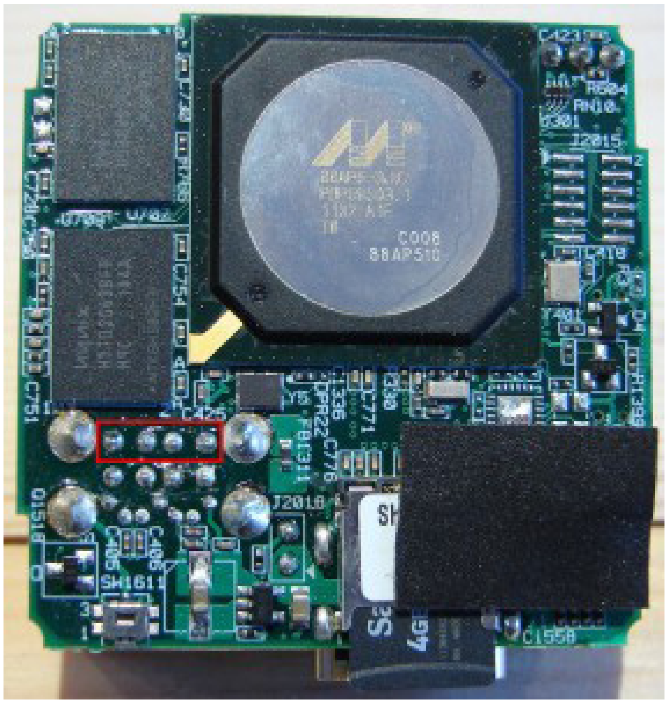


Figure 2. The Marvell system on a chip is tiny, even by cell-phone standards.

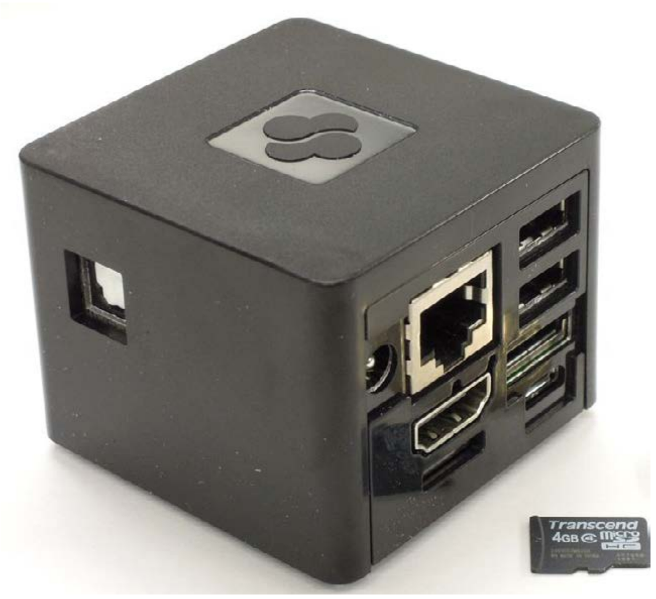


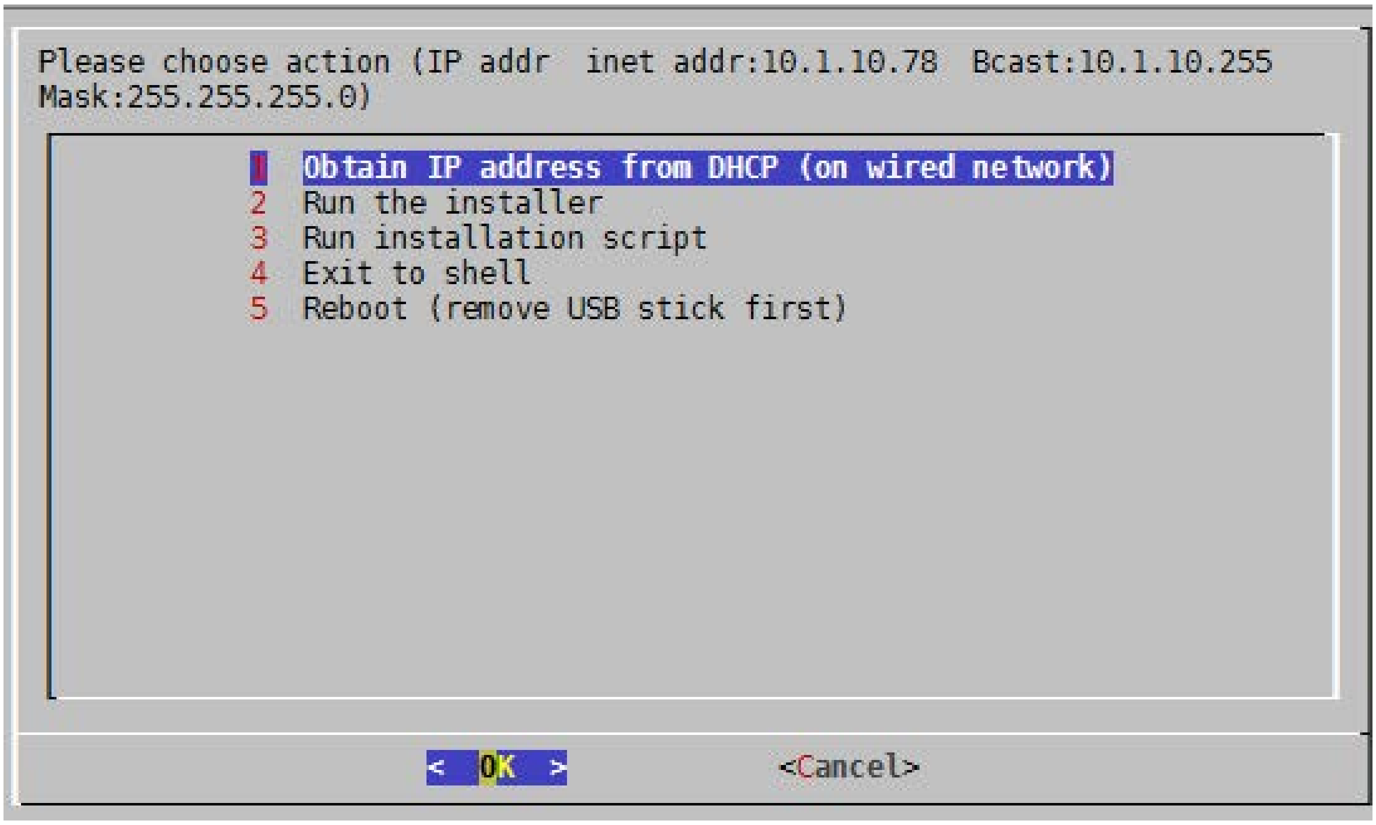
Figure 3. The only strange port placement is the optical audio port on the side. Thankfully, I've been using HDMI only.

How to Make It Go

My first 20 minutes or so with the CuBox were frustrating and intimidating. The instructions I had to configure the CuBox were written for an earlier revision than the one I received. Early on, to get the CuBox to boot, it was required to flash the U-Boot code into firmware, boot to a Micro-USB serial cable and figure out some way to get the firmware to the unit from the Internet. Thankfully, I discovered that the model I received (and all that ship now) came with a recent version of U-Boot and required nothing with a console cable. In fact, the CuBox boasts the simplest installation tools I think I've ever used. SolidRun has created an installer program that does all the heavy lifting, and it allows you to choose from a menu of CuBox-specific distributions and applications easily. The process is so simple, I thought at first I was missing something:

1. Get a FAT/FAT32-formatted USB drive.

2. Unzip the installer tool to a folder named boot on the USB drive.



```
Please choose action (IP addr  inet addr:10.1.10.78  Bcast:10.1.10.255
Mask:255.255.255.0)

          1  Obtain IP address from DHCP (on wired network)
          2  Run the installer
          3  Run installation script
          4  Exit to shell
          5  Reboot (remove USB stick first)

               <  OK  >            <Cancel>
```

Figure 4. The installer is simple, but powerful. It downloads the latest software directly off the Internet during the install, so you always install the latest software.

3. Start CuBox with the USB drive in the top USB slot.

That's it—no Syslinux, no fiddling with the master boot record, no keyboard combinations. Just have the folder named boot with the installation scripts, and the CuBox will boot in to installer mode (Figure 4). From there, the installer scripts will download Ubuntu, XBMC, Debian or any custom installation scripts you can find elsewhere on the Internet. The CuBox installer will format the included MicroSD card and get your system completely ready to go.

Cool Stuff to Do, I: Desktop

The CuBox Pro I received came with Ubuntu preinstalled on the MicroSD card. I've used Raspbian as a desktop computer a couple times, and although it was cool to see the RPi power a fully graphical system, the experience wasn't exactly fun. Because the CuBox has a faster processor and 2 gigs of RAM, it makes sense that it would be better as a desktop replacement—a 3-watt desktop replacement! As it turns out, the CuBox does perform better than a Raspberry Pi, but it's still sluggish enough that I wouldn't want to use it for more than a little while. Plus, since it's ARM-based, there's no possibility of getting Adobe Flash working. Sadly, that's still a showstopper for some of the things I do on a regular basis. If you're looking for a teeny-tiny desktop replacement computer, however, I think the CuBox might fit the bill if:

1. Instead of booting from MicroSD, you connect an SSD via eSATA. The same installation program will install onto an eSATA drive instead of MicroSD, and that would remove what felt like the biggest bottleneck. Once programs were loaded into RAM, the system was quite usable. MicroSD as your main filesystem is just painful.

2. Rather than using Ubuntu proper, you use Xubuntu (the CuBox performs a bit better with Xubuntu). Although the GPU on the system is great for decoding video, it does struggle with dynamic GUI content. Ubuntu's overlays and eye-candy-rich interface make Ubuntu feel slow.

I forced myself to use the CuBox for an afternoon as my main writing computer. It worked, but if I was doing anything other than actual writing, the lag when launching programs really frustrated me. I wish I had an eSATA SSD to test the speed.

Cool Stuff to Do, II: Server

I've mentioned my Raspberry Pi colocated in Austria before. I'm still eternally grateful to Kyle Rankin for tipping me off to the opportunity. As I put services on the RPi, however, I have noticed that as a server, the Raspberry Pi can become overwhelmed fairly quickly. But, with a headless Debian install on the CuBox, I can see the difference 2GB of RAM can make.

The other big limitation with my Raspberry Pi server is storage space. I have a 32GB SD card and a 32GB USB drive in it, but not only are both of those media types rather slow, 64GB also isn't exactly massive cloud storage. With the CuBox, an external eSATA drive means the potential for terabytes of local storage.

Although I'll admit the CuBox as a desktop replacement was a little frustrating, as a server, I was very impressed with it. With Gigabit Ethernet, tons of RAM and visual appeal, it makes a great server for your office. Thanks to the ARM processor, it's perfectly silent and generates very little heat. I would continue to use the CuBox as a server in my office, except....

Cool Stuff to Do, III: Home Theater

GeeXboX is a home-theater software system that at some point started utilizing XBMC as its software. That same awesome installer program that installs Ubuntu or Debian will install the latest version of GeeXboX on the CuBox, already tweaked and configured for the CuBox-specific hardware. At first glance, this doesn't seem like much, but under the hood, there are some cool modifications made to help with menu rendering speeds and screen blanking. Basically, if you use the CuBox installer, XBMC just works (Figure 5).

Running GeeXboX on the CuBox is incredible. Admittedly, there are a few issues with slow menu response (especially while the library is being scanned). If you've ever used XBMC on a Raspberry Pi, however, this performs like a dream. But, that's not even the coolest part. Because the CuBox comes with an infrared receiver built in to its little cube, that means programming the old remote sitting in your junk drawer (an old series one TiVo remote in my case) is simple. Well, okay, maybe "simple" isn't quite true. Still, with a little bit of wiki reading and command-line tweaking, I was able to get my old TiVo remote to work with CuBox and GeeXboX in pretty

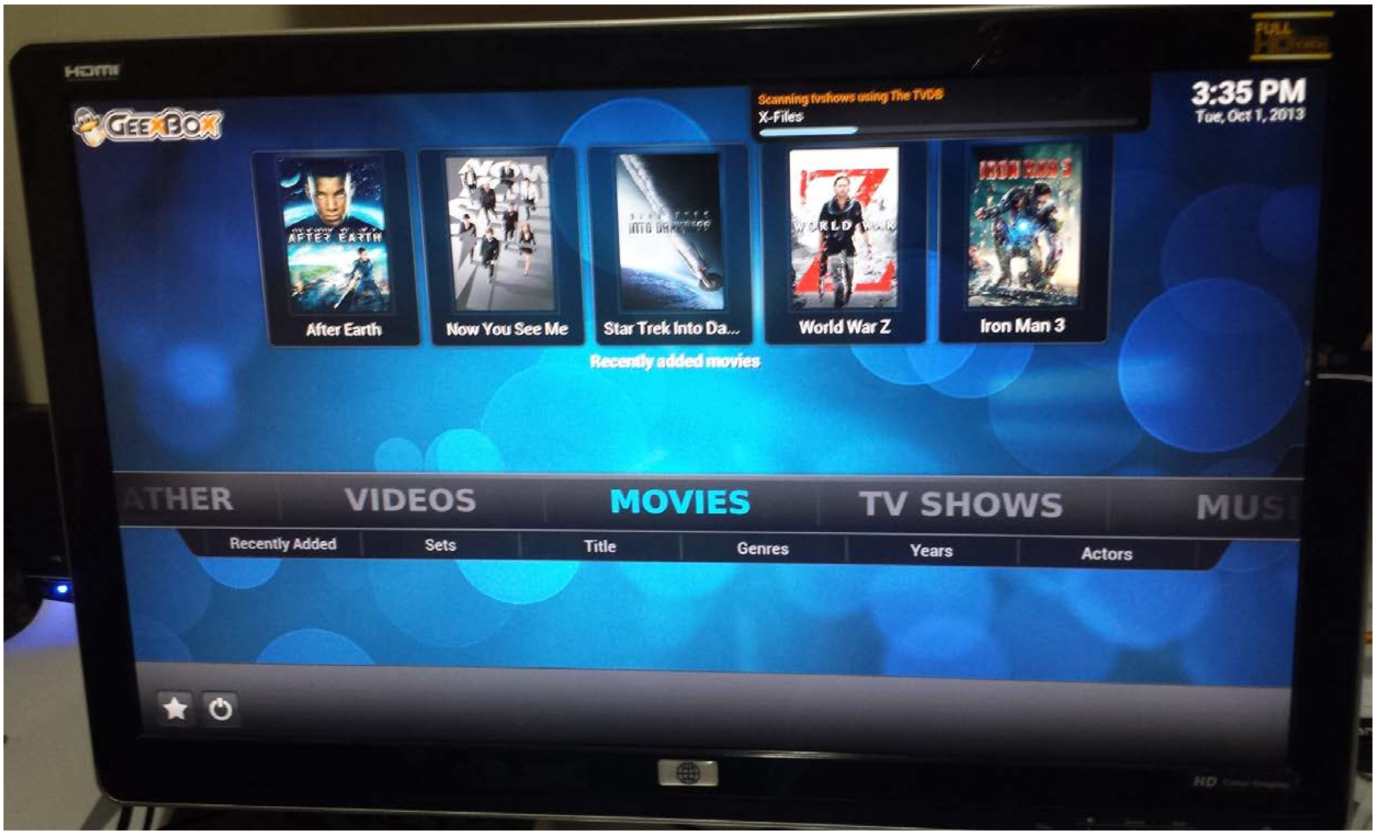


Figure 5. GeeXboX works flawlessly. Even the HDMI and SPDIF are detected correctly and function without additional tweaking.

short order.

If you do try XBMC (or GeeXboX, which is the same thing now it seems), I recommend playing network videos over NFS. There's less overhead than Samba, and high bandwidth media seems to stream better. If you don't have 1080p videos streaming over your network, it's probably not a big issue, but NFS seems to work better for those truly huge videos.

Bonus Stuff I Want to Try Soon

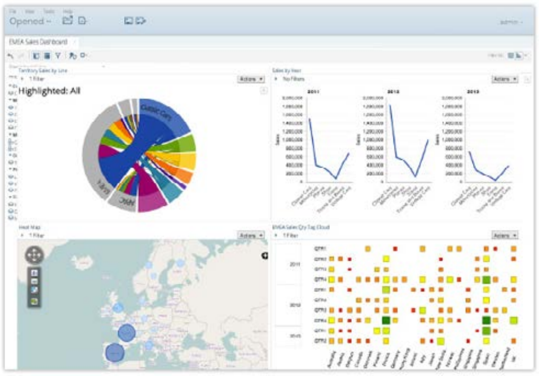
As I mentioned earlier, I'd like to try out the CuBox with an eSATA-connected SSD. I'd also like to explore booting via TFTP. The system supports it, but I haven't looked into how the boot process works. Running something like LTSP over the Gigabit Ethernet port with 2 gigs of RAM might make for an incredibly awesome thin-client environment.

The CuBox has found a semi-permanent home as a GeeXboX/XBMC media player in our home. I would like to see the Plex Home Theater system configured for the CuBox. Although we still use XBMC in our house, the central server-based model Plex offers has features I don't get with XBMC alone.

