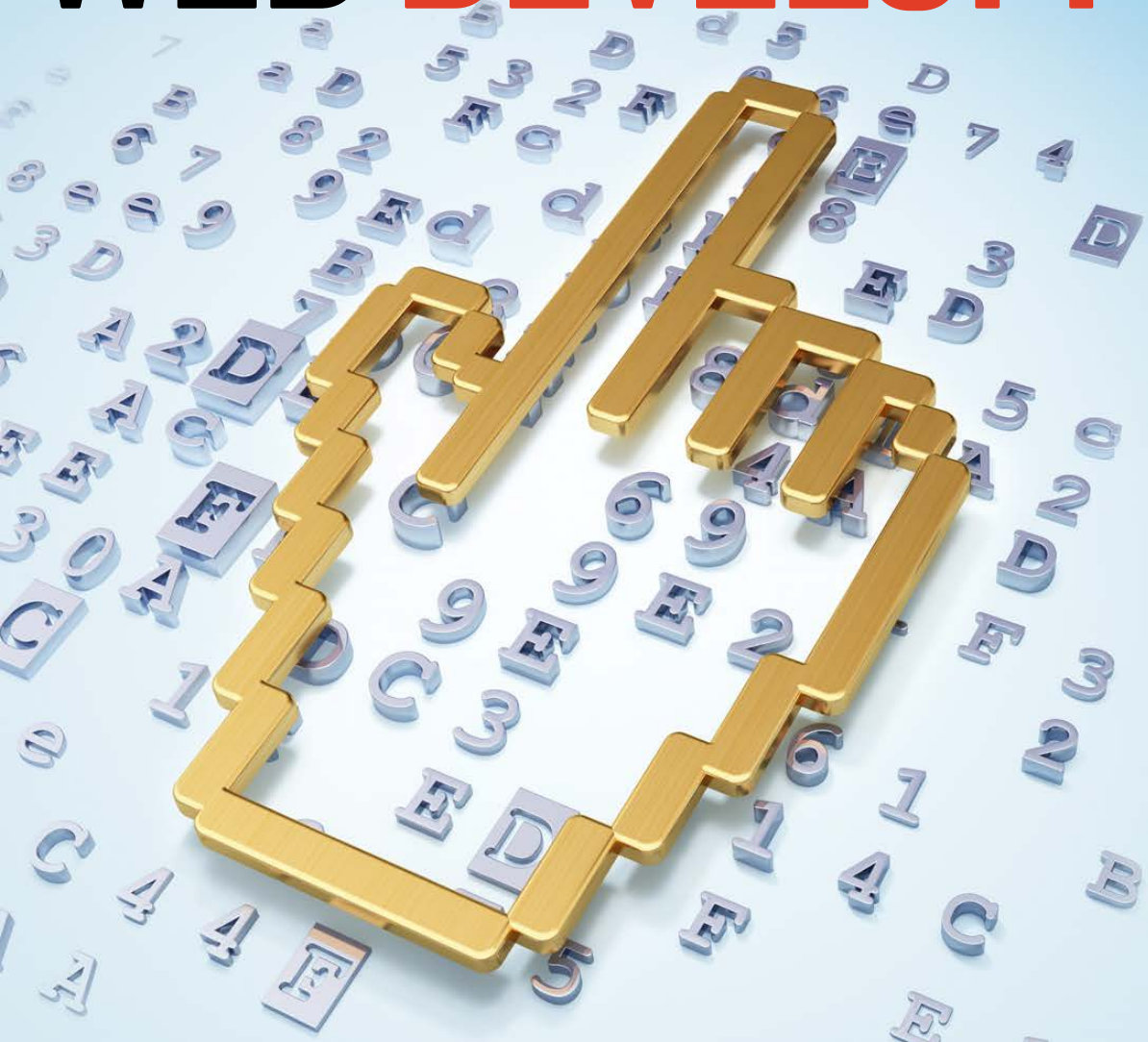
LINUX JOURNAL™
Since 1994: The Original Magazine of the Linux Community

FEBRUARY 2014 | ISSUE 238 | www.linuxjournal.com

WEB DEVELOPMENT



Develop and Deploy Quickly with PaaS Cloud Services

Build a Blog with Django and MongoDB

Keep Your DNS Search History Private

+

GUEST EOF: GIRLS AND SOFTWARE

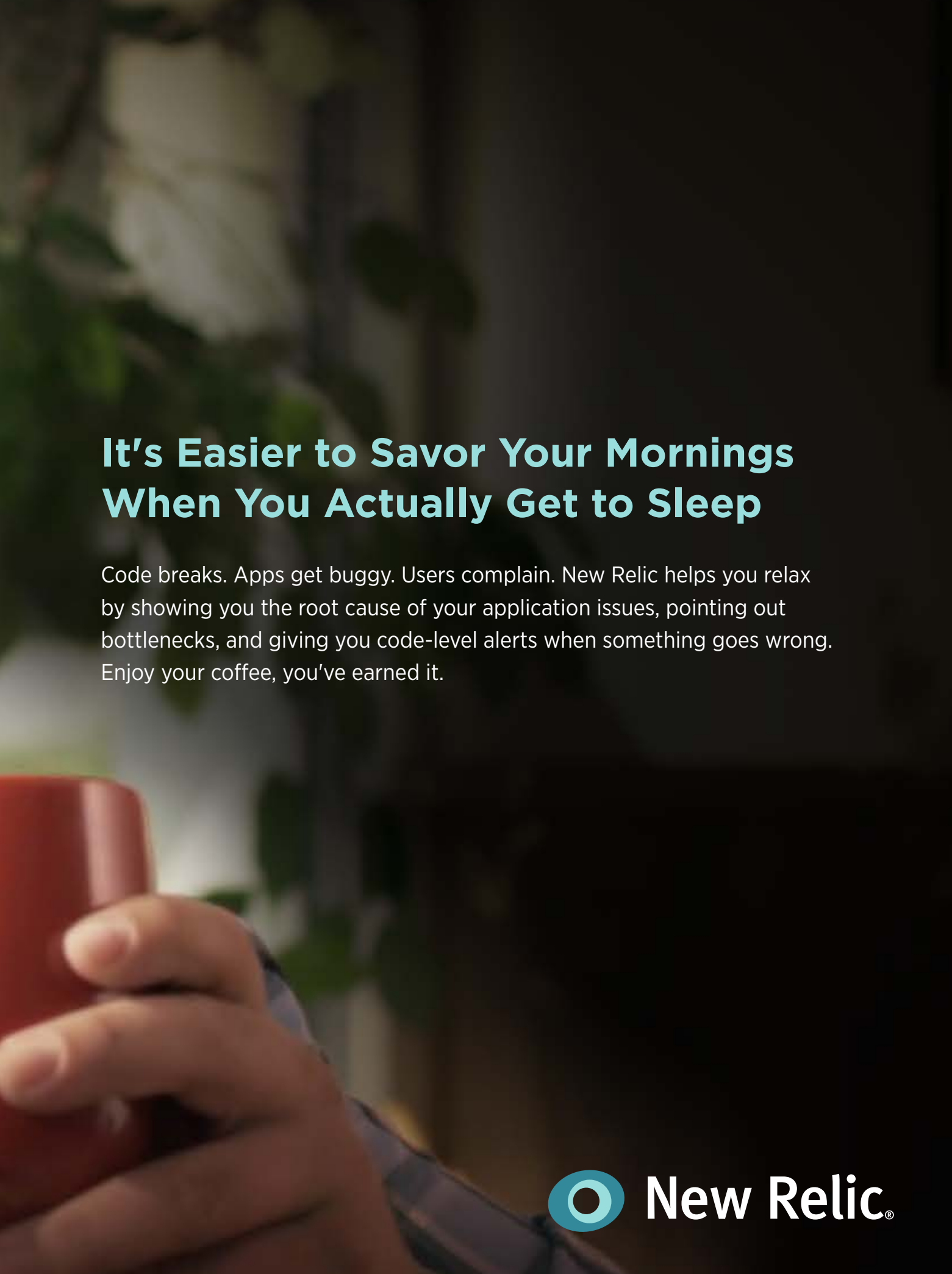
ADD SPLIT TESTING TO YOUR WEB APP

INTRO TO RACK HOSTING AND DEPLOYMENT

SECURITY GUIDELINES FOR WEB DEVELOPMENT

WATCH: ISSUE OVERVIEW


NewRelic.com/sitback

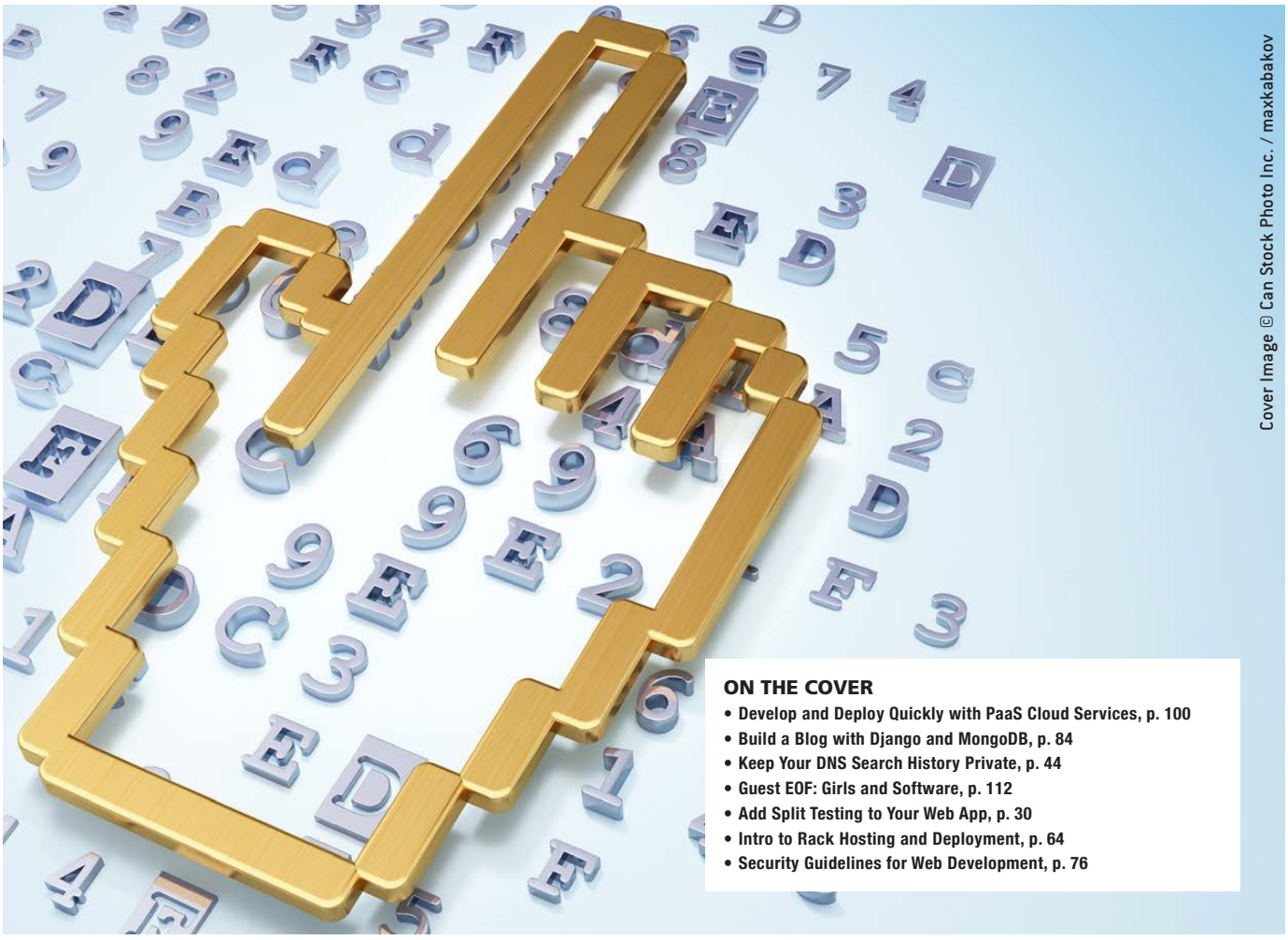

It's Easier to Savor Your Mornings When You Actually Get to Sleep
Code breaks. Apps get buggy. Users complain. New Relic helps you relax by showing you the root cause of your application issues, pointing out bottlenecks, and giving you code-level alerts when something goes wrong. Enjoy your coffee, you've earned it.
New Relic®

CONTENTS

FEBRUARY 2014
ISSUE 238

WEB DEVELOPMENT

FEATURES



Cover Image © Can Stock Photo Inc. / maxkabakov

ON THE COVER

- Develop and Deploy Quickly with PaaS Cloud Services, p. 100
- Build a Blog with Django and MongoDB, p. 84
- Keep Your DNS Search History Private, p. 44
- Guest EOF: Girls and Software, p. 112
- Add Split Testing to Your Web App, p. 30
- Intro to Rack Hosting and Deployment, p. 64
- Security Guidelines for Web Development, p. 76

INDEPTH

COLUMNS

KNOWLEDGE HUB

IN EVERY ISSUE

28 WARZONE 2100

38 IMAGEMAGICK


48 BIRDCAM, ROUND TWO

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

SILICON MECHANICS
SERVERS • STORAGE • HPC

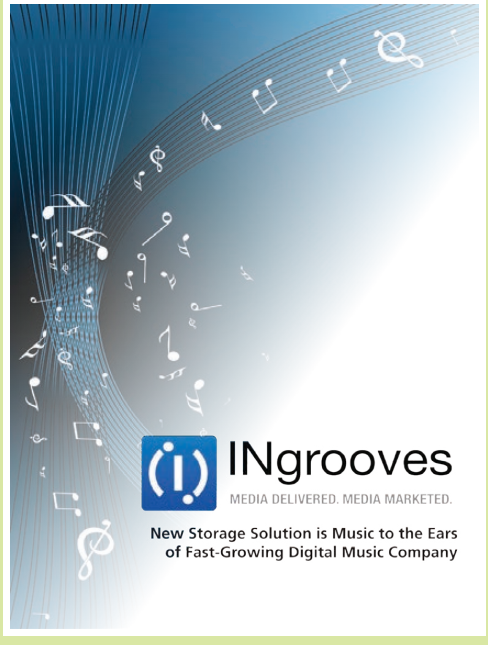
Are you considering software-defined storage?

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.

February Case Study Feature



New Storage Solution is Music to the Ears of Fast-Growing Digital Music Company.

zStax StorCore 64

zStax StorCore 104

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

Developing Webs

SHAWN POWERS

Spiders are really cool. Granted, they're terrifying, but they're still really cool. They keep the pest population down, they create super heroes, and they socialize with little girls eating curds and whey, but most impressively, they make webs. Their ability to develop such intricate and useful constructions with nothing more than spinnerets and a little ingenuity is impressive. Also impressive is the ability for programmers to develop applications for *our* Web, the World Wide Web, and make them accessible instantly to anyone on the planet. Applications usually take more than one evening to build, but with this issue of *Linux Journal*, we hope to make your Web-weaving a little more efficient, and your Web a little more awesome.

We start the Web Development issue with Reuven M. Lerner's column about split testing. When it comes to commercial Web sites, a high "conversion rate" is the ultimate goal for a company, and it's the job of the Web developer to create a site that accomplishes that goal. Reuven shows how to do just that with Ruby, but the principles extend to any platform. Next, Dave Taylor finishes his series on scripting with ImageMagick with a description of how to put frames around images, again from inside a script without the need for user interaction.

Kyle Rankin gives a great lesson on hosting DNS on your own network. With Kyle's article, you'll learn not only how to host your own DNS to help with uptime, but also how to protect your privacy. My column goes in the opposite direction, as I expose my entire backyard to the Internet. Specifically, I revisit BirdCam. I've gotten lots of feedback and questions about my backyard setup, so in this article, I discuss some of the changes I've made, including adding motion detection.

VIDEO:
Shawn Powers runs through the latest issue.

For the past few years, Ruby has been one of the most popular Web development platforms available. I'm not a developer myself, but it's apparently very straightforward for developers. As a sysadmin, I can tell you it's not as easy to manage the underlying hosting system. Fabrizio Soppelsa provides some DevOps insight and describes managing a Ruby system to host those applications. If you need to host Ruby applications, but have struggled to manage the environment, you'll love Fabrizio's article.

Nitish Tiwari follows with a great article on security in Web applications. With the fast-paced world of Web development, it's easy to overlook a security hole. With Nitish's recommendations, you can avoid some of the more common exploits. Rather than plugging holes after the fact, it's better for everyone if the development is done with security in mind from the beginning.

If you're developing a Web site that likely will get lots of traffic, you might be interested in using a NoSQL database instead of the traditional relational-style system. Unfortunately, such a system isn't as common, and many folks aren't sure where to begin. This month, Mihalis Tsoukalos explains how to use Django and MongoDB to create a blog.

"The Cloud" is quite clearly here to stay. It's also evolving in such a way that the concept of "servers" is becoming less and less important. We're all familiar with Software as a Service (SaaS), but going even further down that rabbit hole is Platform as a Service (PaaS). Mitesh Soni explores the benefits of using PaaS and describes how to leverage the concept into your Web development needs. Gone are the days when companies need to manage their own Java platforms. With PaaS in the cloud, Web development platforms are just one more commodity you can buy.

We end this issue with a guest post from Susan Sons, who responded to Doc Searls' December 2013 EOF column on women and Linux. I urge you to read her article, and hopefully it furthers the conversation even more. We've put together a great issue for you this month, and whether or not you're a Web developer, you should find plenty of helpful information between the digital covers.■

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 Audience

First, this is not a hate note, nor any kind of boycott. It's just about something that does not feel right.

Today, I received an e-mail from you guys with a renewal offer. And, I was shocked to see that the promotion appeals to practically all platforms' users, except...well... Linux users!

We complain all the time about how vendors lock out Linux users. For example, there's a lot of hardware out there that is Linux compatible, but you won't see it mentioned on the box. And, then a Linux magazine promo reads like this:

> Receive your subscription as an enhanced digital edition or PDF for your PC or Mac, on your Nook, Kindle or other eReader (.epub and .mobi formats), or via native Android or iPhone/iPad apps.

Just sayin'.

PS. Please, don't say that PCs include Linux. In computer slang, PC refers to Windows, and Mac refers to OS X.
—Eli

You make a fair point, but the funny part is that if the blurb had specified, "Windows PC, Mac OS X, and Linux", we'd have gotten messages pointing out that Windows doesn't have claim to the word "PC".

I think "PCs running Windows or Linux" would be most correct, but then we'd get letters from FreeBSD readers. Thank you for the good nature of your letter though, it's greatly appreciated.—Shawn Powers

Electronic *LJ* Is Very Hard for Me

I subscribed to *Linux Journal* sometime in the first half of the 1990s, so that makes my subscription about 20 years or longer. I read

almost all issues, almost entirely, scanning the difficult articles, zapping through "news items" and learning from comments in letters from other readers. Yes, I wanted to go on following Linux. As a side note (I married since), I changed my subscription to a cheaper delivery: through ACM membership. Apparently they had better EU postage conditions? But, the magazine disappeared from the real, tangible world. There was nothing left to hold while traveling, although I could have risked being "the guy who opens a laptop in the Brussels metro". How long before it falls? I don't see laptops getting stolen, but I don't see laptops on public transport, except on long-distance travel.

I wouldn't mind paying extra to have a paper version again. It can be done: localized digitalized printing, with local postage. Isn't there something like "lulu" and other, often small print-and-ship companies?

A second, but important reason *not* to read a 100+ page magazine on anything *backlit* is eye strain. I am surely not the only *Linux Journal* reader with aging eyes. Teenagers have no problem staring into lamps for hours, but I do. E-paper isn't really a simple alternative. And printing the whole issue on normal printing paper, even with a modern "colour laser" but on A4 paper, is *heavy*.

ACM offers a print subscription, and I read it—not everything (some articles in *ACM* are too difficult), but I read big parts of it. *Linux Journal*, which

is +/- the same size but on "printer paper" is a heavy book.

Is there really no way to find a local printer (in Germany, Belgium, Netherlands, France, Denmark and so on) who can ship the magazine at reasonable over-land rates?
—HansRens

I wish I had a better answer for you, but we don't have any near-future plans for offering paper issues. Honestly, I miss the paper version as well. I'm not sure of the legality of having a personal copy professionally printed, but using something like Lulu or HP MagCloud probably would work, and can be done for single print runs. It's not perfect, but hopefully, it will suffice until full-size, color e-ink becomes a reality.—Shawn Powers

Women in the Workplace

With respect to Doc's article in *Linux Journal*, "Mars Needs Women" in the December 2013 issue, I have a few related (at least to me) comments.

A few years ago, I worked for a woman who used to work for me, and she was an excellent manager for both men and women. She is an anomaly in that she is very much like the mythical super-female manager.

From what I have seen elsewhere and heard from most of my smart female friends, women are not the "be all and end all" in the workplace. One major problem women have in the workplace is that they commonly do not work well together.

A good and bad characteristic of women is that they tend to be more relationship-oriented. Typically, pretty women who are not so smart are particularly threatened by smarter women and then display all the characteristics that are the antithesis of those excellent female managerial tributes not attributed to men. From many men's point of view, women in the workplace is a good thing except when there are women competing with each other. Then you have all sorts of sniping, back-stabbing, mental-health days off, a very unhappy workplace and the resulting significant reduction in production. This is not to say that men aren't the root cause of similar problems, but that is certainly not as common as for women.

An example: one female bar owner I talked to a few years ago said she would much rather hire male waiters

because when she "told them off" or otherwise constructively criticized them, they much less frequently took the criticism personally and got on with the job. The male waiters quickly would forget the incident but still take the criticism to heart.

Don't get me wrong. I like having women in the workplace. I liked working for some women, and I very much disliked working for some very incompetent male managers, and I enjoyed supervising a group of women in a pool. It is interesting that I don't remember experiencing the problems I have noted above, but I did hear, indirectly, that the women very much preferred working for me rather than a female manager. But then I didn't really care about relationships other than I made extra effort to be transparent, and I made sure that I was, as well as was perceived to be, fair and just.

We need more articles, not in *Linux Journal*, that are more honest about women in the workplace and how to get women working better with each other. It must be possible, as there are many examples where there are very successful female businesses and organizations. Note, nursing is *not* one of them.
—Roger

Considering Women in Linux

Regarding Doc Searls' article "Mars Needs Women" in the December 2013 issue: I enjoy reading articles about women and Linux. I would like to share with you my observations of a local Linux club meeting. Although there are three or four female members (out of 300), I am often the only woman. There are 20 or so men, of varying ages, everyone with some sort of laptop. I am introverted and find it difficult to talk to people in general. Most of the men also seem rather shy, but I never sense that I do not belong. I have talked with two members actively, and they were friendly and supportive. My feelings of awkwardness are caused by my inability to interact with them. I go to the meetings in order to have some sort of contact with others who use Linux.

I have been using Linux since a friend installed it on my first computer in 1993 using something like 20 floppy disks. I never have mastered it, and I usually fix problems by re-installing. I've gone to a few Linux conferences here in Europe (I am American living in the Netherlands). I can code in C but have hesitated to get more involved. I tend to work on one project at a time, and if I start a Linux project, all my other projects get ignored.

I am writing this because I feel that

framing the discussion in terms of men and women is not helpful, especially when using Western stereotypes. I rarely read that some women also are introverted and shy. Therefore, I do not fit in either group Doc discussed in his article. It looks like some progress is being made for younger women (I am 53). But one should not forget that not all women match the "Venus" stereotype.
—Rose Dlhopolsky

Doc Searls replies: *Thanks to Rose and others who have reached out in response to my "Mars Needs Women" EOF column in December 2013. I wrote it to stimulate thought and conversation, and I think succeeded at that.*

One more thought: it is an error in statistics to impute cause to correlation. So, is the high ratio of men to women in Linux one that correlates to factors other than gender? If so, should we be talking about those other factors instead? And what are they?

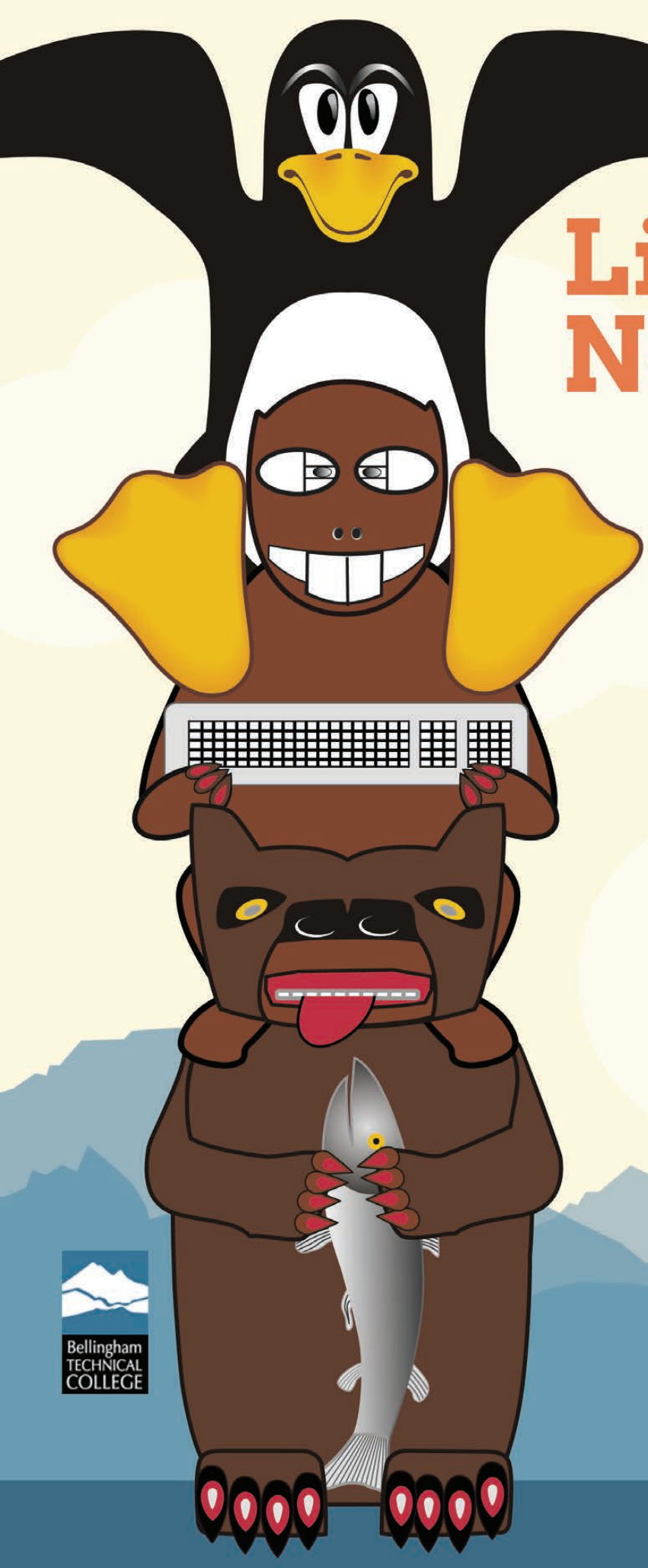
I don't yet know, but I do invite readers to check out our guest EOF this month by Susan Sons, who does a great job of moving this conversation forward.

