

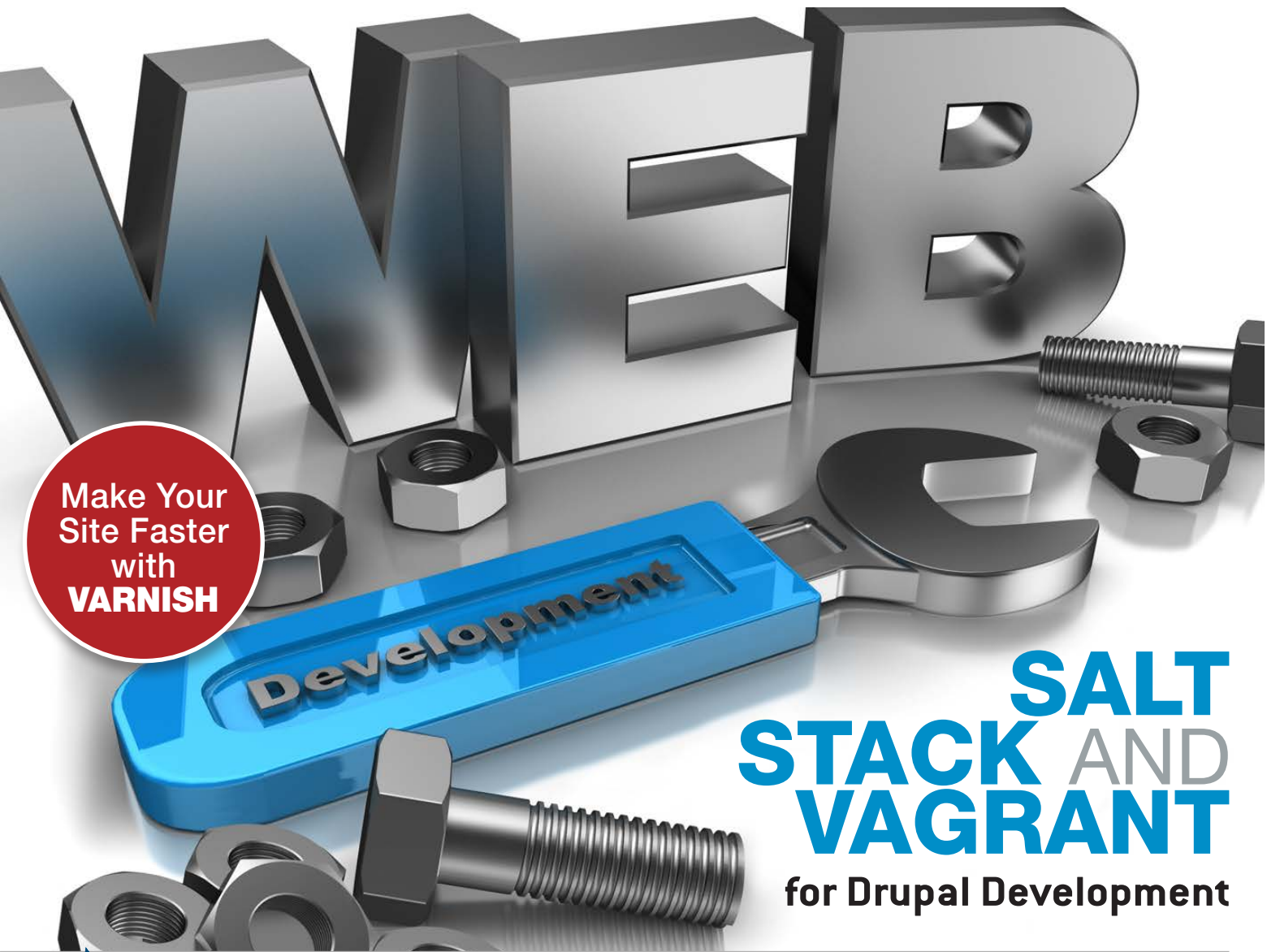
Watir | Android | Salt Stack | Drupal | Varnish | Dart

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

SPONSORED BY
AcquiaTM

MARCH 2013 | ISSUE 227 | www.linuxjournal.com



Make Your
Site Faster
with
VARNISH

**SALT
STACK AND
VAGRANT**

for Drupal Development



HOW -TO:

Get Started
with
Drupal



**A LOOK
AT WATIR**

Web App
Testing in Ruby

**GOOGLE
DART**

Break Away
from JavaScript

ACQUIA™

Inc.
500

2012 **#1** Software Vendor and
#8 Fastest Growing Overall

Score an **A+** with
our **free Drupal tool**
and get a **free shirt**



Insight

Grade your site on performance,
security, SEO, and best practices.