And of course, I've been trying to figure out how to use a CuBox to somehow capture better video of the various bird feeders around our house. The low power draw coupled with impressive hardware specs make the CuBox ideal for a battery/solar-panel-type solution. With a USB Wi-Fi dongle, I could put birdcams all over our woods. Please don't tell my wife. She still rolls her eyes when she sees the cell phones attached to my office window! (See my column in the October 2013 issue for details on BirdCam.)■

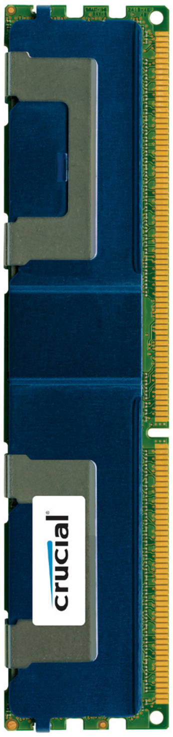
Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing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

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



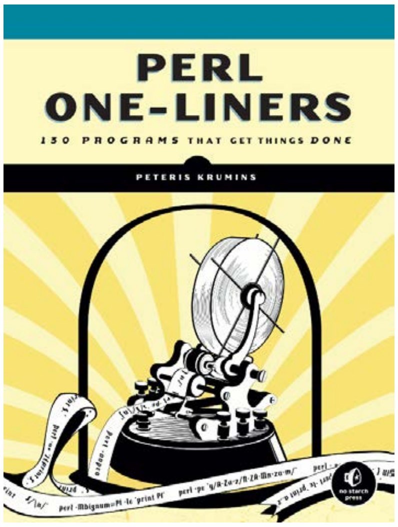
Pentaho Business Analytics

Businesses striving to increase their competitive advantage with big data analytics can do so with help from Pentaho Business Analytics 5.0, a completely redesigned data integration and analytics platform. The result of many years of intensive planning, research and conversations with customers and industry experts, Pentaho 5.0 provides a full spectrum of analytics for today's big data-driven businesses regardless of data type and volume, IT architecture or analysis required. The new, modern interface simplifies the user experience for all those working to turn data into competitive advantage. Highlights of Pentaho's 250+ new features and improvements include blended big data for more accurate insights, simplified analytics and user experience and enterprise-ready big data integration. These features are targeted at five key analytic personas in the enterprise, says Pentaho, and Pentaho 5.0 offers benefits to all these roles and includes functionality to help eliminate many of the common pain points that are holding back big data initiatives in companies of all sizes.
http://www.pentaho.com



Crucial LRDIMMs

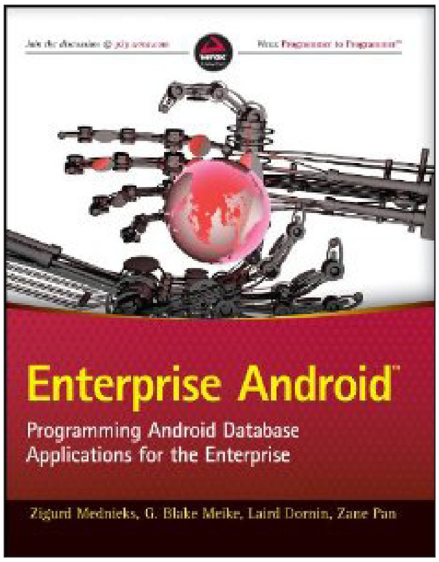
If Crucial could tell you one thing about its new 64GB DDR3L Load-Reduced DIMMs (LRDIMMs) for servers, it would be that they enable more DIMMs per channel for up to twice the installed memory capacity per server. By enabling high-end server environments with demanding workloads to reach the maximum amount of installed memory possible, dramatic performance gains in memory bandwidth and overall server productivity are possible, all while reducing power costs relative to adding additional servers. Utilizing 1.35V (vs. 1.5V) for optimal energy efficiency, Crucial LRDIMMs offer up to a 35% increase in memory bandwidth compared to standard registered DIMMs and eliminate the channel ranking limitation of standard DDR3 registered DIMMs. These new memory modules also are compatible with OEM servers and warranties, allowing users to upgrade their existing server infrastructures without having to purchase an entirely new system. Crucial LRDIMMs fully support the latest Intel Xeon processor E5 family.
http://www.crucial.com/server



Peteris Krumins' *Perl One-Liners* (No Starch)

Perl one-liners are small, super-nifty Perl programs that fit in one line of code and do one thing really well. Peteris Krumins' new book *Perl One-Liners: 130 Programs That Get Things Done* showcases short and compelling lines of code that do all sorts of handy, geeky things like numbering lines in a file; generating random passwords; encoding, decoding and converting strings; calculating factorials; and even checking to see if a number is prime with a regular expression. Krumins' dissections of each bit of code will help readers gain a deeper understanding of the Perl language, and the Perl one-liners contained herein are sure to save time and sharpen skills.
http://www.nostarch.com

Zigurd Mednieks, G. Blake Meike, Laird Dornin and Zane Pan's *Enterprise Android* (Wrox)


Enterprise Android
Programming Android Database Applications for the Enterprise
Zigurd Mednieks, G. Blake Meike, Laird Dornin, Zane Pan

Android devices are becoming ever more established in the mobile device market. With the release of Android 4, they are moving beyond consumer applications into corporate/enterprise use. Developers who want to build data-driven Android applications that integrate with enterprise systems will want to investigate the new Wrox book titled *Enterprise Android: Programming Android Database Applications for the Enterprise*, authored by Zigurd Mednieks, G. Blake Meike, Laird Dornin and Zane Pan. In the tradition of Wrox Professional guides, *Enterprise Android* thoroughly covers sharing and displaying data, transmitting data to enterprise applications and much more. Additional topics include collecting and storing data using SQLite, sharing data using content providers, displaying data using adapters, data migration, transmitting data with Web services and more.
http://www.wrox.com

The Library of Congress' Constitution Annotated App and Web Publication

Our community is packed with power citizens, from free speech advocates to Second Amendment gurus and civil rights activists. To inform your activism, or simply to live up to Thomas Jefferson's ideal of an informed citizenry, the Library of Congress has a new resource for you. *The Constitution of the United States of America: Analysis and Interpretation*, popularly known as the "Constitution Annotated" is now available as a free app and Web publication. The resource, formerly difficult to access by the general public due to its size and update cycle, makes analysis and interpretation of constitutional case law accessible to anyone with a computer or mobile device. The Web publication consists of digitally signed, searchable PDF documents. Meanwhile the app, which adds a wealth of interactive features, is available for the iOS platform; an Android version is currently under development.
http://beta.congress.gov/constitution-annotated

OpenStack Foundation's Training Marketplace

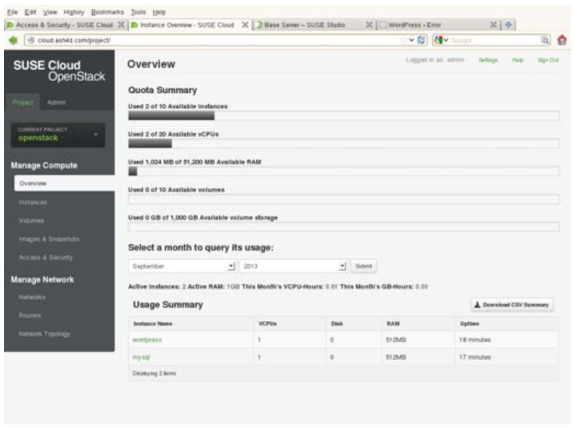
As the OpenStack cloud computing platform infiltrates data centers worldwide, the demand for educated administrators and users to manage these clouds grows in tandem. To help developers and operators gain the valuable skills they need to exploit the burgeoning opportunities, the OpenStack Foundation and its ecosystem partners have responded with the Training Marketplace. The on-line "course catalog" features dozens of courses across 10 countries and 25 cities, from providers at launch time as varied as Aptira, hastexo, The Linux Foundation, Mirantis, Morphlabs, Piston, Rackspace, Red Hat, SUSE and SwiftStack. In addition to paid and free training courses, many community efforts are available to produce helpful documentation, how-to information and new Operations and Security guide books.
http://www.openstack.org/marketplace/training

Zentyal Server

In midst of a cloud push, Zentyal Linux Small Business Server 3.2 represents a drop-in alternative to existing SMB solutions for companies seeking a choice in how to manage their network infrastructure. The all-in-one IT backoffice, which can be set up in less than 30 minutes, is an integration of the complex Samba technologies that provides native interoperability with Microsoft Active Directory and a range of other services. The solution speaks to customers who have logistical, security and connectivity concerns about moving to the cloud. The single most important improvement that Zentyal Server 3.2 introduces, according to its developer, is greater integration of the Samba technology, meaning that it is now possible to introduce Zentyal Server transparently on a Windows environment, migrate network services and users to Zentyal and then turn off unneeded Windows servers without causing inconvenience to users. Zentyal Server 3.2 also supports Group Policy Objects and Organizational Units and secures mobile communications to the private company resources out of the box.

http://www.zentyal.com

SUSE Cloud



Overview

Quota Summary

Used 2 of 10 Available Instances

Used 2 of 20 Available vCPUs

Used 1,024 MB of 51,200 MB Available RAM

Used 0 of 10 Available volumes

Used 0 GB of 1,000 GB Available volume storage

Select a month to query its usage:

Active Instances: 2 Active RAM: 1GB This Month's VCPU-Hours: 0.61 This Month's GB-Hours: 0.00

Usage Summary

Instance Name	VCPUs	Disk	RAM	Uptime
wordpress	1	0	512MB	18 minutes
mysql	1	0	512MB	17 minutes

Displaying 2 items

The headlining new feature of SUSE Cloud 2.0 is the capacity for setting up a mixed-hypervisor private cloud environment that can be deployed rapidly and managed easily, helping enterprises increase business agility and reduce IT costs. SUSE Cloud 2.0 supports KVM and Xen hypervisor environments and is the first OpenStack distribution (namely Grizzly) to add full support for Microsoft Hyper-V. VMware ESXi integration also is included as a technical preview. Other advances in v2.0 include a more robust installation process, integration with the Crowbar deployment framework, all of the features and fixes of OpenStack Grizzly, improved integration with SUSE Studio and SUSE Manager for building and managing cloud-based applications and a broader ecosystem of partner solutions to configure private clouds based on unique IT infrastructure requirements.

http://www.suse.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.

WEBCASTS

A Call to Arms for Private Cloud Builders

Sponsor: ActiveState | **Topic:** Cloud Computing **ON DEMAND**

The era of elastic IT is here. Businesses are realizing that the cloud not only allows cost reduction, but provides opportunities for innovation and growth. Elastic clouds enable next-generation applications that drive revenue opportunities, increase agility, and make IT teams competitive with public cloud systems.

In this presentation, Randy and John talk about the forces driving this change, and outline an action plan for building an elastic cloud infrastructure and dynamic applications using DevOps and Platform-as-a-Service.

> **http://lnxjr.nl/CTACloud**

Private PaaS for the Agile Enterprise

Sponsor: ActiveState | **Topic:** Virtualization

If you already use virtualized infrastructure, you are well on your way to leveraging the power of the cloud. Virtualization offers the promise of limitless resources, but how do you manage that scalability when your DevOps team doesn't scale? In today's hypercompetitive markets, fast results can make a difference between leading the pack vs. obsolescence. Organizations need more benefits from cloud computing than just raw resources. They need agility, flexibility, convenience, ROI, and control.

Stackato private Platform-as-a-Service technology from ActiveState extends your private cloud infrastructure by creating a private PaaS to provide on-demand availability, flexibility, control, and ultimately, faster time-to-market for your enterprise.

> **http://lnxjr.nl/privatepaasAE**

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

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

> **http://lnxjr.nl/IBM5factors**

Linux Backup and Recovery Webinar

Sponsor: Storix | **Topic:** Backup and Recovery

Most companies incorporate backup procedures for critical data, which can be restored quickly if a loss occurs. However, fewer companies are prepared for catastrophic system failures, in which they lose all data, the entire operating system, applications, settings, patches and more, reducing their system(s) to "bare metal." After all, before data can be restored to a system, there must be a system to restore it to.

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

> **http://lnxjr.nl/StorixWebinar**

WHITE PAPERS

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

Sponsor: Red Hat | Topic: Linux Management

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

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

Standardized Operating Environments for IT Efficiency

Sponsor: Red Hat

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

Benefits of an SOE:

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

SOE leads to:

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

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

Scale Out with GlusterFS

Learn how to install, benchmark and optimize this popular, shared-nothing and scalable open-source distributed filesystem.

ALEX DAVIES and ALESSANDRO ORSARIA

GlusterFS is an open-source distributed filesystem, originally developed by a small California startup, Gluster Inc. Two years ago, Red Hat acquired Gluster, and today, it sponsors GlusterFS as an open-source product with commercial support, called Red Hat Storage Server. This article refers to the latest community version of

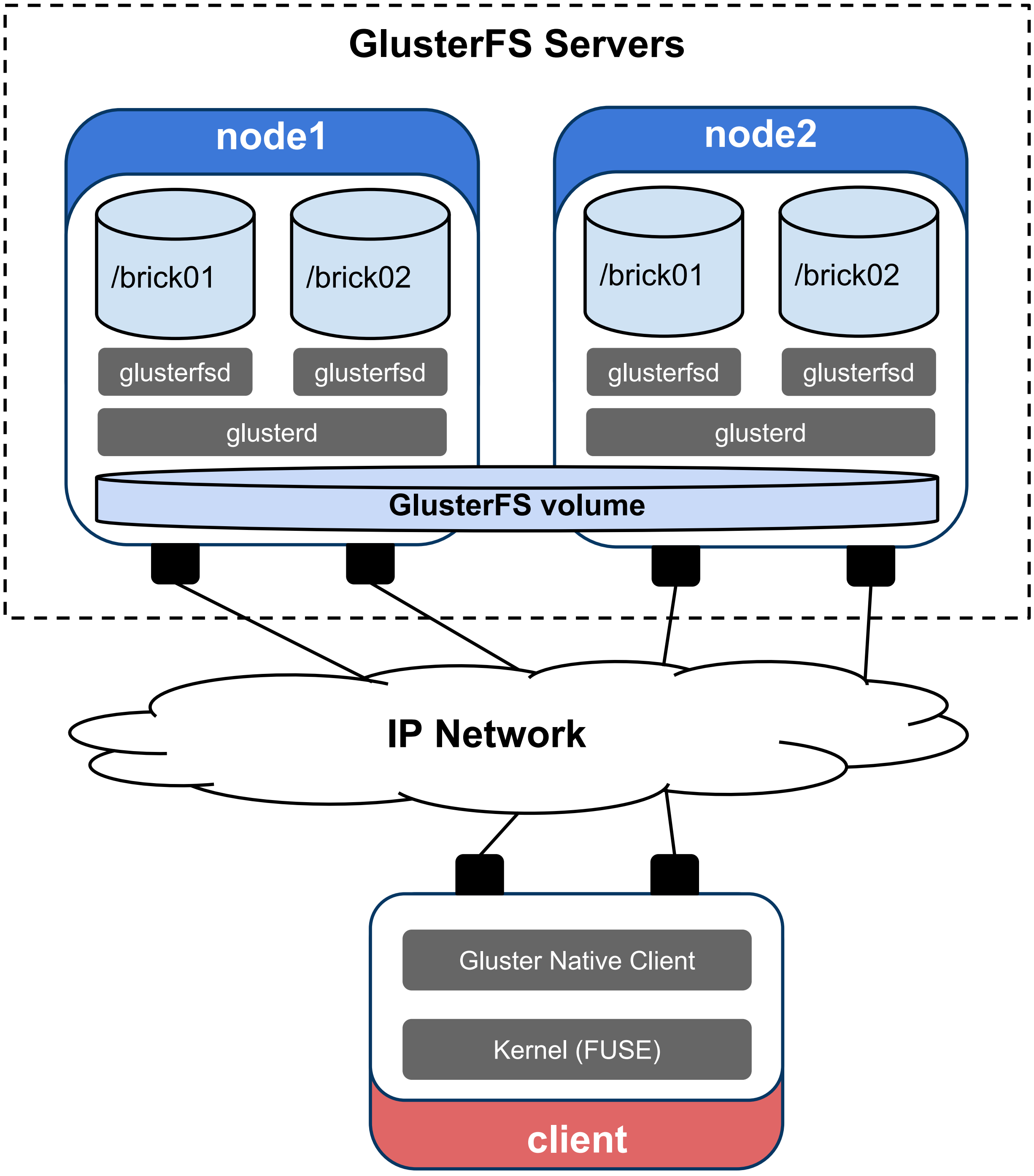


Figure 1. Simplified Architecture Diagram of a Two-Node GlusterFS Cluster and Native Client

Distributed Filesystems and Metadata Servers

While most distributed filesystems include a metadata server, GlusterFS does not. In a filesystem architecture with a metadata server, data is striped among different nodes, with another server keeping track of the location of the metadata and controlling access to the storage nodes. When a client issues an I/O operation to a file, it sends the request to the metadata server, which in turn tells the client where to retrieve the data from. With GlusterFS, native clients deterministically find the correct node where a file is stored via a hashing algorithm. Eliminating the metadata server is a big advantage, as this is typically a single point of contact for clients and often becomes a bottleneck.

GlusterFS for CentOS, which is 3.4 at the time of this writing. The material covered here also applies to Red Hat Storage Server as well as other Linux distributions.

Figure 1 shows a simplified architecture of GlusterFS. GlusterFS provides a unified, global namespace that combines the storage resources from multiple servers. Each node in the GlusterFS storage pool exports one or more bricks via the `glusterfsd` dæmon. Bricks are just local filesystems, usually indicated by the hostname:/directory notation. They are the basic building blocks of a GlusterFS volume, which can be mounted using the GlusterFS Native Client. This typically provides the best performance and is based on the Filesystem In Userspace (FUSE) interface built in to the Linux kernel. GlusterFS also provides object access via the OpenStack Swift API and connectivity through the NFS and CIFS protocols.