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!



LinuxFest Northwest

Bellingham, WA
April 26th & 27th

- Grassroots Linux gathering
- Exhibits of all flavors
- Presentations of all levels
- Prizes and after party
- FREE admission & parking
- FREE open source software
- Bring the whole family!

Bellingham TECHNICAL COLLEGE

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

Recently, **Aldo Iljazi** suggested removing the venerable **menuconfig** build target, on the grounds that **nconfig** was an improvement, and there didn't need to be two menu-based configuration systems in the kernel tree.

nconfig is actually based on the menuconfig code, and they both rely on ncurses to present menus. But nconfig tries to look more modern and gives the user a bit more control using the keyboard.

The idea went nowhere—perhaps it was just too soon, because nconfig has a tiny user base relative to that of menuconfig. But, it was surprising to see how much resistance there was. At one point, **Alexander Holler** raised the objection that nconfig relied on the Fn keys for its operation, which were not available on software keyboards on smartphones, for example. But, even after **Randy Dunlap** pointed out that the regular number keys worked just as well, there still was overwhelming opposition to ditching menuconfig.

It's interesting that certain parts of the kernel—for example, config targets that don't themselves bloat the compiled binary (though they may help select features that do)—are much easier to get into the kernel source tree than actual kernel features. And once in, they are harder to remove. There was no particular need for nconfig, given that it performs a similar function to menuconfig, but there it is in the kernel source tree. These helper projects come and go fairly easily, probably because there's not much cost to having them, and by including them in the tree, they get the chance to show whether they really actually may be better than the alternatives.

Recently, **Jim Lieb** tried to simplify the interface that allowed file servers to impersonate their client user for write operations. This is standard procedure, without which files would have the wrong owners, and things like quotas and access controls would not be able

to tell whether a user violated a given constraint. But the existing implementation used a combination of various system calls, including **setfsuid()**, **setfsgid()**, **setgroups()** and others to accomplish this. Jim wanted to replace the mess with a single **switch_creds()** system call.

As it turned out, there was support for the general idea of cleaning up the existing interface, but no agreement on exactly how it should be done. **Al Viro**, for example, offered his own implementation that avoided some of the complexities of Jim's approach. But both approaches turned out to have significant security gaps, as **Eric W. Biederman** and **Tetsuo Handa** pointed out at various times during the conversation.

Everyone seems agreed on the fact that the current implementation is a bit messy and could use a cleaning. But apparently the security issues are devious and need to be gotten right. The current messy implementation also may turn out to be the best way to deal with those issues—in which case, it really couldn't be considered messy in the first place.

Recently, **Peter Huewe** took over as the primary maintainer of the **TPM** (Trusted Platform Module) device driver. TPM is a hardware authentication system that allows third-party services to confirm that only a trusted operating system and set of software is running on the device.

Many other maintainers were listed, several of whom had not been responsive for a while and there was general agreement that the list should be pruned and kept accurate.

A number of folks tried to contact the various maintainers to ask if they were still interested in working on the project. Ultimately, several folks, such as **Rajiv Andrade** and **Ashley Lai**, said they still were interested in helping out, but they recognized Peter as the project leader. A number of other folks asked to be removed from the maintainers list, as they had moved on to other projects or other companies.

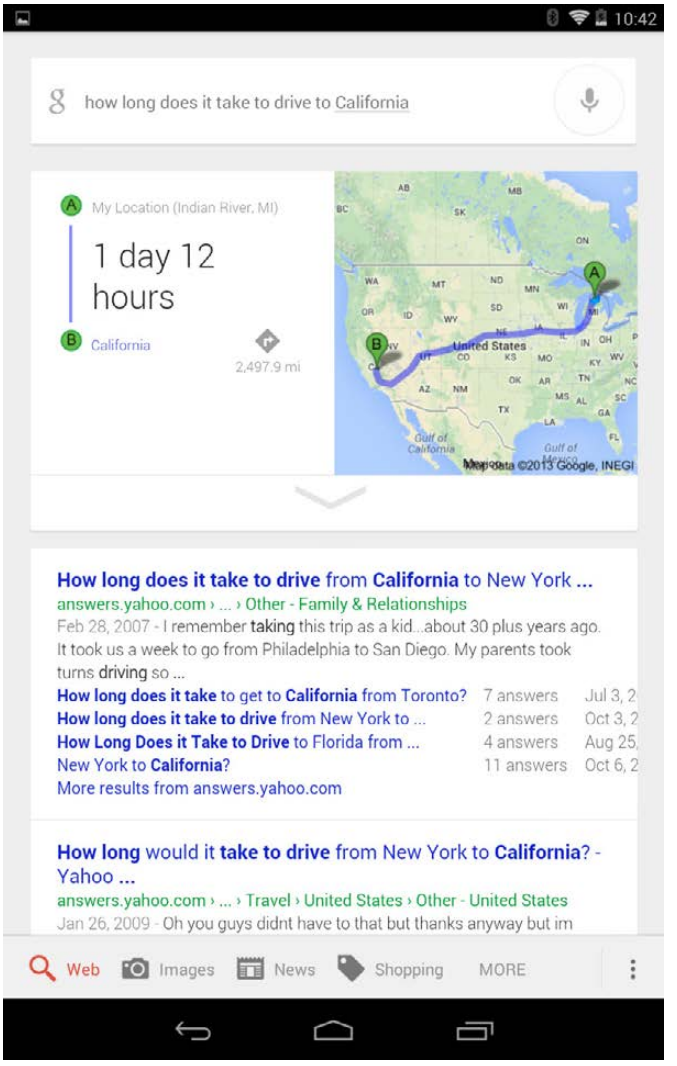
Jason Gunthorpe inaugurated the new maintainer hierarchy by submitting a set of TPM patches that had been waiting for inclusion and remarking, "there are still lots of patches to go before the subsystem meets the current kernel standard." **—ZACK BROWN**

Okay, Google

My favorite scene in *Star Trek IV* is when Scotty tries to use the computer in the 1980s. When he's told he must use the mouse, he responds, "how quaint", and then proceeds to try speaking into the mouse for the computer to respond. With the advent of Siri on iOS and voice recognition on Android, it's beginning to feel like the voice interface portrayed in *Star Trek* isn't too far away.

But it's not here just yet.

I set up my Nexus 7 tablet with the most recent tools from Google (technically, they're not yet available for the Nexus 7, but I'm a nerd, so I was able to find a way). I set my now always-responsive tablet on the window ledge in my office, just out of reach but in easy earshot. I went through the entire day, trying to use the tablet as often as possible without touching it. I discovered a few things:



- Google is really good at giving certain types of feedback. If I asked about the time in London, the current weather or the stock price of a popular stock, I'd get a visual response along with a voice telling me the answer.
- Outside that small list of things Google is really good at answering, it doesn't do anything more than give search results on the tablet. I was hoping for something like, "would you like me to read you the most popular search result?" But alas, it didn't even audibly tell me it heard my question.

- Sending texts and e-mail messages is possible, but frustrating and scary. If you've ever tried to use voice calling with a Bluetooth headset, you've probably had the awkward experience of your phone accidentally trying to call an ex-boyfriend or girlfriend instead of calling the plumber. If you're lucky, you can stop it before it rings on their end, but thanks to caller ID, you're likely in for a very uncomfortable followup call. I found Google's voice-based messaging more cautious than my Bluetooth headset, but still potentially bad. This is especially true because the tablet was across the room, making it hard to dive and press cancel.

So, although we may not be to the point where we can ask Jarvis to order us a pizza while we're flying around in an Ironman suit, we're definitely taking a step in the right direction. The advent of Google Glass will make verbal commands more and more common. Even if you hate Google Glass, you can rejoice in the voice interface improvements it doubtlessly will cause.

Is voice interface more than a novelty for you? Do you successfully send messages to people on a regular basis by dictating only to your smart device? Did you think *Star Trek IV* was awesome too? I'd love to get feedback on your thoughts concerning voice interfaces, Google Glass and the future of interfaces in general. Send me an e-mail at ljeditor@linuxjournal.com. I, for one, look forward to my first cranial implant. (I'd like to wait for version 1.1 though—nobody wants a buggy brain implant!)—**SHAWN POWERS**

They Said It

However beautiful the strategy, you should occasionally look at the results.
—Winston Churchill

Don't bother just to be better than your contemporaries or predecessors. Try to be better than yourself.
—William Faulkner

If women are expected to do the same work as men, we must teach them the same things.
—Plato

Never read a book through merely because you have begun it.
—John Witherspoon

Avoid the crowd. Do your own thinking independently. Be the chess player, not the chess piece.
—Ralph Charell

Anubis, the God of Dead Bitcoin Miners

Figure 1. Anubis gives a nice overview of all the problems with my mining farm.

With the recent resurgence of Bitcoin and the subsequent vitality of other cryptocurrencies (Litecoin, for instance), I've been receiving lots of e-mail messages asking how to mine. I've discussed cryptocurrencies in *LJ* quite a bit during the past few years. Recently, a friend introduced me to Anubis, so I want to mention it briefly here.

Whether you're mining for Bitcoins with ASIC hardware or Litecoins with high-end graphics cards, chances are you're using the cgminer program to do your mining. Although cgminer provides a nice console-based screen for monitoring your miner, there's no easy way to see how all your miners are doing at once. Enter: Anubis.

Anubis is a Web-based program that interacts over the network to all your miners. It then combines the data it collected into a simple monitoring screen so you can check temperature, errors, efficiencies and even change configurations on the fly. If you're running more than one instance of cgminer in your mining farm, you likely will benefit from Anubis. Check it out at **https://github.com/pshep/ANUBIS**.—**SHAWN POWERS**

Full SteamOS Ahead!

(Image from http://www.steampowered.com)

Although its timetable may not always be ideal, Valve has come through for Linux users lately. Not only has it released a native Linux version of Steam (with many native games!), it also has expanded its Linux support as the basis for its standalone SteamBox. The first step toward a Steam-powered console is the operating system. Thankfully for nerds like me, Valve released its operating system (SteamOS) to the public.

SteamOS is in beta testing right now, and unfortunately at the time of this writing, it supports only NVIDIA graphics cards. That limits who can test the OS, but releasing the operating system at all is extremely exciting! Geeks have been creating their own XBMC boxes for years, and now we'll be able to create our own gaming consoles too.

If you haven't tried SteamOS yet, and if you have an NVIDIA graphics card, I urge you to go try it out (**store.steampowered.com/steamos/buildyourown**). Will the SteamBox finally bridge the gap between PC gaming and console gaming? Will its open-source roots help SteamOS become the dominant living room device? It's been a number of years, but Valve definitely has invested into the Linux community. Now if you'll excuse me, I need to go shoot some zombies.—**SHAWN POWERS**

Linux Help for Neuroscientists

In past articles, I have looked at distributions that were built with some scientific discipline in mind. In this article, I take a look at yet another one. In this case, I cover what is provided by NeuroDebian (**http://neuro.debian.net**).

I probably should start by clarifying that NeuroDebian is not strictly a Linux distribution in the classical sense. The people behind NeuroDebian began by working on PyMVPA (**http://www.pymvpa.org**), a Python package to do multivariate pattern analysis of neural data. To make this package easy to deploy, NeuroDebian was created. Over time, more and more packages were added to NeuroDebian to try to create the ultimate integrated environment

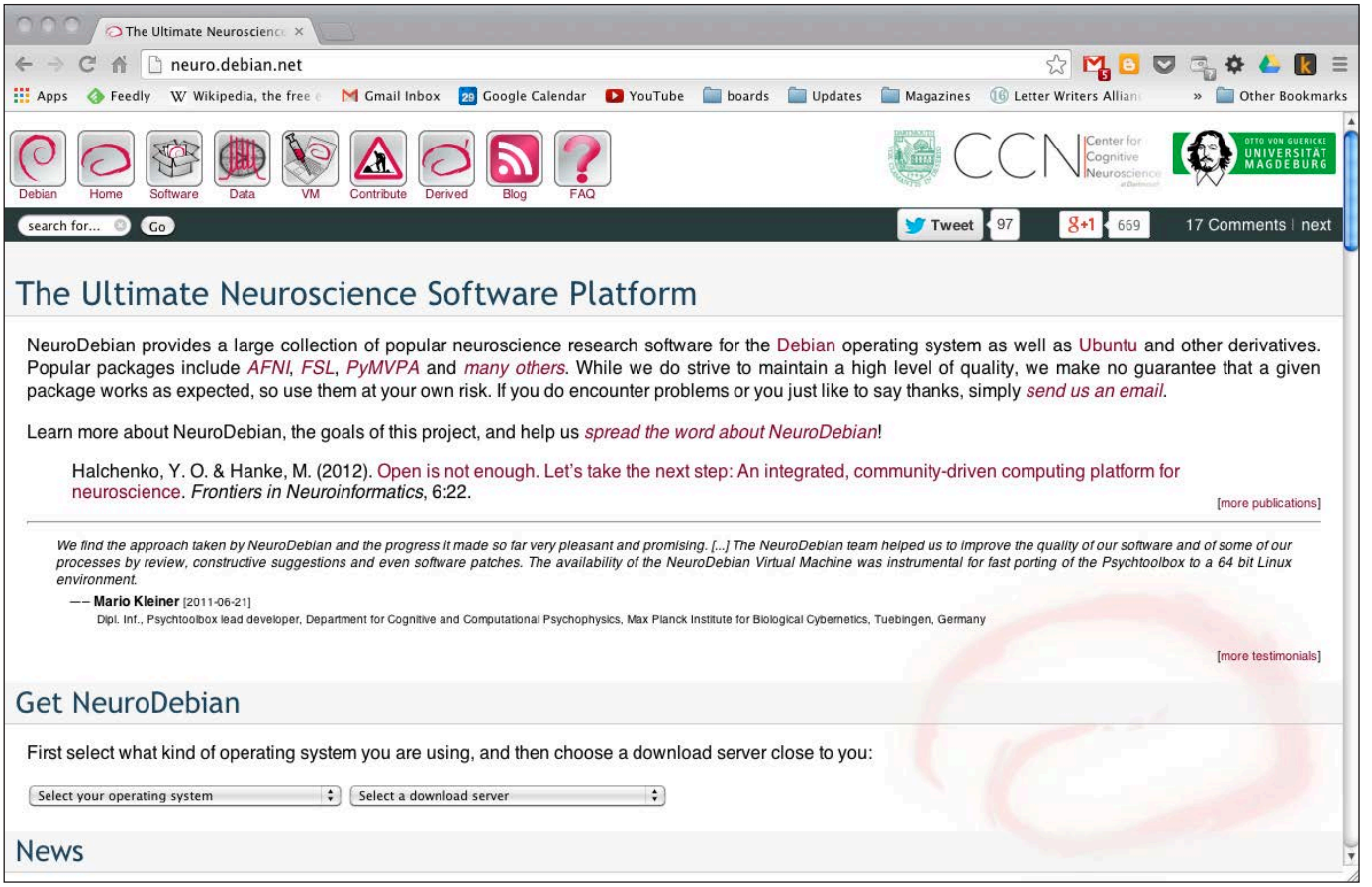


Figure 1. The Main Page for NeuroDebian

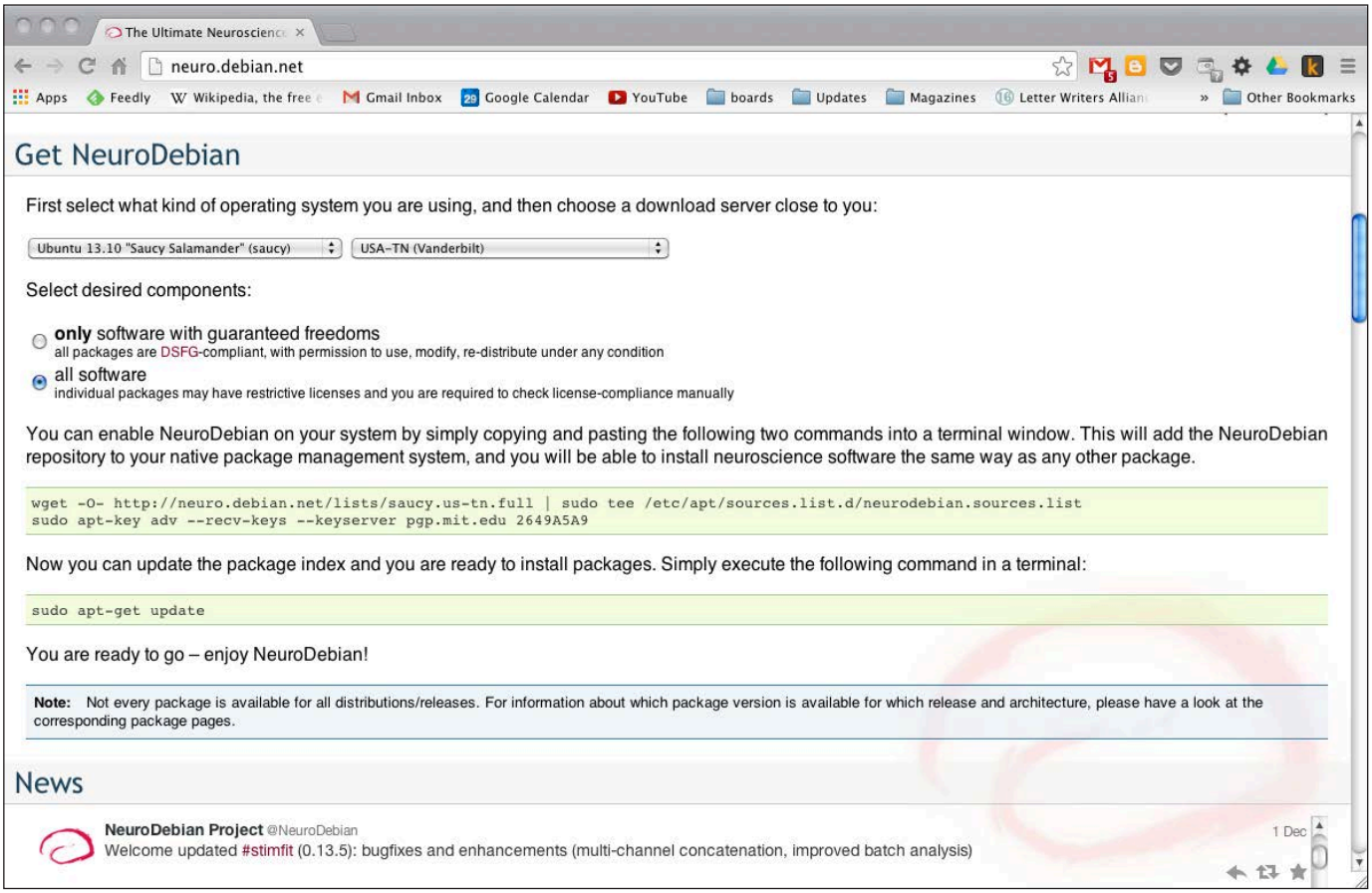


Figure 2. You can get NeuroDebian further down on the main page.

for neuroscience. All of this work is described in a scientific paper, "Open is not enough. Let's take the next step: an integrated, community-driven computing platform for neuroscience" (**http://www.frontiersin.org/Neuroinformatics/10.3389/fninf.2012.00022/full**). This paper is available at the "frontiers in NEUROINFORMATICS" Web site.

Installing NeuroDebian is a bit different from other distributions. On the main home page, there is a section called Get NeuroDebian. Here you can select which distribution you use as your desktop and the mirror from which you want to download.

You then get a couple commands that you need to run on your system. The first one is a wget command meant to download an entry for APT and store it in a source file in the directory /etc/apt/sources.list.d/. The second command uses apt-key to go out to the MIT PGP key server to download and install the key used to verify the NeuroDebian packages. Once these two commands have been run, you then can do:

```
sudo apt-get update
```

to download the package definitions for everything provided by NeuroDebian.

This works well if you already are running some version of Debian, or a derivative like Ubuntu, as your desktop operating system. But, what can you do if you are running Windows or Mac OS X? The NeuroDebian project provides a virtual machine option for those situations.

If you select either Windows or Mac OS X as the operating system in the download section, you will be provided with a link to download an OVA file. This type of file is a standard file format for virtual machines. For example, you can import this file into Virtual Box (Figure 3). This virtual machine uses Debian 7, or Wheezy, as the core operating system. The main

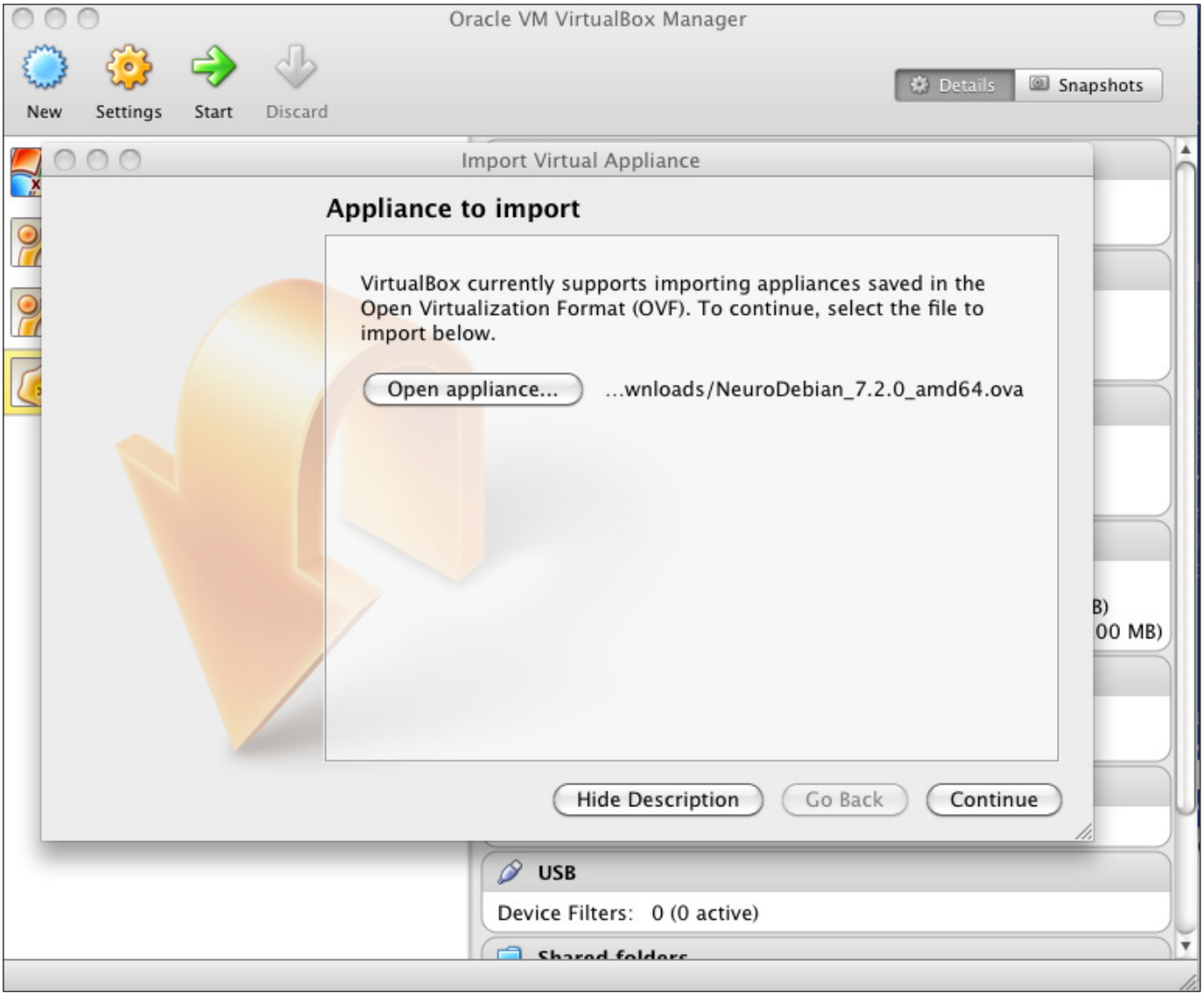


Figure 3. The OVA file can be imported in Virtual Box to get a complete environment.

Web site says that GNOME is used as the desktop environment.

However, when I actually installed the latest version of the virtual machine, the desktop environment that is used is XFCE. You even could use this on your Linux desktop in a virtual machine. This way, you always have a stable, complete computing environment for neuroscience that you know will not change or be broken.

When you first start up this virtual machine, you will be presented with some configuration steps. The first step is to do an update of the installed packages. After this, you will be asked whether you want to take part in an application survey. If you are using NeuroDebian regularly, you may want to take part in order to provide feedback to the team. Several tools require environment variables to be set. The next step asks you whether you want these to be set automatically in the default profile settings. The next step allows you to select several extra packages like Emacs, a PyMVPA tutorial and R. Be prepared for a bit of a wait, as there are several packages to be downloaded. In my case, I ended up with a download of 625MB of extra packages.

After all of the configuration steps are completed, you can click on the Applications Menu button in the top right-hand corner and go down to the NeuroDebian menu entry. Here you will find all of the particular applications specifically selected for neuroscience. They are broken down into categories for electrophysiology, medical imaging, psychophysics and a section of support links to access the relevant mailing lists. There is also an entry to re-run the setup wizard for the virtual machine.

Now that you have NeuroDebian installed and set up, let's take a quick look at some of the provided tools.

The core reason for the creation of NeuroDebian was to deploy PyMVPA, so let's start there. PyMVPA provides a set of tools to do multivariate pattern analysis on large data sets. This is very useful in neuroimaging. Several processing steps are usually involved in this type of work flow, such as data preparation, classification, feature selection and generalization testing. PyMVPA provides high-level abstraction of these processes.