Go to <https://acquia.com/instant-insight>,
enter your URL & get a free site scan

SILICON MECHANICS



visit us at www.siliconmechanics.com or call us toll free at 888-352-1173

RACKMOUNT SERVERS STORAGE SOLUTIONS HIGH-PERFORMANCE COMPUTING

**“Just because
it’s badass,
doesn’t mean
it’s a game.”**

Pierre, our new Operations Manager, is always looking for the right tools to get more work done in less time. That’s why he respects NVIDIA® Tesla® GPUs: he sees customers return again and again for more server products featuring hybrid CPU / GPU computing, like the Silicon Mechanics Hyperform HPCg R2504.v3.

We start with your choice of two state-of-the-art processors, for fast, reliable, energy-efficient processing. Then we add four NVIDIA® Tesla® GPUs, to dramatically accelerate parallel processing for applications like ray tracing and finite element analysis. Load it up with DDR3 memory, and you have herculean capabilities and an 80 PLUS Platinum Certified power supply, all in the space of a 4U server.



When you partner with Silicon Mechanics, you get more than stellar technology - you get an Expert like Pierre.

Expert included.

CONTENTS

MARCH 2013

ISSUE 227

WEB DEVELOPMENT

FEATURES

70 Things a Front-End Developer Should Know about Drupal

Explore how to get started with a custom theme and work with templates, JavaScript and CSS. Plus, a look at considerations for mobile, performance tips and what's next in Drupal 8.

Alexander Castillo

84 Using Salt Stack and Vagrant for Drupal Development

Build instant Drupal development environments with Vagrant and Salt Stack.

Ben Hosmer

96 Dart: a New Web Programming Experience

Introducing Dart, the new Web language from Google.

James Slocum



COVER IMAGE: © Can Stock Photo Inc. / mmaxer

INDEPTH

110 Speed Up Your Web Site with Varnish

Your Web server called to say it's tired. Use Varnish to give it the break it deserves.