You can increase capacity or IOPS in a GlusterFS storage pool by adding more nodes and expanding the existing volumes to include the new bricks. As you provision additional storage nodes, you increase not only the total storage capacity, but network bandwidth, memory and CPU resources as well. GlusterFS scales "out" instead of "up".

Unlike traditional storage arrays, there is no hard limit provided by the array controller; you are limited by the sum of all your storage nodes. This scalability results in a small latency penalty per I/O operation. Therefore, GlusterFS works very well where nearline storage access is required, such as bandwidth-intensive HPC or general-purpose archival storage. It is less suited for latency-sensitive applications, such as highly transactional databases.

Getting Started with GlusterFS

Installing GlusterFS is straightforward. You can download packages for the most common distributions, including CentOS, Fedora, Debian and Ubuntu. For CentOS and other distributions with the yum package manager, just add the corresponding yum repo, as explained in the official GlusterFS Quick Start Guide (see Resources). Then, install the glusterfs-server-* package on the storage servers and glusterfs-client-* on the clients. Finally, start the `glusterd` service on the storage nodes:

```
# service glusterd start
```

The next task is to create a trusted storage pool, which is made from the bricks of the storage nodes. From one of the servers, for instance node1, just type:

```
# gluster peer probe node2
```

And, repeat that for all the other storage nodes. The command `gluster peer status` lists all the other peers and their connection status.

Provisioning a Volume

The first step to provision a volume consists of setting up the bricks. As mentioned previously, these are mountpoints on the storage nodes, with XFS being the recommended filesystem. GlusterFS uses extended file attributes to store key/value pairs for several purposes, such as volume replication or striping across different bricks. To make this work with XFS, you should format the bricks using a larger inode size—for example, 512 bytes:

```
# mkfs.xfs -i 512 /brick01
```

GlusterFS volumes can belong to several different types. They can be *distributed*, *replicated* or *striped*. In addition, they can be a combination of these—for instance, *distributed replicated* or *striped replicated*. You can specify the type of a volume using the `stripe` and `replica` keywords of the `gluster` command. There is no keyword to indicate that a volume is distributed, as this is the default configuration.

Distributed volumes store whole files across different bricks. This is similar to striping, but the minimum "unit of storage" in a brick for a distributed volume is a single file. Conversely, striped volumes split files in chunks (of 128KB by default) and stripe those across bricks. Although striped volumes increase complexity, you might use them if you have very large files that do not fit in a single brick or in scenarios where a lot of file contention is experienced. The last volume type, replicated, provides redundancy by synchronously replicating files across different bricks.

As an example, let's create a distributed replicated volume using the bricks mounted on the /brick01 and /brick02 directories on two different storage nodes:

```
# gluster volume create MYVOL replica 2 \
> node{1,2}:/brick01 node{1,2}:/brick02
# gluster volume start MYVOL
```

The `replica 2` option tells GlusterFS to dedicate half the bricks for two-way replication. Keep in mind that the order of the bricks *does* matter. In the previous example, `node1:/brick01` will mirror `node2:/brick01`. The same logic applies to /brick02.

Finally, you can make the new volume available on other machines using the GlusterFS native client:

```
# mount -t glusterfs node1:/MYVOL /mnt/DIR
```

You can pass the `-o backupvolfile-server` option on the previous command to specify an alternative server from which to mount the filesystem, should node1 be unavailable.

GlusterFS offers more flexibility and scalability than a traditional monolithic storage solution.

Scaling GlusterFS

GlusterFS offers more flexibility and scalability than a traditional monolithic storage solution. But, this comes at a complexity cost, as you now have more control and need to consider several aspects for best performance.

To check whether performance meets your expectations, start with a baseline benchmark to measure throughput on the newly created GlusterFS volume. You can use several tools to do this, such as `iozone`, `mpirun` or `dd`. For example, run the following script after changing directory into a GlusterFS mountpoint:

```
#!/bin/bash
for BS in 64 128 256 512 1024 2048 4096 8192 ; do
  iozone -R -b results-${BS}.xls -t 16 -r $BS -s 8g \
    -i 0 -i 1 -i 2
done
```

The previous code runs `iozone` with different block sizes and 16 active client threads, using an 8GB file size to test. To minimize the effects of caching, you should benchmark I/O performance using a file size at least twice the RAM installed on your servers. However, as RAM is cheap and many new servers come equipped with a lot of memory, it may be impractical to test with very large file sizes. Instead, you can limit the amount of RAM seen by the kernel to 2GB—for example, by passing the `mem=2G` kernel option at boot.

Generally, you will discover that throughput is either limited by the network or the storage layer. To identify where the system is constrained, run `sar` and `iostat` in separate terminals for each GlusterFS node, as illustrated below:

```
# sar -n DEV 2 | egrep "(IFACE|eth0|eth1|bond0)"
# iostat -N -t -x 2 | egrep "(Device|brick)"
```

Keep an eye on the throughput of each network interface reported by the `sar` command above. Also, monitor the utilization column of the `iostat` output to check whether any of the bricks are saturated. The commands `gluster volume top` and `gluster volume profile` provide many other useful performance metrics. Refer to the documentation on the Gluster community Web site for more details.

Once you have identified where the bottleneck is on your system, you can do some simple tuning.

Tuning the Network Layer

GlusterFS supports TCP/IP and RDMA as the transport protocols. TCP on IPoIB (IP over InfiniBand) is also a viable option. Due to space limitations, here we mostly refer to TCP/IP over Ethernet, although some considerations also apply to IPoIB.

There are several ways to improve network throughput on your storage servers. You certainly should minimize per-packet overhead and balance traffic across your network interfaces in the most effective way, so that none of the NIC ports are underutilized.

Under normal conditions, no particular TCP tuning is required, as the Linux kernel enables TCP window scaling by default and does a good job of automatically adjusting buffer sizes for TCP performance.

To reduce packet overhead, you should put your storage nodes on the same network segment as the clients and enable jumbo frames. To do so, all the interfaces on your network switches and GlusterFS servers must be set up with a larger MTU size. Typically a maximum value of 9,000 bytes is allowed on most switch platforms and network cards.

You also should ensure that TSO (TCP segment offload) or GSO (generic segment offload) is enabled, if supported by the NIC driver. Use the `ethtool` command with the `-k` option to verify the current settings. TSO and GSO allow the kernel to manipulate much larger packet sizes and offload to the NIC hardware the process of segmenting packets into smaller frames, thus reducing CPU and packet overhead. Similarly, large receive offload (LRO) and generic receive offload (GRO) can improve throughput in the inbound direction.

Another important aspect to take into consideration is NIC bonding. The Linux bonding driver provides several load-balancing policies, such as active-backup or round-robin. In a GlusterFS setup, an active-backup policy may result in a bottleneck, as only one of the Ethernet interfaces will be used to transmit and receive packets. Conversely, a pure round-robin policy provides per-packet load balancing and full utilization of the bandwidth of all the interfaces if required. However, depending on your network topology, round-robin may cause out-of-order packet delivery, and your system would require extra CPU cycles to re-assemble them.

Better alternatives in this case are IEEE 802.3ad dynamic link aggregation (selected with `mode=4` in the bonding driver) or adaptive load balancing (`mode=6`). To deploy IEEE 802.3ad link aggregation, an essential requirement is that the switches connected to the servers must also support this protocol. Then, the `xmit_hash_policy` parameter of the Linux kernel bonding driver specifies the hash policy used to balance traffic across the interfaces, when 802.3ad is in operation. For better results, this parameter should be set to `layer3+4`. This policy uses a hash of the source and destination IP addresses and TCP ports to select the outbound interface. In this way, traffic is load-balanced per flow, rather than per packet, and the end result is that packets from a certain connection socket always will be delivered out of the same network interface.

In GlusterFS, when a client mounts a volume using the native protocol, a TCP connection is established from the client for each brick hosted on the storage nodes. On an eight-node cluster with two bricks per node, this results in 16 TCP connections. Hence, the "layer3+4" hash transmit policy typically would contain enough "entropy" to split those connections evenly across the network interfaces, resulting in an efficient and balanced usage of all the NIC ports.

A final consideration concerns management traffic. For security purposes, you may want to separate GlusterFS and management traffic (such as SSH, SNMP and so on) onto different subnets. To achieve best performance, you also should use different physical interfaces; otherwise, you may get contention between storage and management traffic over the same NIC ports. If your servers come with only two network interfaces, you still can split storage and management traffic onto different subnets by using VLAN tagging.

Figure 2 illustrates a potential configuration to achieve this result. On a server with two physical network

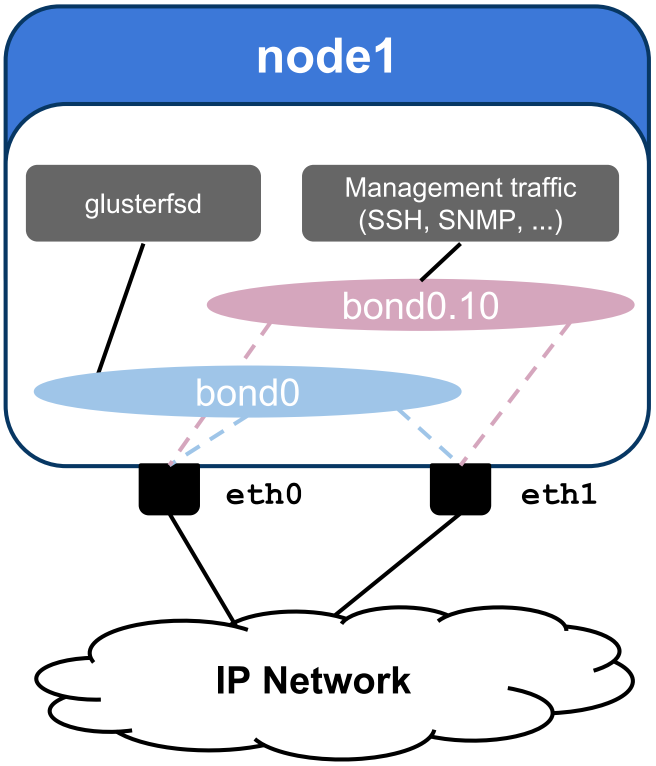


Figure 2. Network Configuration on a GlusterFS Server with Multiple Bond Interfaces and VLAN Tagging

ports, we have aggregated those into a bond interface named bond0 for GlusterFS traffic. The corresponding configuration is shown below:

```
# cat /etc/sysconfig/network-scripts/ifcfg-bond0

DEVICE=bond0

BONDING_OPTS="miimon=100 mode=4 xmit_hash_policy=layer3+4"

MTU=9000

BOOTPROTO=none

IPADDR=192.168.1.1

NETMASK=255.255.255.0
```

We have also created a second, logical interface named bond0.10. The configuration is listed below:

```
# cat /etc/sysconfig/network-scripts/ifcfg-bond0.10
DEVICE=bond0.10
ONPARENT=yes
VLAN=10
MTU=1500
IPADDR=10.1.1.1
NETMASK=255.255.255.0
GATEWAY=10.1.1.254
```

This interface is used for traffic management and is set up with a standard MTU of 1,500 bytes and a VLAN tag. Conversely, bond0 is not set with any VLAN tag, so it will send traffic untagged, which on the switches will be assigned to the native VLAN of the corresponding Ethernet trunk.

Tuning the Storage Layer

As of today, the commodity x86 servers with the highest density of local storage can fit up to 26 local drives in their chassis. If two drives are dedicated to the OS, this leaves 24 drives to host the GlusterFS bricks. You can add additional drives via Direct Attached Storage (DAS), but before scaling "up", you should check whether you have enough spare network bandwidth. With large sequential I/O patterns, you may run out of bandwidth well before exhausting the throughput of your local drives.

The optimal RAID level and LUN size depend on your workload. If your application generates lots of random write I/O, you may consider aggregating all the drives in a single RAID 10 volume. However, efficiency won't be the best, as you will lose 50% of your capacity. On the other hand, you can go with a RAID 0 configuration for maximum efficiency and performance, and rely on GlusterFS to provide data redundancy via its replication capabilities.

It is not surprising that the best approach often lies in the middle. With 24 drives per node, Red Hat's recommended setup consists of 2 x RAID 6 (10+2) volumes (see Resources). A single volume with a RAID 50 or 60 configuration also can provide similar performance.

On a LUN spanning a RAID 6 volume of 12 SAS hard disk drives, a typical sustained throughput is 1–2GB per second, depending on the type and model of the disk drives. To transfer this amount of data across the network, a single 10Gbps interface easily would become a bottleneck. That's why you should load-balance traffic across your network interfaces, as we discussed previously.

It is important to configure a local LUN volume properly to get the most of the parallel access benefits provided by the RAID controller. The chunk size (often referred to as element stripe size) is the "atomic" block of data that the RAID controller can write to each disk drive before moving onto the next one. If the average I/O request size issued by an application is smaller than the chunk size, only one drive at a time would handle most of the I/O.

The optimal chunk size is obtained by dividing the average I/O request size by the number of data-bearing disks. As an example, a RAID 6 (10+2) volume configuration gives ten data-bearing disks. If the average I/O request issued by your application is 1MB, this gives a chunk size (approximated to the nearest power of 2) of 128KB. Then, the minimum amount of data that the RAID controller distributes across the drives is given by the number of data-bearing disks multiplied by the chunk size. That is, in this example, 10 x 128 KB = 1280 KB, slightly larger than the average I/O request of 1MB that we initially assumed.

Next, ensure that LVM properly aligns its metadata based on the stripe value that was just calculated:

```
# pvcreate --dataalignment=1280k /dev/sdb
```

When formatting the bricks, `mkfs` can optimize the distribution of the filesystem metadata by taking into account the layout of the underlying RAID volume. To do this, pass the `su` and `sw` parameters to `mkfs.xfs`. These indicate the chunk size in bytes and the stripe width as a multiplier of the chunk size, respectively:

```
# mkfs.xfs -d su=128k,sw=10 -i 512 /brick01
```

Another setting to consider is the default *read ahead* of the brick devices. This tells the kernel and RAID controller how much data to pre-fetch when sequential requests come in. The actual value for the kernel is available in the /sys virtual filesystem and is set to 128KB by default:

```
# cat /sys/block/sdb/queue/read_ahead_kb
128
```

A typical SAS hard drive with 15k RPM of rotational speed has a standard average response time of around 5ms and can read 128KB (the default read ahead) in less than a millisecond. Under I/O contention, the drive may spend most of its time positioning to a new sector for the next I/O request and use only a small fraction of its time to transfer data. In such situations, a larger read ahead can improve performance. The rationale behind this change is that it is better to pre-fetch a big chunk of data in the hope that it will be used later, as this will reduce the overall seek time of the drives.

Again, there is no optimal value for all workloads, but as a starting point, it may be worth noting that Red Hat recommends to set this parameter to 64MB. However, if your application I/O profile is IOPS-intensive and mostly made of small random requests, you may be better off tuning the read ahead size to a lower value and deploying solid disk drives. The commercial version of GlusterFS provides a built-in profile for the `tuned` dæmon, which automatically increases read ahead to 64MB and configures the bricks to use the deadline I/O elevator.

Conclusion

GlusterFS is a versatile distributed filesystem, well supported on standard Linux distributions. It is trivial to install and simple to tune, as we have shown here. However, this article is merely a short introduction. We did not have space to cover many of the most exciting features of GlusterFS, such as asynchronous replication, OpenStack Swift integration and CIFS/NFS exports—to mention just a few! We hope this article enables you to give GlusterFS a go in your own environment and explore some of its many capabilities in more detail.■

Alex Davies and Alessandro Orsaria worked together for a large Web firm before moving into technology within the algorithmic and hedge-fund industries. When they are not trekking in Scotland, mountain climbing in the Alps, playing the piano or volunteering, they are typically in front of a computer and can be reached at alex@davz.net and alex@linux.it.

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

Resources

GlusterFS Download Page: **http://download.gluster.org**

GlusterFS Quick Start Guide: **http://www.gluster.org/community/documentation/index.php/QuickStart**

B. England and N. Khare, *Best Practices for Red Hat Storage Server Performance*: slides available from **http://www.redhat.com/summit/2013/presentations**

Dell Drive Characteristics and Metrics: **http://www.dell.com/downloads/global/products/pvaul/en/enterprise-hdd-sdd-specification.pdf**

Linux Ethernet Bonding Driver HOWTO: **https://www.kernel.org/doc/Documentation/networking/bonding.txt**

8-10th Nov. GOTHENBURG SWEDEN

FSCONS

SEMANTICS OF FREEDOM

fscons.org

Zato Agile ESB, SOA, REST and Cloud Integrations in Python

Learn how to integrate applications with an elegant and agile platform that is quick to adapt to ever-faster business changes. This how-to introduces the methodology and includes an integration example of a treasury.gov's financial API with two external apps.

Dariusz Suchojad

Zato is a Python-based platform for integrating applications and exposing back-end services to front-end clients (**https://zato.io/docs/index.html**). It's an ESB (Enterprise Service Bus) and an application server focused on data integrations. The platform doesn't enforce any limits on architectural style for designing systems and can be used for SOA (Service Oriented Architecture), REST (Representational State Transfer) and for building systems of systems running in-house or in the cloud.

ZATO PROMOTES LOOSE COUPLING, REUSABILITY OF COMPONENTS AND HOT-DEPLOYMENT.

At its current version of 1.1 (at the time of this writing), Zato supports HTTP, JSON, SOAP, SQL, AMQP, JMS WebSphere MQ, ZeroMQ, Redis NoSQL and FTP. It includes a browser-based GUI, CLI, API, security, statistics, job scheduler, HAProxy-based load balancer and hot-deployment. Each piece is extensively documented from the viewpoint of several audiences: architects, admins and programmers.