A tutorial is available at the PyMVPA project Web site that walks you through the core concepts and processes involved in using it. A full description of what you can do would require a whole article on its own.

PyMVPA isn't the only software included, however. Going to the package list at the NeuroDebian home page gives a full listing, broken down into the following categories: distributed computing, electrophysiology, magnetic resonance imaging, modeling of neural systems, neuroscience datasets and psychophysics.

An interesting piece of software is under the educational category: virtual-mri-nonfree. This package provides a virtual MRI scanner to simulate running an MRI. This way, you can learn about how scanner parameters affect your images—a very cool tool.

Software is not the only thing provided by the NeuroDebian distribution. There is also a rather large set of data packages available, all in one location. These include items like brain atlases, fMRI data from face and object processing in the ventral temporal cortex, and an MRI-based brain atlas of the anatomy of a normal human brain.

There are tutorials for PyMVPA and MRI analysis that require sample data sets. These also are available from NeuroDebian. Additionally, there is a blog on the NeuroDebian Web site where you can find articles on specific tools and help with particular techniques.

If you do work in computational neuroscience, you could do worse than starting with NeuroDebian. This distribution gives you a full set of tools to get you started. There even are further derivatives of NeuroDebian, built to support classwork or to have a specific subset of the tools available for well-defined work flows. Maybe other research communities might be tempted to do a similar project? If you have the ability, you should consider offering some of your skills back to the project in order to help it grow. Of course, this is true of all-open source projects.

—JOEY BERNARD

The Best SharePoint Training in the World!

Choose from more than 80 classes and tutorials!

"This is the place to be if you want to know about SharePoint. Not just technical, but business-related information as well."

—David Pileggi, Lead Consultant, Portal Solutions

"SPTechCon has lots of excellent real-world knowledge, bonding opportunities and the ability to interface with leaders in the space."

—David Miller, Director, IT, Cubic Transportation Systems

SPTechCon

The SharePoint Technology Conference

April 22-25, 2014

San Francisco Hilton

Bolster your career by becoming a SharePoint Master!

- Learn from SharePoint experts, including dozens of SharePoint MVPs and Certified SharePoint Professionals
- Master document management
- Study SharePoint governance
- Find out about SharePoint 2013
- Learn how to create applications for SharePoint that solve real business problems
- Exchange SharePoint tips and tricks with colleagues
- Test-drive SharePoint solutions in the Exhibit Hall

If you or your team needs Microsoft SharePoint training, come to SPTechCon San Francisco!

Register Early and SAVE! → www.sptechcon.com

A BZ Media Event @SPTechCon

A Look at *Warzone 2100*

I'm not really much of a computer gamer. That said, I'm both ashamed and oddly proud of the hours (probably thousands!) I spent playing *Dune 2000* back when it was cutting-edge gaming technology. There's just something about real-time strategy games that appeals to those of us lacking the reflexes for the more action-packed first-person shooters. If you also enjoy games like *Dune 2000*, *Starcraft*, *Warcraft*, *Civilization* or other RTS classics, *Warzone 2100* will be right up your alley.

Warzone 2100 reminds me very much of my beloved *Dune 2000*. The landscapes, the missions and even the look of the game pieces resemble that old RTS game I spent so much time playing. *Warzone 2100* is far better than *Dune 2000* ever was, however, thanks to its amazing set of features:

- Cross-platform, supporting Windows, Mac and Linux.
- Single-player missions.
- Multiplayer gameplay.

- Network and Internet hosting/playing.

Warzone 2100 truly excels at being a fun, easy-to-learn game. The coolest part, at least in my opinion, is its history. *Warzone 2100* started as a commercial game. Much like the *Quake* engine was open-sourced, *Warzone 2100* was released to the public as an open-source project back in 2004. Then, in 2008, the rights of that license were clarified, and the in-game videos and soundtrack also were released. Now the game is under active development, and it has a healthy community releasing maps and mods.

The game is available for direct download either at SourceForge or its Web site: **http://www.wz2100.net**. It's also available using Desura, the Linux-native game manager (similar to Steam) that we've covered before in *Linux Journal*. Due to its fun and relevant gameplay, cross-platform availability and awesome history, *Warzone 2100* is this month's Editors' Choice.—**SHAWN POWERS**

REUVEN M. LERNER

Split Testing

Adding split testing to your Web application is easy, fun and potentially quite profitable.

It's nice to have many people visit your Web site. It's even better when people don't just come to your site, but also enjoy your content. But, best of all is when visitors to your site do what you would like them to do—sign up for your newsletter, register for your SaaS application or buy one of your products.

The rate at which visitors become customers is called the "conversion rate", and it's probably the top priority for Web-based businesses. If you can convert 10% of your visitors into customers, you're doing twice as well as the person who can convert 5% of visitors into customers.

What leads people to convert more often? That's a question to which I'm still learning the answer, as is an entire industry of Internet marketers and "conversion optimization" experts. The thing is, it's often difficult to know what will work. Should you use a green button or a red one? Should your headline be a question or a statement?

One of the most popular, and effective, ways to check the effectiveness of your copy is to do "split testing", sometimes known as "A/B testing". Although I heard about A/B testing years ago, I have only recently actually started to apply it.

In this article, I describe a simple way to run split tests on your Web pages. The examples I show here are written in Ruby, but there are A/B testing libraries for many different languages. There also are third-party businesses that can help you with A/B testing, either on your own Web site or on your mailing lists, lead pages and other products.

It's All about Conversion

The most important thing to keep in mind when you're doing split testing is that you're trying to get users to do something. What that something is depends on your site. The goal at Amazon is to get you to buy things. The goal at Google is to get you to click on ads. And the goal for a mailing-list subscription form is to get people to sign up.

Once visitors have achieved your

goal, they have been "converted" into customers. The goal is to increase the number of such conversions—giving you more customers, subscribers or users of your system. The key insight with split testing is that by changing the text, graphics and even layout of your page, the number of conversions will change as well. The question is, which of your various ideas will work best?

Split testing uses the scientific method, backed by simple statistics, to try to answer that question. In science, you can test a hypothesis by using a control and an experiment. Split testing works the same way. You take your existing text and an alternate text, and display them to your users. One of them probably will work better than the other. You analyze the numbers to understand which worked better, and then you use the best one. Then you start all over again, trying to find another improvement via split testing.

In order for split testing to work, you need to do several things:

- Define what counts as a conversion. This often is described in terms of the user arriving at a particular page on the site, such as a "thank you" for shopping that appears after a successful sale.
- Define the control and alternate texts.
- Wait for enough users to see both.
- Analyze the numbers.

Introducing Split

Regular readers of this column know that I tend to use Ruby on Rails for most of my Web development work. Rails isn't the only Web framework written in Ruby, although it certainly is the most widely used. All modern Ruby-based Web frameworks use "Rack" to communicate with the HTTP server software that invokes them. This is analogous to the Python world's WSGI. The idea in both of these cases is that the HTTP server doesn't need to know much about the Ruby or Python application (or framework) and vice versa. This means libraries that know how to work with Rack can operate with any framework, not just with Rails. In the specific case of Ruby, Rack-sensitive gems thus can work with Rails, Sinatra or anything else that follows the Rack API.

The Split gem, written by Andrew Nesbitt (with a number of contributors), is one such package of code, able to provide split-testing execution and analysis to any Rack-compatible application server. It is based on a number of earlier split-testing gems, including the well-known A/Bingo gem by Patrick Mackenzie. The idea behind Split is you identify your conversion

target, automatically try two or more variations on users, collect statistics on which of them actually managed to achieve more conversions, and then start to test things again.

Split takes care of much of this work for you, allowing you to concentrate on the portion of your site you want to change, rather than the mechanics of setting up experiments and reporting their results. Split also implements a number of features that ensure the statistical power of your results. For example, it will involve each participant in only a single experiment at a time, and it will compare results only after at least 30 people have seen it. These options can be changed, of course, but the defaults are more than good enough for most basic tests that you'll want to run.

You install split as you would any other Ruby gem:

```
gem install split -V
```

Note that I use rvm, the Ruby version manager, which allows me to work with multiple versions of Ruby at any given time. One side effect of using rvm is that gems are installed in my own home directory, rather than on the system. As a result, I don't need to preface the `gem` command with `sudo`; depending on your system configuration, you might need to do so.

After installing the gem, you also will need to install and start Redis, an extremely fast and popular key-value store. I have used Redis in a number of projects through the years, and I never cease to be amazed by its utility for caching. In the case of Split, Redis is used to keep track of the experiments you are running. You can start Redis by executing:

```
redis-server
```

Running a Split Test

Now that you have installed the gem and Redis, you're ready to perform

Listing 1. linksplit.rb, the Main Sinatra Application

```
#!/usr/bin/env ruby

class LinkSplit < Sinatra::Base
  enable :sessions
  helpers Split::Helper

  get '/foo' do
    erb :foo
  end

  get '/bar' do
    finished('click_text')
    erb :bar
  end
end
```

some experiments. Let's create a simple Web site using Sinatra, in which the page displays a link. While Sinatra has a reputation for being simple and for allowing you to create a single-page application, I've generally gone for a configuration that uses several files: the application itself, a Rack configuration file (setup.ru) and a Gemfile to list all the Ruby gems I want to use (via the "Bundler" management gem). These are shown in Listings 1, 2 and 3.

Let's review them briefly. The setup.ru file is the way that Rack runs your application. If you want to run the application without any external HTTP server, you can use the `rackup` command:

```
rackup config.ru
```

Rack then will run an HTTP server (defaulting to the built-in WEBrick server that comes with Ruby), acting as the glue between the server and your application. The config.ru basically bootstraps your application, loading the gems listed in the Gemfile (thanks to Bundler), then loading your application and the special code needed to run the Split dashboard.

Then you do something interesting. Rather than simply running your

Listing 2. config.ru, the Rack Configuration File for LinkSplit

```
Bundler.require

require './linksplit.rb'
require 'split/dashboard'

run Rack::URLMap.new("/" => LinkSplit.new,
                     "/split" => Split::Dashboard.new)
```

Listing 3. Gemfile, Used by Bundler

```
source 'https://rubygems.org'

gem "sinatra"
gem 'split', github: 'andrew/split'
```

application, you tell Rack that it should route requests to different places. Anything starting with a / should go to LinkSplit.new, meaning a new instance of the Sinatra application. But, anything starting with /split will be routed to a completely separate Sinatra application, Split::Dashboard.new, part of the Split gem.

The Sinatra application defines two different URLs, both of which are available via HTTP GET requests. The first, /foo, displays the contents of the foo.erb file (located in views/foo.erb), as shown in Listing 4. While this file seems, on the surface, to be no different from any other ERb (embedded Ruby) document, it contains the telltale sign of a split test:

```
<% ab_test("click_text", "Click on me!",
 ➥"Click here!") do |click_text| %>
    <%= click_text %>
<% end %>
```

The ab_test method, loaded by the line:

```
helpers Split::Helper
```

in linksplit.rb does several things at once: it defines a split test (the first parameter), and it provides the two alternatives that you are interested in testing. In this example, you can see that you're testing whether "Click on me!" (the control) converts more frequently than "Click here!" (the experiment). You then pass a block to the ab_test method. The block can contain any text you want, and it can be as long or short as you want. The key thing to realize is that the block parameter (`click_text` in this case) will contain one of the two text strings.

The above split test, thus, will compare the efficacy of two different links. But, you easily could switch the class of an HTML tag (thus giving different styles, including colors and fonts), a different location, the addition of a picture or anything else.

Once this is in place, the Split gem will produce an experiment, displaying one of these text strings to each of your users. Typically, you'll want to show them equally. There are ways to change the ratios, so that you're showing your experimental text to only a small proportion of your users.

However, showing these different texts isn't quite enough. You also need to be able to tell Split when visitors have "converted"—that is, when they have achieved the goal that you set out. In this particular case, you want to know which text is more effective at getting users to click on the link. So, you should report a conversion back to Split when the

Listing 4. views/foo.erb

```
<html>
  <head>
    <title>Foo</title>
  </head>
  <body>
    <h1>Foo</h1>
    <p>
      <a href="/bar">
        <% ab_test("click_text", "Click on me!",
         ➥"Click here!") do |click_text| %>
          <%= click_text %>
        <% end %>
      </a>
    </p>
  </body>
</html>
```

user has clicked on the link. In other words, in the Sinatra application's handler for the "/bar" method, you invoke the "finished" method, passing it the name of the experiment you want to indicate was finished:

```
finished('click_text')
```

Once you have put a split test in place, you can sit back and wait for users to come to your site. Some will get the first test, and some will get the second. How do you know which was more effective? By using the Split dashboard, which you connected to /split when you set up routing for Rack. It will show the percentage of users who responded to each of your experimental texts, so you can see which was more effective. Split also will tell you the confidence that it has in the results of this test. As a general rule, the more people you test, the more confident you can be in the results. But, statistics also show that even a (surprisingly) small sample size can give you interesting and meaningful results.

The dashboard is useful in several ways. First, it allows you to look at your various experiments, seeing how many people are viewing each of your experimental texts, and how many did and didn't complete the test.

It also allows you, after examining the data, to use one of the two experimental texts permanently. This is useful in several ways. It means that a nonprogrammer can control and decide on each text. But, it also means that you don't have to go back into the code and remove your experiment when it is over. You can take some time, letting the Split system handle it for you. You also can reset the statistics, allowing you to try new ideas or throw out old ones.

Conclusion

Split testing is a powerful tool for helping ensure that your goals are met—whether those goals are selling something on-line or just getting people to sign up for your mailing list. The Split gem works with any Rack-powered Web site, and it makes it extremely easy to implement and then act on your optimization experiments. I encourage you to try some experiments on your own Web applications and see if you can find ways to get your users to convert into customers. If so, congratulations! You've discovered that modern, on-line marketing requires an understanding of programming as much as an understanding of the customer.■

Reuven M. Lerner, a longtime Web developer, consultant and trainer, is completing his PhD in learning sciences at Northwestern University. You can learn about his on-line programming courses, subscribe to his newsletter or contact him at http://lerner.co.il.

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

Resources

The Split gem described here is at **https://github.com/andrew/split**. As indicated in this article, Split works with any Rack-based Web application, which basically means Ruby on Rails and Sinatra.

You can read more about Sinatra at **http://sinatrarb.com**.

Patrick Mackenzie, the author of the A/Bingo gem, has written extensively on the subject of conversion optimization. You can read his articles on the subject at **http://www.kalzumeus.com**. In particular, look for the "conversion optimization" heading on his "greatest hits" page.

The 12th Annual Southern California Linux Expo

SCALE 12x

Keynote by Brendan Gregg: What Linux Can Learn from Solaris Performance, and Vice Versa.

http://www.socallinuxexpo.org
Use Promo Code LJAD for a 30% discount on admission to SCALE

February 21-23, 2014
Hilton Hotel @ LAX
Los Angeles, CA

DAVE TAYLOR

Framing Images with ImageMagick

For the final article in his series on ImageMagick, Dave shows how to add a fancy 3-D frame to images, resize them to fit a certain size and add a very attractive caption, all in a quick, succinct script.

I've come to the end of my journey with ImageMagick, and in this article, I show some really slick techniques for creating attractive frames around images, and then I'm done. I'm not done with the column—don't get too excited—but with this topic. Which means, yes, you need to communicate with me on ideas and suggestions for future columns (send e-mail via http://www.linuxjournal.com/contact).

Right. Now let's proceed.

For the past few months, I've dug into the many amazing capabilities of the ImageMagick command-line image manipulation suite, open-source software you can download from http://www.imagemagick.org.

I've described analyzing, resizing and even adding watermarks to existing images in bulk. All are useful tasks, particularly when you can apply the transformation to a set of images or even a folder (oops, sorry, "directory") full of images too.

This time, I'm first going to look at simply pushing the edges of the image with empty space, then I explore some of the slick capabilities of ImageMagick's `convert` command. The basic image file I'm using is shown in Figure 1.

The image you're seeing, by the way, is the "Submarine" Spitfire MK XVI fighter plane from WWII. This particular image was taken in 2006 and is of a carefully restored replica, featured on Wikipedia UK.

The most basic form of adding a

Figure 1. The Base Image for Manipulation

border is to specify the size of the border in horizontal and vertical pixels and the border color. This is done like so:

```
$ convert spitfire.jpg -bordercolor black -border 10x10
➥spitfire-with-border.jpg
```

This creates a new version of the spitfire.jpg file with a black 10-pixel wide and 10-pixel high frame, as shown in Figure 2.

The only problem with what I've done here is that the 10px frame would be barely visible in a huge image, and it would overpower a tiny thumbnail. ImageMagick's got that covered though, because in addition to specifying pixel size for a border with `-border 10x10`, you also can specify `-border 10%x10%`. Neat!

But really, that is one boring frame, so let's do something more interesting and switch from the `-border` parameter to the `-frame` parameter.

For example, here's a nice framing effect:

```
$ convert spitfire.jpg -frame 25x25+5+5 spitfire-3d.jpg
```

Figure 2. A black 10px frame has been added.

Figure 3. A 3-D frame greatly improves the image presentation.

The results are shown in Figure 3—very nice.

Want to change the color of the frame from the default gray? Use `-mattecolor` followed by a color name. Try using `-mattecolor DarkBlue` as a parameter.

One more interesting combination lets you add a frame and a text caption. Name the file smartly, and you can use its name as the caption automatically too.

Here's a considerably more complicated example:

```
$ convert spitfire.jpg -mattecolor grey -background grey
➥-frame 20x20+0+3 -gravity South -splice 0x15 -pointsize
➥33 -annotate 0x0 "Spitfire MK XVI" -frame 6x6+3+0
➥spitfire-frame-caption.jpg
```

You've seen much of this already, but here's what's new: `-gravity`, which indicates where on the image the subsequent elements should be positioned; `-splice`, which adds the current background color into the image, making space for the label; `-pointsize`, which specifies using 33pt type, not the default (18pt, I believe); and `-annotate`, which as

Figure 4. Image with 3-D Frame and Label Too

you can see takes the legend desired.

Then, there's a second frame to ensure that there's a beveled edge on the top of the area with the text and, finally, the output. The result is shown in Figure 4.

Confusing? Yeah, this stuff confuses me too, truth be told, but the good thing about ImageMagick is that tons of examples are available on-line that show all sorts of complicated transformations, and you can tap into those to get the formula you need for your own work.

With that last example in mind, let's write a script that creates a new version of every .jpg file it finds in a given directory that includes the name of the file too:

```
#!/bin/sh
for name in *jpg
do
  newname=$(echo $name | sed 's/.jpg/-new.jpg/')
  convert $name -mattecolor grey -background grey -frame
   ➥20x20+0+3 -gravity South -splice 0x15 -pointsize 33
   ➥-annotate 0x0 "$name" -frame 6x6+3+0 $newname
done
exit 0
```

That's pretty easy, but the resulting label isn't very attractive. What if the filenames were created with the intention of making useful and informative labels? So instead of "spitfire.jpg", it could be "Spitfire-MK-XVI.jpg"? And, how about removing the filename suffix and any dashes or underscores in the filename as well? That's easily done:

```
label=$(echo $name | sed 's/.jpg//;s/-/ /g;s/_/ /g')
```

So, given a filename like "Spitfire-Mk-XVI.jpg", the resulting image label will be the far more visually pleasing "Spitfire Mk XVI". Now, what else could you do with this sort of script? Well, for one thing, you can combine a few months' worth of ideas and have a script that first normalizes the size of the image if it's bigger than, say, 800 pixels in width, then applies these transforms. That code snippet would be:

```
width=$(identify $name | cut -d\   -f3 | cut -dx -f2)
if [ $width -gt 800 ] ; then
  smaller=$(echo $name | sed 's/.jpg/-800.jpg/')
  convert $name -resize 800 $smaller
  name=$smaller
fi
```

Let's use the ImageMagick `identify` command to ascertain the width of the current image (with some fiddling to extract just the datum needed), then test to see if it's greater than 800. If it is, then you use `convert` to resize it to exactly 800 pixels wide, knowing that the program

automatically will reduce the height proportionally at the same time. The challenge here is source integrity. As with most scripts, you never want to touch the original file. It's not only a smart policy, it's also a life-saver, because it's quite easy to mess up a script when you're developing it, and this way, you're sure that it's not completely destructive.

In the previous code block, notice that yet another temp version of the file is created in the `$smaller` variable, a version that has the ".jpg" replaced by "-800.jpg" to differentiate itself. Then, that's the one to which the caption is added, producing yet another file called "-new.jpg". At that point, you really won't want three versions of your file, so the last step is to remove the `$smaller` interim version at the tail end.

All told, here's the resize-to-800px and add-filename-based-caption script:

```
#!/bin/sh
# Caption images, reducing to max 800px wide as needed

for name in *.jpg
do
  newname=$(echo $name | sed 's/.jpg/-new.jpg/')
  # first, let's shrink it if it's more than 800px wide
  width=$(identify $name | cut -d\   -f3 | cut -dx -f2)
  if [ $width -gt 800 ] ; then
    smaller=$(echo $name | sed 's/.jpg/-800.jpg/')
    convert $name -resize 800 $smaller
    name=$smaller  # caption smaller image, not larger
    echo "(reduced original image $name to 800px wide)"
  fi
  echo "Adding caption to $name, saved as $newname"
  convert $name -mattecolor grey -background grey
   ➥-frame 20x20+0+3 -gravity South -splice 0x15
   ➥-pointsize 33 -annotate 0x0 "$name" -frame 6x6+3+0 $newname
   ➥rm $smaller # we don't need it anymore
done
exit 0
```

See how that all fits together? It's neat and succinct, thanks to the amazing powers of the ImageMagick suite. And in this case, I'm using only `identify` to ascertain width and `convert` to reduce, as needed, and add the annotated frame.

That's it for the journey into ImageMagick. Next month, I'll delve into some other interesting scripting project. Have an idea for something cool? Drop me a note here at the magazine!■

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.

KYLE RANKIN

Own Your DNS Data

Why make it easy for people to capture your complete Internet browsing history?

I honestly think most people simply are unaware of how much personal data they leak on a daily basis as they use their computers. Even if they have some inkling along those lines, I still imagine many think of the data they leak only in terms of individual facts, such as their name or where they ate lunch. What many people don't realize is how revealing all of those individual, innocent facts are when they are combined, filtered and analyzed.

Cell-phone metadata (who you called, who called you, the length of the call and what time the call happened) falls under this category, as do all of the search queries you enter on the Internet (more on how to secure that in a future column).

For this column, I discuss a common but often overlooked source of data that is far too revealing: your DNS data. You see, although you may give an awful lot of personal marketing data to Google with every search query you type, that still doesn't capture all of the sites you visit outside Google searches either directly, via RSS readers or via links your friends send you. That's why the implementation of Google's free DNS service on 8.8.8.8 and 8.8.4.4 is so genius—search queries are revealing, but when you capture all of someone's DNS traffic, you get the complete picture of every site they visit on the Internet and beyond that, even every non-Web service (e-mail, FTP, P2P traffic and VoIP), provided that the service uses hostnames instead of IP addresses.

Let me back up a bit. DNS is one of the core services that runs on the Internet, and its job is to convert a hostname, like www.linuxjournal.com, into an IP address, such as 76.74.252.198. Without DNS, the Internet as we know it today would cease to function, because basically every site we visit in a Web browser, and indeed, just about every service we use on the Internet, we get to via its hostname and not its IP. That said,

the only way we actually can reach a host on the Internet is via its IP address, so when you decide to visit a site, its hostname is converted into an IP address to which your browser then opens up a connection. Note that via DNS caching and TTL (Time To Live) settings, you may not have to send out a DNS query every time you visit a site. All the same, these days TTLs are short enough (often ranging between one minute to an hour or two—www.linuxjournal.com's TTL is 30 minutes) that if I captured all your DNS traffic for a day, I'd be able to tell you every Web site you visited along with the first time that day you visited it. If the TTL is short enough, I probably could tell you every time you went there.

Most people tend to use whatever DNS servers they have been provided. On a corporate network, you are likely to get a set of DNS servers over DHCP when you connect to the network. This is important because many corporate networks have internal resources and internal hostnames that you would be able to resolve only if you talked to an internal name server.

Although many people assume very little privacy at work, home is a different matter. At home, you are most likely to use the DNS servers your ISP provided you, while others use Google's DNS servers because the IPs are easy to remember. This means even if others can't intercept your traffic (maybe you are sending it through a VPN, or maybe that kind of line tapping simply requires more legal standing), if they can get access to your DNS logs (I could see some arguing that this qualifies as metadata), they would have a fairly complete view of all the sites you visit without your ever knowing.

This is not just valuable data from a surveillance standpoint, or a privacy standpoint, but also from a marketing standpoint. Even if you may be fine with the government knowing what porn sites you browse, where you shop, where you get your news and what e-mail provider you use, you may not want a marketing firm to have that data.

Recursive DNS vs. DNS Caching

The key to owning your DNS data and keeping it private is to run your own DNS server and use it for all of your outbound DNS queries. Although many people already run some sort of DNS caching programs, such as dnsmasq to speed up DNS queries, what you want isn't simply a DNS cache, but something that can function as a recursive DNS resolver. In the case of dnsmasq, it is configured to use upstream recursive DNS servers to do all of the DNS heavy lifting (the documentation recommends you use whatever DNS servers you currently have in /etc/resolv.conf). Thus, all of your DNS queries for

www.linuxjournal.com go to your DNS caching software and then are directed to, for instance, your ISP's DNS servers before they do the traditional recursive DNS procedure of starting at root name servers, then going to com, then finally to the name servers for linuxjournal.com. So, all of your queries still get logged at the external recursive DNS server.

What you want is a local DNS service that can do the complete recursive DNS query for you. In the case of a request for www.linuxjournal.com, it would communicate with the root, com and linuxjournal.com name servers directly without an intermediary and ultimately cache the results like any other DNS caching server. For outside parties to capture all of your DNS logs, they either would have to compromise your local, personal DNS server on your home network, set up a tap to collect all of your Internet traffic or set up a tap at all the root name servers. All three of these options are either illegal or require substantial court oversight.

Install and Configure Your DNS Server

So, even when you rule out pure DNS caching software, there still are a number of different DNS servers you can choose from, including BIND, djbdns and unbound, among others. I personally have the most experience with BIND, so that's what I prefer, but any of those would do the job. The nice thing about BIND, particularly in the case of the Debian and Ubuntu packages, is that all you need to do is run:

```
$ sudo apt-get install bind9
```

and after the software installs, BIND automatically is configured to act as a local recursive DNS server for your internal network. The procedure also would be the same if you were to set this up on a spare Raspberry Pi running the Raspbian distribution. On other Linux distributions, the package may just be called bind.