Pablo Graziano

COLUMNS

42 Reuven M. Lerner's At the Forge

Watir

50 Dave Taylor's Work the Shell

Cribbage: Calculating Hand Value

54 Kyle Rankin's Hack and /

Temper Pi

60 Shawn Powers' The Open-Source Classroom

Dynamic DNS—an Object Lesson in Problem Solving

130 Doc Searls' EOF

Android for Independence

IN EVERY ISSUE

8 Current_Issue.tar.gz

12 Letters

20 UPFRONT

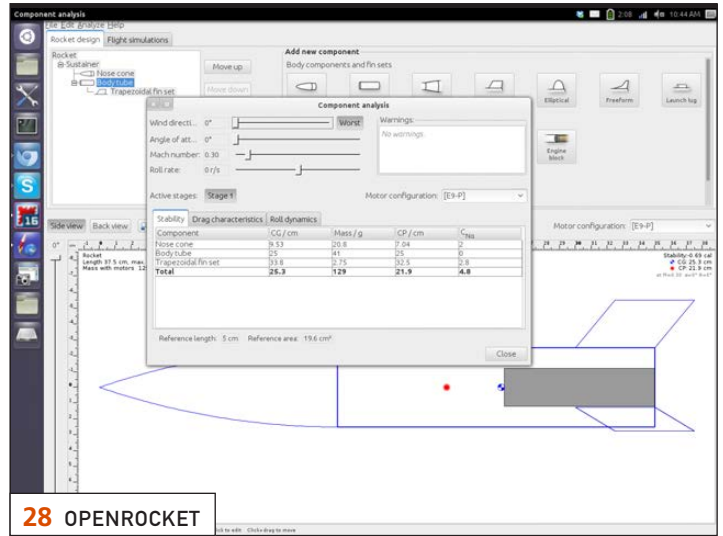
40 Editors' Choice

66 New Products

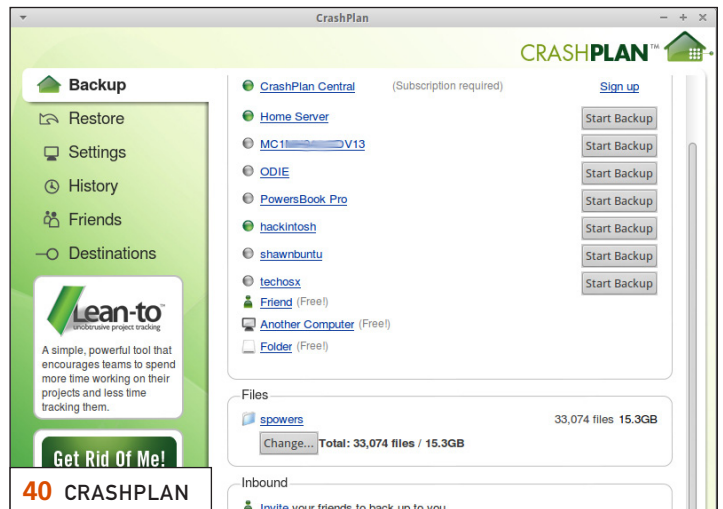
135 Advertisers Index

ON THE COVER

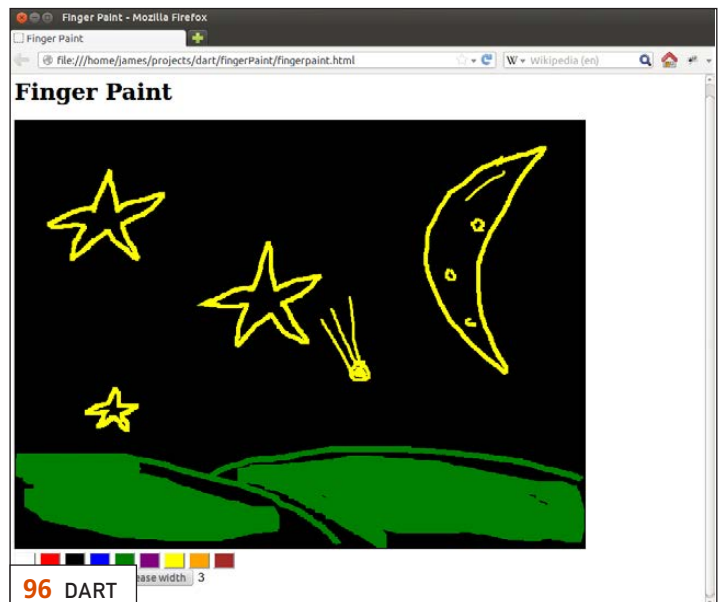
- Make Your Site Faster with Varnish, p. 110
- Salt Stack and Vagrant for Drupal Development, p. 84
- How-To: Get Started with Drupal, p. 70
- A Look at Watir—Web App Testing in Ruby, p. 42
- Google Dart—Break Away from JavaScript, p. 96



28 OPENROCKET



40 CRASHPLAN



96 DART

LINUX JOURNAL™

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



ENJOY:

Timely delivery

Off-line reading

Easy navigation

Phrase search
and highlighting

Ability to save, clip
and share articles

Embedded videos

Android & iOS apps,
desktop and
e-Reader versions

SUBSCRIBE TODAY!

LINUX JOURNAL

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

Publisher	Carlie Fairchild publisher@linuxjournal.com
------------------	--

Director of Sales	John Grogan john@linuxjournal.com
--------------------------	--------------------------------------

Associate Publisher	Mark Irgang mark@linuxjournal.com
----------------------------	--------------------------------------

Webmistress	Katherine Druckman webmistress@linuxjournal.com
--------------------	--

Accountant	Candy Beauchamp acct@linuxjournal.com
-------------------	--

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

PO Box 980985, Houston, TX 77098 USA

Editorial Advisory Panel

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

Advertising

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

Subscriptions

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

LINUX is a registered trademark of Linus Torvalds.

TrueNAS™ Storage Appliances Harness the Cloud



Unified. Scalable. Flexible.

Thanks to the Intel® Xeon® Processor 5600 series and high-performance flash, every TrueNAS Storage appliance delivers the utmost in throughput and IOPS.

As IT infrastructure becomes increasingly virtualized, effective storage has become a critical requirement. iXsystems' TrueNAS Storage appliances offer high-throughput, low-latency backing for popular virtualization programs such as Hyper-V, VMWare®, and Xen®. TrueNAS hybrid storage technology combines memory, NAND flash, and traditional hard disks to dramatically reduce the cost of operating a high performance storage infrastructure. Each TrueNAS appliance can also serve multiple types of clients simultaneously over both iSCSI and NFS, making TrueNAS a flexible solution for your enterprise needs.