Zato servers are built on top of gevent and gunicorn frameworks that are responsible for handling incoming traffic using asynchronous notification libraries, such as libevent or libev, but all of that is hidden from programmers' views so they can focus on their job only.

Servers always are part of a cluster and run identical copies of services deployed. There is no limit on how many servers a single cluster can contain.

Each cluster keeps its configuration in Redis and an SQL database. The former is used for statistics or data that is frequently updated and mostly read-only. The latter is where the more static configuration shared between servers is kept.

Users access Zato through its Web-based GUI (**https://zato.io/docs/web-admin/intro.html**), the command line (**https://zato.io/docs/admin/cli/index.html**) or API (**https://zato.io/docs/public-api/intro.html**).

Zato promotes loose coupling, reusability of components and hot-deployment. The high-level goal is to make it trivial to access or expose any sort of information. Common integration techniques and needs should be, at most, a couple clicks away, removing the need to reimplement the same steps constantly, slightly differently in each integration effort.

Everything in Zato is about minimizing the interference of components on each other, and server-side objects you create can be updated easily, reconfigured on fly or reused in other contexts without influencing any other.

This article guides you through the process of exposing complex XML data to three clients using JSON, a simpler form of XML and SOAP, all from a single code base in an elegant and Pythonic way that doesn't require you to think about the particularities of any format or transport.

To speed up the process of retrieving information by clients, back-end data will be cached in Redis and updated periodically by a job-scheduled service.

The data provider used will be US Department of the Treasury's real long-term interest rates (**http://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=reallongtermrate**). Clients will be generic HTTP-based ones invoked through curl, although in practice, any HTTP client would do.

The Process and IRA Services

The goal is to make it easy and efficient for external client applications to access long-term US rates information. To that end, you'll make use of several features of Zato:

- Scheduler will be employed to invoke a service to download the XML offered by treasury.gov and parse interesting data out of it.
- Redis will be used as a cache to store the results of parsing the XML.
- Zato's SimpleIO (SIO, **https://zato.io/docs/progguide/sio.html**) will allow you to expose the same set of information to more than one client, each using a different message format, without any code changes or server restarts.

Zato encourages the division of each business process into a set of IRA services (**https://zato.io/docs/intro/esb-soa.html**)—that is, each service exposed to users should be:

- *Interesting*: services should provide a real value that makes potential users pause for a moment and, at least, contemplate using the service in their own applications for their own benefit.

- *Reusable*: making services modular will allow you to make use of them in circumstances yet unforeseen—to build new, and possibly unexpected, solutions on top of lower-level ones.

- *Atomic*: a service should have a well defined goal, indivisible from the viewpoint of a service's users, and preferably no functionality should overlap between services.

The IRA approach closely follows the UNIX philosophy of "do one thing and do it well" as well as the KISS principle that is well known and followed in many areas of engineering.

When you design an IRA service, it is almost exactly like defining APIs between the components of a standalone application. The difference is that services connect several applications running in a distributed environment. Once you take that into account, the mental process is identical.

Anyone who already has created an interesting interface of any sort in a single-noded application written in any programming language will feel right like home when dealing with IRA services.

From Zato's viewpoint, there is no difference in whether a service corresponds to an S in SOA or an R in REST; however, throughout this article, I'm using the the former approach.

WHEN YOU DESIGN AN IRA SERVICE, IT IS ALMOST EXACTLY LIKE DEFINING APIS BETWEEN THE COMPONENTS OF A STANDALONE APPLICATION.

Laying Out the Services

The first thing you need is to diagram the integration process, pull out the services that will be implemented and document their purpose. If you need a hand with it, Zato offers its own API's documentation as an example of how a service should be documented (see **https://zato.io/docs/progguide/documenting.html** and **https://zato.io/docs/public-api/intro.html**):

- Zato's scheduler is configured to invoke a service (update-cache) refreshing the cache once in an hour.

- update-cache, by default, fetches the XML for the current month, but it can be configured to grab data for any date. This allows for reuse of the service in other contexts.

- Client applications use either JSON or simple XML to request long-term rates (get-rate), and responses are produced based on data cached in Redis, making them super-fast. A single SIO Zato service can produce responses in JSON, XML or SOAP. Indeed, the same service can be exposed independently in completely different channels, such as HTTP or AMQP, each using different security definitions and not interrupting the message flow of other channels.

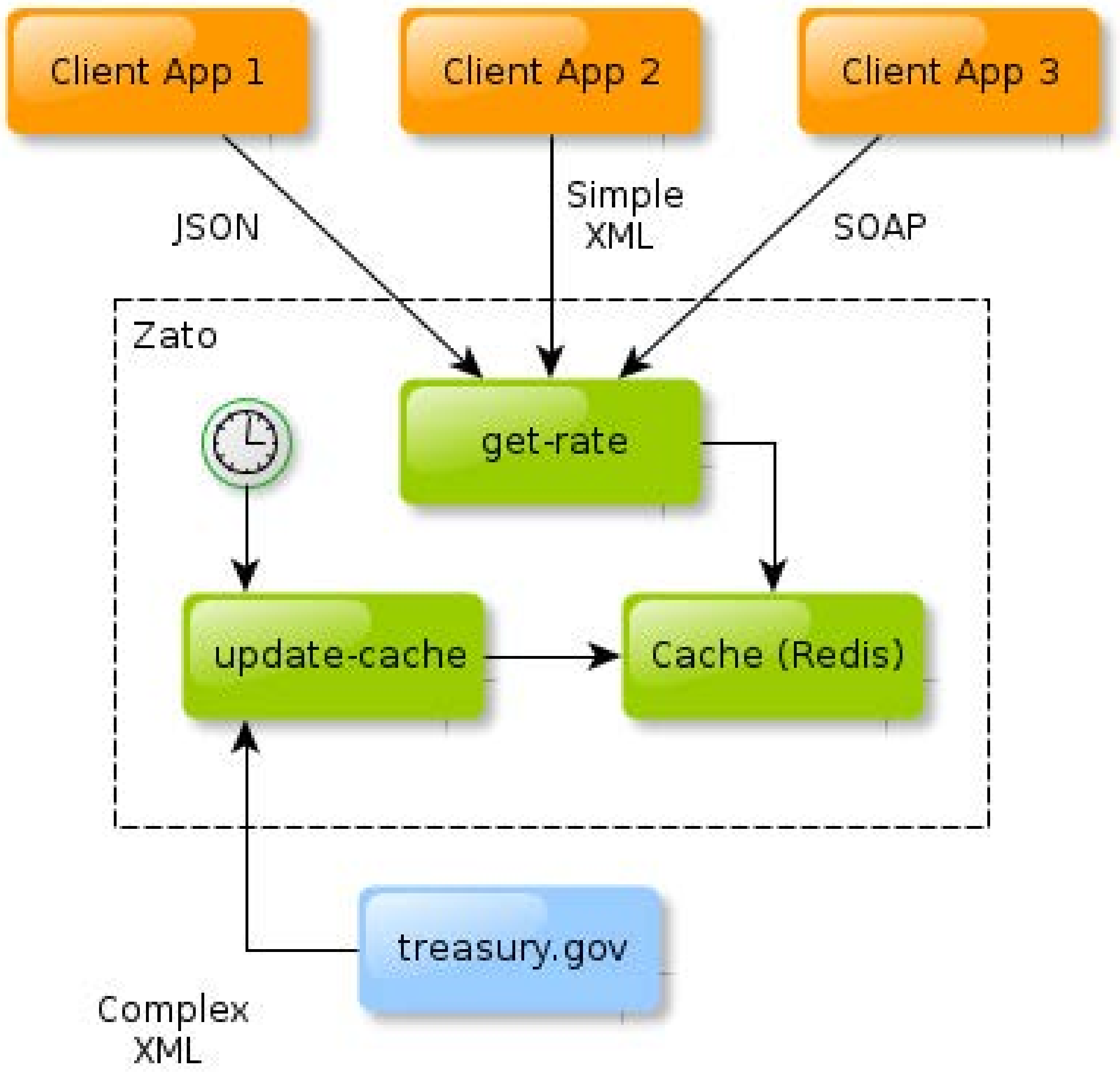


Figure 1. Overall Business Process

WHEN USING ZATO SERVICES, YOU ARE NEVER REQUIRED TO HARD-CODE NETWORK ADDRESSES.

Implementation

The full code for both services is available as a gist on GitHub, and only the most interesting parts are discussed (**https://gist.github.com/dsuch/6440366**).

linuxjournal.update-cache

Steps the service performs are:

- Connect to treasury.gov.
- Download the big XML.
- Find interesting elements containing the business data.
- Store it all in Redis cache.

Key fragments of the service are presented below.

When using Zato services, you are never required to hard-code network addresses. A service shields such information and uses human-defined names, such as "treasury.gov"; during runtime, these resolve into a set of concrete connection parameters. This works for HTTP and any other protocol supported by Zato. You also can update a connection definition on the fly without touching the code of the service and without any restarts:

```
# Fetch connection by its name
out = self.outgoing.plain_http.get('treasury.gov')

# Build a query string the backend data source expects
query_string = {
  '$filter':'month(QUOTE_DATE) eq {} and year(QUOTE_DATE) eq
{}'.format(month, year)
}

# Invoke the backend with query string, fetch
  # the response as a UTF-8 string
# and turn it into an XML object
response = out.conn.get(self.cid, query_string)
```

lxml (**http://lxml.de**) is a very good Python library for XML processing and is used in the example to issue XPath queries against the complex document (**http://data.treasury.gov/feed.svc/DailyTreasuryRealLongTermRateAverageData?$filter=month%28QUOTE_DATE%29%20eq%208%20and%20year%28QUOTE_DATE%29%20eq%202013**) returned:

```
xml = etree.fromstring(response)

# Look up all XML elements needed (date and rate) using XPath
elements = xml.xpath('//m:properties/d:*/text()',
 ➥namespaces=NAMESPACES)
```

For each element returned

by the back-end service, you create an entry in the Redis cache in the format specified by `REDIS_KEY_PATTERN`—for instance, `linuxjournal:rates:2013:09:03` with a value of 1.22:

```
for date, rate in elements:

  # Create a date object out of string
  date = parse(date)

  # Build a key for Redis and store the data under it
  key = REDIS_KEY_PATTERN.format(
      date.year, str(date.month).zfill(2),
      ➥str(date.day).zfill(2))
  self.kvdb.conn.set(key, rate)

  # Leave a trace of our activity
  self.logger.info('Key %s set to %s', key, rate)
```

linuxjournal.get-rate

Now that a service for updating the cache is ready, the one to return the data is so simple yet powerful that it can be reproduced in its entirety:

```
class GetRate(Service):
""" Returns the real long-term rate for a given date
(defaults to today if no date is given).
"""
class SimpleIO:
    input_optional = ('year', 'month', 'day')
    output_optional = ('rate',)

def handle(self):
    # Get date needed either from input or current day
    year, month, day = get_date(self.request.input)

    # Build the key the data is cached under
    key = REDIS_KEY_PATTERN.format(year, month, day)

    # Assign the result from cache directly to response
    self.response.payload.rate = self.kvdb.conn.get(key)
```

A couple points to note:

- SimpleIO was used—this is a declarative syntax for expressing simple documents that can be serialized to JSON or XML in the current Zato version, with more to come in future releases.

- Nowhere in the service did you have to mention JSON, XML or even HTTP at all. It's all working on a high level of Python objects without specifying any output format or transport method.

This is the Zato way. It promotes reusability, which is valuable because a generic and interesting service, such as returning interest rates, is bound to be desirable in situations that cannot be predicted.

As an author of a service, you are not forced into committing to a particular format. Those are configuration details that can be taken care of through a variety of means, including a GUI that Zato provides. A single service can be exposed simultaneously through multiple access channels each using a different data format, security definition or rate limit independently of any other.

Installing Services

There are several ways to install a service:

- Hot-deployment from the command line.
- Hot-deployment from the browser.
- Adding it to services-sources.txt—you can specify a path to a single module, to a Python package or a Python-dotted name by which to import it.

Let's hot-deploy what you have so far from the command line, assuming a Zato server is installed in /opt/zato/server1. You can do this using the `cp` command:

```
$ cp linuxjournal.py /opt/zato/server1/pickup-dir
$
```

Now in the server log:

```
INFO - zato.hot-deploy.create:22 - Creating tar archive
INFO - zato.hot-deploy.create:22 - Uploaded package id:[21],
 ➥payload_name:[linuxjournal.py]
```

Here's what just happened:

- The server to be deployed was stored in an SQL database, and each server from a cluster was notified of the deployment of new code.
- Each server made a backup of currently deployed services and stored it in the filesystem (by default, there's a circular log of the last 100 backups kept).
- Each server imported the service and made it available for use.

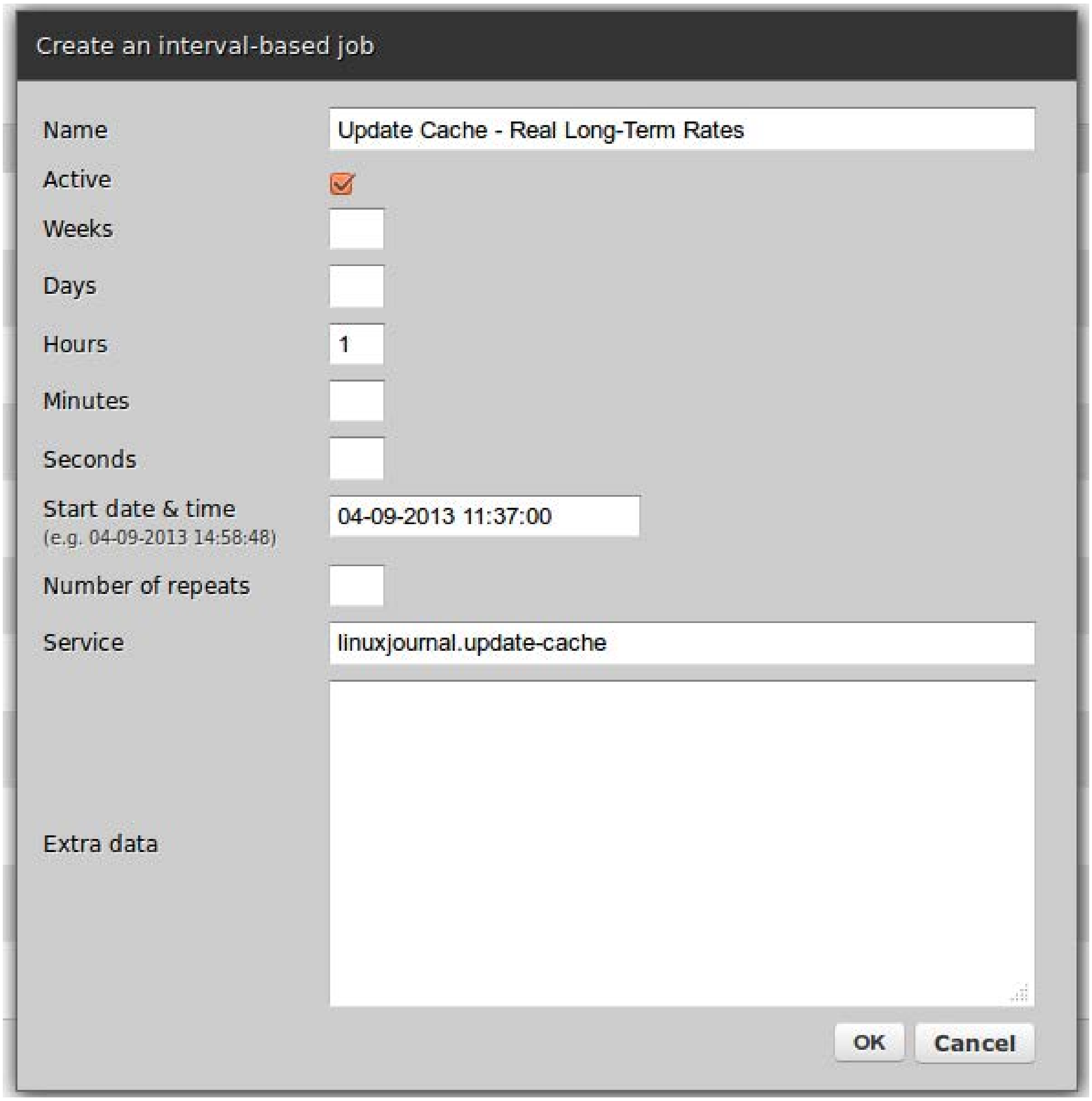
All those changes were introduced throughout the whole cluster with no restarts and no reconfiguration.

Using the GUI to Configure the Resources Needed

Zato's Web admin is a GUI that can be used to create server objects that services need quickly, check runtime statistics or gather information needed for debugging purposes.

The Web admin is merely a client (**https://zato.io/docs/progguide/clients/python.html**) of Zato's own API, so everything it does also can be achieved from the command line or by user-created clients making API calls.