If BIND isn't automatically configured as a local recursive DNS server on your particular Linux distribution and doesn't appear to work out of the box, just locate the options section of your BIND config (often in /etc/bind/named.conf, /etc/bind/named.conf.options or /etc/named/named.conf, depending on the distribution), and if you can't seem to perform recursive queries, add the following line under the `options{}` section:

```
options {
  allow-recursion { 10/8; 172.16/12; 192.168/16; 127.0.0.1; };
. . .
}
```

This change allows any hosts on those networks (internal RFC1918 IP addresses) to perform recursive queries on your name server without allowing the world to do so.

Once you have BIND installed, you'll want to test it. If you installed BIND on your local machine, you could test this out with the `dig` command:

```
$ dig @localhost www.linuxjournal.com

; <<>> DiG 9.8.1-P1 <<>> @localhost www.linuxjournal.com

; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17485

;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0


;; QUESTION SECTION:

;www.linuxjournal.com.          IN    A


;; ANSWER SECTION:

www.linuxjournal.com.    1800    IN    A    76.74.252.198


;; AUTHORITY SECTION:

linuxjournal.com.        30479   IN    NS   ns66.domaincontrol.com.

linuxjournal.com.        30479   IN    NS   ns65.domaincontrol.com.


;; Query time: 31 msec

;; SERVER: 127.0.0.1#53(127.0.0.1)

;; WHEN: Wed Dec 18 09:37:13 2013

;; MSG SIZE  rcvd: 106
```

Otherwise, replace localhost with the IP address of your Raspberry Pi or whatever machine on which you installed BIND. To use this name server for all of your requests, update your /etc/resolv.conf file so that it contains:

```
nameserver 127.0.0.1
```

as its only `nameserver` line. Replace 127.0.0.1 with the IP address of the machine you installed BIND on if it isn't on the same machine. On some modern distributions, there are external tools that tweak /etc/resolv.conf for you, so in those cases, you may have to edit your dhclient.conf or other network configuration files so that you can override the provided list of name servers. Once you do that though, really that's all there is to it. Now you can use DNS knowing that all of your DNS search data sits on a machine under your control.■

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

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

SHAWN POWERS

BirdCam, Round Two

BirdCam is back—now with HD video and archives.

In the October 2013 issue, I described the hardware and software I used to create my "BirdTopia Monitoring Station", more commonly called BirdCam. If you've been visiting BirdCam recently, which a surprising number of folks have been doing, you'll notice quite a few changes (Figure 1). In this article, I describe the upgrades, the changes and some of the challenges along the way. If you like fun projects like these involving

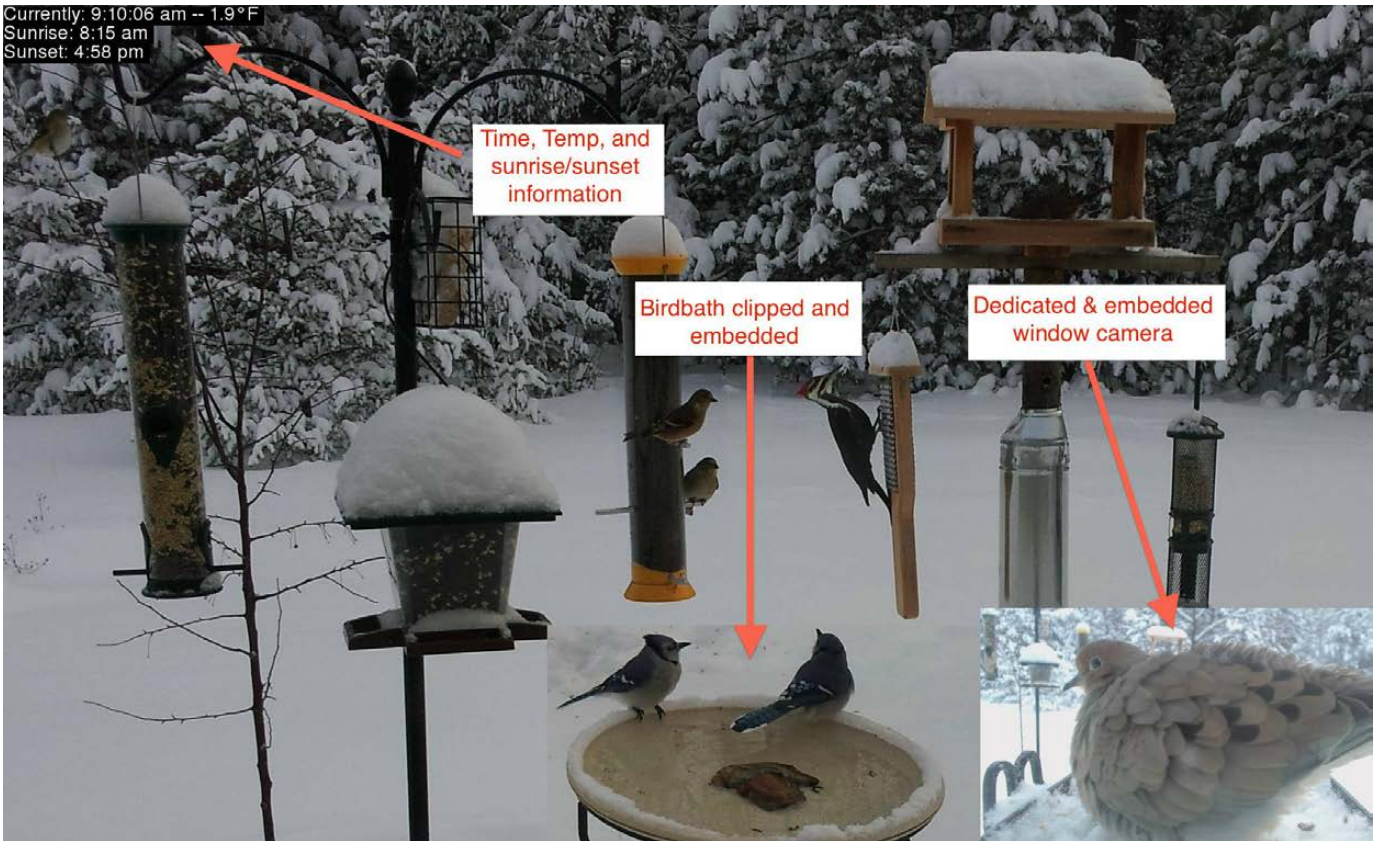


Figure 1. BirdCam has changed a lot. Here, the biggest changes are highlighted. Also, *look at all those birds!*

Linux, please read on and join in my birdy obsession!

Slicing and Dicing

One of the first changes I wanted to make to BirdCam was to zoom in a bit on the feeders. Yes, the enormous photo provided by the Galaxy S2 phone mounted in the window is nice, but for displaying on a computer screen (or HDTV, as I'll talk about later), a 1920x1080-size snapshot is really ideal. Unfortunately, when I crop the photo, it leaves out the birdbath. Because I spent the money on a heated birdbath this winter, I didn't want to miss out on any candid water shots. You can see in Figure 2 how I planned to zoom in on the bird feeders and then relocate the birdbath onto what was left of the photo. Although it took a bit of trial

Figure 2. My old cell phone takes really high-resolution photos. I was able to clip out the birdbath and overlay it nicely to get a closeup of the feeders.

and error, the code for doing this was remarkably easy. I used the `convert` program from the ImageMagick suite. It might be possible to include the crop and relocate into a single command, but I just created a temp file and then overlayed that temp file later on:

```
convert /dev/shm/original.jpg -crop 640x360+1800+1425 \
        /dev/shm/birdbath.jpg
convert /dev/shm/original.jpg -crop 1920x1080+220+130 \
        /dev/shm/birdbath.jpg -gravity southeast
-composite \
        /dev/shm/final.jpg
```

In the code snippet above, I crop out the small birdbath photo from the original camera photo and save it as birdbath.jpg. Then, with another `convert` command, I crop the original photo to that 1080p size I mentioned earlier and overlay the birdbath onto the photo with the `-composite` flag. In this little example, I use the `-gravity` flag to put the birdbath in the corner. You can be more precise with the `-geometry` flag, which you'll see in my final script (see Resources).

Time and Temp

The original BirdCam article showed how to add a timestamp to the top of the photo, but I didn't mention how I got the temperature. I've since added sunrise and sunset information to the annotation, and made it a little more readable. (Figure 1 shows the annotation in the upper-left corner.)

Although the method for overlaying the information isn't much different (you can check it out in the final script—see Resources for the link), getting the temperature and sunrise/sunset information was challenging. To get the current temperature, I use a little command-line program called weather-util, available in most distro repositories. The program actually gives much more information than is required, so to extract just the temperature, you need to do a little grep-fu:

```
weather-util -i KPLN | grep Temperature | awk '{print $2}'
```

This requires a bit of explanation. You need to figure out what your four-letter weather ID is. The best place to find that is at **http://weather.noaa.gov**. Find the closest location to you, and then look in the URL at the top for the four-letter code (usually an airport). For example, the Pellston Regional Airport is my closest, so my ID is KPLN. If you're thinking it would be better to get the actual temperature from my backyard, I agree. In fact, I'll be working on setting up my own weather station in the

months to come for that very reason.

The rest of the code does two things. Grepping for "Temperature" returns the line of the weather-util results containing the temperature, and then the awk command extracts just the Fahrenheit temperature from that line. I set up a cron job to save the temperature into a text file every few minutes, and I use that file to annotate the BirdCam photo. The sunrise/sunset information is similar, but for that, I use the Yahoo weather information. You'll need your WOEID information, which again is available in the URL when you go to **http://weather.yahoo.com** and enter your location information. For example, my weather URL is **http://weather.yahoo.com/united-states/michigan/indian-river-2426936**, so my WOEID is 2426936. The rest is fairly easy using more grep-fu. Here's the code for sunrise:

```
echo `l=2426936;curl -s http://weather.yahooapis.com/
➥forecastrss?w=$l|grep astronomy| awk -F\" '{print $2}'`
```

And the code for sunset:

```
echo `l=2426936;curl -s http://weather.yahooapis.com/
➥forecastrss?w=$l|grep astronomy| awk -F\" '{print $4}'`
```

Much like the temperature, I have a cron job that stores these values into a text file that gets used during the annotation of the final BirdCam graphic. Once I get a local weather station from which I can pull information, I might add wind speed and such. For now, I think it's a useful bit of information.

The WindowCam

If you've been following my blog or my Twitter feed, you may have seen a few different iterations of WindowCam. I tried weatherproofing a USB Webcam (Figure 3), which worked until the wind packed snow into the "hood" I created with a Dixie cup. For a long time, I leaned an old iPhone against the window and just extracted a photo to embed into the final BirdCam photo. The iPhone's resolution was only 640x480, so it was fairly simple to embed that image (downloaded directly from the iPhone) using `convert`.

But of course, I wanted more. My current window camera is a Logitech C-920 USB Webcam connected to a computer running Linux. At first I connected it to a Raspberry Pi, but the RPi couldn't do more than five frames per second. I wanted 15 frames per second, full HD, so I'm using a full-blown computer. See, not only do I want to embed the window camera into BirdCam, but I also want to have

Figure 3. My Dixie-cup weatherproofing did an okay job, until the snow flew. Now it's inside looking through the window.

an archive video of the birds that visit my feeder every day.

The Magic of Motion

Much of the feedback I got about my original BirdCam article included "why didn't you use motion?" Quite simply, I didn't have a computer nearby I could attach to a camera. Now I do. Like I mentioned earlier, I started with a Raspberry Pi. If you have simple needs, or a low-resolution camera, the RPi might be plenty of horsepower. Since the procedures are the same regardless of what computer you're using, just pick whatever makes sense for you.

Motion is a frustratingly powerful program. By that, I mean it will do so many things, it's often difficult to know where to begin. Thankfully, the default configuration file is commented really well, and very few tweaks are required to get things going. For my WindowCam, I have motion do three things:

1. Save a snapshot every second. This is for BirdCam, so that it can embed a thumbnail of WindowCam in the corner. (See Figure 1, with the huge Mourning

Dove in the corner.)

2. When motion is detected, save full-resolution images, 15 frames every second. Note, this is a lot of photos, especially with a busy WindowCam. I easily get 100,000 photos or more a day.

3. Record videos of motion detected. I actually don't do this anymore, because it's hard to deal with hundreds of short videos. I just take the photos saved in step two and create a daily archival video (more on that later).

Motion will do much, much more. It will support multiple cameras. It can handle IP cameras. It can fire off commands when motion is detected (turn on lights, or alarms, or text you and so on). But I'll leave those things for BirdCam 3.0! Let's configure motion.

Motion: Choosing a USB Camera

You want a camera that is UVC (USB Video-Compatible). It would be nice if cameras had a nice "USB Video-Compatible!" sticker on the box, but sadly, they never do. The good news is that many cameras are compatible, even if they don't brag about it. You can google for a specific camera model to see if other folks have used it, or you can check an on-line database before buying. See **http://www.ideasonboard.org/uvc** for a list of devices known to work, but if you're getting an off-brand model, you might just have to try it to find out. Interesting to note, if you see a "Certified for Windows Vista" sticker, the camera is UVC-compliant, as that's a requirement for Windows Vista certification.

I chose the Logitech C920 USB camera, which supports 1920x1080 (1080p) video at 30fps. By default, the camera autofocuses, which sounds like a great idea. Unfortunately, I found the camera almost never focuses on what I want, especially if it's very close to the camera, which the birds tend to be. If you have issues with autofocusing, you might want to try tweaking the camera on the command-line. I have the following added to my startup scripts to turn off autofocus, and I set the manual focus to just outside my window:

```
uvcdynctrl -s "Focus, Auto" 0
uvcdynctrl -s "Focus (absolute)" 25
```

You'll need to play around with the focus value to get it just right, but if your camera supports manual focus, you should be able to tweak it

for sharp images that don't need to autofocus every time a bird lands.

Motion: Configuring the Config

Installing motion is usually nothing more than finding it in your repositories and installing. You'll probably have to edit a file in order to get it to start at boot (like the /etc/default/motion file in Ubuntu), but it's not too painful. It's important to understand a few things about the default config:

1. Usually the "root" folder motion uses by default is in the /tmp folder. This could cause issues on your system if you just leave it alone and let it run—it could fill your /tmp partition and make your system break.

2. Motion uses /dev/video0 by default. This is good, as it most likely is the device name of your USB camera. If you have a funky camera though, you might need to change the device setting.

3. When you make a change to the configuration file, you need to restart motion for it to take effect. That generally means running `sudo service motion restart` or something similar.

Next, open the config file (as root), and make some changes. I'll name the directive to search for below, then talk a bit about the settings. These configuration directives should all be in your /etc/motion/motion.conf file. You'll just need to modify these values:

- `width`: I use `width 1280` for my camera. Even though it supports 1080p, I actually only record at 720p (1280x720). It saves space and still produces HD quality content.

- `height`: in order to get the proper aspect ratio, I set this to `height 720`. Note, my camera will take photos only with the 16x9 aspect ratio, even if I set it to something else.

- `framerate`: this is how many frames are captured per second. I use `framerate 15`, which provides fairly smooth motion while saving the most disk space. With the Raspberry Pi, I could get around 5fps with a 640x480 camera before the CPU load maxed out.

- `threshold`: this is how many pixels must change before the program triggers a motion event.

I left the default, then tweaked it later. My setting currently is `threshold 2500`.

- `minimum_motion_frames`: by default, this is set to 1. I found that occasional camera glitches would fire off a motion event, even if nothing moved. Setting this to 2 or 3 will make sure there's actual motion and not a camera glitch. Mine is `minimum_motion_frames 2`.

- `quality`: the default quality of 75 is probably fine, but I really wanted a sharp image, especially since I spent almost $100 on a camera. I set this to `quality 95`.

- `ffmpeg_cap_new`: this is the setting that will tell motion to record videos. I had this "on" at first (it defaults to "off"), but I ended up with hundreds of short little videos. If you turn it on, fiddle with the other ffmpeg options as well.

- `snapshot_interval`: this will capture a single image every *N* seconds. I have it set to `snapshot_interval 1` and use the image for embedding into BirdCam.

- `target_dir`: this setting is very important. This directory will be the "root" directory of all files created by motion. Both photo and video files are kept in this directory, so although using the ramdisk might sound like a good idea, it probably will fill up quickly.

- `snapshot_filename`: this will be the periodic snapshot filename. I set it to a single filename so that it is overwritten constantly, but you can leave it with the datestring stuff so it will keep all the files. This filename is relative to `target_dir`, so if you try to set an absolute path, it still will be relative to `target_dir`. You can, however, add a directory name. On my system, `target_dir` is /home/birds, and I have a symbolic link inside /home/birds named ram, which points to a folder inside the /dev/shm ramdisk. I then set `snapshot_filename ram/windowcam`, and it overwrites the windowcam.jpg file in my ramdisk every second. This saves wear and tear on my hard drive.

- `jpeg_filename`: this directive is similar to `snapshot_filename`, but instead, it's where the 15 images per second are stored. This

is one you don't want to redirect to a ramdisk, unless you have an enormous ramdisk. I kept the default string for naming the files. My setting puts the photos in a separate folder, so it looks like this: `jpeg_filename photos/%Y-%m-%d_%H.%M.%S-%q`.

- `movie_filename`: this is what determines the name and location of the ffmpeg videos. I'm not saving ffmpeg videos anymore, so my setting is moot, but this is where you assign folder and name location. Even though mine isn't used anymore, my setting from when I did record video is `movie_filename movies/%Y-%m-%d_%H.%M.%S`.

- `webcam_port`: by default, this is set to 0, which means the Webcam is disabled. On a Raspberry Pi, I don't recommend turning this on, but on my Intel i5-based system, I have it set to port 8081. It creates a live MJPEG stream that can be viewed from a browser or IP camera-viewing software (like from an Android tablet).

- `webcam_maxrate`: this determines frames per second. I have this set to 10, which is fast enough to see motion, but it doesn't tax the server.

- `webcam_localhost`: by default, this is set to "on", which means only the localhost can view the Webcam stream. If you want to view it remotely, even from other computers on your LAN, you'll need to change this to "off".

Although that seems like a huge number of options to fiddle with, many of the defaults will be fine for you. Once you're happy with the settings, type `sudo service motion restart`, and the server should re-read your config file and start doing what you configured it to do.

100,000 Photos?

I probably could talk about the fun things I do with my BirdCam server for two or three more articles. Who knows, maybe I'll visit the topic again. Before I close this chapter though, I want to share a cool thing I do to archive the bird visits for a given day. I mentioned that I end up with more than 100,000 photos a day from detected motion, and quite honestly, looking through that many photos is no fun. So at the end of every day, I create an MP4 video using the images captured. I've tried

a few different tools, but the easiest to configure is mencoder. If you get mencoder installed (it should be in your repositories), you can turn those photos into a movie like this:

```
mencoder "mf:///home/birds/photos/*.jpg" \
  -o /home/birds/archive/`date +"%Y-%m-%d"`-WindowCam.avi -fps 15 \
  -ovc lavc -lavcopts vcodec=mpeg4:mbd=2:trell:vbitrate=7000
```

I run that command from cron every evening after sundown. Then I delete the images, and I'm ready for the next day. What I end up with is a 2–3 hour video every day showing all the bird visits at my window. It might seem like a lame video, but I usually scrub through it the next morning to see if there are any new or exotic birds visiting. The videos are 4–6GB each, so if storage space is an issue for you, it's something to keep in mind—which brings me to the next, and final, cool addition to BirdCam.

The Most Boring YouTube Channel Ever

I want to be able to watch my archived video from anywhere. It's possible to stream my local media via Plex or something direct like that, but it would mean I had to connect to my home network in order to see the birds. Now that YouTube has removed the 10 or 15 minute limit on YouTube accounts, it means I can upload a three-hour archive video of birds at my window to YouTube, and have them store and display my video in the cloud—and for free!

Because this is Linux, I wanted to find a way to script the uploading and naming of my daily video. Thankfully, there's a really cool program called ytu, which stands for YouTube Uploader. It's simple, but very powerful. Download it at **http://tasvideos.org/YoutubeUploader.html**.

The most difficult part of running ytu is that you need a developer key. You can become a developer and get a developer key at **https://code.google.com/apis/youtube/dashboard/gwt/index.html**. It's not difficult, but the URLs keep changing, so I can't give you a more specific link. Once you have your developer key (a long string of letters and numbers), you can fill out the credentials.txt file with:

- Google e-mail address.
- Google password in plain text.
- Developer key.

Make sure only the account executing the ytu binary has access to

the file, as your password is stored in plain text. If this security problem isn't acceptable, you can, of course, upload videos to YouTube manually. I actually have a separate Google account for uploading BirdCam videos, so I'm not as concerned with the security.

Once the credentials file is set, you invoke ytu from the command line, or in my case, from the same cron job that creates the video. After the video is created, I immediately start ytu to upload it to YouTube. After several hours of uploading and processing, the video is ready to watch. If you'd like to see my archived videos, just surf over to http://snar.co/windowcam and see all the past videos.

It Doesn't Have to Be Birds!

Obviously, I have a mild obsession with birdwatching. We all have our quirks. With this article, and my last BirdCam article, hopefully you can figure out a way to adapt the power of Linux to scratch that video-capturing itch you have. If you do end up with something similar, or even something drastically different (dogcam on the local fire hydrant?), I'd love to hear about it! Well, maybe not the fire-hydrant cam...that's a little too strange, even for me.■

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.

Resources

BirdCam: **http://birds.brainofshawn.com**

BirdCam Script: **http://snar.co/birdcamscript**

WindowCam Archive Channel: **http://snar.co/windowcam**

YTU Program: **http://tasvideos.org/YoutubeUploader.html**

YouTube Developer Dashboard: **https://code.google.com/apis/youtube/dashboard/gwt/index.html**

O'REILLY®

Fluent CONFERENCE

The Web Platform

San Francisco, CA | March 11–13, 2014

"Fluent is one of the best informational and networking events in technology. From the keynote presentations to the after-hours meet and greets, every event was well thought out and executed. I will be back next year."

–**Aaron Biggs,** *University of Oklahoma*

Does Your Future Depend on the Web Platform?

You bet it does. If you're building web applications, designing for mobile devices, or working with the web's evolving infrastructure, you need to keep up with the enormous proliferation of web technologies. At Fluent, you'll find workshops, tutorials, and sessions on all aspects of the Web Platform, including JavaScript, HTML5, WebGL, CSS3, mobile APIs, Node.js, AngularJS, ECMAScript 6, and more. We're even partnering with WebPlatform.org and hosting a Document Sprint. It's everything you need to stay competitive.

Don't miss your chance to be a part of the web's evolution. Build your custom Fluent schedule around exciting talks on topics like:

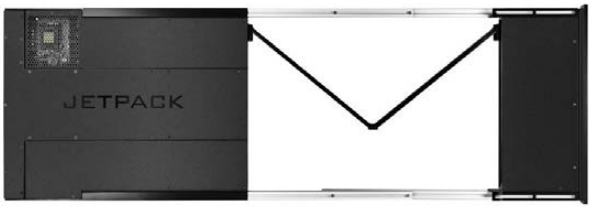
- Front End Frameworks and Libraries
- HTML5, CSS3, and Browser Technologies
- Node.js
- Pure Code and JavaScript
- The Leading Edge
- The Server Side
- Tools, Platforms, and APIs
- User Interface / User Experience
- HTML5 Gaming

Fluent is closer than you think.