For growing businesses that are consolidating infrastructure, the **TrueNAS Pro** is a powerful, flexible entry-level storage appliance. iXsystems also offers the **TrueNAS Enterprise**, which provides increased bandwidth, IOPS and storage capacity for resource-intensive applications.

Call **1-855-GREP-4-IX**, or go to **www.iXsystems.com**

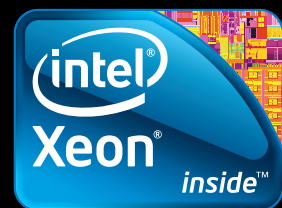
- ✓ Supports iSCSI and NFS exports simultaneously
- ✓ Compatible with popular Virtualization programs such as Hyper-V, VMware, and Xen
- ✓ 128-bit ZFS file system with up to triple parity software RAID

TrueNAS Pro Features

- One Six-Core Intel® Xeon® Processor 5600 Series
- High Performance Write Cache
- Up to 480GB MLC SSD Cache
- Up to 220 TB SATA Capacity
- Quad Gigabit Ethernet
- 48GB ECC Memory

TrueNAS Enterprise Features

- Two Six-Core Intel® Xeon® Processors 5600 Series
- Extreme Performance Write Cache
- Up to 1.2TB High Performance ioMemory
- Up to 500TB SATA or 320TB SAS Capacity
- Dual Ten Gigabit Ethernet
- 96GB ECC Memory





SHAWN POWERS

Remembering Spidey

Back before Google was born, and even longer before it became a verb, the World Wide Web was often searched by a little spider on a surfboard. Webcrawler was to many of us in the mid 1990s what Google is to the entire planet now. Of course, now I use Google to search the Internet, but in its day, that little spider was the gateway to knowledge. Times have changed, and the Internet has grown. Likewise, Web development in general has changed drastically during the past 20 years. Gone are the blink tags and “under construction” animated GIFs. Gone is Geocities. Even the giant AOL is a shadow of its former self. This month, we look at Web development as it is today, built on the best Web development platform available, which in our opinion, is Linux.

Our own Katherine Druckman starts the issue with a look at Drupal and what to expect with Drupal 8—specifically from the perspective of Drupal community members from all over the world. It’s no

secret Katherine loves Drupal, and she’s able to share with us the views of kindred spirits. Feeling right at home himself this issue, Reuven M. Lerner teaches how to use Watir, a tool for browser-testing Ruby code without the need to start up and navigate the various browsers manually. If you test your code (and you should), Watir is worth checking out.

Dave Taylor continues his series on *Cribbage* this month, and turns a game I have always called “that one with the pegs” into an interesting and fairly complex script. If you like math, you’ll fall in love with Dave’s version of *Cribbage*. Kyle Rankin deals with Pi this month, but unlike Dave, Kyle’s Pi is of the raspberry variety. He returns to his beer fridge, and makes it both more efficient and more modern. We’re sure there are things the Raspberry Pi can’t do, but so far, they elude us.

Even I get into the spirit of Web development a bit this month. Granted my contribution is about three lines of PHP, but it’s an integral

part of my column on problem solving with Linux. My dynamic DNS service decided to delete my account one day, so I decided to use Linux tools and scripting hacks to fix the problem on my own. If you need a primer on thinking outside the box, I guarantee my method doesn't exist in any box.

Alexander Castillo describes all the things a front-end developer should know about Drupal. Every version of Drupal brings new features, and for the last few releases, the learning curve has been declining steadily. Whether you're creating entire Drupal themes or just want to customize some CSS, Alexander's article is invaluable.

Developing code on your own dev box is a noble tradition and one that works well for many environments. Once the environment grows to include multiple developers, multiple environments for test and production and multiple locations, keeping things similar can be a nightmare. Ben Hosmer looks at Salt Stack and Vagrant this month. Although the two might sound like pirate names, they're actually a set of applications that can keep your development environments similar, with very little effort.