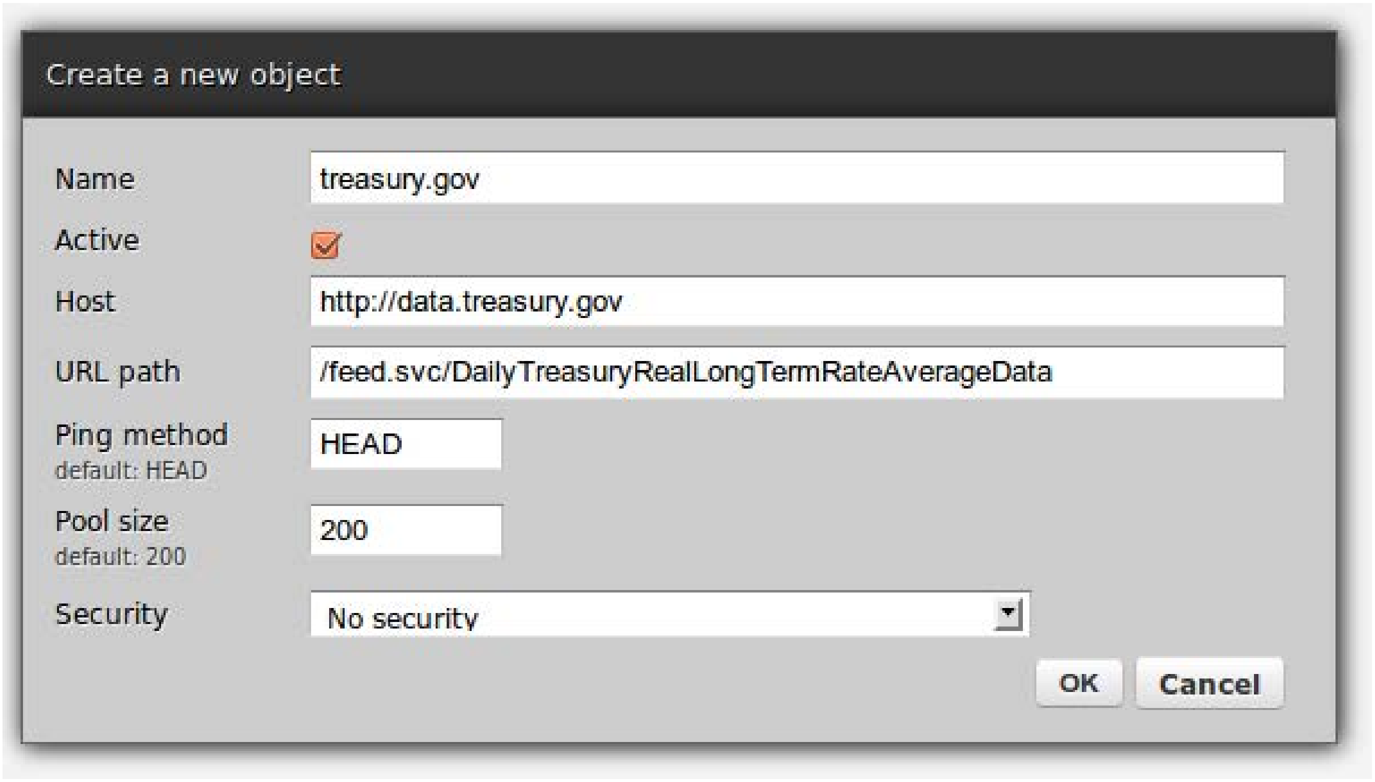
On top of that, server-side objects can be managed "en masse" (**https://zato.io/docs/admin/guide/enmasse.html**) using a JSON-based configuration that can be kept in a config repository for versioning and diffing. This allows for interesting workflows, such as creating a base



Create an interval-based job

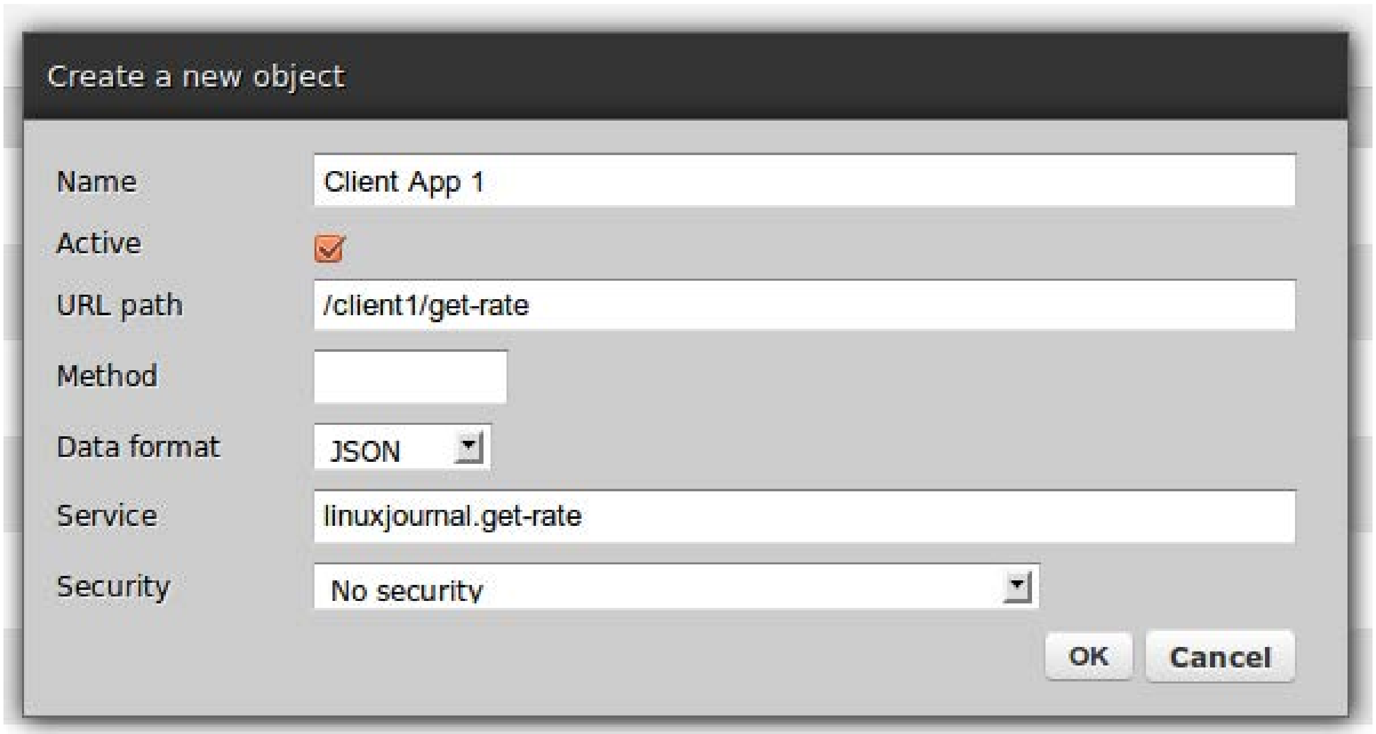
Name	Update Cache - Real Long-Term Rates
Active	✓
Weeks	
Days	
Hours	1
Minutes	
Seconds	
Start date & time (e.g. 04-09-2013 14:58:48)	04-09-2013 11:37:00
Number of repeats	
Service	linuxjournal.update-cache
Extra data	

Figure 2. Scheduler Job Creation Form


Create a new object

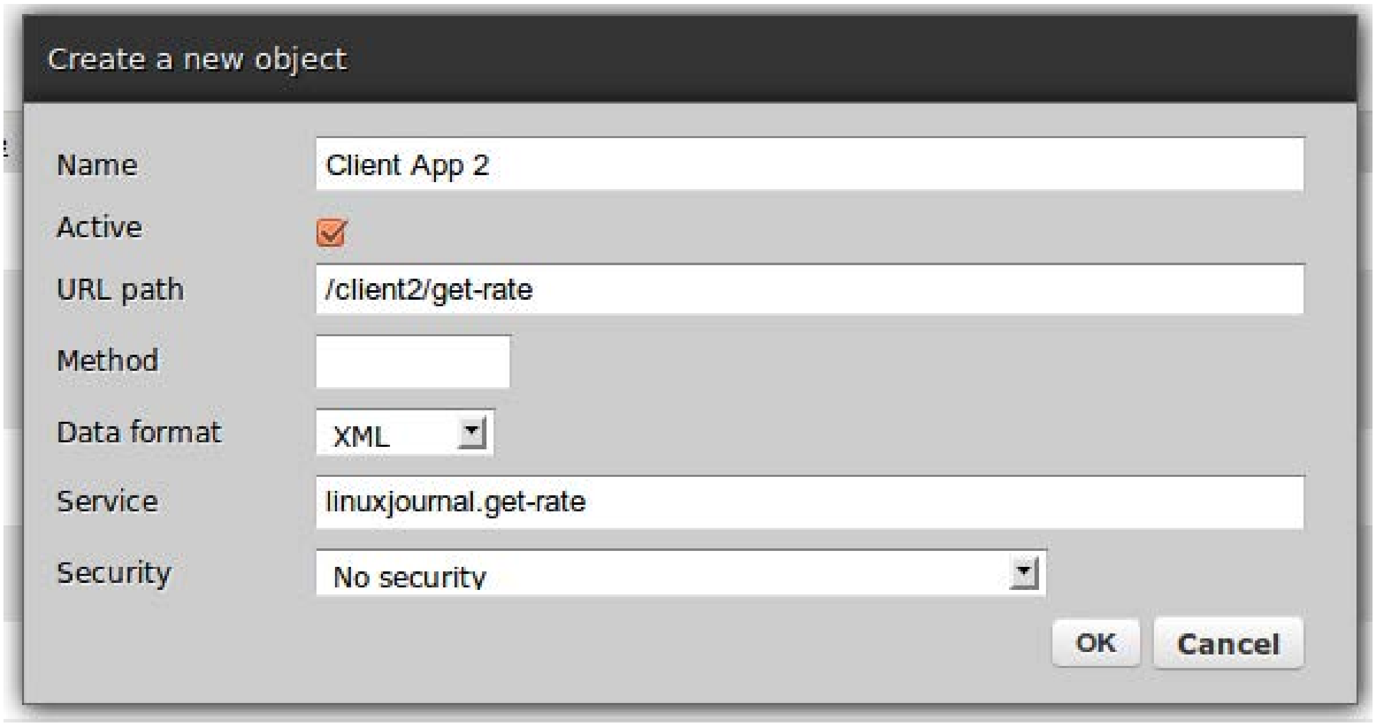
Name	treasury.gov
Active	✓
Host	http://data.treasury.gov
URL path	/feed.svc/DailyTreasuryRealLongTermRateAverageData
Ping method default: HEAD	HEAD
Pool size default: 200	200
Security	No security

Figure 3. Outgoing HTTP Connection Creation Form


Create a new object

Name	Client App 1
Active	✓
URL path	/client1/get-rate
Method	
Data format	JSON
Service	linuxjournal.get-rate
Security	No security

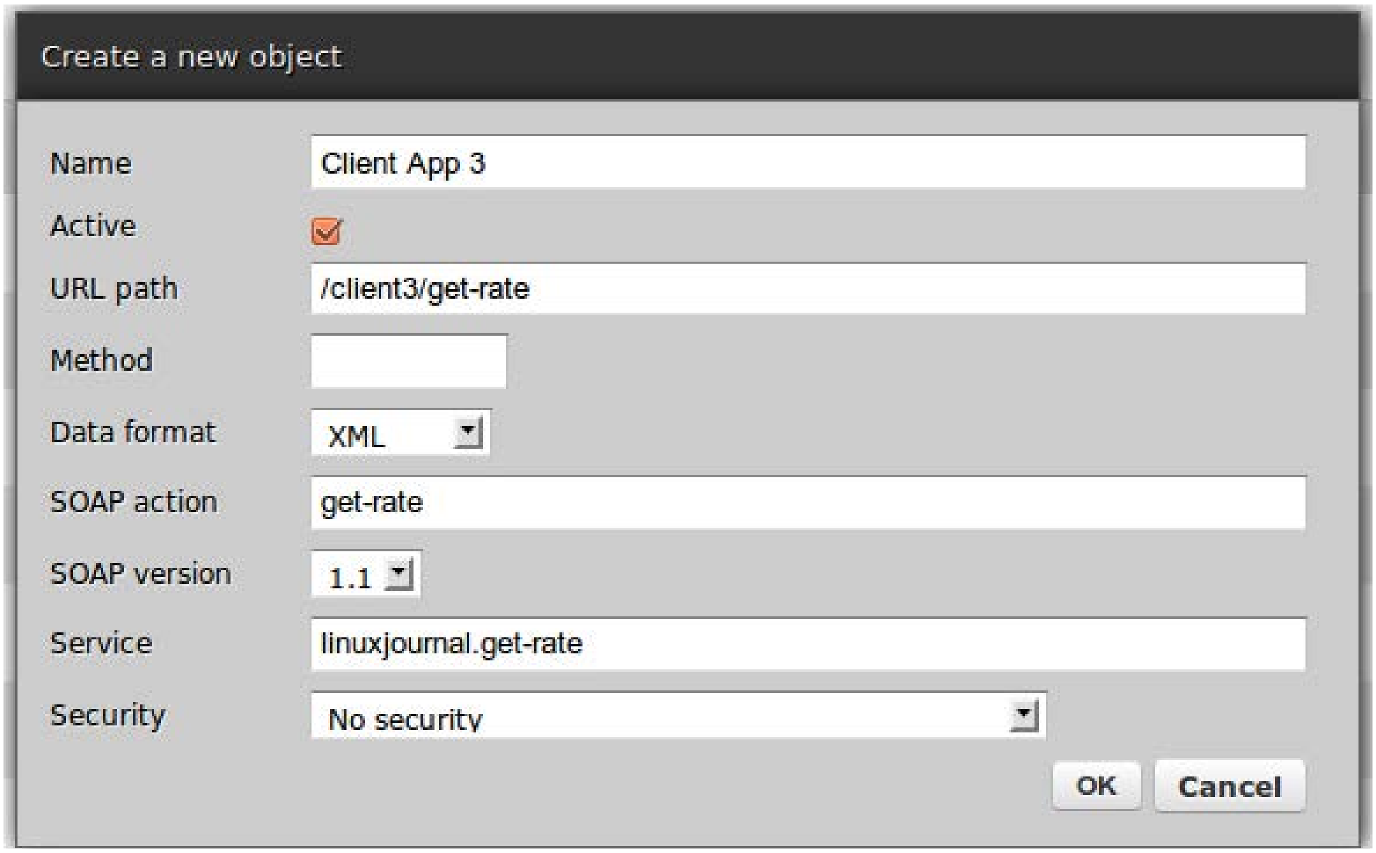
Figure 4. JSON Channel Creation Form



Create a new object

Name	Client App 2
Active	✓
URL path	/client2/get-rate
Method	
Data format	XML
Service	linuxjournal.get-rate
Security	No security

Figure 5. Plain XML Channel Creation Form



Create a new object

Name	Client App 3
Active	✓
URL path	/client3/get-rate
Method	
Data format	XML
SOAP action	get-rate
SOAP version	1.1
Service	linuxjournal.get-rate
Security	No security

Figure 6. SOAP Channel Creation Form

configuration on a development environment and exporting it to test environments where the new configuration can be merged into an existing one, and later on, all that can be exported to production.

Figures 2–6 show the following configs:

- Scheduler's job to invoke the service updating the cache.
- Outgoing HTTP connection definitions for connecting to treasury.gov.
- HTTP channels for each client—there is no requirement that each client be given a separate channel but doing so allows one to assign different security definitions to each channel without interfering with any other.

Testing It

update-cache will be invoked by the scheduler, but Zato's CLI offers the means to invoke any service from the command line, even if it's not mounted on any channel, like this:

```
$ zato service invoke /opt/zato/server1 linuxjournal.update-cache
 ➥--payload '{}'
(None)
$
```

There was no output, because the service doesn't produce any. However, when you check the logs you notice:

```
INFO - Key linuxjournal:rates:2013:09:03 set to 1.22
```

Now you can invoke get-rate from the command line using curl with JSON, XML and SOAP. The very same service exposed through three independent channels will produce output in three formats, as shown below (output slightly reformatted for clarity).

Output 1:

```
$ curl localhost:17010/client1/get-rate -d
 ➥'{"year":"2013","month":"09","day":"03"}'
 ➥{"response": {"rate": "1.22"}}
$
```

Output 2:

```
$ curl localhost:17010/client2/get-rate -d '
  <request><year>2013</year><month>09</month><day>03</day></request>'
<response>
 <zato_env>
  <cid>K295602460207582970321705053471448424629</cid>
  <result>ZATO_OK</result>
 </zato_env>
 <item>
  <rate>1.22</rate>
 </item>
</response>
$
```

Output 3:

```
$ curl localhost:17010/client3/get-rate \
    -H "SOAPAction:get-rates" -d '
  <soapenv:Envelope xmlns:soapenv=
➥"http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:z="https://zato.io/ns/20130518">
    <soapenv:Body>
      <z:request>
        <z:year>2013</z:year>
        <z:month>09</z:month>
        <z:day>03</z:day>
      </z:request>
    </soapenv:Body>
  </soapenv:Envelope>'
<?xml version='1.0' encoding='UTF-8'?>
 <soap:Envelope
  ➥xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   ➥xmlns="https://zato.io/ns/20130518">
  <soap:Body>
   <response>
    <zato_env>
     <cid>K175546649891418529601921746996004574051</cid>
    <result>ZATO_OK</result>
   </zato_env>
   <item>
    <rate>1.22</rate>
   </item>
  </response>
 </soap:Body>
</soap:Envelope>
$
```

IRA Is the Key

IRA (Interesting, Reusable, Atomic) is the key you should always keep in mind when designing services that are to be successful.

Both the services presented in the article meet the following criteria:

- *I*: focus on providing data interesting to multiple parties.
- *R*: can take part in many processes and be accessed through more than one method.
- *A*: focus on one job only and do it well.

In this vein, Zato makes it easy for you to expose services over many channels and to incorporate them into higher-level integration scenarios, thereby increasing their overall attractiveness (I in IRA) to potential client applications.