Save 20% with code **LINUX20**

fluentconf.com

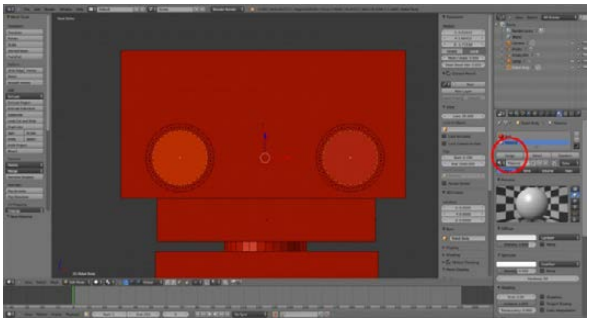
PiixL's Jetpack



The Jetpack, says its maker PiixL, is an open gaming hardware platform designed for gamers who value both performance and interior design. With Jetpack, raw, unprecedented—and hackable—gaming power meets living-room entertainment with a beautiful design. After releasing its EdgeCenter product, PiixL has become an expert at designing TV-centric PC hardware and is now set to break new ground with this latest release. With more than 500 watts of thermal capacity available to house the fastest components, Jetpack promises to be the most powerful machine offered in such a compact package, asserts PiixL. Designed around PiixL's leading-edge cooling architecture, Jetpack uses a new generation of centrifugal fans to stay ultra-quiet at all times while being able to host factory overclocked Core i7 processors, up to one Terabyte of SSD storage, and the latest graphics cards from NVIDIA, including Titan and GTX780s. Jetpack is optimized to complement Linux and Windows, with a special focus on SteamOS, which combines the rock-solid architecture of Linux with a gaming experience built for the big screen.
http://www.piixl.com

Linutop PC

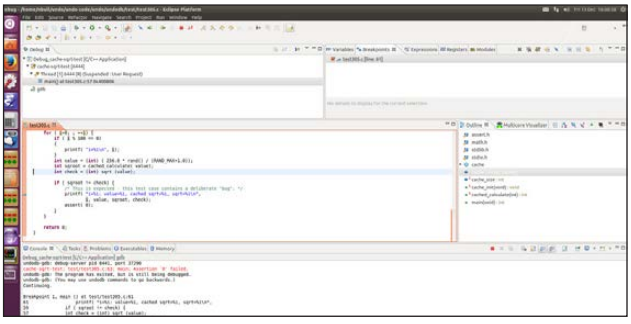
Packing additional punch into a more petite package is the new Intel Atom-based Linutop 5 miniature, energy-efficient PC. Linutop 5 is a ready-to-use small PC, designed to reduce maintenance costs. Compact and powerful with a 1.6GHz fanless Intel ATOM processor, Linutop 5 is designed for applications, such as no-maintenance professional use, public Internet access, office applications and digital signage. The 4GB of internal Flash memory replaces the hard drive, and energy consumption has dropped to a mere 14 Watts, the same as a compact fluorescent light bulb. A new security feature can lock the Linutop Operating System and protect the Flash memory from wearing out by limiting write cycles. Once set up, the Linutop software is protected. In lock mode, Linutop can recover its state at each startup, minimizing maintenance costs. Linutop 5 is powered by the new Linutop OS based on the latest long-term service Ubuntu version 12.04.
http://www.linutop.com



The Hello World Program

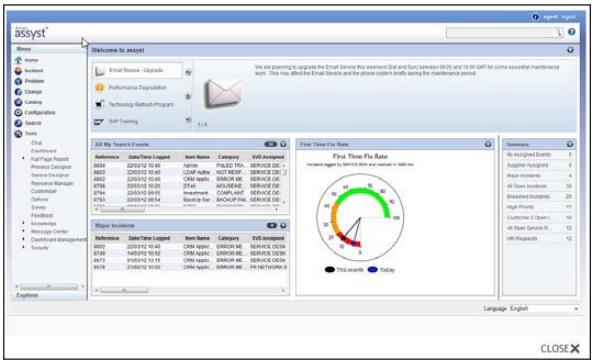
Learning about computer science and programming tends to be dry and not all that accessible to kids. In an effort to make Linux and computer programming more accessible for young people—not to mention really fun—a pair of LA-based brothers have developed The Hello World Program, a kid-friendly educational video puppet show that covers tech topics as diverse as Python, Linux, Web design, video editing, 3-D modeling and animation, graphic design and, of course, puppet making! Hello World is the creative effort of brothers Jared and JR Nielsen who grew up in a rural area and made their own fun by staging puppet shows, making their own toys and shooting short stop-motion videos. "We don't want a boring future, and we think the best way to avoid that is by encouraging and educating the young and old to actively produce their own media", says Jared Nielsen. At the time of this writing, the Nielsen bothers are pitching Hello World on Kickstarter in order to leapfrog it to the next level. "We will definitely be continuing with the project whether or not we reach our funding goal", commented JR Nielsen. "Our time line and distribution may change as a result", he added, "but we are still set to produce the same content regardless of the Kickstarter outcome." A boring future has indeed been preempted.
http://www.thehelloworldprogram.com

Undo Software's UndoDB

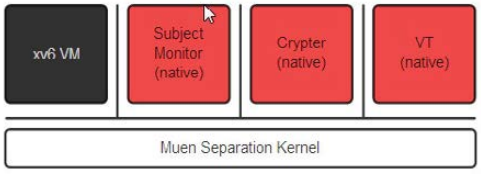


As devices and systems become more capable, the software becomes ever more important and complex. As such, software developers bear an ever-growing burden in terms of both quality and time to market. An updated—and groovy—solution to these issues is version 4.0 of Undo Software's UndoDB, a reversible debugger for Linux. UndoDB allows Linux software developers to record their program's execution and then "wind the tape" back and forth in real time to get a clear picture of their program's execution, significantly reducing the cost of bugs to software vendors. The most notable new features in the new version 4.0 are support for ARM processors and Android Native. Undo also notes that Linux developers have an alternative way to utilize its solution: ARM recently integrated UndoDB into its flagship software development studio, ARM DS-5 Professional Edition.
http://undo-software.com

Axios Systems' assyst

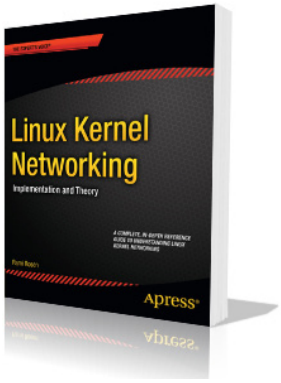


Because this latest update of the assyst IT Service Management (ITSM) solution makes it more akin to social media, the folks at Axios Systems believe that it is now destined to revolutionize the ITSM industry. assyst is a purpose-built solution, designed to transform IT departments from technology-focused cost centers into profitable business-focused customer service teams. Axios says that assyst enables faster, less costly delivery and support of IT services, allowing its clients to offer unparalleled multichannel support. This latest release of assyst now offers guided help, enhanced reporting, enhancements to the assystNET self-service portal and, most notably, IT Resource Performance Management (IT RPM). IT RPM promotes collaboration and innovation within IT, helping to establish IT as a business driver while reducing incoming calls to the service desk. It also allows a business to leverage staff as a valuable asset through rapid capture and transfer of knowledge as an integrated part of the service management processes through the aforementioned social-media tools, namely crowdsourcing, leaderboards and social profiles.
http://www.axiossystems.com



University of Applied Science Rapperswil's Muen Separation Kernel

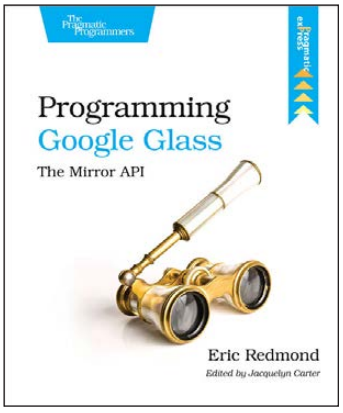
As initiatives like the new Muen Separation Kernel mature, open source is destined to play an ever greater role in the development of safe and secure systems. To this end, the Institute for Internet Technologies & Applications at the University of Applied Science Rapperswil (Switzerland), together with corporate partner AdaCore, announced a preview release of the Muen Kernel, which enforces a strict and robust isolation of components. The isolation shields security-critical functions from vulnerable software running on the same physical system. To achieve the necessary level of trustworthiness, the Muen team used the SPARK language and toolset to prove the absence of runtime errors formally. The Muen developers used SPARK with a zero-footprint runtime, a mode where no runtime environment, and only a minimum of supporting code, is required. The name "Muen" is a Japanese term that means "unrelated" or "without relation", reflecting the main objective for a separation kernel: ensuring the isolation between components.
http://muen.codelabs.ch, http://www.adacore.com



Rami Rosen's *Linux Kernel Networking* (Apress)

Because Linux-kernel networking is a complex topic, you don't want to be bothered with extraneous miscellanea. This is the approach you'll find in Rami Rosen's *Linux Kernel Networking: Implementation and Theory*. Publisher Apress says that readers will not be overloaded with cumbersome line-by-line code walk-throughs not directly related to the topic at hand. Readers will find exactly what they need, with in-depth explanations in each chapter and a quick reference at the end of each chapter. Apress also claims that the title is the only up-to-date reference guide to understanding how networking is implemented in the latest version of the Linux kernel. In years to come, the book is sure to be indispensable, because so many devices now use Linux or operating systems based on Linux, and because Linux is so prevalent in the data center, for example, due to Xen.
http://www.apress.com

Eric Redmond's *Programming Google Glass* (Pragmatic Bookshelf)



If the government frantically regulates a widget before it's even out, you know you gotta' get your hands on one. Today's widget in question is Google Glass, a programmable, wearable Android-powered computer with an optical head-mounted display. The new book *Programming Google Glass: The Mirror API* by Eric Redmond exists to kick-start users' Glassware development. Core topics include exploring interfacing with Glass, developing a Glass application via the Mirror API, tracking a Glass's geolocation, creating rich interactions by responding to user inputs and capturing or serving up user images and videos. This is the book to read for a shortcut to this brave new world. Now, says publisher Pragmatic Bookshelf, is the best time to be an early adopter of a technology that quickly will become more advanced, nuanced and ubiquitous.
http://www.pragprog.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.

A Shining Ruby in Production Environments

Rails, the popular Ruby framework, is straightforward in development but hard to manage. Read on to learn how to set up Ruby hosting and automatically deploy a Rails application.

FABRIZIO SOPPELSA

Even the most beautiful Rails application can lose its elegance if not deployed correctly. Like other Ruby frameworks or languages, such as Sinatra, Rails is based on the Rack interface. This article provides a basic introduction to Rack hosting and Rack-based application deployments.

When Rails first was released in 2005, developers exulted. Finally, a comprehensive open-source framework for Web applications was available, packed with a set of tools making Web development fast, productive and fun. Rails has the reputation of being a "heaven for developers", but despite the many facilities it provides for avoiding typical and repetitive tasks, there is still a weak spot: deployment. Deploying a Rails application is not a smooth matter. Everyone knows that Rails applications will be published on-line one day, but not precisely how.

Platform as a Service (PaaS)

Developers often choose to purchase hosting space as Platform as a Service (for example, Heroku, OpenShift or EngineYard). PaaS is marvelous as it provides a ready-to-use environment containing a full stack of software dependencies. Publishing on a PaaS platform is, as a rule, easy, fast and everything tends to work (almost) immediately. But there are at least two cases when PaaS won't fit your needs: when applications must be kept in the customer's private infrastructure or when applications have superior hardware or software requirements—for instance, when you need a specific software service not supported by your PaaS provider.

In such situations, you must implement custom virtual server configurations and custom deployment procedures. You can deploy Rails applications on servers or on virtual machines. The availability of entire cloud services like Amazon Web Services (AWS), which allow you to create complex infrastructures made of several Web servers, database servers and front-end balancing machines, is hugely growing in popularity. This approach is very flexible, although you must access, install and manage the operating system and the distribution packages, configure the network, activate the services, and so on and so forth. In this article, I describe the Rack-based hosting software requirements and some basic example configurations to implement automated Ruby hosting on a GNU/Linux server.

RVM comes packed with a set of scripts that helps you install and update the Ruby ecosystem.

RVM

First, if you want to host Ruby software, you must install the Ruby platform. You can install Ruby and gems with apt-get or yum. It's easy, but when your application requires specific gem versions or specific interpreter versions, you will face a common problem. How can you satisfy these requests if your GNU/Linux distribution doesn't package those specific versions? Furthermore, how can you maintain multiple Ruby versions in a clean and repeatable manner?

You may think you can just download the Ruby platform and compile it manually. It's guaranteed that you can install the interpreter versions and the gem versions you need. Unfortunately, this is totally inconvenient. This kind of software management makes your configuration hard to update.

There are several solutions for overcoming these common issues. The one I find more reliable for server environments is named Ruby enVironment Manager (RVM). RVM comes packed with a set of scripts that helps you install and update the Ruby ecosystem.

Download RVM by issuing the following command as root:

```
# \curl -L https://get.rvm.io | bash -s stable
```

Despite the fact that it's recommended that you work with RVM using security facilities as sudo, the `rvm` executable must be available in your root $PATH environment, so install it as root. For a multiuser RVM installation, typical for servers, the software is kept by default in the /usr/local/rvm directory, so you can remove the whole distribution safely with an `rm -fr /usr/local/rvm` command.

Before proceeding with the Ruby installation, make sure your system is ready to compile Ruby. Check that you have the `rvm` command available in your PATH (if not, log out and log in again or reload your shell with `bash -s`), and execute the following command:

```
$ sudo rvm requirements
```

RVM will install, through yum or

apt-get, the required packages to compile the Ruby distribution. In this article, I use the stable official Ruby distribution called MRI, Matz Ruby Interpreter (derived from the name of Ruby's creator, Yukihiro Matsumoto).

Now, you'll likely need to add to your future Rubies some basic libraries typically needed by some complex gems or software. Setting up such libraries immediately will guarantee that the Ruby software will never complain that the system libraries are old or incompatible, generating annoying errors. Previously, you would have installed these extra packages via the `rvm pkg install <pkg>` command, but now RVM deprecates this. Instead, simply enable autolibs to delegate to RVM the responsibility to build coherent and not-buggy distributions:

```
$ sudo rvm autolibs enable
```

You finally are ready to provide your environment a full Ruby distribution. For example, let's install the latest stable version of the official MRI interpreter, the 2.0.0 version:

```
$ sudo rvm install 2.0.0
```

If everything goes well, the distribution is available for root and the system users. If not, it's commonly a $PATH problem, so adjust it in the /etc/profile.d, and also to avoid deployment pitfalls, verify that the $GEM_HOME variable is exported to the correct gem path. In practice, if something is not working properly, set the following variables like this:

```
if [ -d "/usr/local/rvm/bin" ] ; then
    PATH="/usr/local/rvm/gems/ruby-2.0.0-p353@global/bin:
➥/usr/local/rvm/bin:$PATH"
    GEM_HOME="/usr/local/rvm/gems/ruby-2.0.0-p353@global"
fi
```

You can list the available Ruby versions with this command:

```
$ rvm list known
```

On a system running multiple Rubies, users and system processes may load other environment versions with a command like this:

```
$ rvm use jruby-1.7.1
```

And set the default system distribution in this way:

```
$ rvm --default use 2.0
```

The Web Server

Ruby on Rails, like Sinatra and many other popular Ruby frameworks

or Domain Specific Languages, is based on an interface named Rack. Rack provides the minimal abstraction possible between Web servers supporting Ruby and Ruby frameworks. Rack is responsible for invoking the main instance of your application as specified in the startup file, config.ru.

So, a Web server hosting Ruby Web applications will have to understand how Rack talks. With a stable and clean Ruby environment, you're ready to build your Web server that is capable of speaking Rack.

With Ruby, you can choose between many Web servers. You may have heard of Mongrel, Unicorn, Thin, Reel or Goliath. For typical Rails deployments, Passenger is one of the most popular choices. It integrates well with Apache and Nginx, so in this example, let's set up an Apache + Passenger configuration.

Passenger Installation

Passenger, developed by Phusion, also formerly known as mod_rails or mod_rack, is a module that allows you to publish Ruby applications in the popular Web server containers Apache or Nginx. Passenger is available as a "community" free edition and as an enterprise release, which includes commercial support and advanced features.

If you chose to install Ruby through packages, Passenger is conveniently available through RPM or DEB repositories, and yum or apt-get will install all the required software.

On an RVM-customized system, to install the free version of Passenger, you need to add the gem through Ruby gems:

```
$ sudo gem install passenger
```

Now you can install the server module (the latest version at the time of this writing is 4.0.33) by executing a script provided by the gem:

```
# passenger-install-apache2-module
```

Let's select Ruby only, and let's skip Python, Node.js and Meteor support. If your system misses software requirements, the script will give you a tip to the exact command line for yum or apt-get to meet those dependencies.

After some compile time, you will be introduced to Passenger configuration with useful and self-explanatory output. Specifically, copy to the directives that load Passenger into Apache in your main Apache configuration file (apache2.conf

If your goal is to host one or more Ruby applications on the same server, you should activate each instance as a virtual host.

or httpd.conf):

```
LoadModule passenger_module
/usr/local/rvm/gems/ruby-2.0.0-p353/gems/passenger-4.0.33/
➥buildout/apache2/mod_passenger.so
PassengerRoot /usr/local/rvm/gems/ruby-2.0.0-p353/gems/
➥passenger-4.0.33
PassengerDefaultRuby /usr/local/rvm/wrappers/ruby-2.0.0-p353/ruby
```

Finally, restart Apache. *Et voilà*, now you can host Ruby Web applications.

Virtual Hosts

If your goal is to host one or more Ruby applications on the same server, you should activate each instance as a virtual host. The most significant directive with Ruby hosting is the DocumentRoot. It's mandatory that it points to the public/ directory in the application's root project directory. The public/ directory is the default public path of a Rails application. So let's say you have a Kolobok application made in Rails, and you have to deploy it to the DNS zone kolobok.example.com on the kolobok.example.com server. Here is an example VirtualHost:

```
<VirtualHost *:80>
      ServerName kolobok.example.com
      DocumentRoot /srv/www/kolobok/public
      <Directory /srv/www/kolobok/public>
         # This relaxes Apache security settings.
         AllowOverride all
         # MultiViews must be turned off.
         Options -MultiViews
      </Directory>
</VirtualHost>
```

Now, if you have put your application in /srv/www/kolobok, and it's well configured (configured and binded to the database and so on), enable the virtual host, reload Apache, and your application is published.

Automating Software Deployments

Ages ago, it was common to deploy Web applications by doing a bulk copy of files via FTP, from the developer's desktop to the server hosting space, or by downloading through Subversion or Git. Although this approach still works for simpler PHP applications, it won't fit more complex projects made using more complex frameworks, such as Rails.

In fact, a Rails application is not made only of the source code files. To make a Rails application ready, you have to download and compile its dependencies as gems (by running bundle), safely manage database access and other configurations, migrate the database (create the database and the schema by executing a list of files containing SQL instructions in the Ruby language), adjust paths for shared content (like images, videos and so on), precompile the assets (that is, optimizing static content, such as JavaScript and CSS), and perform many other steps in a large and complex work flow. You can execute these steps by writing your own scripts, maybe in Ruby or bash, but this task is tedious and wastes your time. You should instead invest your time by writing good tests.

The Ruby community provides several ways to accomplish the whole deploy task, and one very popular method uses Capistrano. Capistrano lets you write a set of "recipes" that will "cook" your application in the production environment. Common tasks executed by Capistrano are: 1) pulling the source code from a git or svn repository; 2) putting it in the right location; 3) checking if a bundle is needed and, if yes, bundling your gems; 4) checking if migrations are required and, if yes, running them; 5) checking if assets precompile is required and, if yes, precompiling; and 6) checking other Rake tasks you have defined and running them in order. If the whole recipe fails, Capistrano will keep the current software release in production; otherwise, it will substitute the latest release with the one you've just deployed. Capistrano is a largely tested and very reliable tool. You definitely can trust it.

Configuring Capistrano

To use Capistrano, you just need to install it through Ruby gems on the system where the deploy will be done (not on the server):

```
$ gem install capistrano
```

When Capistrano is available, you'll have two new binaries in your PATH: capify and cap. With capify, you build your deploy skeleton. So, `cd` to the project directory and type:

```
$ capify .
```

This command creates a file named Capfile and a config/deploy.rb file. Capfile tells Capistrano where the right deploy.rb configuration file is. This is the file that includes your recipes, and typically it's kept in the

project's config/ directory.

Next, verify that Capistrano is installed correctly, and see the many useful tasks it comes with:

```
$ cap -T
cap deploy                # Deploys your project.
cap deploy:check          # Tests deployment dependencies.
cap deploy:cleanup        # Cleans up old releases.
cap deploy:cold           # Deploys and starts a 'cold' application.
cap deploy:create_symlink # Updates the symlink to the most recently
                          # deployed...
cap deploy:migrations     # Deploys and runs pending migrations.
cap deploy:pending        # Displays the commits since your last
                          # deploy.
cap deploy:pending:diff   # Displays the 'diff' since your last
                          # deploy.
cap deploy:rollback       # Rolls back to a previous version and
                          # restarts.
cap deploy:rollback:code  # Rolls back to the previously deployed
                          # version.
cap deploy:setup          # Prepares one or more servers for
                          # deployment.
cap deploy:symlink        # Deprecated API.
cap deploy:update         # Copies your project and updates the
                          # symlink.
cap deploy:update_code    # Copies your project to the remote
                          # servers.
cap deploy:upload         # Copies files to the currently deployed
                          # version.
cap invoke                # Invokes a single command on the remote
                          # servers.
cap link_shared           # Link cake, configuration, themes, upload,
                          # tool
cap shell                 # Begins an interactive Capistrano session.
```

The user that will deploy the application will need valid SSH access to the server (in order to perform remote commands with Capistrano) and write permissions to the directory where the project will be deployed. The directory structure created on the server in this directory allows you to maintain software releases. In the project's document root, Capistrano keeps two directories, one that contains the released software (releases/, by default it keeps the latest ten releases), and another that contains shared or static data (shared/). Moreover, Capistrano manages a symbolic link named current that always points to the most recent successfully deployed release.

In practice, each time Capistrano is invoked to deploy an application, it connects via SSH, creates a temporary release directory named with the current timestamp (for example, releases/20140115120050), and runs the process (pull, bundle, migrate and so on). If it finishes with no errors, as final step, Capistrano links the symlink "current" to releases/20140115120050. Otherwise, it keeps "current" symlinked with the latest directory where the deploy was successful.

So with Capistrano, the system administrator will set the virtual

server DocumentRoot directive to the current directory of the released application version:

```
DocumentRoot /srv/www/kolobok/current/public
```

The Anatomy of a deploy.rb File

A deploy.rb file is virtually made of two parts: one that defines the standard configurations, like the repository server or the path to deploy files physically, and another that includes custom tasks defined by the developer responsible for deploying the application.

Let's deploy the Kolobok application. Open the kolobok/config/deploy.rb file with your favourite editor, delete the example configuration and begin to code it from scratch. A deploy.rb file is programmed in Ruby, so you can use Ruby constructs in your tasks, beyond the Capistrano "keywords".

Let's start by requiring a library:

```
require "bundler/capistrano"
```

This statement orders Capistrano to do the gem bundle each time it's necessary. Good gem files separate required dependency gems in this way:

```
group :test do
  gem 'rspec-rails'
  gem 'capybara'
  gem 'factory_girl_rails'
end

group :production do
  gem 'execjs'
  gem 'therubyracer'
  gem 'coffee-rails', '~> 3.1.1'
end
```

Only the gems common to all environments and included in the :production group are bundled. Gems belonging to :development and :test environments are not. And the first time you deploy your application, a bundle install is executed to bundle all the requirements as specified. The next time you deploy the software, gems are downloaded, compiled or removed only if the Gemfile and the Gemfile.lock have changed. The complete bundle is installed in shared/ and soft-linked into the current instance. By following this approach, less disk space is required.

Then, from Rails 3.1, it's common to release applications with the assets pipeline. The pipeline is active if in config/environments/production.rb the following variable is set to true:

```
config.assets.compile = true
```

If your application will use the pipeline, you need to precompile

it. The Rake task to precompile assets is `bundle exec rake assets:precompile`. To insert this task into your work flow and keep the generated assets pipeline in shared/ and linked into the current release, load the standard assets functionality:

```
load "deploy/assets"
```

After loading the main requirements, specify the application name, the path on the server where it will be deployed, and the user allowed to SSH:

```
set :application, "kolobok"
set :deploy_to, "/srv/www/kolobok"
et :user, "myuser"
```

With Rails > 3, it's recommended to invoke Rake (it's used to do the database migrations and to precompile the assets pipeline) with the correct bundled Rake version in the bundle. So, specify the exact `rake` command:

```
set :rake, 'bundle exec rake'
```

Now it's time to configure the repository from which to pull the project source code:

```
set :scm, :git
set :branch, "master"
set :repository, "git://github.com/myusername/kolobok.git"
```

Finally, set the server names:

```
role :web, "kolobok.example.com"
role :app, "kolobok.example.com"
role :db,  "mydb.example.com", :primary => true
```

`web` is the address of the responding Web server, and `app` is where the application will be deployed. These roles are the same if the application runs on only one host rather than on a cluster. `db` is the address of the database, and `primary => true` means that migrations will be run there.

Now you have a well-defined deploy.rb and the right server configurations. Begin by creating the structure tree (releases/ and static/) on the server, from the desktop host:

```
$ cap deploy:setup
```

Releasing Software

After having set up the project directory on the server, run the first deploy:

```
$ cap deploy:cold
```

The actions performed by Capistrano follow the standard pattern: git checkout, bundle, execute migrations, assets precompile. If everything is fine, your application is finally published as a reliable versioned release, with a current symlink.

The actions performed by Capistrano follow the standard pattern: git checkout, bundle, execute migrations, assets precompile.

Normal deploys (skipping the first Rails app configuration, such as creating the database) will be done in the future by invoking:

```
$ cap deploy
```

If you notice that some errors occurred with the current application in production, you immediately can roll back to the previous release by calling Capistrano like this:

```
$ cap deploy:rollback
```

Easy, reliable and smart, isn't it?

Custom Tasks

When you deploy a more complex application, you'll normally be handling more complex recipes than the standard Capistrano procedure. For example, if you want to publish an application on GitHub and release it open source, you won't put configurations there (like credentials to access databases or session secret tokens). Rather, it's preferable to copy them in shared/ on the server and link them on the fly before modifying the database or performing your tasks.

In Capistrano, you can define hooks to actions to force the tool to execute required actions before or after other actions. It might be useful, for instance, to link a directory where users of kolobok have uploaded files. If you move the current directory to another release path, you might discover that those files are no longer available to users. So, you can define a final task that, after having deployed code, links the shared/uploads into your current release in public/uploads directory. Notice how this can be managed with ease by exploiting the presence of the shared_path and release_path paths variables:

```
desc "Link uploaded directory"
task :link_uploads do
  run "ln -nfs #{shared_path}/uploads
   ➥#{release_path}/public/uploads"
end
```

Finally, another common task to perform is to restart the application instance into the server container.

In case of Passenger, it's enough to `touch` the tmp/restart.txt file. So, you can write:

```
desc "Restart Passenger"
task :restart do
  run "cd #{current_path} && touch tmp/restart.txt"
end
```

You execute these two tasks automatically by hooking them at the end of the deploy flow. So add this extra line just before the tasks definitions:

```
after "deploy:update_code", :link_uploads, :restart
```

Performance Issues?

People often complain of Rails' performance in production environments. This is a tricky topic. Tuning servers and application responsiveness are rather hard tasks that cannot be discussed briefly, so I don't cover them here. To make your application faster, you should involve several technologies and engineering patterns, like setting intermediate caching services, serving static and dynamic content with different server containers and monitoring the application with tools like New Relic to find bottlenecks. After having set up the right environment to host the application, this is the next challenge—optimizing. Happy deploys!■

Fabrizio Soppelsa works as a sysadmin for an Italian provider. His areas of interest include Ruby hosting, automated Ruby deployments, Ruby application performance in production environments, Platform as a Service, and scalable and event-driven Web patterns. He is passionate about espresso coffee and lives in Italy with his cat. He is a Red Hat-certified engineer, and you can follow him on Twitter: @f_soppelsa.

Resources

Rack: **http://rack.github.com**

RVM: **https://rvm.io**

Phusion Passenger: **https://www.phusionpassenger.com**

Capistrano: **https://github.com/capistrano/capistrano**

SIMPLE WAYS TO ADD SECURITY TO WEB DEVELOPMENT

Can't afford to lose time in code refactoring for security? Why not make everything secure in the first place? Read on!

NITISH TIWARI

As a software developer myself, I have seen developers rushing to finish the feature they are assigned to, with little or no consideration for security in the code—no security guidelines, no coding standards, just a mad dash to finish the feature. Next comes the security review, in which the software obviously fails, and then comes the security-hardening phase.

Although trying to improve code's security obviously is a nice thing to do, the time when it commonly is done is often in the final code development phase, and as with the basic nature of software development, changing the code almost always leads the software away from maturity. So, the software that has almost ended its development phase is again pushed to instability during the security-hardening phase. Is this really necessary?