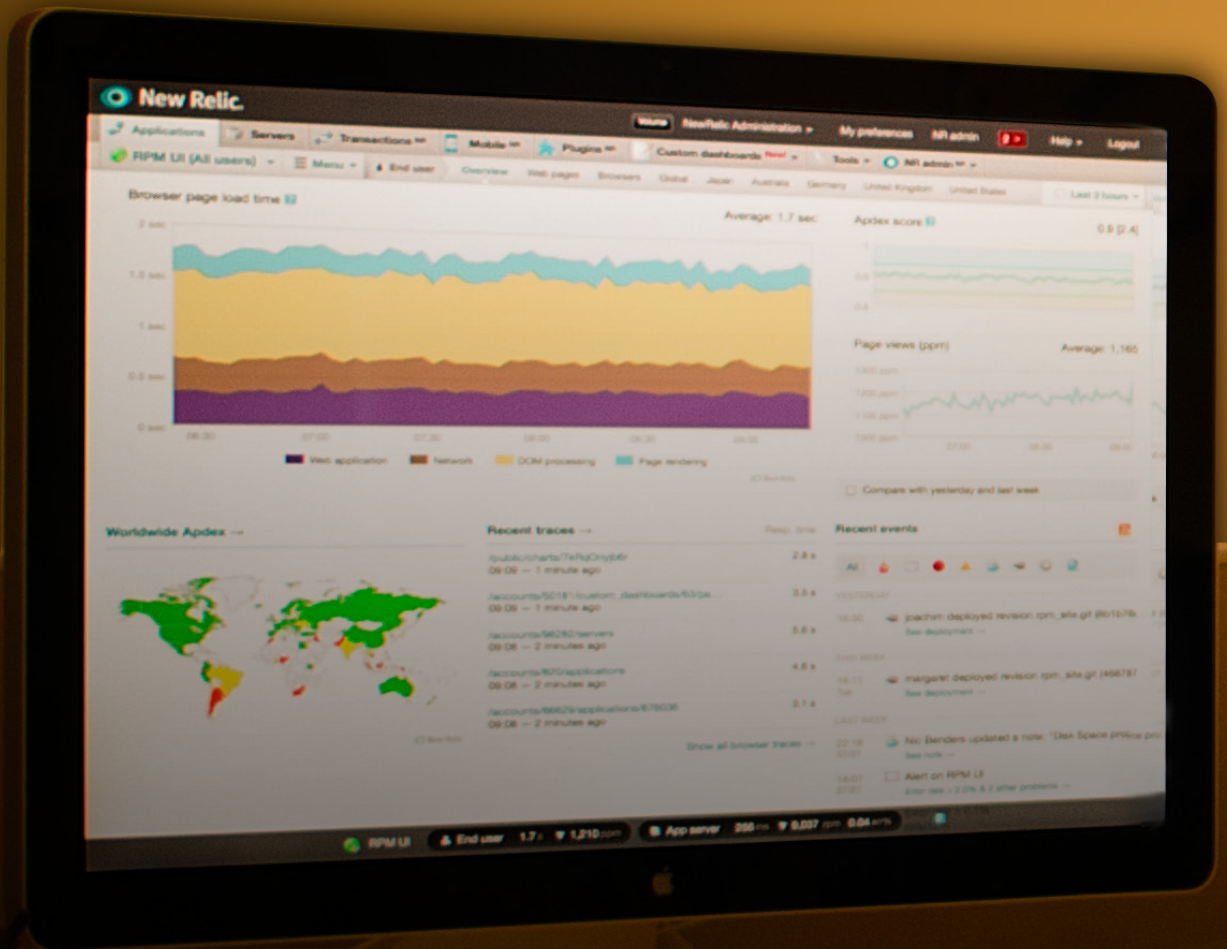
If you think JavaScript has had its day in the sun, and it's time for something newer, you're in luck. This month, James Slocum demos a completely different