It may be helpful to think of a few ways not to design services:

- **Anti-I**: update-cache could be turned into two smaller services. One would fetch data and store it in an SQL database; the other would grab it from SQL and put it into Redis. Even if such a design could be defended by some rationale, neither of the pair of services would be interesting for external applications. A third service wrapping these two should be created and exposed to client apps, in the case of it being necessary for other systems to update the cache. In other words, let's keep the implementation details inside without exposing them to the whole world.

- **Anti-R**: hard-coding nontrivial parameters is almost always a poor idea. The result being that a service cannot be driven by external systems invoking it with a set of arguments. For instance, creating a service that is limited to a specific year only ensures its limited use outside the original project.

- **Anti-A**: returning a list of previous queries in response to a request may be a need of one particular client application, but contrary to the needs of another. In cases when a composite service becomes necessary, it should not be obliged upon each and every client.

Designing IRA services is like designing a good programming interface that will be released as an open-source library and used in places that can't be predicted initially.

Born Out of Practical Experience

Zato it not only about IRA but also about codifying common admin and programming tasks that are of a practical nature:

- Each config file is versioned automatically (**https://zato.io/docs/admin/guide/install-config/overview.html#config-files-are-versioned**) and kept in a local bzr repository (**http://bazaar-vcs.org**), so it's always possible to revert to a safe state. This is completely transparent and needs no configuration nor management.

- A frequent requirement before integration projects are started, particularly if certain services already are available on the platform, is to provide usage examples in the form of message requests and responses. Zato lets you specify that one-in-*n* invocations of a service be stored for a later use, precisely so that such requirements can be fulfilled by admins quickly.

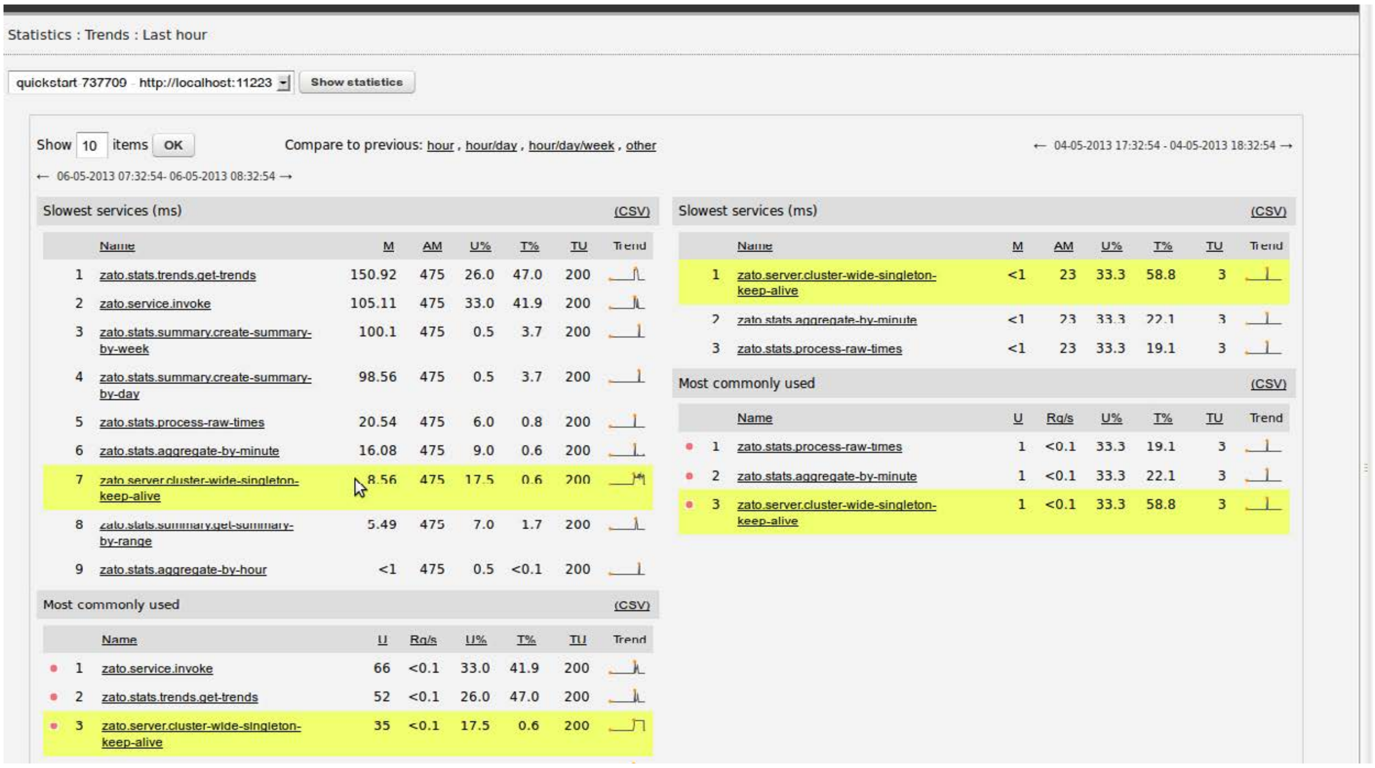
Two popular questions asked regarding production are: 1) What are my slowest services? and 2) Which services are most commonly used? To answer these, Zato provides statistics (**https://zato.io/docs/stats/guide.html**) that can be accessed via Web admin, CLI or API. Data can be compared over arbitrary periods or exported to CSV as well.

Summary

Despite being a relatively new project, Zato is already a lightweight yet complete solution that can be used in many integration and back-end scenarios. Regardless of the project's underlying integration principles, such as SOA or REST, the platform can be used to deliver scalable architectures that are easy to use, maintain and extend. ■

Dariusz Suchojad is a systems architect specializing in SOA/ESB/EAI/BPM/SSO with 12 years of experience completing projects for enterprise customers in telecommunications and banking. Dissatisfied with existing solutions used for systems integrations, a couple years ago, he quit his job and now focuses solely on developing Zato—a Python-based platform for integrating applications and back-end systems.

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



Statistics : Trends : Last hour

quickstart 737709 http://localhost:11223 | Show statistics

Show 10 items OK   Compare to previous: hour, hour/day, hour/day/week, other

← 06-05-2013 07:32:54- 06-05-2013 08:32:54 →

Slowest services (ms) (CSV)

	Name	M	AM	U%	T%	TU	Trend
1	zato.stats.trends.get-trends	150.92	475	26.0	47.0	200	
2	zato.service.invoke	105.11	475	33.0	41.9	200	
3	zato.stats.summary.create-summary-by-week	100.1	475	0.5	3.7	200	
4	zato.stats.summary.create-summary-by-day	98.56	475	0.5	3.7	200	
5	zato.stats.process-raw-times	20.54	475	6.0	0.8	200	
6	zato.stats.aggregate-by-minute	16.08	475	9.0	0.6	200	
7	zato.server.cluster-wide-singleton-keep-alive	8.56	475	17.5	0.6	200	
8	zato.stats.summary.get-summary-by-range	5.49	475	7.0	1.7	200	
9	zato.stats.aggregate-by-hour	<1	475	0.5	<0.1	200	

Most commonly used (CSV)

		Name	U	Rq/s	U%	T%	TU	Trend
●	1	zato.service.invoke	66	<0.1	33.0	41.9	200	
●	2	zato.stats.trends.get-trends	52	<0.1	26.0	47.0	200	
●	3	zato.server.cluster-wide-singleton-keep-alive	35	<0.1	17.5	0.6	200	

← 04-05-2013 17:32:54 - 04-05-2013 18:32:54 →

Slowest services (ms) (CSV)

	Name	M	AM	U%	T%	TU	Trend
1	zato.server.cluster-wide-singleton-keep-alive	<1	23	33.3	58.8	3	
2	zato.stats.aggregate-by-minute	<1	23	33.3	22.1	3	
3	zato.stats.process-raw-times	<1	23	33.3	19.1	3	

Most commonly used (CSV)

		Name	U	Rq/s	U%	T%	TU	Trend
●	1	zato.stats.process-raw-times	1	<0.1	33.3	19.1	3	
●	2	zato.stats.aggregate-by-minute	1	<0.1	33.3	22.1	3	
●	3	zato.server.cluster-wide-singleton-keep-alive	1	<0.1	33.3	58.8	3	

Figure 7. Sample Statistics

drupalize.me

Instant Access to Premium Online Drupal Training

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

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

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

```
 *
 * 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
```

SIDUS the Solution for Extreme Deduplication of an Operating System

Probe corrupted computers without disassembling anything, and provide users with a full-featured environment in just a few seconds.

Emmanuel Quemener and **Marianne Corvellec**

SIDUS (Single-Instance Distributing Universal System) was developed at Centre Blaise Pascal (Ecole normale supérieure de Lyon, Lyon, France), where one administrator alone is in charge of 180 stations. Emmanuel Quemener started SIDUS in February 2010, and he significantly cut his workload for administering this park of stations. SIDUS is now in use at the supercomputing centre PSMN (Pôle Scientifique de Modélisation Numérique) of the Ecole normale supérieure de Lyon.

SIDUS offers users a single given environment that is easily configurable at any moment.

With SIDUS, you can provide a new user with a complete functional environment in just a few seconds. You can probe corrupted computers without disassembling anything. You can test new equipment without installing an OS on them. You can make your life so much easier when managing hundreds of cluster nodes, of workstations or of self-service stations. You drastically can reduce the amount of storage needed for the OS on these machines.

Disclaimer

SIDUS is not LTSP. LTSP is a solution for the simplified management of thin terminals through X11 or RDP access to a server. Thus, all the processing load is on the latter server. On the contrary, SIDUS makes full use (or partial use, as the user wishes) of the station's resources. Only the OS is stored remotely.

SIDUS is not FAI. FAI or Kickstart offer full simplified installs so that administration can be reduced or dismissed altogether. On the contrary, SIDUS offers a single system in a tree that integrates the base system as well as all manually installed applications.

SIDUS is flexible. When organizing IT-training sessions, you might want to give participants a specific virtual environment. But once they download it, you cannot modify it for them. SIDUS offers users a single given environment that is easily configurable at any moment.

SIDUS is not exotic. SIDUS makes use of services available with any distribution (DHCP, PXE, TFTP, NFSroot, DebootStrap and AUFS). You can install SIDUS knowing only these few keywords. Besides, SIDUS makes use of distribution tricks from live CDs. SIDUS works on Debian, all the way from version Etch.

How Good Is SIDUS?

SIDUS is:

- Universal: platform-independent, x86 or x86_64 architectures.
- Efficient: installing takes a few minutes, and booting takes a few seconds.
- Energy-saving: it takes only one core, 1GB of RAM, 40GB in disk space and an Ethernet (Gbit) network.
- Scalable: tested successfully on a hundred nodes.
- Multipurpose: we chose to use Debian as it comes with broad integration of open-source scientific software.

Installing SIDUS on Your System

It takes a little preparation for your system to host SIDUS. We have several services at our disposal in order to deploy our clients: DHCP, TFTP and NFS servers. Now, either you are on great terms with your own IT staff, or you are able to access freely the well-defined LDAP and DNS servers:

- DHCP service provides the client with one IP address but propagates two complementary pieces of information: IP address of the TFTP server (variable "next-server") and the name of the PXE binary, often called pxelinux.0.
- TFTP service then comes into play. Booting the system is enabled by TFTP, through the binary pxelinux.0, the kernel and startup of the client's system. If you need to give a client some parameters, you just build a dedicated file whose name stems from the client's MAC address (prefixing with 01 and replacing : with -).
- NFS service now enters the loop: it gives the system's root via its protocol (NFSroot). Accordingly, you will install your client system in this root—for example, /src/nfsroot/sidus.

In our configuration, we have used isc-dhcp-server, tftpd-hpa and nfs-kernel-server for the servers DHCP, TFTP and NFS, respectively. Let's look into this configuration.

For DHCP, the configuration file (/etc/dhcp/dhcpd.conf) reads:

```
next-server 172.16.20.251;
```

```
filename "pxelinux.0";
allow booting;
```

For TFTP, there are three files and one directory (pxelinux.cfg) in /srv/tftp:

```
./pxelinux.0
./vmlinuz-Sidus
./initrd.img-Sidus
./pxelinux.cfg
```

The pxelinux.0 file comes from the syslinux-common package. In pxelinux.cfg, there is the file called default.

To boot, you need the following: the kernel vmlinuz-Sidus, the system initrd.img-Sidus and the server NFSroot 10.13.20.13 with the mountpoint /srv/nfsroot/sidus.

Below is an example of a boot file. It takes two inputs: tmpfs and iscsi (we'll come back to the iscsi input later on):

```
DEFAULT tmpfs

LABEL tmpfs
KERNEL vmlinuz-Sidus
APPEND console=tty1 root=/dev/nfs
    initrd=initrd.img-Sidus
    nfsroot=10.13.20.13:/srv/nfsroot/sidus,
    rsize=8192,wsize=8192,tcp ip=dhcp aufs=tmpfs
LABEL iscsi
KERNEL vmlinuz-Sidus
APPEND console=tty1 root=/dev/nfs
    initrd=initrd.img-Sidus
    nfsroot=10.13.20.13:/srv/nfsroot/wheezy64,
    rsize=8192,wsize=8192,tcp ip=dhcp aufs=iscsi
    ISCSI_TARGET_IP=10.13.20.14
    ISCSI_INITIATOR=iqn.2013-04.zone.sidus.target:
    default root=LABEL=ISCSI
```

Regarding the NFS server, it takes one line in the file /etc/exports to configure it:

```
/srv/nfsroot/sidus
10.13.20.0/255.255.255.0(ro,no_subtree_check,async,no_root_squash)
```

Here, we open a read-only access to stations with IP between 10.13.20.1 and 10.13.20.254.

Once you have configured these three services (DHCP, TFTP and NFS), you can install a full SIDUS. Note that you also will need a root for user accounts (via NFSv4) and a process enabling their identification/authentication (via LDAP or Kerberos). We have deployed SIDUS on environments where these services are provided by third-party servers but also on standalone environments. Installing an OpenLDAP server with SSL or a Kerberos server is off-topic, so we simply show the client configuration files for our infrastructure (again, LDAP for identification/authentication and NFSv4 for user folders).

Install the Debian Base with Debootstrap

With Debootstrap, you can install a system in an extra root location. Debootstrap needs you to specify parameters, such as the install root, the hardware architecture, the distribution and the FTP or HTTP Debian archive to use for downloading on Debian worldwide mirror sites.

Warning: this is where we get Debian-specific. Debootstrap is a familiar tool for all Debian-like distributions (typically, it is available on Ubuntu). It is not too difficult to make it happen on Red Hat-like distributions though. There is a clone for Fedora called febootstrap; we have not tested it though.

Debootstrap (**http://wiki.debian.org/Debootstrap**) also takes as input a list of archives—as we all know, Debian is very particular about distinguishing the main archive area from the contrib and non-free ones—a list of packages to include and a list of packages to dismiss. We wish we could specify the latter two lists, but you cannot handle everything with Debootstrap. We install from the very beginning a set of tools we deem necessary (such as the kernel, some firmware and auditing tools).

We define environment variables corresponding to the root of our SIDUS system. We define a command that enables the execution of commands via chroot, with a specific option for package install. The variable `$MyInclude` corresponds to the (comma-separated) list of packages you want, and `$MyExclude` corresponds to the list of packages you do not want:

```
export SIDUS=/srv/nfsroot/sidus
time debootstrap --arch amd64
    --components='main,contrib,non-free'
    --include=$MyInclude --exclude=$MyExclude
    wheezy $SIDUS http://ftp.debian.org/debian
```

Precautions before Moving on with the Install

After running the last-mentioned command, you should be a little cautious. A Debian package normally starts after install. You need to define a hook to inhibit the booting of services. After completion of the install, you can remove this hook:

```
printf '#!/bin/sh\nexit 101\n' >
     ${SIDUS}/usr/sbin/policy-rc.d
```

```
chmod +x ${SIDUS}/usr/sbin/policy-rc.d
```

Some packages require access to the list of processes, system, peripherals, peripheral pointers and virtual memory. Hence, you should bind the mounting of these host system folders to SIDUS:

```
alias sidus="DEBIAN_FRONTEND=noninteractive chroot
    ${SIDUS} $@"
sidus mount -t proc none /proc
sidus mount -t sysfs sys /sys
mount --bind /run/shm ${SIDUS}/run/shm
mount --bind /dev/pts ${SIDUS}/dev/pts
```

Install Additional Packages (Scientific Libraries)

To make it simpler when installing packages of the same family, Debian ships with meta-packages. In our case, we are interested in the scientific ones: their names are prefixed by "science". For example, we have "science-chemistry", including all chemistry packages. You install all scientific packages with only one command:

```
time sidus apt-get install --install-suggests -f
    -m -y --force-yes science-*
```

Because we are talking about a *full-featured* OS, we also install the suggested packages: the option `--install-suggests` is available from Wheezy onward (released May 5, 2013).

When installing, the costliest phase is downloading packages and configuring certain components (Perl and LaTeX). In the best-case scenario, it takes 45 minutes for a 32GB full tree. There is a price to pay for this install craze. Some packages do not install well, and you will want to purge some, such as a M*tlab installer:

```
time sidus apt-get purge -y -f --force-yes matlab-*
```

Local Environment

Usually, you will want to adapt the system to a local environment (authentication and user sharing). The default is US, so you may want to configure:

- ${SIDUS}/etc/locale.gen.
- ${SIDUS}/etc/timezone.
- ${SIDUS}/etc/default/keyboard.