Why can't developers make the code secure in the first place? What can be done to make developers more aware of application security policies, so they are more informed and alert when they develop their next application? In this article, I discuss how developers can do so effectively.

One simple way is to change developers' coding styles and make them write code that is inherently secure. Also, following simple policies related to application security can make a lot of difference. This is sometimes not a very easy thing to do, but if the practices to follow are simple and easy to adopt, it is not very difficult.

Let's look at some security concerns/flaws typically found in software and the corresponding security mechanisms and policies that can be applied to counter them. These mechanisms generally can be implemented in all programming languages and follow the OWASP code development guidelines. But, for the sake of open-source culture, I use PHP as the language for the examples in this article.

SQL Injection

Let's start with the most famous of the lot. It is also one of the most widely used and one of the most simple for unleashing attacks on the Web. What many people don't know, however, is that it's easy to prevent as well. Let's first consider what an SQL injection attack is.

Suppose you have a text box in your application for a user name field. As the user fills it in, you take the data to the back end and fire a query to the

database—something like this:

```
<Input Type = "Text" value ="username" name = "username">
```

```
<?php $username = $_POST['username']; ?>
```

Then, the SQL query:

```
SELECT * FROM table WHERE name = '" + $username + '"
```

A simple way to attack this system would be to type "'" or "'1'='1" in the text box. The resulting database query now will be:

```
SELECT * FROM table WHERE name = '  ' or  '1'='1'
```

As you can see, this condition always is true and when executed, the query will just split out all the rows in the table. This was a simple example, but in real-life scenarios, such attacks are very severe and can take down a whole application in a few seconds, because they are targeted directly at the database.

So, how can you prevent this? Simple logic is that instead of passing the input taken from the front end directly, it should be checked thoroughly and only then sent to the database as a part of the query. Here are the most common and effective ways to do that:

Parameterized Queries: such queries result in exactly the same way as normal SQL queries, but the difference is that here you need to define the SQL code first and then pass the parameters to the query later. So, even if someone tries to attack by passing malicious data to the query, the query searches for the exact match of whatever is sent as input. For example, if someone tries to pass ' or '1=1 as the data, the query will look up the DB for a literal match of the data.

Here is an example of how to write parameterized queries in PHP (see your programming language manual for more about parameterized queries):

```
/* Prepared statement, stage 1: prepare */
if (!($stmt = $mysqli->prepare("INSERT INTO test(id) VALUES (?)"))) {
 echo "Prepare failed: (" . $mysqli->errno . ") " . $mysqli->error;
}

/* Prepared statement, stage 2: bind and execute */
$id = 1;

if (!$stmt->bind_param("i", $id)) {
    echo "Binding parameters failed: (" . $stmt->errno . ") " .
$stmt->error;
}

if (!$stmt->execute()) {
    echo "Execute failed: (" . $stmt->errno . ") " . $stmt->error;
}
```

So, the next time you need to look up the database, use a parameterized query for it. But beware, this approach has a downside as well. In some cases, doing this can harm performance, because parameterized queries need server resources. In situations where an application is performance-critical, there are other ways to counter SQL injection attacks.

Stored procedures: this is another commonly used method for countering SQL injection attacks. It works the same as parameterized queries, the only difference being that the procedure or the method is itself stored in the database and called by the application when required. Here's how to write a stored procedure in PHP for MySQL:

```
/* Create the stored procedure */
if (!$mysqli->query("DROP PROCEDURE IF EXISTS p") ||
    !$mysqli->query("CREATE PROCEDURE p(IN id_val INT)
    ➥BEGIN INSERT INTO
test VALUES(id_val); END;")) {
    echo "Stored procedure creation failed: (" . $mysqli->errno . ") " .
$mysqli->error;
}

/* Call the stored procedure */
if (!$mysqli->query("CALL p(1)")) {
    echo "CALL failed: (" . $mysqli->errno . ") " . $mysqli->error;
}
```

This approach is equally effective in preventing SQL injection as the parameterized queries method I mentioned earlier, so you can decide which is better for your situation.

Escaping user supplied input: in this approach, user input is manually (or sometimes with the help of DBMS escaping mechanisms) escaped for valid strings, thus minimizing any chance of SQL injection attacks. Although it is bit weaker than other approaches, it can be useful in cases where you want better performance or are rewriting legacy code and want to finish with lesser effort.

PHP provides an automatic input escape mechanism called magic_quotes_gpc that you can use before sending the input to the back end. But, it would be better to use the escaping mechanism provided by your database, because in the end, the query comes to the database, and the database will know better about what is a valid query. MySQL provides the mysql_real_escape_string() method to escape the input. Check your database documentation to find which escape function is supported.

Session Handling

As soon as legitimate users log in to a site with their credentials, a session is started and maintained until they

log out. The problem begins when someone impersonating a genuine user tries to sneak in. Obviously, the results can be very severe—users' money or even their identities can be stolen. Let's explore how you can change your coding style so that the session is handled safely.

Session management implementation: you always should use the built-in session management feature that comes out of the box with your Web development framework. Not only does this save critical development time and cost, it generally is safer as well, because many people are using and testing it.

Another thing to take care of while implementing session management is keeping track of what method the application uses to send/receive the session ID. It may be a cookie or URL rewriting or a mixture of both, but you generally should limit it and accept the session ID only via the mechanism you chose in the first place.

Cookie management: whenever you plan to use cookies, be aware that you are sending out data about the user/session, which potentially can be intercepted and misused. So, you need to be extra careful while handling cookies. Always add the cookie attributes securely with HttpOnly, because this ensures that the cookie always is sent only via an HTTPS connection and doesn't allow scripts to access the cookie. These two attributes will reduce the chances of cookies being intercepted.

Other attributes like domain and path always should be set to indicate to the browser where exactly the cookie should be sent, so that it reaches only the exact destination and not anywhere else.

Last but definitely not least, the expire and max age attributes should be set in order to make the cookie nonpersistent, so it is wiped off once the browser instance is closed.

Session expiry management: a timeout should be enforced over and above the idle time out, so that if users intend to stay longer, they should authenticate themselves again. This generates a new session ID, so attackers have less time to crack the session ID.

Also, when the session is being invalidated, special care should be taken to clear the browser data and cookie, if used. To invalidate a cookie, set the session ID to empty and the expires date to a past date. Similarly, the server side also should close and invalidate the session by calling proper session handling methods, for example session_destroy()/unset() in PHP.

Web Service Security

In layman's terms, a Web service can be defined as a software system designed to support communication between two electronic devices over the Internet. In real-life scenarios, however, things are not that simple. As the Internet grows, the threat of misuse of these services grows. Let's look at some important tips you should keep in mind while developing Web services.

Schema validation: whatever SOAP payloads you expect your Web service to handle, they should be validated against the associated XML schema definition. The XSD should at least define the maximum length and character set of every parameter allowed in the Web service. Also, if you expect a fixed-format parameter, such as e-mail ID, phone numbers and so on, define the validation pattern in the XSD.

XML denial of service prevention: denial of service attacks try flooding the Web server with a large number of requests so that it eventually crashes. To protect your Web service from such attacks, be sure to optimize the configuration for maximum message throughput and limit SOAP message size. Also, validate the requests against recursive or oversized payloads, because such payloads generally can be malicious.

Safe URL Redirects

Many times you will need to redirect to a new page based on user input. But, the redirects can take a dangerous turn if not done properly. For example, if attackers get to redirect the application to a URL of their choice, they then can launch a phishing attack and compromise user data. A safe URL redirect is one where you have hard-coded the URL in the code like this:

```
<?php header("Location: http://www.mywebsite.com") ?>
```

Cases like the following where you have to go to a URL passed by the user should be avoided:

```
<?php $url = $_GET['inputURL'];
header("Location: " . $url); ?>
```

If you can't avoid this, keep reading.

Don't use the URL as the input: even if you want to redirect to a URL based on user input, it is better not to allow the user to enter the URL. Instead, you can use other input mechanisms like a button or a direct link. This way you can prevent users from entering any random link.

Validate the input before the redirect: in cases where you simply can't avoid user input, make sure you validate the input against exactly the same site or a list of hosts or a regex. Also notify users with an indication of the site they are being redirected to.

Cross-Site Scripting

Such attacks are targeted to the end user's browser. A common way to attack is to inject a malicious script, in the form of JavaScript, Flash or even HTML to a genuine Web site. When the user accesses that site, the browser has no clue whether script is genuine and just executes it. Such scripts, once executed, can access the cookies, session data or other sensitive information stored in the browser, because they are sent via a genuine Web site that the user tried to access.

To counter such attacks/injections in your Web site, OWASP suggests treating the Web pages as templates with certain slots for the untrusted data. For example, say you are creating a home page, and on the top left, you have a slot for the user name, which is retrieved by the application and displayed to the user while the Web page renders. These slots can be for one of several components—for example, JavaScript, HTML or CSS. For each component, there are preventive measures that help make sure others can't inject their code. Let's look at all the rules.

First, you need to define the slots that should be present in the Web page. Then, make sure you don't allow any untrusted data into the document, unless it is within one of the slots you already defined.

Untrusted data in HTML tags and attributes: when you need to insert untrusted data in HTML tags like div, p, b, td and so on, make sure it is escaped before being used. This way, even if attackers somehow manage to send in their code, escaping the untrusted data will ensure that the code doesn't cause much harm. For example, characters like <, >, & and so on should be changed to &lt, &gt and &amp, respectively. Also attribute fields in HTML tags like name, id, width and so on sometimes can be required to take variable values, so you may need to pass untrusted data to such fields. In this case, make sure to escape the data before using it as well. Take a look at the ESAPI reference implementation of HTML entity escaping and un-escaping.

Here is a sample usage of the API:

```
String safe = ESAPI.encoder().encodeForHTMLAttribute(
➥request.getParameter("input" ) );
```

Untrusted data in CSS: in situations where you need to put untrusted data in CSS, it is important to take care that it is done only in a property value and nowhere else. Especially when you need to pass a URL, check if it starts with "http". Then, except for alphanumeric characters, escape all the other characters with an ASCII value less than 256 (the ESAPI also supports CSS escaping and un-escaping):

```
String safe = ESAPI.encoder().encodeForCSS(
➥request.getParameter( "input" ) );
```

Untrusted data in an HTTP GET parameter: URLs like "http:www.mysite.com/value=data" can be possible targets of attack if the GET parameter is not being escaped before execution. In such cases, make sure to escape all characters with an ASCII value less than 256 with the %HH escaping format. The ESAPI for URL validation looks like this:

```
String safe = ESAPI.encoder().encodeForURL(
➥request.getParameter( "input" ) );
```

Untrusted data in JavaScript data values: if you want to place untrusted data in JavaScript, the only safe place is inside the quoted data value. Because otherwise it is very easy to change the execution context and execute the statement, which actually is meant to be data. Again, the escaping mechanism to follow is the same as in previous cases—that is, escape all the characters with an ASCII value less than 256. The ESAPI for this is:

```
String safe = ESAPI.encoder().encodeForJavaScript(
➥request.getParameter("input" ) );
```

Conclusion

Finally, it is best to acquaint yourself with the OWASP security guidelines if you are a Web developer or plan to become one. This not only will save you a lot of work in design changes and security hardening, but it also makes sure your end users' data and identity are safe.■

Nitish Tiwari is based out of Bangalore, India. He currently is working as a developer in a FOSS-based startup. He also helps enterprises implement open-source tools based on their needs. In his free time, he likes to try out and test open-source tools. You can reach him at nitish.tiwari@technocube.in.

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

Using Django and MongoDB to Build a Blog

MongoEngine is an ORM for working with MongoDB from Python.

MIHALIS TSOUKALOS

This article shows how to create a simple blog site using the MongoDB Document Database and the Django Web framework.

Mongo Basics

MongoDB is an open-source document-oriented database, not a traditional relational database, written in C++ by Dwight Merriman and Eliot Horowitz. Being a document database does not mean storing Microsoft Word documents, but rather it means storing semi-structured data. You can input arbitrary binary JSON objects (BSON) into a MongoDB database. It runs on UNIX machines as well as Windows and supports replication and sharding.

Your Linux distribution probably includes a MongoDB package, so go ahead an install it if you have not done so already. Alternatively, you can download a precompiled binary or get the MongoDB source code from **http://www.mongodb.org** and compile it yourself.

On a Debian 7 system, you can install MongoDB with the following command:

```
# apt-get install mongodb
```

After installing MongoDB, start the MongoDB server process with:

```
# service mongodb start
```

Similarly, you can stop the running MongoDB server with:

```
# service mongodb stop
```

After installation, type `mongo --version` in your UNIX shell to find the MongoDB version you are using, and type `mongo` to enter the MongoDB shell and check whether the MongoDB server process is running.

By default, the MongoDB server process listens to localhost using the 27017 port. You can change it if you want, but if both the MongoDB server and the Django installation are on the same machine, it is more secure to leave it as it is.

The configuration file for MongoDB is /etc/mongodb.conf. Nevertheless, if you want to run multiple MongoDB servers on the same UNIX machine, you can bypass the /etc/mongodb.conf file and utilize command-line options that allow you to use a different port number, a different IP or even a

SQL Term	MongoDB Term
Database	Database
Table	Collection
Index	Index
Row	BSON document
Column	BSON field
Primary Key	_id field
Group by	Aggregation
Join	Embedding and Linking

Figure 1. MongoDB Terminology

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

different MongoDB configuration file.

Figure 1 shows the most useful MongoDB terms in relation to their respective SQL terms.

Starting the MongoDB server process (mongod) on a Linux machine without any parameters and without root privileges should generate output similar to the following:

```
$ mongod
mongod --help for help and startup options
Fri Sep 27 23:21:33 [initandlisten] MongoDB starting :
 ➥pid=7991 port=27017 dbpath=/data/db/ 64-bit host=mail
Fri Sep 27 23:21:33 [initandlisten] db version v2.0.6,
 ➥pdfile version 4.5
Fri Sep 27 23:21:33 [initandlisten] git version: nogitversion
Fri Sep 27 23:21:33 [initandlisten] build info: Linux z6
 ➥3.8-trunk-amd64 #1 SMP Debian 3.8.3-1~experimental.1
 ➥x86_64 BOOST_LIB_VERSION=1_49
Fri Sep 27 23:21:33 [initandlisten] options: {}
Fri Sep 27 23:21:33 [initandlisten] exception in initAndListen:
 ➥10296 dbpath (/data/db/) does not exist, terminating
Fri Sep 27 23:21:33 dbexit:
Fri Sep 27 23:21:33 [initandlisten] shutdown: going to close
 ➥listening sockets...
Fri Sep 27 23:21:33 [initandlisten] shutdown: going to
 ➥flush diaglog...
Fri Sep 27 23:21:33 [initandlisten] shutdown: going to
 ➥close sockets...
Fri Sep 27 23:21:33 [initandlisten] shutdown: waiting
 ➥for fs preallocator...
Fri Sep 27 23:21:33 [initandlisten] shutdown: lock for
 ➥final commit...
Fri Sep 27 23:21:33 [initandlisten] shutdown: final commit...
Fri Sep 27 23:21:33 [initandlisten] shutdown: closing all files...
Fri Sep 27 23:21:33 [initandlisten] closeAllFiles() finished
Fri Sep 27 23:21:33 dbexit: really exiting now
```

Django Basics

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It allows you to build Web applications quickly. Instagram, Mozilla and Pinterest use Django.

Simply put, Django is a collection of libraries written in Python. In order to create a site using Django, you basically write Python code that uses the Django libraries. If you already have a good working knowledge of Python, you have to understand only how the Django libraries work.

Django follows a slightly changed version of the MVC (Model View

Controller) design pattern called Model Template View (MTV). The MTV handles the Controller work by the core, and all the other work is done in Models, Templates and Views. According to Django's philosophy, what is truly important is not terminology but getting things done.

On a Debian 7 system, you can install Django with the following command:

```
# apt-get install python-django
```

To make sure that everything works as expected, type the following Django command, which prints the version of Django:

```
# django-admin version
1.5.1
```

How Python and Django Communicate with MongoDB

You will need a Python module called PyMongo to talk to MongoDB from Python. On a Debian 7 system, you can install it as follows:

```
# apt-get install python-pymongo
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  python-bson python-bson-ext python-gridfs python-pymongo-ext
The following NEW packages will be installed:
  python-bson python-bson-ext python-gridfs python-pymongo
  ➥python-pymongo-ext
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 212 kB of archives.
After this operation, 928 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

The following instructional Python code (saved as connect.py) connects to a MongoDB database, prints the available databases of the mongodb://localhost:27017 server and closes the MongoDB connection:

```
import pymongo
# Open the MongoDB connection
connMongo = pymongo.Connection('mongodb://localhost:27017')
# Print the available MongoDB databases
print connMongo.database_names()
# Close the MongoDB connection
connMongo.close()
```

You can run the small Python script as follows:

```
$ python connect.py
[u'LJ', u'local', u'test']
```

The output shows that at the time of running the script, three databases exist called LJ, local and test. Although PyMongo will not be used directly in the rest of the article, it is useful to know about it for testing

Generally speaking, Django has a wrapper for every relational database it supports, but Mongo is a non-relational database, so you need some external help.

and troubleshooting purposes.

Generally speaking, Django has a wrapper for every relational database it supports, but Mongo is a non-relational database, so you need some external help. You need the MongoEngine Python package in order to utilize MongoDB. Other options are Ming, MongoKit, django-mongodb and django-nonrel. In my opinion, MongoEngine is the best option.

MongoEngine is an object-document mapper made for MongoDB, following Django's ORM style. You can install it by executing this command:

```
# apt-get install python-mongoengine
```

MongoEngine is based on PyMongo, and that's why you need to know some basic things about PyMongo.

For those of you who are familiar with Django, you should know that when you are using MongoEngine, you lose both the Django Admin panel and the `python manage.py syncdb` command. Losing the Django Admin panel is a major drawback, but MongoDB offers features that relational databases cannot provide.

The Problem

Imagine you registered a new domain to host your personal site. The site also will have a blog. Instead of using a CMS, such as Joomla! or WordPress, for creating the blog, you want more control over the site, so you decide to use Django for creating the blog and MongoDB for storing the blog data.

The nice thing about this solution is that if you already are familiar with Django, it will not take more than two hours to develop, test and deliver a complete version of the blog site.

Note: the solution presented here tries to be as Django-"native" as possible. The only thing different from the usual Django way is the use of MongoDB.

The Solution

If you try to access a MongoDB that does not already exist, MongoDB

will create it. The same happens if you try to write to a MongoDB collection (table) that does not exist. So, you do not need to execute any commands on MongoDB, but you should be very careful not to have any typos in your code.

Do the following steps on Django.

1) Create a new project called LJ:

```
$ django-admin.py startproject LJ
$ cd LJ
```

The manage.py script is created for every Django project and is a wrapper around django-admin.py. You do not need to make any changes to it.

2) Run the test development Web server to see if everything is okay:

```
$ python manage.py runserver
```

By going to http://localhost:8000/ (or http://127.0.0.1:8000/), you will see Figure 2.

The test development server restarts automatically every time you make changes to the Django project.

3) Create the app for the blog called LJblog:

```
$ python manage.py startapp LJblog
```

4) Add the name of the LJblog app to the list of the INSTALLED_APPS in the LJ/settings.py file. If you do not "install" the app, you will not be able to use it. So, the `INSTALLED_APPS` variable should have the following values:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'LJblog',
    'django_extensions',
)
```

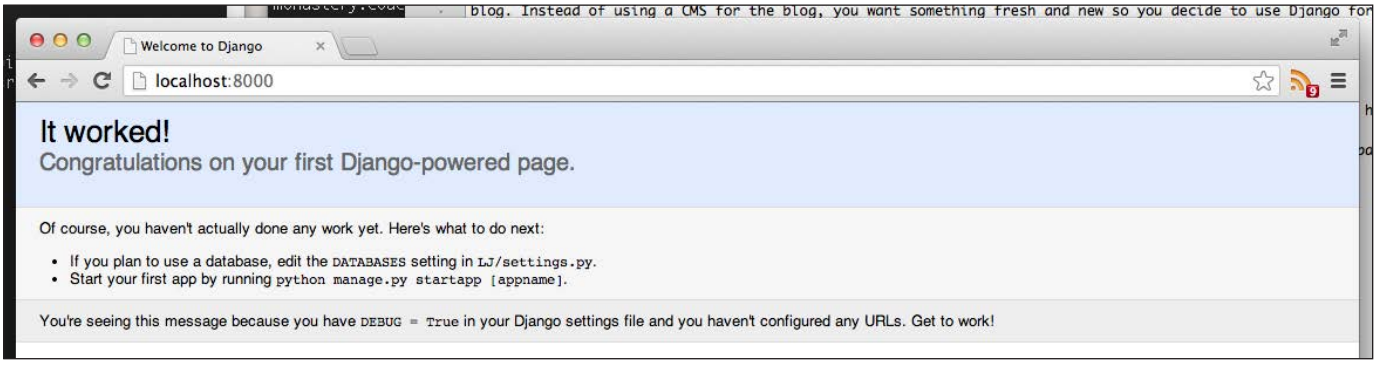


Figure 2. The Test Development Web Server

As you can see here, it also is required to install a package called django-extensions. If your Linux distribution does not provide a ready-to-install package, visit the Django Extensions site for instructions about installing it.

4) Many other changes had to be made in the LJ/settings.py file. The following `diff` output shows the changes:

```
$ diff settings.py{,.orig}
3,5d2
< import os
< PROJECT_ROOT = os.path.abspath(os.path.dirname(__file__))
<
14a12,23
> DATABASES = {
>   'default': {
>       'ENGINE': 'django.db.backends.', # Add 'postgresql_psycopg2',
                                         # 'mysql', 'sqlite3' or
                                         # 'oracle'.
>       'NAME': '',      # Or path to database file if using sqlite3.
>       # The following settings are not used with sqlite3:
>       'USER': '',
>       'PASSWORD': '',
>       'HOST': '',      # Empty for localhost through domain sockets
                         # or '127.0.0.1' for localhost through TCP.
>       'PORT': '',      # Set to empty string for default.
>   }
> }
>
44c53
< MEDIA_ROOT = os.path.join(PROJECT_ROOT, '..', 'media')
---
> MEDIA_ROOT = ''
49c58
< MEDIA_URL = '/media/'
---
> MEDIA_URL = ''
55c64
< STATIC_ROOT = os.path.join(PROJECT_ROOT, '..', 'static')
---
> STATIC_ROOT = ''
63d71
<         os.path.join(PROJECT_ROOT, 'static'),
103d110
<         os.path.join(PROJECT_ROOT, 'templates'),
148,159d154
<
< AUTHENTICATION_BACKENDS = (
<     'mongoengine.django.auth.MongoEngineBackend',
< )
<
< SESSION_ENGINE = 'mongoengine.django.sessions'
<
< MONGO_DATABASE_NAME = 'LJ_blog'
<
< from mongoengine import connect
< connect(MONGO_DATABASE_NAME)
<
```

Note: the name of the MongoDB database is defined and stored in the `MONGO_DATABASE_NAME` variable.

5) The contents of the LJ/urls.py file should be the following:

```
from django.conf.urls import patterns, include, url
```

```
from django.conf import settings
from LJblog.views import PostListView

urlpatterns = patterns('',
        url(r'^$', PostListView.as_view(), name='list'),
    url(r'^post/', include('LJblog.urls'))
)
```

6) Inside the LJ/LJ directory, you need to create two directories, called static and templates, and copy some files and directories inside them. The project uses the Twitter Bootstrap set of tools for creating Web sites and Web applications.

The following output shows the full contents of the static directory:

```
$ ls -lR static/
total 0
drwxr-xr-x@ 6 mtsouk  staff  204 Sep 21 14:13 bootstrap

static//bootstrap:
total 0
drwxr-xr-x@ 6 mtsouk  staff  204 Jan  5  2013 css
drwxr-xr-x@ 4 mtsouk  staff  136 Jan  5  2013 img
drwxr-xr-x@ 4 mtsouk  staff  136 Jan  5  2013 js

static//bootstrap/css:
total 544
-rwxr-xr-x@ 1 mtsouk  staff   21751 Jan  5  2013
 ➥bootstrap-responsive.css
-rwxr-xr-x@ 1 mtsouk  staff   16553 Jan  5  2013
 ➥bootstrap-responsive.min.css
-rwxr-xr-x@ 1 mtsouk  staff  124223 Jan  5  2013
 ➥bootstrap.css
-rwxr-xr-x@ 1 mtsouk  staff  103314 Jan  5  2013
 ➥bootstrap.min.css

static//bootstrap/img:
total 56
-rwxr-xr-x@ 1 mtsouk  staff   8777 Jan  5  2013
 ➥glyphicons-halflings-white.png
-rwxr-xr-x@ 1 mtsouk  staff  12799 Jan  5  2013
 ➥glyphicons-halflings.png

static//bootstrap/js:
total 184
-rwxr-xr-x@ 1 mtsouk  staff  58516 Jan  5  2013
 ➥bootstrap.js
-rwxr-xr-x@ 1 mtsouk  staff  31596 Jan  5  2013
 ➥bootstrap.min.js
```

The templates directory contains two files, as the output of the `ls -lR` command shows:

```
$ ls -lR templates/
total 16
-rwxr-xr-x@ 1 mtsouk  staff  1389 Sep 25 21:25 base.html
-rwxr-xr-x@ 1 mtsouk  staff   148 Jan  5  2013 messages.html
```

The base.html file contains project-specific information that you can change.

7) The connection to the MongoDB database happens inside the LJ/settings.py file. Django needs the following two commands to connect to a

MongoDB database:

```
from mongoengine import connect
connect(MONGO_DATABASE_NAME)
```

8) Next, you should create four HTML files inside the templates/LJblog directory. They are the displayed Web pages for the Create, Read, Update and Delete operations:

- create.html
- detail.html
- list.html
- update.html

The selected filenames must match the parameters found inside the LJblog/urls.py file.

9) The contents of the LJblog/urls.py file are the following:

```
from django.conf.urls import patterns, url
from views import PostCreateView, PostDetailView,
 ➥PostUpdateView, PostDeleteView

urlpatterns = patterns('',
    url(r'^add/$', PostCreateView.as_view(), name='create'),
    url(r'^(?P<pk>[\w\d]+)/$', PostDetailView.as_view(),
    ➥name='detail'),
    url(r'^(?P<pk>[\w\d]+)/edit/$', PostUpdateView.as_view(),
    ➥name='update'),
    url(r'^(?P<pk>[\w\d]+)/delete/$', PostDeleteView.as_view(),
    ➥name='delete'),
)
```

10) Next, you need to edit the models.py file. This file is where data models are defined.

Using Django's ORM (Object-Relational Mapper) is one of the project's goals. ORM allows the Python classes that were defined inside models.py to access the selected database without requiring you to deal with the database directly. ORM is a major advantage of Django.

The Post Python class is defined as follows:

```
class Post(Document):
    user = ReferenceField(User, reverse_delete_rule=CASCADE)
    title = StringField(max_length=200, required=True)
    text = StringField(required=True)
    text_length = IntField()
    date_modified = DateTimeField(default=datetime.now)
    is_published = BooleanField()
```

As you will see later in this article, the Post MongoDB table has a direct connection to the Post Python class.

11) Then, you need to edit the forms.py file. The forms.py file allows Django to access user-submitted form data.

12) Last but not least, you should edit the views.py file. This file includes the functions that handle data as well

as various other things.

The directory structure of the project as well as the included files are shown in Figure 3. You also will notice files ending with .pyc. These are byte-code files created by the Python interpreter that are executed by the Python virtual machine.

You can examine the contents of the LJ_blog collection using MongoDB commands:

```
> use LJ_blog;
switched to db LJ_blog
> show collections;
post
system.indexes
> db.post.find();
{ "_id" : ObjectId("523d83de8491973b242e2772"), "title" :
 ➥"First blog entry!", "text" : "This is my first
 ➥blog entry.\r\Mihalis Tsoukalos", "text_length" : 47,
 ➥"date_modified" : ISODate("2013-09-21T06:32:46.289Z"),
 ➥"is_published" : true }
{ "_id" : ObjectId("523d83f88491973b242e2773"), "title" :
 ➥"Another post", "text" : "Just another blog post!",
 ➥"text_length" : 23, "date_modified" :
 ➥ISODate("2013-09-21T06:33:12.321Z"), "is_published" : true }
{ "_id" : ObjectId("523d86f58491973b9e3c8c78"), "title" :
 ➥"Just another test!", "text" : "Just another test!\r\nLJ",
 ➥"text_length" : 22, "date_modified" :
 ➥ISODate("2013-09-21T06:45:57.092Z"), "is_published" : true }
>
```

Note: every time you insert a BSON document in MongoDB,

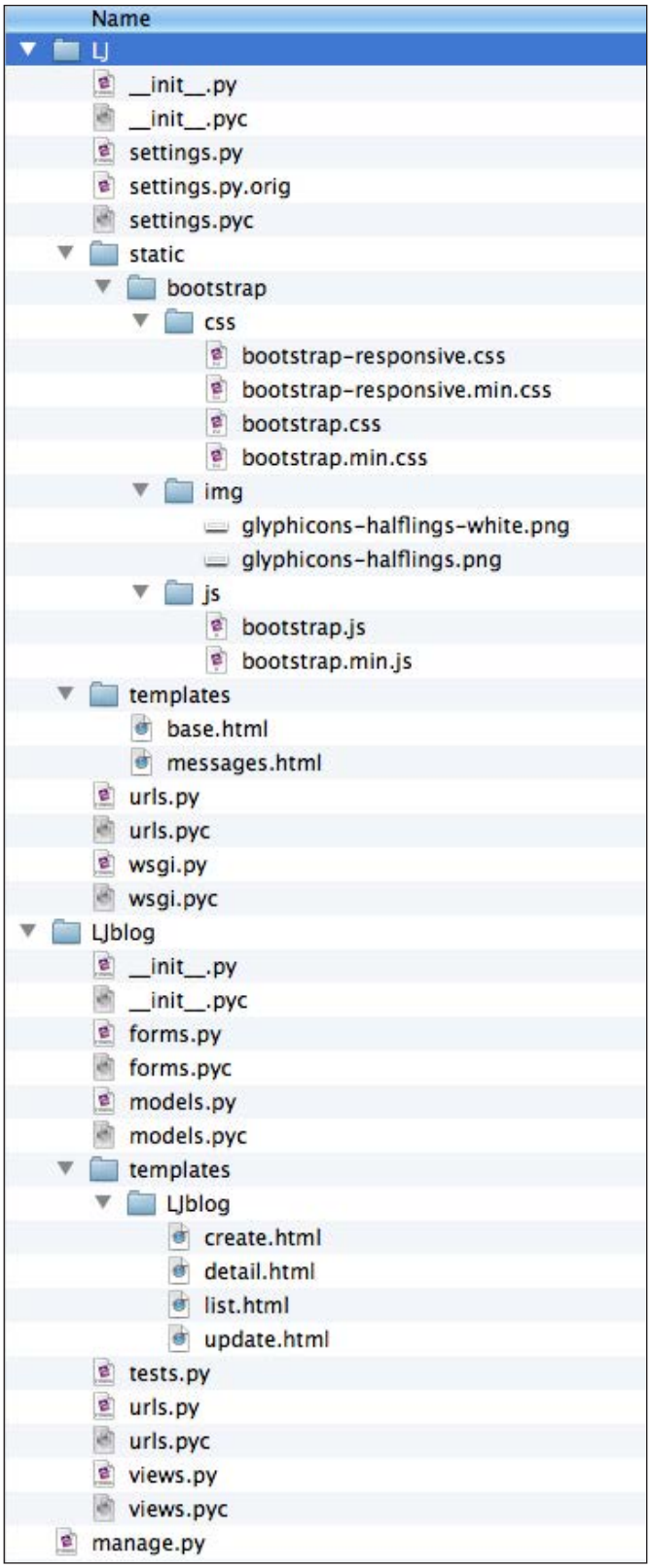


Figure 3. The Directory Structure of the Django Project

MongoDB automatically generates a new field called _id. The _id field acts as the primary key and is always 12 bytes long.

Now, you should check that everything is fine by running the test development server and trying to connect to http://localhost:8000/:

```
$ python manage.py runserver
Validating models...

0 errors found
September 21, 2013 - 07:25:07
Django version 1.5.1, using settings 'LJ.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

If everything is okay, you will see something similar to Figure 4 after visiting http://127.0.0.1:8000/ or http://localhost:8000/.

While using the application, the output from the test development server is updated and will look similar to the following:

```
[25/Sep/2013 12:18:28] "GET / HTTP/1.1" 200 1320
[25/Sep/2013 12:18:28] "GET /static/bootstrap/css/bootstrap.min.css
➥HTTP/1.1" 200 103314
```

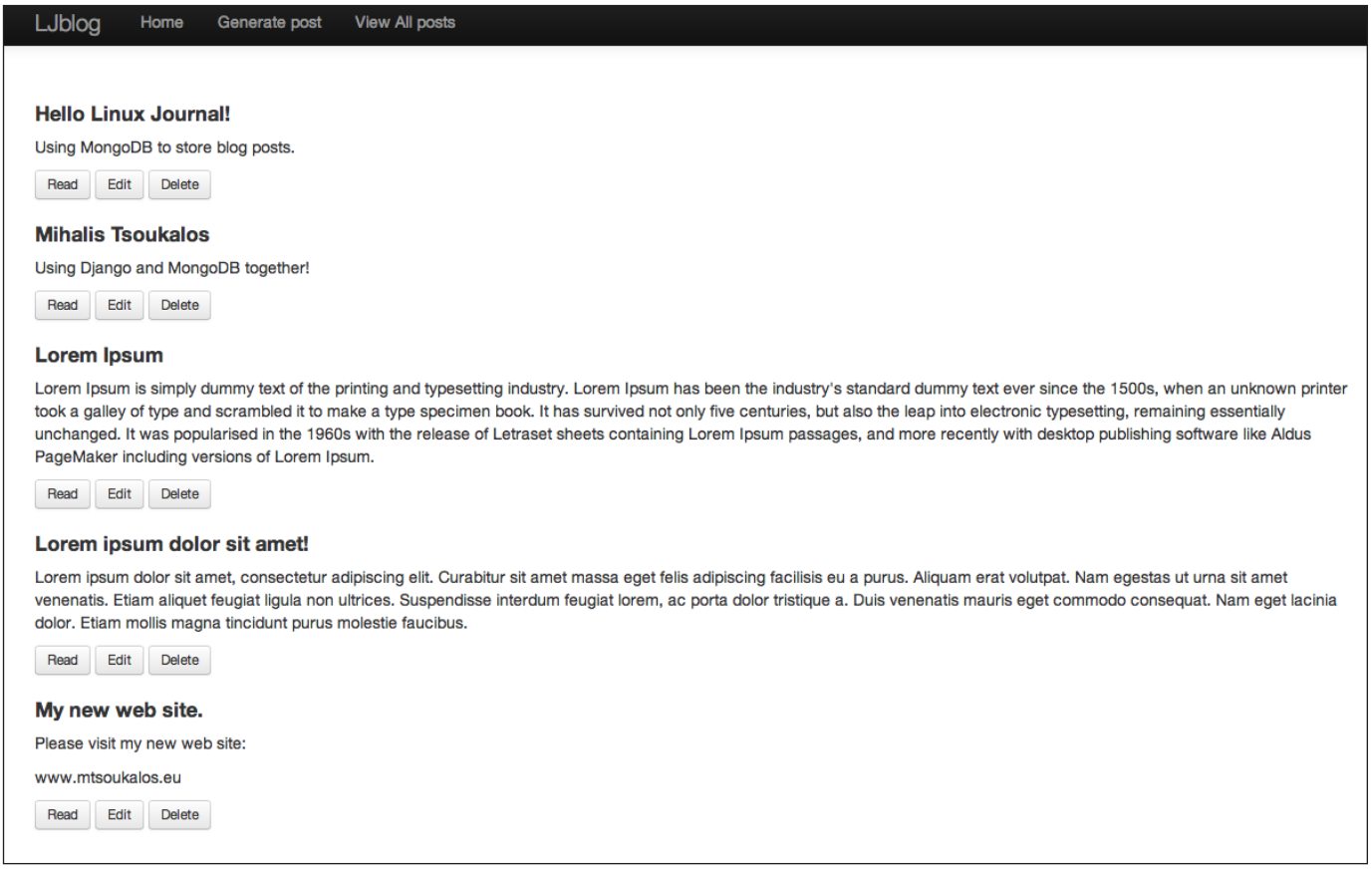


Figure 4. The LJblog application is up and running.

```
[25/Sep/2013 12:18:28] "GET /static/bootstrap/js/bootstrap.min.js
 ➥HTTP/1.1" 200 31596
[25/Sep/2013 12:18:28] "GET /static/bootstrap/css/
➥bootstrap-responsive.css HTTP/1.1" 200 21751
[25/Sep/2013 12:18:32] "GET / HTTP/1.1" 200 1320
[25/Sep/2013 12:18:33] "GET /post/add/ HTTP/1.1" 200 1823
[25/Sep/2013 12:18:34] "GET /?all_posts HTTP/1.1" 200 1320
[25/Sep/2013 16:01:10] "GET /post/5243295f8491976bd8f016d0/edit/
 ➥HTTP/1.1" 200 1841
[25/Sep/2013 16:01:18] "GET /post/5243295f8491976bd8f016d0/delete/
 ➥HTTP/1.1" 302 0
```

The output is useful for debugging purposes, especially when you don't get the expected results on your Web browser.

If you want to delete the LJ_blog collection in order to start your blog from scratch, use the following command with care:

```
> db.post.drop()
true
> db.post.find();
> show collections;
system.indexes
```

Deploying the Django Web Site to a Production Server

Explaining the full deployment process is outside the scope of this article, but I want to give some useful tips. When you try to run your Django project on a production server, keep the following things in mind:

1) Turn off Debug mode inside LJ/settings.py:

```
DEBUG = False
TEMPLATE_DEBUG = DEBUG
```

2) Change the ADMINS setting inside LJ/settings.py to something useful:

```
ADMINS = (
    ('Mihalis', 'someEmail@Domain.GR'),
)
```

3) Install and activate the mod_python Apache module.

4) You can use mod_wsgi instead of mod_python.

Why Use MongoDB Instead of a Relational Database?

You may wonder why you should use a NoSQL database, such as MongoDB, instead of a traditional DBMS like MySQL or PostgreSQL. Although it would be possible to use a relational database, here are the reasons for preferring MongoDB:

- MongoDB is generally faster.
- MongoDB is better for high-volume traffic sites.
- MongoDB supports sharding. Sharding (aka horizontal

partitioning) is the process of separating a single database across a cluster of machines.

- MongoDB supports replication.
- Your data schema may change without downtime when using MongoDB.
- Depending on the application, it may feel more natural to develop in a document-oriented database.
- MongoDB has an easy-to-use protocol for storing large files and file metadata called GridFS.

Summary

As I've explained here, MongoDB and Django can indeed work together. However, two things are missing to be 100% Django-native: support for the Django Admin Panel and support for the `syncdb` command.

A link to the full code for this project is listed in the Resources section of this article.

Acknowledgement

I would like to thank Josh Ourisman for answering some questions that I had while writing this article.■

Mihalis Tsoukalos is a UNIX administrator who also focuses on programming, databases and mathematics. You can reach him at tsoukalos@sch.gr and @mactsouk (Twitter). Contact him if you are looking for a UNIX system administrator.

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

Resources

Code for This Article: **http://www.mtsoukalos.eu/FILES/MongoDjango.zip**
MongoDB: **http://www.mongodb.org**
Pymongo: **http://api.mongodb.org/python/current**
BSON: **http://bsonspec.org**
Django Web Page: **https://www.djangoproject.com**
MongoEngine: **http://mongoengine.org**
Django Extensions: **https://django-extensions.readthedocs.org/en/latest**
The Django Book: **http://www.djangobook.com/en/2.0/index.html**
Twitter Bootstrap: **http://en.wikipedia.org/wiki/Twitter_Bootstrap**
MongoKit: **http://namlook.github.io/mongokit**
mod_wsgi: **http://code.google.com/p/modwsgi**

LINUX JOURNAL
ARCHIVE DVD

NOW AVAILABLE

Save $10.00 by using discount code DVD2013 at checkout.

Coupon code expires 2/3/2013

www.linuxjournal.com/dvd

WEBCASTS

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

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

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

Modernizing SAP environments with minimum risk—a path to Big Data

Sponsor: SAP | Topic: Big Data

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

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

WHITE PAPERS

White Paper: JBoss Enterprise Application Platform for OpenShift Enterprise

Sponsor: DLT Solutions

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

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

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

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

WHITE PAPERS

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

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

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

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

Standardized Operating Environments for IT Efficiency

Sponsor: Red Hat

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

Benefits of an SOE:

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

SOE leads to:

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

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

Cloud Computing Basics—Platform as a Service (PaaS)

PaaS is a cloud service model that allows you to develop, deploy and run business applications swiftly.

MITESH SONI

Generally, good programming is considered to be the measured application of an art form, craft or discipline, with the objective of producing a competent and evolving business solution. In traditional environments, computer programming is a practice that has multiple phases, such as designing, developing, testing, debugging and maintaining application code. We programmers use programming languages like C, C++, C#, Java, Python and Smalltalk to create business applications. The process of writing code often requires proficiency in many diverse subjects, including knowledge of the application domain, runtime environment, specific algorithms, programming languages and proper logic.

In addition, developing and deploying applications is a complex, expensive and time-consuming task. Business applications require hardware resources, operating systems, databases, middleware, Web servers and other software. Once the technology stack and hardware resources are available, a team of developers needs to navigate frameworks, such as J2EE and .NET, for development. A dedicated team of network, database and IT management experts keeps resources and applications available.

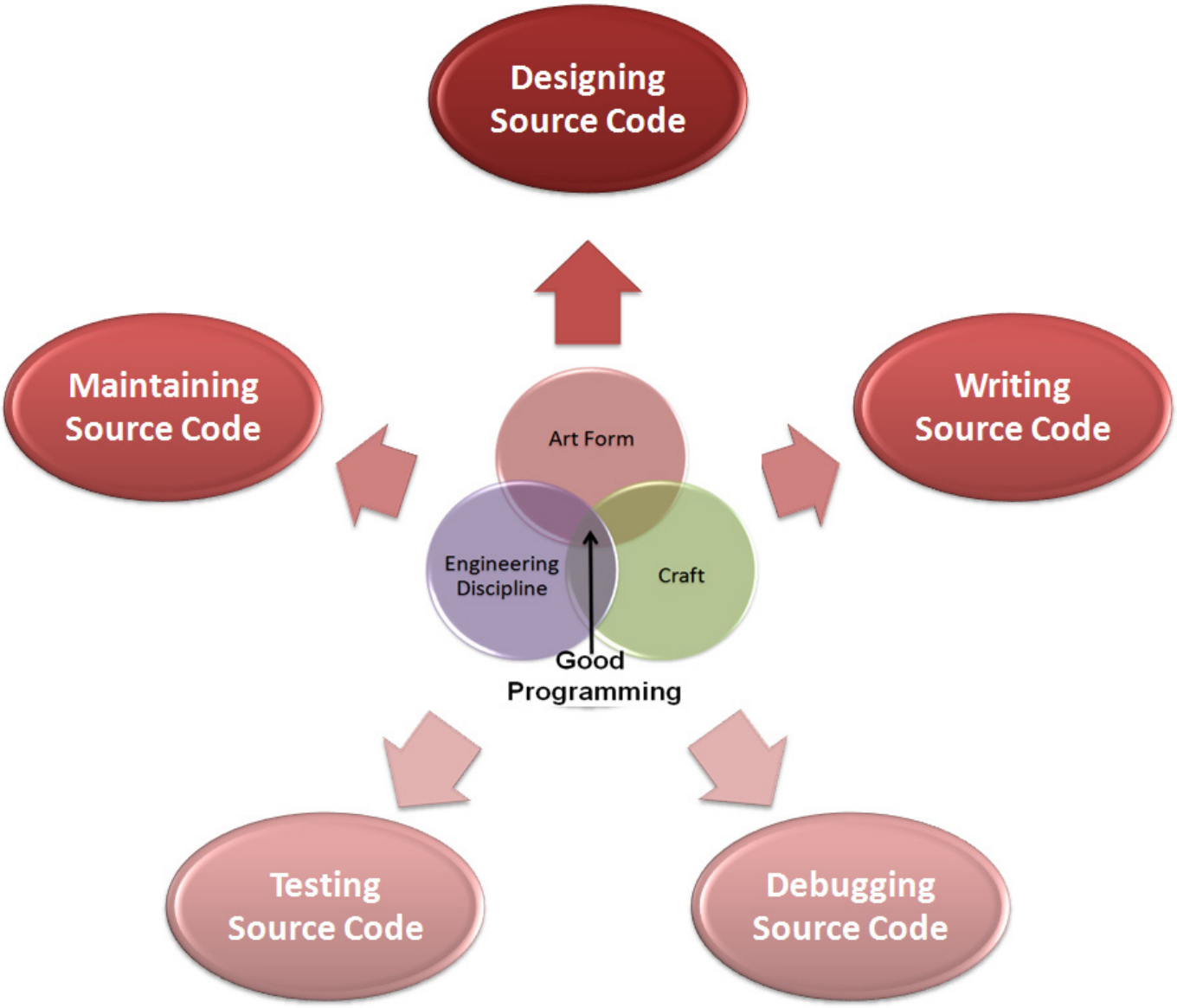


Figure 1. Programming

Inevitably, business requirements will require changes in the application, which results in a lengthy development, testing and re-deployment cycle. Furthermore, large organizations need specialized facilities to house their data centers and a team to maintain them. A gigantic amount of electricity is used to power the servers as well as to keep them cool.

If you think the complexities end here, then wait. A failover site holds significant importance to mirror the data center so that information and resources can be replicated in case of disaster. Applications built with these complexities are difficult

to scale for usage spike demands, fragile to update, and difficult to make mobile and social as business needs change.

Modern Application Programming and Its Challenges

Modern programming has changed in a different manner due to the explosion of devices. Expanding data and new business requirements need different approaches from the previous era. Considering either the traditional approach or modern approach, business applications must satisfy some fundamental properties, such as automation, reliability, robustness, performance and availability. Unfortunately, even in modern times, automated build and test environments don't exist in many organizations, and best practices are not applied everywhere in the software development life cycle.

It is important to understand that an application's reliability depends on the accuracy of algorithms, fewer programming mistakes, the implementation of security best practices, and avoiding logic errors, such as division by zero. Anticipating errors, such as incorrect or compromised data and unavailability of resources, increases robustness. How well application developers and IT teams manage robustness holds more significance in achieving the application's objectives. Efficiency, performance, maintainability, portability and availability of an application are equally important in modern environments. Cloud computing is a revolutionary approach that provides a ray of hope for organizations dealing with these modern challenges.

Introducing PaaS

Cloud computing is still an evolving paradigm, but it is one of the most disruptive innovations in the past few years. According to the definition from NIST, it is the model to enable convenient and on-demand network access to a shared pool of configurable computing resources, such as compute, storage and network, that can be provisioned rapidly and released with minimal

PaaS providers manage underlying infrastructure resources, such as operating systems, virtual servers, networks, Web servers, application servers, databases, backup and disaster recovery.

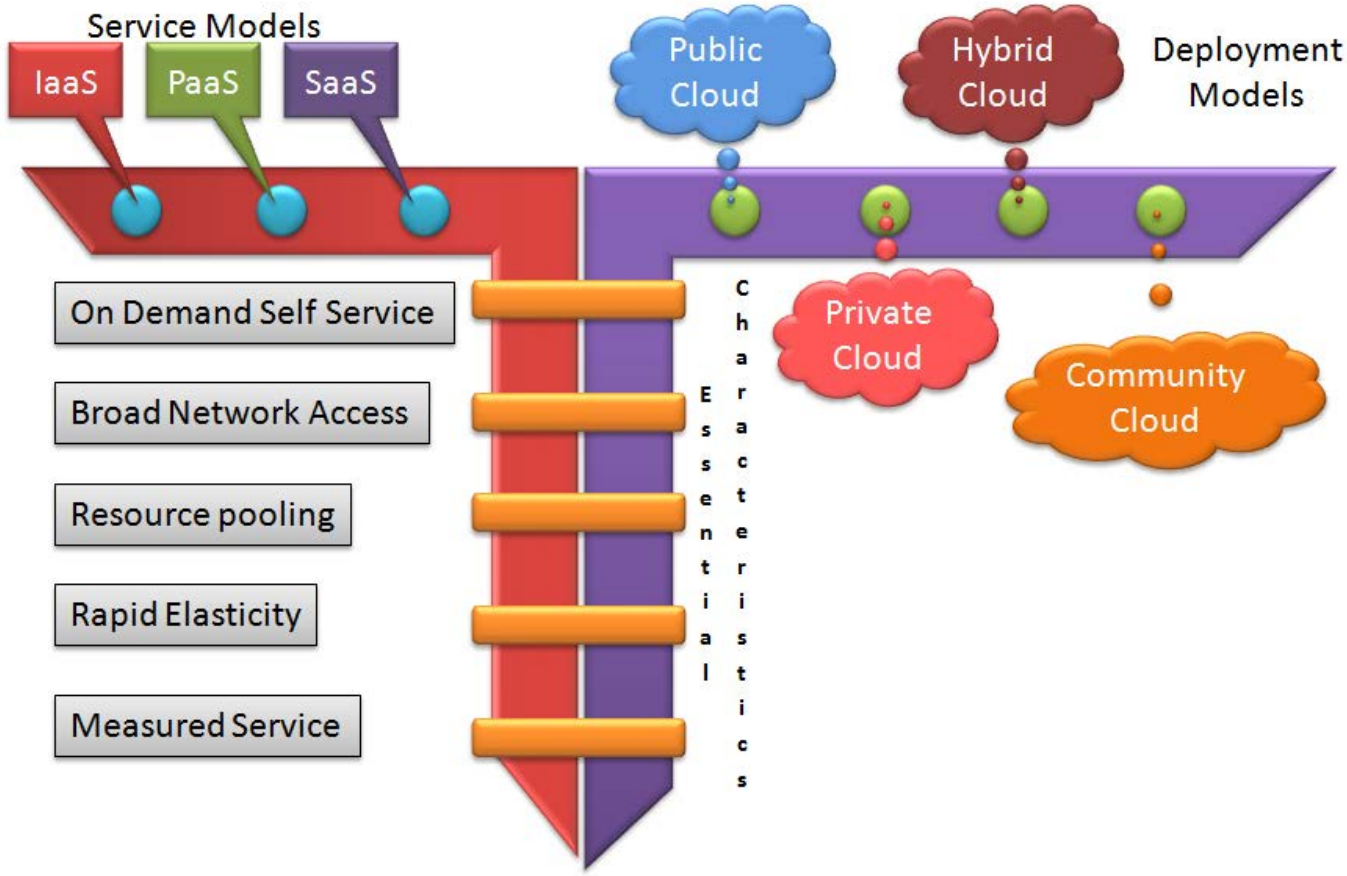


Figure 2. NIST Definition of Cloud Computing

management effort.

Cloud computing is composed of three service models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). This article focuses on PaaS.

With PaaS, you can deploy applications into the cloud infrastructure using supported programming languages (such as Java, PHP, Ruby and .Net) and platforms/tools (such as Web servers/application servers and databases). This enables PaaS users or organizations to focus on their business and application maintenance rather than having to worry about managing resources, platforms and software versions.

PaaS resides within the space between SaaS and IaaS. IaaS provides network, storage and compute processing capabilities. Examples of IaaS offerings include Amazon EC2, Windows Azure VM Role and RackSpace Cloud Servers. SaaS delivers business software capabilities, such as CRM.

PaaS includes not only the deployment environment, but it also includes repositories, build

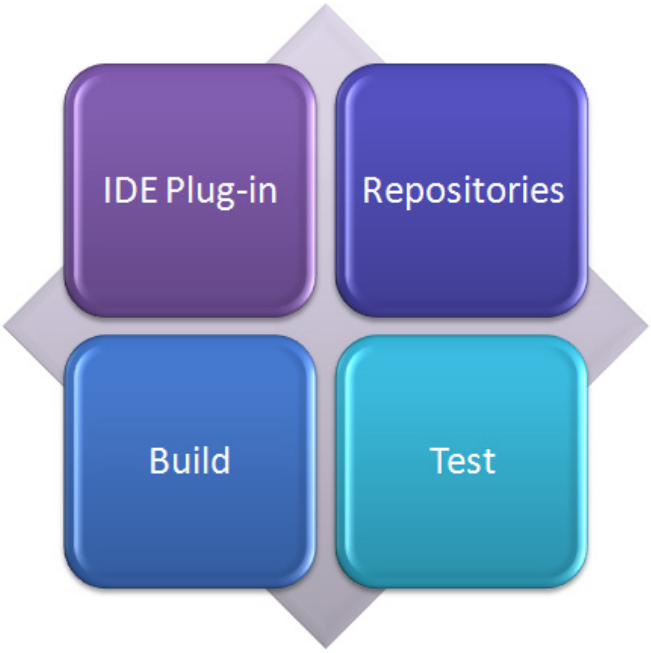


Figure 3. PaaS

environments, testing environments, performance management, mail services, log services, database services, big data services, search services, enterprise messaging services and application performance management for modern application architectures and code inspection services.

PaaS is becoming popular because it eliminates the cost and complexity of acquisition, installation, configuration, evaluation, experimentation and management of all the hardware and software resources needed to run business applications. PaaS provides the infrastructure and platform needed to develop and run applications. By using PaaS, organizations can utilize budgets to develop applications that provide real business value.

PaaS is driving a new era of innovation and business agility. Development and IT teams use PaaS to design, experiment, build, test and deliver customized applications. Hence, application developers and IT teams can focus on application and domain expertise for their business, rather than managing complex hardware and software resources.

Benefits of PaaS

Two significant benefits of using PaaS are cost benefits and faster development and deployment cycles. PaaS provides agility, flexibility and faster time to market to the development, testing and deployment cycles and, hence, focus remains on the application and not in managing resources. It ensures access to resources from anywhere in the world. By using PaaS, user satisfaction increases, and at the same time, resource utilization improves. It provides underlying software and hardware resources on a pay-as-you-go billing model, so it reduces the capital expenditure associated with large amounts of server and storage space, power, cooling, management and maintenance of software updates and changes, and skilled

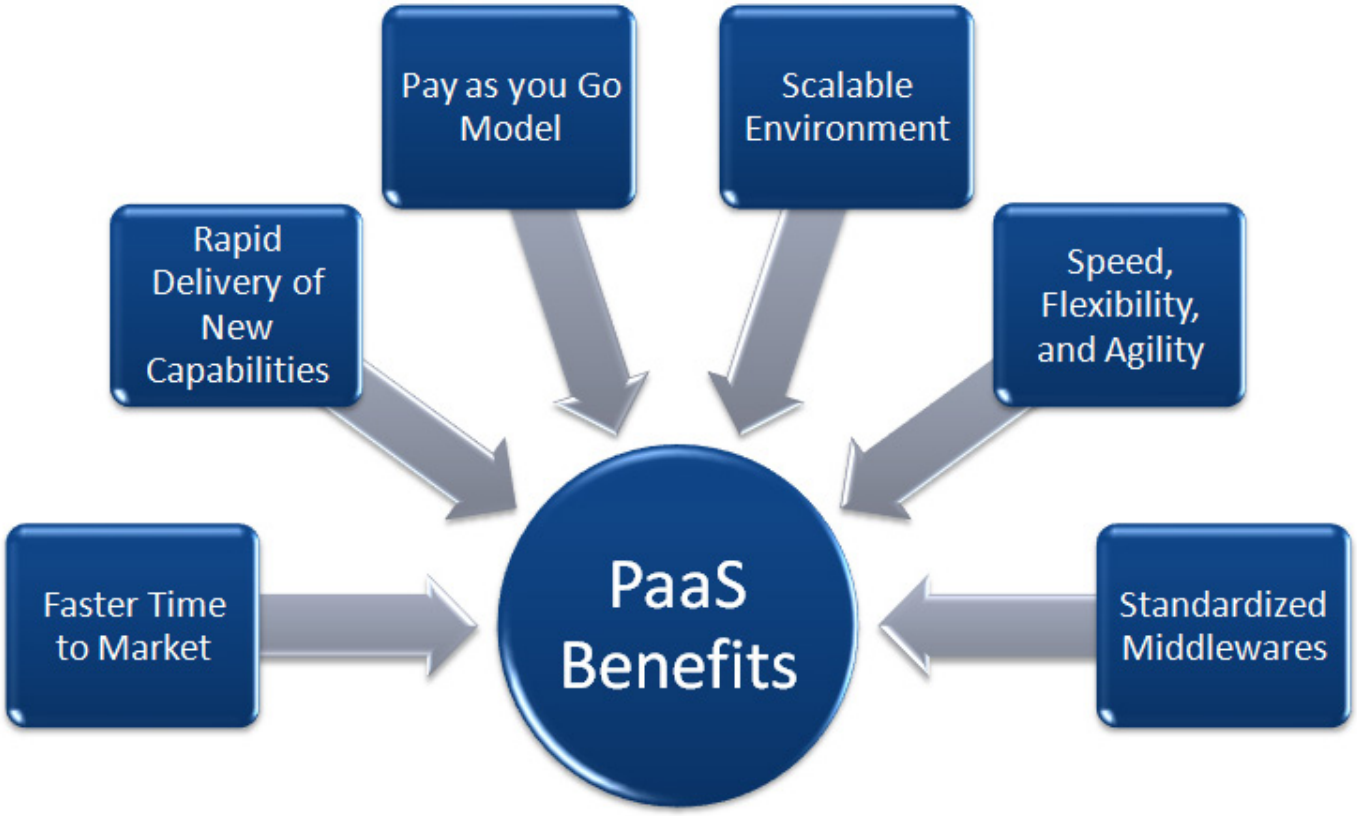


Figure 4. PaaS Benefits

personnel. With almost zero capital expenditure, horizontal or vertical scaling features can increase the application's performance. It does not involve local installation, so adoption speed is usually high. The PaaS platform ensures that consumers don't need to keep investing in OS upgrades and maintenance. It is the PaaS provider's responsibility to manage infrastructure and platform resources so organizations don't need to worry about licenses, versions of software, patch management and so on. Furthermore, flexibility and availability of resources improves collaboration between the development and testing teams.

PaaS offers surprising benefits in deployment and management tasks considering the fact that most organizations opt for PaaS solutions based on the environment that is already standardized in their internal environment.

How to Deploy Applications in PaaS

Here are the basic steps:

- Select the application type, programming language for

developing the application, platform to run the application, build environment and so on.

- Create business or Web applications using an IDE, such as Eclipse, NetBeans or any other IDE.
- Create a database using available database products.
- Change the database configuration accordingly in the application.
- Create an archive file—for example, a WAR or EAR file.
- Upload the archive file to the PaaS Portal.
- Configure log, e-mail, backup and scaling services.
- Access your application.

Major PaaS Providers and Offerings in Cloud Services

PaaS services provide an opportunity to increase customers, boost revenue, add value to existing services and gain broader adoption of cloud services among business customers. Application development is a core capability, while additional capabilities often play to the cloud provider's brand building and strengths. To prepare for the PaaS, cloud providers are developing flexible cloud platforms to support a variety of functions for the cloud environment and processes. Examples of PaaS include, but are not limited to, Force.com, Microsoft Azure, Engine Yard, Heroku, CloudBees and Google App Engine.

Red Hat OpenShift Red Hat OpenShift is a PaaS with Apache License 2.0. It has built-in support for Java, Python, PHP, Perl, Node.js, Ruby and extensible functionality to add languages. OpenShift supports MySQL, PostgreSQL and MongoDB. It supports Web-application frameworks, such as Rack for Ruby, WSGI for Python and PSGI for Perl. For Java, it covers end-to-end support for Java EE6, CDI/Weld, Spring, Liferay, Scala/Play!, JBoss AS7, JBoss EAP6, Tomcat 6 and 7 (JBoss EWS 1.0 and 2.0), Glassfish as DIY, Jetty as DIY, Eclipse, JBoss Tools, Jenkins, Cloud9 IDE, Appcelerator Titanium, Git, SSH access, Maven 3 and Ant. Three versions are available: OpenShift Online, OpenShift Enterprise and OpenShift Origin. A free tier is available for OpenShift Online with 512MB of RAM and 1GB disk. OpenShift Enterprise is a private cloud version from Red Hat.

CloudBees CloudBees supports Java SE and Java EE. It also includes Tomcat, MySQL, commercial relational databases, BIG data—MongoDB and CouchDB in its stack. CloudBees brings quite a few unique features into the Java PaaS landscape, especially continuous integration—an entire development/testing/deployment management in the cloud. Application deployment requires no special framework, and it is easy to migrate applications in or out and to or from CloudBees. Developers can use the Jenkins service to have CloudBees automatically and continuously build, test, check in and check out code in the repository. CloudBees supports command-line tools, IDE tools (such as Eclipse), a Web-based console, Web access to logs, third-party developer/testing services and API access. New Relic monitoring is a part of the CloudBees ecosystem. Users can turn monitoring on for any of the applications with a few clicks. In CloudBees, the New Relic monitoring agent is deployed automatically into your application at the time of deployment. CloudBees also supports sending mail from applications with the SendGrid Service for RUN@cloud.

Google App Engine Google App Engine (GAE) is designed to run Java, Python, Go or PHP applications on the Google infrastructure. In GAE, applications run within a secure environment with limited access to the underlying operating system; thus, existing applications may require significant changes—for example, applications can't write to the filesystem. It provides native support for Google Cloud SQL and Google Cloud Storage. It supports automatic scaling and load balancing. The Google App Engine SDK for Java, Python, PHP and Go are available. Users can use the Google plugin for Eclipse to develop and deploy applications. App Engine Datastore provides a NoSQL datastore, with a query engine and atomic transactions. Google Cloud SQL provides a relational database service based on the MySQL RDBMS, while Google Cloud Storage provides a storage service for objects and files. For free usage, applications can use up to 1GB of storage and adequate CPU and bandwidth to support a capable application, which can serve around 5 million requests a month. It also provides simulation of Google App Engine on your computer.

Cloud Foundry Cloud Foundry is an open-source PaaS developed by VMware and released under the Apache License 2.0. It is written

in Ruby. Cloud Foundry supports the Java, Ruby, Node.js and Scala languages with runtime environments Java 6, Java 7, Ruby 1.8, Ruby 1.9, Node.js, Spring Framework 3.1, Rails and Sinatra. Cloud Foundry supports the MySQL and vFabric Postgres relational databases and the MongoDB document-based database. Users can use RabbitMQ, a reliable, scalable and portable messaging system for applications. Micro Cloud Foundry is a downloadable version of Cloud Foundry that can run on a developer's machine.

Heroku Heroku is a polyglot cloud application platform that supports Clojure, Facebook, Java, Spring, Play, Node.js, Python, Django, Ruby on Rails and Scala. It supports PostgreSQL and MongoDB as SQL and NoSQL databases, respectively. Heroku aggregates three categories of logs: app logs, system logs and API logs. It is available in the US and EU. Users can use the maintenance mode to disable access to their applications for some duration of time, where it will serve a static page to all users. Users can enable SSL to ensure that all information is transmitted securely. Its production check feature is useful to verify application configuration against recommended criteria to ensure maximum uptime.

Windows Azure Windows Azure is a PaaS offering from Microsoft. Microsoft released it in February 2010. Application developers can write code for it in different programming languages; there are definite SDKs started by Microsoft for Java, Python, Node.js and .NET. Microsoft publishes the source code for the client libraries on GitHub. For data storage, SQL Azure is a cloud-based scalable and highly available database service built on SQL server. Windows Azure Blobs provide storage for unstructured binary data. AppFabric simplifies connecting to cloud services and on-premise applications. For queued messaging, queues and service bus features are available. With the use of the caching and content delivery network, application performance can be enhanced. Broadcasters successfully used Windows Azure Media Services to stream the London 2012 Olympics. Windows Azure Active Directory manages user information similar to Windows Server Active Directory.

Amazon Elastic Beanstalk Elastic Beanstalk is a PaaS offering from AWS to deploy and manage applications quickly in the AWS cloud. Elastic Beanstalk manages auto-scaling, load balancing and application monitoring. AWS Elastic Beanstalk supports Java, PHP, Python, Node.js, Ruby and .NET

Web applications with the Apache Tomcat, Apache HTTP Server, Nginx, Passenger and Microsoft IIS 7.5 development stacks, respectively. It runs on the Amazon Linux AMI, Windows Server 2008 R2 AMI and the Windows Server 2012 AMI. Consumers can create 25 applications and 500 application versions. The application size can be up to 512MB. Users can use Eclipse and Visual Studio to deploy applications to AWS Elastic Beanstalk. Users can leverage the AWS Toolkit for Eclipse and the AWS Toolkit for Visual Studio for Java applications and .NET applications, respectively. Application files and optionally server log files are stored in Amazon S3. Amazon RDS, DynamoDB and SimpleDB can be used as a datastore, or users can use Oracle, Microsoft SQL Server or any other relational databases running on Amazon EC2.

Security in PaaS

It is essential that application developers are thoroughly experienced in application security best practices. This can include secure coding practices in the proffered language as well as in secure design principles. Unlike traditional application development, PaaS offers a shared development environment, so authentication, authorization and access control can be combined to ensure consumers' data and application security. It is also important to scan Web applications for common security issues, such as cross-site scripting (XSS) and SQL injection. To refine development efforts to produce secure and robust applications, application threat modeling is critical. OWASP threat modeling can be used to create secure applications.

Future of PaaS

Many IaaS providers are moving up in the stack of service models. AWS is one of the examples. Today, the market for PaaS may be the smallest proportion of the overall public cloud. But, it will have huge implications in application developers' roles and responsibilities. Nearly half of respondents (49.6%) to the 2013 TechTarget Cloud Pulse Survey said they chose the PaaS they did because it was part of a cloud ecosystem they already were using.

According to 451 Research, PaaS is the fastest growing area of cloud computing and is projected to attain a 41% compound annual growth rate (CAGR) through 2016—24% of total cloud revenues.

Similarly, 43% of respondents said they based their decision on how well the PaaS would integrate with existing architecture. Now, PaaS providers have realized that it will be more beneficial from a market and competition point of view to run services across several IaaS providers' architectures. Some providers are providing private versions of their PaaS offering, so organizations can use the same services in their existing facilities to utilize existing resources in a better manner.■

Mitesh Soni has been associated with the Cloud Services Team for the past three years, which is a part of the Research and Innovation Group of iGATE. Currently, he is working there in the capacity as Technical Lead. He loves to write on technical and social subjects, and he also enjoys photography. His technical blog is at https://abcdofcloud.wordpress.com, and his photography blog is at http://myvividvisions.com.

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

Resources

The NIST Definition of Cloud Computing, version 15: **http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf**

Peeling Back the Layers of the Platform as a Service Market: **http://searchcloudcomputing.techtarget.com/tip/Peeling-back-the-layers-of-the-Platform-as-a-Service-market**

451 Research: Platform-as-a-Service Fastest Growing Area of Cloud Computing: **http://www.cloudcomputing-news.net/blog-hub/2013/aug/23/451-research-platform-as-a-service-paas-fastest-growing-area-of-cloud-computing/?utm_medium=referral&utm_source=t.co**

OpenShift: **https://www.openshift.com**

AppEngine: **https://developers.google.com/appengine/docs**

Heroku: **https://www.heroku.com**

Windows Azure: **http://www.windowsazure.com**

Amazon Elastic Beanstalk: **http://aws.amazon.com/elasticbeanstalk**

Cloud Foundry: **http://www.cloudfoundry.com**

CloudBees: **http://www.cloudbees.com**

SUSAN SONS

Girls and Software

December 2013's EOF, titled "Mars Needs Women", visited an interesting fact: that the male/female ratio among *Linux Journal* readers, and Linux kernel developers, is so lopsided (male high, female low) that graphing it would produce a near-vertical line. I was hoping the piece would invite a Linux hacker on the female side of that graph to step up and move the conversation forward. And sure enough, here we have Susan Sons aka @HedgeMage. Read on.—Doc Searls

Yep, I said "girls". Since men were once boys, but women sprang from the head of Zeus full-grown and fighting like modern-day Athenas, you can start flaming me now for using that nasty word...unless you'd like to see the industry through the eyes of a girl who grew up to be a woman in the midst of a loose collection of open-source communities.

Looking around at the hackers I know, the great ones started before puberty. Even if they lacked computers, they were taking apart alarm clocks, repairing pencil sharpeners or tinkering with ham radios. Some of them built pumpkin launchers or LEGO trains. I started coding when I was six years old, sitting in my father's basement office, on the machine he used to track inventory for his repair service. After a summer of determined trial and error, I'd managed to make some gorillas throw things other than exploding bananas. It felt like victory!

When I was 12, I got my hands on a Slackware disk and installed it on my computer—a Christmas gift from my parents in an especially good year for my dad's company—and I found a bug in a program. The program was in C, a language I'd never seen. I found

my way onto IRC and explained the predicament: what was happening, how to reproduce it and where I thought I'd found the problem.

I was pretty clueless then—I hadn't even realized that the reason I couldn't read the code well was that there was more than one programming language in the world—but the channel denizens pointed me to the project's issue tracker, explained its purpose and helped me file my first bug report.

What I didn't find out about until later was the following private message exchange between one of the veterans who'd been helping me and a channel denizen who recognized my nickname from a mailing list:

> coder0: That was a really well-asked question...but why do I get the feeling he's a 16yo boy?
>
> coder1: Because she's a 12yo girl.
>
> coder0: Well...wow. What do her parents do that she thinks like that?
>
> coder1: I think she's on a farm somewhere, actually.

When coder1 told me about the conversation, I was sold on open source. As a little girl from farm country who'd repeatedly been excluded from intellectual activities because she wasn't wealthy or urban or old enough to be wanted, I could not believe how readily I'd been accepted and treated like anybody else in the channel, even though I'd been outed. I was doubly floored when I found out that coder0 was none other than Eric S. Raymond, whose writings I'd devoured shortly after discovering Linux.

Open source was my refuge because it was a place were nobody cared what my pedigree was or what I looked like—they cared only about what I did. I ingratiated myself to people who could help me learn by doing dull scutwork: triaging issues to keep the issue queues neat and orderly, writing documentation and fixing code comments. I was the helpful kid, so when I needed help, the community was there. I'd never met another programmer in real life at this point, but I knew more about programming than some college students.

It Really Is about Girls (and Boys)

Twelve-year-old girls today don't generally get to have the experiences that I did. Parents are warned to

keep kids off the computer lest they get lured away by child molesters or worse—become fat! That goes doubly for girls, who then grow up to be liberal arts majors. Then, in their late teens or early twenties, someone who feels the gender skew in technology communities is a problem drags them to a LUG meeting or an IRC channel. Shockingly, this doesn't turn the young women into hackers.

Why does anyone, anywhere, think this will work? Start with a young woman who's already formed her identity. Dump her in a situation that operates on different social scripts than she's accustomed to, full of people talking about a subject she doesn't yet understand. Then tell her the community is hostile toward women and therefore doesn't have enough of them, all while showing her off like a prize poodle so you can feel good about recruiting a female. This is a recipe for failure.

Young women don't magically become technologists at 22. Neither do young men. Hackers are born in childhood, because that's when the addiction to solving the puzzle or building something kicks in to those who've experienced that "victory!" moment like I had when I imposed my will on a couple electronic primates.

Unfortunately, our society has set girls up to be anything but technologists. My son is in elementary school. Last year, his school offered a robotics class for girls only. When my son asked why he couldn't join, it was explained to him that girls need special help to become interested in technology, and that if there are boys around, the girls will be too scared to try.

My son came home very confused. You see, he grew up with a mom who coded while she breastfed and brought him to his first LUG meeting at age seven weeks. The first time he saw a home-built robot, it was shown to him by a local hackerspace member, a woman who happens to administer one of the country's biggest supercomputers. Why was his school acting like girls were dumb?

Thanks so much, modern-day "feminism", for putting very unfeminist ideas in my son's head.

There's another place in my life, besides my home, where the idea of technology being a "guy thing" is totally absent: my hometown. I still visit Sandridge School from time to time, most recently when my old math teacher invited me in to talk to students about STEM careers. I'm fairly sure I'm the only programmer anyone in that town has met in

person...so I'm something of the archetypal computer geek as far as they are concerned. If anything, some folks assume that it's a "girl thing".

Still, I don't see the area producing a bunch of female hackers. The poverty, urbanization and rising crime aside, girls aren't being raised to hack any more in my hometown than they are anywhere else. When I talked to those fifth-grade math classes, the boys told me about fixing broken video game systems or rooting their phones. The girls didn't do projects—they talked about fashion or seeking popularity—not building things.

What's Changed?

I've never had a problem with old-school hackers. These guys treat me like one of them, rather than "the woman in the group", and many are old enough to remember when they worked on teams that were about one third women, and no one thought that strange. Of course, the key word here is "old" (sorry guys). Most of the programmers I like are closer to my father's age than mine.

The new breed of open-source programmer isn't like the old. They've changed the rules in ways that have put a spotlight on my sex for the first time in my 18 years in this community.

When we call a man a "technologist", we mean he's a programmer, system administrator, electrical engineer or something like that. The same used to be true when we called a woman a "technologist". However, according to the new breed, a female technologist might also be a graphic designer or someone who tweets for a living. Now, I'm glad that there are social media people out there—it means I can ignore that end of things—but putting them next to programmers makes being a "woman in tech" feel a lot like the Programmer Special Olympics.

It used to be that I was comfortable standing side by side with men, and no one cared how I looked. Now I find myself having to waste time talking about my gender rather than my technology...otherwise, there are lectures:

- The "you didn't have a woman on the panel" lecture. I'm on the panel, but I'm told I don't count because of the way I dress: t-shirt, jeans, boots, no make-up.

- The "you desexualize yourself to fit in; you're oppressed!" lecture. I'm told that deep in my female heart I must really love make-up and fashion. It's not that I'm a geek who doesn't much care how she looks.

- The "you aren't representing women; you'd be a better role model for girls if you looked the part" lecture. Funny, the *rest* of the world seems very busy telling girls to look fashionable (just pick up a magazine or walk down the girls' toy aisle). I don't think someone as bad at fashion as I am should worry about it.

With one exception, I've heard these lectures only from women, and women who can't code at that. Sometimes I want to shout "you're not a programmer, what are you *doing* here?!"

I've also come to realize that I have an advantage that female newcomers don't: I was here before the sexism moral panic started. When a dozen guys decide to drink and hack in someone's hotel room, I get invited. They've known me for years, so I'm safe. New women, regardless of competence, don't get invited unless I'm along. That's a sexual harassment accusation waiting to happen, and no one will risk having 12 men alone with a single woman and booze. So the new ladies get left out.

I've never been segregated into a "Women in X" group, away from the real action in a project. I've got enough clout to say no when I'm told I should be loyal and spend my time working on women's groups instead of technology. I'm not young or impressionable enough to listen to the likes of the Ada Initiative (**http://adainitiative.org**) who'd have me passive-aggressively redcarding (**http://singlevoice.net/redyellow-card-project**) anyone who bothers me or feeling like every male is a threat, or that every social conflict I have is because of my sex.

Here's a news flash for you: except for the polymaths in the group, hackers are generally kind of socially inept. If someone of any gender does something that violates my boundaries, I assume it was a misunderstanding. I calmly and specifically explain what bothered me and how to avoid crossing that boundary, making it a point to let the person know that I am not upset with them, I just want to make sure they're aware so it doesn't happen again. This is what adults do, and it works. Adults don't look for ways to take offense, silently hand out "creeper cards" or expect anyone to read their minds. I'm not a child, I'm an adult, and I act like one.

My Boobs Don't Matter

I came to the Open Source world because I liked being part of a community where my ideas, my skills

and my experience mattered, not my boobs. That's changed, and it's changed at the hands of the people who say they want a community where ideas, skills and experience matter more than boobs.

There aren't very many girls who want to hack. I imagine this has a lot to do with the fact that girls are given fashion dolls and make-up and told to fantasize about dating and popularity, while boys are given LEGOs and tool sets and told to do something. I imagine it has a lot to do with the sort of women who used to coo "but she could be so pretty if only she didn't waste so much time with computers". I imagine it has a lot to do with how girls are sold on ephemera—popularity, beauty and fitting in—while boys are taught to revel in accomplishment.

Give me a young person of any gender with a hacker mentality, and I'll make sure they get the support they need to become awesome. Meanwhile, buy your niece or daughter or neighbor girl some LEGOs and teach her to solder. I love seeing kids at LUG meetings and hackerspaces—bring them! There can never be too many hackers.

Do not punish the men simply for being here. "Male privilege" is a way to say "you are guilty because you

don't have boobs, feel ashamed, even if you did nothing wrong", and I've wasted too much of my time trying to defend good guys from it. Yes, some people are jerks. Call them out as jerks, and don't blame everyone with the same anatomy for their behavior. Lumping good guys in with bad doesn't help anyone, it just makes good guys afraid to interact with women because they feel like they can't win. I'm tired of expending time and energy to protect good men from this drama.

Do not punish hackers for non-hackers' shortcomings. It is not my fault some people don't read man pages, nor is it my job to hold their hand step-by-step so they don't have to. It is not my place to drag grown women in chains to LUG meetings and attempt to brainwash them to make you more comfortable with the gender ratio, and doing so wouldn't work anyway.

Most of all, I'm disappointed. I had a haven, a place where no one cared what I looked like, what my body was like or about any ephemera—they cared about what I could do—and this culture shift has robbed me of my haven. At least I had that haven. The girls who follow me missed out on it.

I remember in those early days, in my haven, if someone was rude or tried to bully me, the people around me would pounce with a resounding "*How dare you* be mean to someone we like!" Now, if a man behaves badly, we're bogged down with a much more complex thought process: "Did this happen because she's a woman?" "Am I white knighting if I step in?" "Am I a misogynist if I don't?" "What does this say about women in technology?" "Do I really want to be part of another gender politics mess?" It was *so* much simpler when we didn't analyze so much, and just trounced on mean people for being mean.■

Susan Sons' passion for education has driven her open-source efforts with Debian Edu, Edubuntu and her own initiative Frog and Owl, which helps technologists connect with educators to build more useful educational tools. She co-authored the first edition of *The Edubuntu Handbook* and *The Definitive Guide to Drupal 7*. Susan has served as a staff member for the Freenode network and founding president of Drupal Group Indianapolis. She designed and implemented a program to help preteens and teens from under-resourced rural communities learn computer science through experimentation and open-source contribution. When not coding or writing, Susan can be found studying Shorei Goju-Ryu Karate, backpacking and geocaching with her son, and volunteering with abuse victims and at-risk populations. She is also an amateur radio operator.

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