For LDAP authentication, you may want to configure: ${SIDUS}/etc/nsswitch.conf, ${SIDUS}/etc/libpam_ldap.conf, ${SIDUS}/etc/libnss-ldap.conf and ${SIDUS}/etc/ldap/ldap.conf.

How do you share SIDUS without duplicating it? We found the best solution to be via live CD.

As for the mounting of NFS user folders:

- ${SIDUS}/etc/default/nfs-common, ${SIDUS}/etc/default/idmapd.conf and ${SIDUS}/etc/fstab (for NFSv4).

- ${SIDUS}/etc/fstab (for NFSv3).

Set Up the Boot Sequence

How do you share SIDUS without duplicating it? We found the best solution to be via live CD. The boot sequence includes the two layers you need—that is, a read-only layer (on media for live CD and on NFS in our case) and a read-write layer (on TMPFS). The two layers are linked via AUFS (the successor of UnionFS). Everything is taken care of by a single hook upon boot (the script called rootaufs). It operates in five steps:

1. Creates the temporary files /ro, /rw and /aufs.

2. Moves the root of NFSroot from the original mountpoint to /ro.

3. Mounts the local or remote partition.

4. Superimposes /ro and /rw into /aufs.

5. Moves /aufs into the original mountpoint. rootaufs goes into ${SIDUS}/etc/initramfs-tools/scripts/init-bottom.

The original script is inspired by the rootaufs project by Nicholas A. Schembri (**http://code.google.com/p/rootaufs**). We adapted it to a large extent to match our infrastructure. A version is available at **http://www.cbp.ens-lyon.fr/sidus/rootaufs**:

```
wget -O
    ${SIDUS}/etc/initramfs-tools/scripts/init-bottom
    http://www.cbp.ens-lyon.fr/sidus/rootaufs
```

The system is not functional yet. You need to create an initrd specific to your NFS boot. Add aufs in ${SIDUS}/etc/initramfs-tools/modules and force eth0 as DEVICE in ${SIDUS}/etc/initramfs-tools/initramfs.conf:

```
sidus update-initramfs -k all -u
```

Then, you just copy the kernel and bootloader in the definition:

```
cp ${SIDUS}/vmlinuz /srv/tftp/vmlinux-Sidus
cp ${SIDUS}/srv/nfsroot/boot/initrd
    /srv/tftp/initrd-Sidus
```

How can you take advantage of SIDUS while keeping a given configuration from one boot to the next? Mounting NFS on each node separately is very costly. It is preferable to mount iSCSI on each node.

Originally, we investigated how to offer a second NFS share in read-write mode to ensure persistence of client-related changes from one boot to the next. This version, although functional, required an atomized NFS—one for each client. This was not sustainable for the server.

Therefore, we decided on another solution to ensure persistence. We create an iSCSI share for each client. The settings for mounting the iSCSI disk are defined in the line command.

So we use a network drive from iSCSI technology. In the config file /srv/tftp/pxelinux.cfg/default, we have the definition `LABEL=iscsi`. Each SIDUS client needs its own iSCSI storage space to ensure persistence. For reasons of simplicity, in the initrd booting sequence, the SIDUS clients fetch the volumes that bear their respective IPs. The rootaufs file contains a default login/password.

A few tricks:

- Erase /etc/hostname to set the hostname through DHCP.
- Set /etc/resolv.conf with a hard-coded definition.
- Define a loopback in /etc/network/interfaces.
- Change the booting of GDM3 so it starts only after NSCD is launched.
- Set /etc/security/limits.conf (essential in an HPC environment).
- Set /etc/fstab with input from the NFS server of user accounts.
- For VirtualBox-based virtual systems, install VBoxLinuxAdditions.run in the SIDUS system.
- For systems with an InfiniBand card, force loading of modules in /etc/modules and regenerate initrd. In /etc/rc.local, execute a script that gets the Ethernet IP address and builds an IP address for the InfiniBand card.

- For systems with an NVIDIA card: with most NVIDIA cards, packages offered with Debian Wheezy let you install the necessary proprietary drivers and the OpenGL, Cuda and OpenCL libraries. Be careful if you want to use the OpenCL ICD (Installable Client Loader) for AMD to operate your processors and your graphics board simultaneously. To be able to do so, we had to install the entire environment—drivers, Cuda and OpenCL—from scratch.

- For systems with an AMD ATI card: with the most ATI cards, packages offered with Debian Wheezy let you install the necessary proprietary drivers and the OpenGL, Cuda and OpenCL libraries.

At the present time at CBP, we use the technique "NFSroot + iSCSI = AUFS" on SIDUS stations that require persistence, such as DiStoNet nodes (**http://forge.cbp.ens-lyon.fr/projects/distonet**). Otherwise, we use "NFSroot + TMPFS = AUFS".

Install Wrap-up

We unmount all system folders necessary for installation:

```
umount ${SIDUS}/run/shm
umount ${SIDUS}/dev/pts
sidus umount /proc/sys/fs/binfmt_misc
sidus umount /proc
sidus umount /sys
```

We activate the startup dæmons:

```
rm -f ${SIDUS}/usr/bin/policy-rc.d
cp /usr/sbin/start-stop-daemon
   ${SIDUS}/usr/sbin/start-stop-daemon
```

We remove all process references launched by the install process:

```
rm -r ${SIDUS}/run/* ${SIDUS}/tmp/*
```

This purges all processes related to SIDUS.

Adapting to Heterogeneity

The park of computers may consist of cluster nodes (with fast network equipment), workstations (with embedded GPUs) or virtual machines (which require data sharing and GPU acceleration). For a large park, you do not want persistence. You should use boot scripts, a separate SIDUS tree or install third-party components.

Administering the system is not as easy as installing it. The gain you experience in installing the system more than makes up for the pain you experience in administering the system though. With SIDUS, every administration phase abides by the installation mechanisms: protection against booting and mounting of system folders.

Administration techniques are similar to those for the initial install. We use a script so that commands executed in SIDUS are surrounded by pre/post operations. We use this script either automatically (typically for updates) or manually. At the end of the day, these additional commands represent a negligible burden with respect to the benefits we get. We now have several (many) stations that are bit-for-bit identical to a given base system. Other benefits:

- SIDUS works on user stations. The individual workstations can be considered shared. We started with a dozen Neoware light clients that were memory-enhanced and overclocked. We now have about 20 of those.

- SIDUS works on cluster nodes. In March 2010, we had a proof of concept with 24 nodes. Nowadays, SIDUS serves 86 permanent nodes over four different hardware architectures.

- SIDUS works on virtual stations. Every year since 2011, Université Joseph Fourier organizes a summer school on scientific computing. For ten busy days, students train hands-on. Thus, it's necessary to offer them a homogeneous environment in no time. Two virtual images are offered: a persistent one they can use after the summer school and another one via SIDUS. This way, teachers can adapt materials and activities day by day. Since summer 2012, this solution has been used at the Laboratory of Chemistry of ENS de Lyon as well.

- SIDUS works on suspicious stations. Booting via the network enables the investigation of the shutdown system mass storage. There is no need for a live CD—always short of your ideal forensic tool.

- SIDUS works on loan stations. Hardware manufacturers usually offer assessment equipment. The install phase can be tedious on recent equipment. Using SIDUS, the system boots just like on other (in use) equipment—for example, it takes a few minutes for 20 nodes.

Conclusion

Who should use SIDUS and why?

- Users: you choose the resources on which you want to boot your station. Therefore, workstations can be segmented at will. The VirtualBox version of SIDUS has been tested successfully on Linux, MS Windows and Mac OS. GPU acceleration and sharing with the host are available. Users find themselves in the same environment as the nodes'. This makes code integration tremendously easier. Performance-wise, losses due to virtualization vary between 10% and 20% for VirtualBox and about 5% for KVM.

- Administrators: a given operation propagates to the entire infrastructure, as if simply syncing over the SIDUS tree. The install takes a few tens of minutes for a full-featured system. To work out minor differences between systems, simple scripts or puppets do the job. In the case of more important differences, just build another SIDUS tree. The SIDUS tree might even just be cloned instantaneously using snapshot tools (LVM or, better, ZFSonLinux).

- For the sake of experiments: the SIDUS environment offers scientists and system engineers a framework for conducting reproducible experiments. Two nodes booting on the same SIDUS base do run the exact same system. This way, even if the stations are not actually identical, relevant tests still can be carried out.

How much does it cost in terms of resources? To get an idea, the clusters' server (also gateway) at CBP hosts DHCP, DNS, TFTP and NFS services, as well as a batch server OAR. When booting the entire infrastructure (88 nodes), the NFS server takes in 900Mb/s.

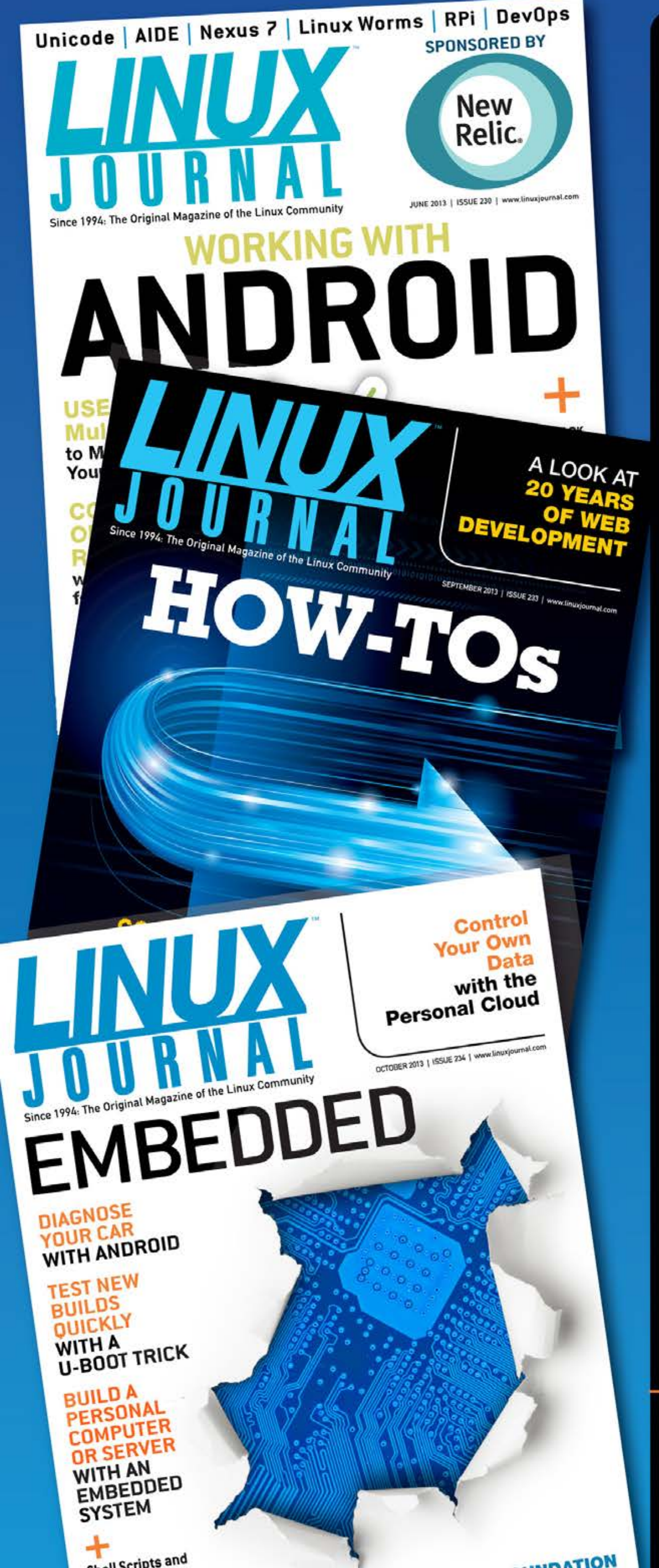
To conclude, you will want to use SIDUS on a variety of environments, be they HPC nodes, workstations or virtual machines. SIDUS gives unprecedented flexibility to both users and administrators. It is so energy-efficient and it propagates so rapidly, you won't want to live without it!■

Emmanuel Quemener defines his job as an "IT test pilot". His work at the HPC "Centre Blaise Pascal" (Lyon, France) involves software integration, storage, scientific computing with GPUs and technology transfer in science.

Marianne Corvellec is a physicist who gratefully was exposed to the wonderful world of free/libre and open-source software. She wants to make computation science a better place, promoting best practices and interacting with software experts.

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

If You Use Linux, You Should Be Reading LINUX JOURNAL™



Unicode | AIDE | Nexus 7 | Linux Worms | RPi | DevOps
SPONSORED BY New Relic.
LINUX JOURNAL
Since 1994: The Original Magazine of the Linux Community
JUNE 2013 | ISSUE 230 | www.linuxjournal.com
WORKING WITH ANDROID
LINUX JOURNAL
A LOOK AT 20 YEARS OF WEB DEVELOPMENT
SEPTEMBER 2013 | ISSUE 233 | www.linuxjournal.com
HOW-TOs
LINUX JOURNAL
Control Your Own Data with the Personal Cloud
OCTOBER 2013 | ISSUE 234 | www.linuxjournal.com
EMBEDDED
DIAGNOSE YOUR CAR WITH ANDROID
TEST NEW BUILDS QUICKLY WITH A U-BOOT TRICK
BUILD A PERSONAL COMPUTER OR SERVER WITH AN EMBEDDED SYSTEM

- » In-depth information providing a full 360-degree look at featured topics relating to Linux
- » Tools, tips and tricks you will use today as well as relevant information for the future
- » Advice and inspiration for getting the most out of your Linux system
- » Instructional how-tos will save you time and money

Subscribe now for instant access! For only $29.50 per year—less than $2.50 per issue—you'll have access to *Linux Journal* each month as a PDF, in ePub format, in Mobi format, on-line and through our Android and iOS apps. Wherever you go, *Linux Journal* goes with you.

SUBSCRIBE NOW AT:
WWW.LINUXJOURNAL.COM/SUBSCRIBE

Introduction to OpenStack

You've probably heard of OpenStack. It's that cloud software that's getting a lot of attention from a lot of big names in the IT industry and major users like CERN, Comcast and PayPal. However, did you know that it's more than that? It's also the fastest-growing Open Source community in the world, and a very interesting collaboration among technology vendors.

TOM FITFIELD

OpenStack is really truly open. If you want to test it out, you can stop reading this article (come back after—we'll still be here), visit **http://status.openstack.org/release**, and get a real-time list of features that are under development. Select one, follow it to the code review system, and give your comments. This is an example of Open Development, just one of the "Four Opens" on which OpenStack was founded. Of course, you likely already know that OpenStack is fully released under the Open Source Apache 2 license, no bits reserved, but did you also know about the principle of "Open Design"?

The software is released on a six-month cycle, and at the start of each cycle, we host a Design Summit for the contributors and users to gather and plan out the roadmap for the next release. The Design Summit has become part of an increasingly large conference that also hosts workshops for newcomers, inspiring keynotes and some fairly amazing user stories. However, somewhere tucked away are rooms with chairs arranged in semicircular layouts ensconcing dozens of developers engaged in robust discussion, taking notes on a collaborative document displayed on a projector. This is where the roadmap of

OpenStack is determined, pathways for implementations of features are aligned, and people volunteer to make it reality. This is the Open Design process, and we welcome your participation.

OpenStack may have started with just two organizations, NASA and Rackspace, but now there are hundreds, and with each additional member, the community grows stronger and more cohesive. Every morning the OpenStack developer awakes to a torrent of e-mail from the discussions in other countries, and the momentum is best described as "intense".

OVERALL, OPENSTACK'S STANDOUT FEATURE IS THIS STRONG COMMUNITY.

Overall, OpenStack's standout feature is this strong community. It's extremely diverse, composed of very different technical backgrounds (Python developers to packagers to translators) and different philosophical backgrounds (free software evangelists to hard-core capitalists). It's also very widespread, with the OpenStack Foundation claiming membership from 130 countries as of October 2013. So, dear reader, chances are there is a place for you.

Becoming a Contributor

During a 12-month period, OpenStack typically has more than 1,000 software developers contributing patches. Despite this, we always need more.

You can find detailed instructions on how to get started on the wiki (**http://wiki.openstack.org/HowToContribute**), but let me run through a couple key aspects:

- OpenStack uses Launchpad for bugs and GitHub for code hosting, but neither of those places for contributing code. That's right, no GitHub pull requests are accepted. Don't panic yet—this is for a good reason: all patches to OpenStack go through an extensive code review and testing process (**https://wiki.openstack.org/wiki/Gerrit_Workflow**).

- Every code change in OpenStack is seen by at least three people (the owner and two of the core reviewers for the project), but often many more, and test cases and PEP8 compliance is required.

The patches also are run through the continuous integration system, which effectively builds a new cloud for every code change submitted, ensuring that the interaction of that piece of software is as expected with all other parts of OpenStack. As a result of this quest for quality, many have found that contributing has improved their Python coding skills.

Of course, not everyone is a Python developer. However, don't worry; there's still a place for you to work with us to change the face of cloud computing:

- Documentation: in many cases, the easiest way to become an OpenStack contributor is to participate in the documentation efforts. It requires no coding, just a willingness to read and understand the systems that you're writing about. Because the documentation is treated like code, you will be learning the mechanics necessary to make contributions to OpenStack itself by helping with documentation. Visit **https://wiki.openstack.org/wiki/Documentation/HowTo** for more information.

- Translation: if you know how to write in another language, translations of OpenStack happen through an easy-to-use Web interface (**https://www.transifex.com/projects/p/openstack**). For help, contact the Internationalization team (**https://wiki.openstack.org/wiki/I18nTeam**).

- File bugs: OpenStack is one of the few projects you will work on that loves to hear your angry rants when something does not work. If you notice something off, consider filing a bug report with details of your environment and what you experienced at **http://docs.openstack.org/trunk/openstack-ops/content/upstream_openstack.html**.

- Asking questions: a StackOverflow-style board for questions about OpenStack is available at **http://ask.openstack.org**. Feel free to use it to ask yours, and if you have the ability, stick around and try to answer someone else's (or at least vote on the ones that look good).

- Evangelism: do you think OpenStack is pretty cool? Help us out by telling your friends; we'd really appreciate it. You can find some materials to help at **http://openstack.org/marketing**, or join the marketing mailing list to find out about some cool events to attend.

- User groups: join your local user group (**https://wiki.openstack.org/wiki/OpenStackUserGroups**). They're in about 50 countries so far. Attend to learn, or volunteer to speak. We'd love to have you.

Navigating the Ecosystem: Where Does Your OpenStack Journey Begin?

One of the unique aspects about OpenStack as an open-source project is that there are many different levels at which you can begin to engage with it. You don't have to do everything yourself.

Starting with *Public Clouds*, you don't even need to have an OpenStack installation to start using it. Today, you can swipe your credit card at eNovance, HP, Rackspace and others, and just start migrating your applications.

Of course, for many people, the enticing part of OpenStack is building their own private clouds, and there are several ways to do that. Perhaps the simplest of all is an *appliance-style* solution. You purchase a "thing", unbox it, plug in the power and the network, and it just *is* an OpenStack cloud.

However, hardware choice is important for many applications, so if that applies to you, consider that there are several software distributions available. You can, of course, get enterprise-supported OpenStack from Canonical, Red Hat and SUSE, but take a look also at some of the specialized distributions, such as those from Rackspace, Piston, SwiftStack or Cloudscaling.

OF COURSE, FOR MANY PEOPLE, THE ENTICING PART OF OPENSTACK IS BUILDING THEIR OWN PRIVATE CLOUDS, AND THERE ARE SEVERAL WAYS TO DO THAT.

If you want someone to help guide you through the decisions from the hardware up to your applications, perhaps adding in a few features or integrating components along the way, consider contacting one of the system integrators with OpenStack experience like Mirantis or Metacloud.

Alternately, if your preference

is to build your own OpenStack expertise internally, a good way to kick start that might be to attend or arrange a training session. The OpenStack Foundation recently launched a Training Marketplace (**http://www.openstack.org/marketplace/training**), where you can look for events nearby. There's also a community training effort (**https://wiki.openstack.org/wiki/Training-manuals**) underway to produce open-source content for training.

To derive the most from the flexibility of the OpenStack framework, you may elect to perform a DIY solution. In which case, we strongly recommend getting a copy of the OpenStack Operations Guide (**http://docs.openstack.org/ops**), which discusses many of the decisions you will face along the way. There's also a new OpenStack Security Guide (**http://docs.openstack.org/sec**), which is an invaluable reference for hardening your installation.

DIY-ing Your OpenStack Cloud

If, after careful analysis, you've decided to construct OpenStack yourself from the ground up, there are a number of areas to consider.

Storage One of the most fundamental underpinnings of a cloud platform is the storage on which it runs. In general, when you select storage back ends, ask the following questions:

- Do my users need block storage?
- Do my users need object storage?
- Do I need to support live migration?
- Should my persistent storage drives be contained in my compute nodes, or should I use external storage?
- What is the platter count I can achieve? Do more spindles result in better I/O despite network access?
- Which one results in the best cost-performance scenario I'm aiming for?
- How do I manage the storage operationally?
- How redundant and distributed is the storage? What happens if a storage node fails? To what extent can it mitigate my data-loss disaster scenarios?
- Which plugin do I use for block storage?

For many new clouds, the object storage and persistent/block storage are great features that users want. However, with OpenStack, you're not forced to use either if you want a simpler deployment.

Many parts of OpenStack are pluggable, and one of the best examples of this is Block Storage, which you are able to configure to use storage from a long list of vendors (Coraid, EMC, GlusterFS, Hitachi, HP, IBM, LVM, NetApp, Nexenta, NFS, RBD, Scality, SolidFire, Windows Server and Zadara).

Network If this is the first time you are deploying a cloud infrastructure in your organization, after reading this section, your first conversations should be with your networking team. Network usage in a running cloud is vastly different from traditional network deployments, and it has the potential to be disruptive at both a connectivity and a policy level.

For example, you must plan the number of IP addresses that you need for both your guest instances as well as management infrastructure. Additionally, you must research and discuss cloud network connectivity through proxy servers and firewalls.

One of the first choices you need to make is between the "legacy" nova-network and OpenStack Networking (aka Neutron). Nova-network is a much simpler way to deploy a network, but it does not have the full software-defined networking features of Neutron, and it will be deprecated after 12–18 months.

Object Storage's network patterns might seem unfamiliar at first. Consider these main traffic flows:

- Among object, container and account servers.
- Between those servers and the proxies.
- Between the proxies and your users.

Object Storage is very "chatty" among servers hosting data—even a small cluster does megabytes/second of traffic, which is predominantly "Do you have the object?"/"Yes I have the object." Of course, if the answer to the aforementioned question is negative or times out, replication of the object begins.

Consider the scenario where an entire server fails, and 24TB of data needs to be transferred "immediately" to remain at three copies—this can put significant load on the network.

Another oft-forgotten fact is that when a new file is being uploaded, the proxy server must write out as many streams as there are replicas—giving a multiple of network traffic. For a three-replica cluster, 10Gbps in means 30Gbps out. Combining this with the previous high-bandwidth demands of replication is what results in the recommendation that your private network is of significantly higher bandwidth than your public need be. Oh, and OpenStack Object Storage communicates internally with unencrypted, unauthenticated sync for performance—you do want the private network to be private.

The remaining point on bandwidth is the public-facing portion. Swift-proxy is stateless, which means that you easily can add more and use HTTP load-balancing methods to share bandwidth and availability between them.

More proxies mean more bandwidth, if your storage can keep up.

"Cloud Controller" To achieve maximum scalability via a shared-nothing/distributed-everything architecture, OpenStack does not have the concept of a "cloud controller". Indeed, one of the biggest decisions deployers face is exactly how to segregate out all of the "central" services, such as the API endpoints, schedulers, database servers and the message queue.

For best results, acquiring some metrics on how the cloud will be used is necessary. Although of course, with a proper automated configuration management system, it will be possible to scale as operational experience is gained. Key questions to consider may include:

- How many instances will run at once?
- How many compute nodes will run at once?
- How many users will access the API?
- How many users will access the dashboard?
- How many nova-API services do you run at once for your cloud?
- How long does a single instance run?
- Does your authentication system also verify externally?

When choosing the size of compute node hardware is mainly dependent on the types of virtual machines running, "central service" machines can be more difficult. Contrast two clouds running 1,000 virtual machines. One is mainly used for long-running Web sites, and in the other the average lifetime is more akin to an hour. With so much churn in the latter, it certainly will need a more heavyset API/database/message queue.

Scaling Given that "scalability" is a key word in OpenStack's mission, it's no surprise that there are several methods dedicated to assisting the expansion of your cloud by segregating it—in addition to the natural horizontal scaling of all components.

The first two are aimed at very large, multisite deployments. Compute cells are designed to allow running the cloud in a distributed fashion without having to use more complicated technologies or being invasive to existing nova installations. Hosts in a cloud are partitioned into groups called cells. Cells are configured in a tree. The top-level cell (API cell) has a host that runs the API service, but no hypervisors. Each child cell runs all of the other typical services found in a regular installation, except for the API service. Each cell has its own message queue and database service, and it also runs the cell service, which manages the communication between the API cell and child cells.

This allows for a single API server to be used to control access to multiple cloud installations. Introducing a second level of scheduling (the cell selection), in addition to the regular nova-scheduler selection of hosts, provides greater flexibility to control where virtual machines are run.

Contrast this with regions. Regions have a separate API endpoint per installation, allowing for a more discrete separation. Users wishing to run instances across sites have to select a region explicitly. However, the additional complexity of a running a new service is not required.

Alternately, you can use availability zones, host aggregates or both to partition a compute deployment.

Availability zones enable you to arrange OpenStack Compute hosts into logical groups, and they provide a form of physical isolation and redundancy from other availability zones, such as by using separate power supplies or network equipment.

You define the availability zone in which a specified Compute host resides locally on each server. An availability zone is commonly used to identify a set of servers that have a common attribute. For instance, if some of the racks in your data center are on a separate power source, you can put servers in those racks in their own availability zone. Availability zones also can help separate different classes of hardware.

When users provision resources, they can specify from which availability zone they would like their instance to be built. This allows cloud consumers to ensure that their application resources are spread across disparate machines to achieve high availability in the event of hardware failure.

Host aggregates, on the other hand, enable you to partition OpenStack Compute deployments into logical groups for load balancing and instance distribution. You can use host aggregates to partition an availability zone further. For example, you might use host aggregates to partition an availability zone into groups of hosts that either share common resources, such as storage and network, or have a special property, such as trusted computing hardware.

A common use of host aggregates is to provide information for use with the compute scheduler. For example, you might use a host aggregate to group a set of hosts that share specific images.

Another very useful feature for scaling is Object Storage Global Clusters. This adds the concept of a Region and allows for scenarios like having one of the three replicas in an off-site location. Check the SwiftStack blog for more information (**http://swiftstack.com/blog/2012/09/16/globally-distributed-openstack-swift-cluster**).

Summary

If you've looked at the storage options, determined which types of storage you want and how they will be implemented, planned the network carefully (taking into account the different ways to deploy it and how it will be managed), acquired metrics to design your cloud controller, then considered how to scale your cluster, you probably are now an OpenStack expert. In which case, we'd encourage you to customize your deployment and share your findings with the community. ■

After learning about scalability in computing from particle physics experiments like ATLAS at the Large Hadron Collider, Tom led the creation of the NeCTAR Research Cloud. Tom is currently harnessing his passion for large-scale distributed systems by focusing on their most important part, people, as Community Manager at the OpenStack Foundation. Tom is a co-author of the OpenStack Operations Guide, a documentation core reviewer and a contributor in a small way to Compute (Nova), Object Storage (Swift), Block Storage (Cinder), Dashboard (Horizon) and Identity Service (Keystone).

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

This is your career

Any questions?

Turbocharge your career with OpenStack training by Mirantis. As the #1 pure-play OpenStack company, we're the leading provider of OpenStack education. We've trained thousands of students with our rigorous, proven curriculum.

Courses are available worldwide. Go to training.mirantis.com to sign up for a class, or to arrange for us to come to your company and delivery a class at your site.

http://training.mirantis.com

openstack® TRAINING

MIRANTIS
Pure Play OpenStack

DOC SEARLS

Life on the Forked Road

We are analog and digital. One is old, the other new. Civilizing the latter will take some work.

On a panel at the last LinuxCon, Linus was asked if the US government ever wanted a backdoor added to Linux (**http://www.eweek.com/developer/linus-torvalds-talks-linux-development-at-linuxcon.html**). He nodded "yes" while saying "no". While we're sure Linus meant what he said, his answer calls to mind the immortal words of Yogi Berra (**http://en.wikipedia.org/wiki/Yogi_Berra**): "When you come to a fork in the road, take it." We are at that fork now. We arrived when Edward Snowden began revealing to muggles what the wizards among us always knew, or at least suspected: that government spying on ordinary data communications among private individuals was not only possible, but happening.

I was talking the other day to a high-ranking US military officer who referred to our new collective condition as "post-Snowden". It is a permanent state. We aren't going back.

The central problem of post-Snowden life is that our lives are now both analog and digital. Snowden revealed the forked road we had already taken and made clear how hard it is to travel both at once.

The difference between the two sides of the fork is binary. The analog and the digital are as different and complementary as zero and one, which are also the only numbers we need for binary math. For better and worse, we now cohabitate a world that is both analog and base-2.

Today's use of base-2 mathematics traces back to Gottfried Leibniz (**http://en.wikipedia.org/wiki/Gottfried_Leibniz**), who wrote, "Now one can say that nothing in the world can better present and demonstrate this power than the origin of numbers, as it is presented here through the simple and unadorned presentation of one and zero or nothing." The thing "presented here" was the Chinese *I Ching*, which is wrapped around the more ancient notion of *yin* and *yang*, drawn by Taoists as the familiar symbol shown in Figure 1.

Berra-ists might prefer another Eastern symbol, the *endless knot* shown in Figure 2 (**http://en.wikipedia.org/wiki/Endless_knot**).

Both convey embrasure of irony: one in a state, the other in a path.

I'd love to see a symbol for this: *embodied* and *disembodied*. The symbol shown in Figure 3 will have to do for now.

Our analog selves are embodied. Our digital ones aren't. Yet we manifest both ways to each other.

As fleshy creatures, being analog is our permanent fate. Being digital isn't, but it's what we also are now. We need to learn what that is and how to deal with it. And we have a long way to go before being digital comes as naturally as being analog. If it ever does.

In *Philosophy in the Flesh*, George Lakoff and Mark Johnson say, "The mind is inherently embodied." Because of this, they continue, our bodies supply the primary metaphors we use to make sense of the world. For example, we say *good is up* and *bad is down* because we walk upright. We say good is light and bad is dark because we are diurnal. If we were nocturnal, those might be reversed. We speak of life in terms of travel (*birth is arrival*, *death is departure* or *passing on*, *careers are paths*) because we are ambulatory animals. To live is to move.

It is also fairly easy to own things in the analog world, or at least to

Figure 1. I Ching

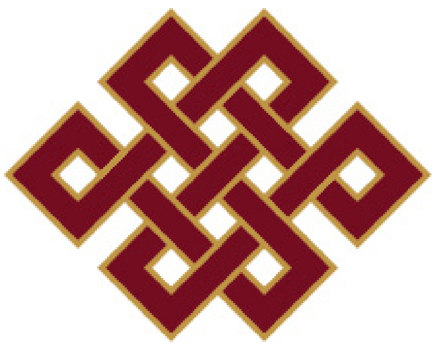


Figure 2. Endless Knot

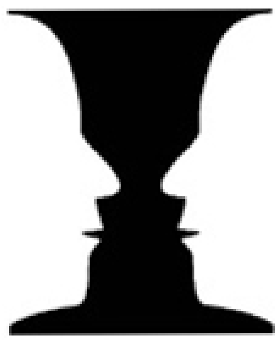


Figure 3. Embodied and Disembodied

claim them as our own. "Possession is nine-tenths of the law" because it is nine-tenths of the three-year-old. She says "It's mine!", because she has mastery of her opposable thumbs. *Possession is holding*.

Our senses also extend outward beyond our bodies through a capacity called *indwelling*. When a carpenter speaks of "my hammer", and drivers speak of "my engine", it is because their senses dwell in those things as extensions of their bodies.

But how do we dwell in the disembodied place we call the Internet? It isn't a place, even though it has *sites*, *locations* and *domains* that we *visit* and *browse*. Our bodies, even in public places, tend to be clothed, with things we call "privates" concealed. On the Net—so far—we are exposed in ways that are foreign to our bodily experience. Even the muggles among us knew that, to some degree, in pre-Snowden time. Now just about all of us know our degrees of personal exposure could be...anything, to a degree that is incalculable in the limited terms of our analog experience.

This near-absolute range of possibility on the disembodied digital side of our forked road is encapsulated in this single line from Kevin Kelly: "The Internet is a copy machine" (**http://www.kk.org/thetechnium/archives/2008/01/better_than_fre.php**). When you send me a digital document over the Net, you don't yield it, as you would in the analog way. Instead, you contribute to the proliferation of everything.

T.Rob says the difference "is one of atoms versus bits" (**https://ioptconsulting.com/industry-still-puzzling-over-consumer-reaction-to-tracking**). He goes on to explain:

> When surveillance was physical Newtonian physics limited what could be done. We didn't need laws or policies stating that you couldn't surveil all of the people all of the time because to do so wasn't physically possible. Because we have never had that capability before, we do not have any experience with it from a policy-making standpoint.
>
> Having always been constrained along that boundary by what was possible, it was difficult to overstep the line of what was ethical. Today we have the reverse. The line of possibility has withdrawn so far that it no longer constrains us from crossing the line of what is ethical. But we are so unused to

having to exercise self-restraint along this boundary that many are not recognizing the ethical line when they cross it. Many are in fact treating it like a land rush, grabbing all the territory between the ethical line and the possible line before policy and legislation can step in to claim it.

I suspect in the end it will come down to manners. There will be things that we do not do, even though we can. I see hope in the word *protocol*, which is well understood in technical circles, as well as among diplomatic and other practices back in the analog world.

But I don't have faith. Not yet. Not while the land rush is still on for marketers and government agencies, trampling our analog sensibilities. Snowden's revelations may have marked Peak Surveillance, but they also marked the point at which we start to work seriously on civilizing our digital selves.■

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.

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

Advertiser Index

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

ADVERTISER	URL	PAGE #
AnDevCon Fall	http://www.andevcon.com	2
Drupalize.me	http://drupalize.me	99
Emac, Inc.	http://www.emacinc.com	11
EmperorLinux	http://www.emperorlinux.com	31
FSCONS 2013	https://fscons.org/2013/	83
iXsystems, Inc.	http://www.ixsystems.com	7
Manage Engine	http://www.manageengine.com	57
Mirantis	http://www.mirantis.com	121
New Relic	http://newrelic.com	34, 35
OVH	http://www.ovh.com	45
Silicon Mechanics	http://www.siliconmechanics.com	3
USENIX LISA	https://www.usenix.org/conference/LISA2013	47

ATTENTION ADVERTISERS

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