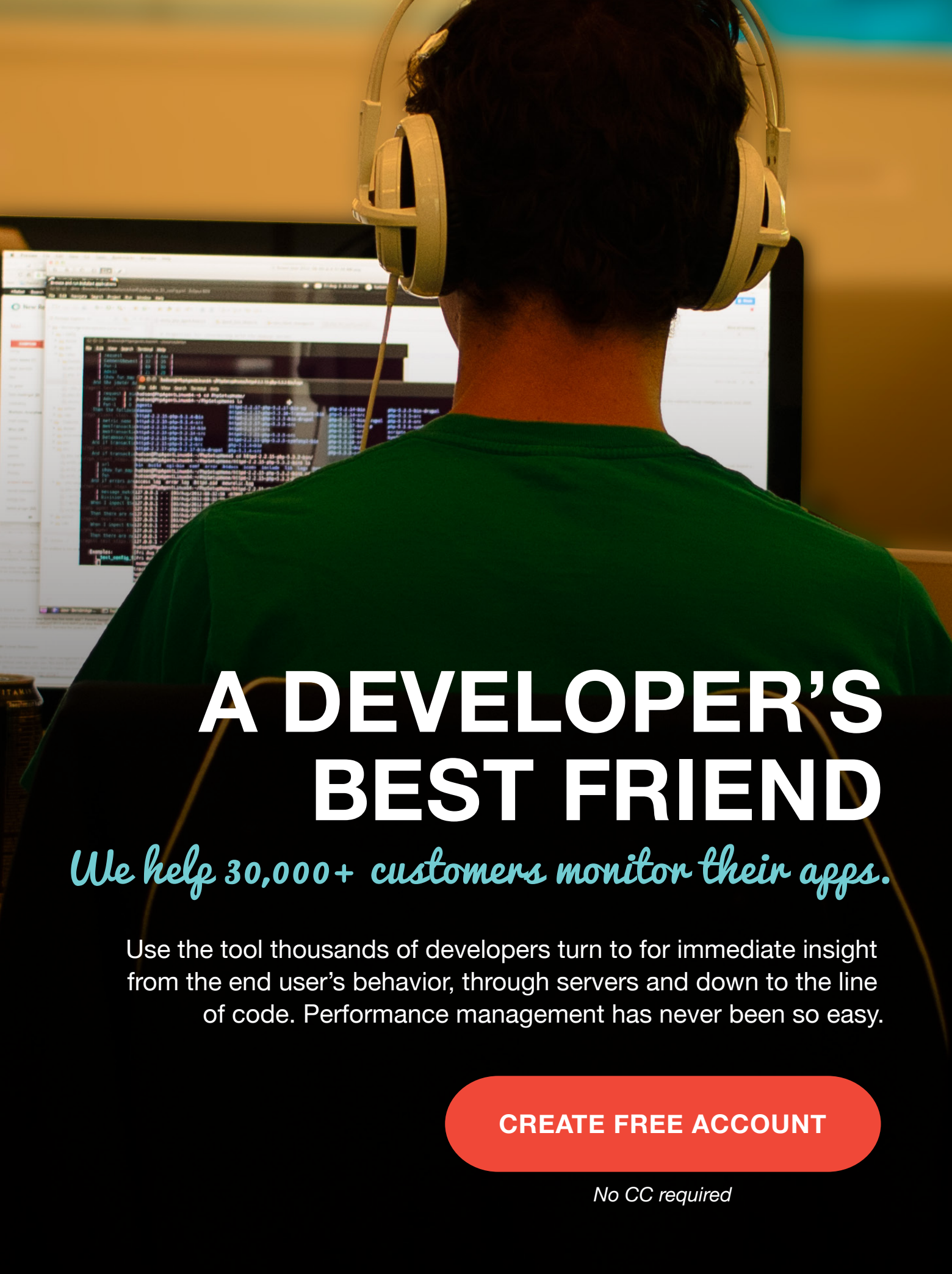
tool for interactive Web programs—Dart. The language is from Google, which implies it's not a fly-by-night idea. Dart is still very new, but the idea is exciting. James explains how it works and even gives a demonstration of it in action.

The Internet and the World Wide Web aren't going away any time soon. In fact, as Web sites become more and more complex, scaling to meet demand is a challenge. Pablo Graziano shows how to use Varnish as a reverse-caching proxy to speed up server response time and help scale heavy loads. Pablo walks through setting up, configuring and tweaking your system to squeeze every bit of performance possible out of your Web servers.

When Spidey the surfboard-riding Webcrawler was first introduced, it indexed a whopping 4,000 Web sites. The Web has grown significantly since then, and thanks to Web developers, its usefulness has grown as well. Whether your first search engine was Webcrawler or you were born after Google became a verb, this issue should be interesting. I know we liked putting it together. ■

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.





A DEVELOPER'S BEST FRIEND

We help 30,000+ customers monitor their apps.

Use the tool thousands of developers turn to for immediate insight from the end user's behavior, through servers and down to the line of code. Performance management has never been so easy.

[CREATE FREE ACCOUNT](#)

No CC required

letters



Bash Notational Shortcuts II

Regarding the “Bash Notational Shortcut” letter in the December 2012 issue: whitespace

is allowed in Fortran variable names, as is in Fortran keywords. It is simply ignored. Here’s a small example:

```
hoel@pchoel:~> cat lj224.f
  p r o g r a m l j 2 2 4
  w r i t e (*,*) "Hello LinuxJournal"
  e n d
hoel@pchoel:~> ifort lj224.f
hoel@pchoel:~> ./a.out
  Hello LinuxJournal
hoel@pchoel:~> gfortran lj224.f
hoel@pchoel:~> ./a.out
  Hello LinuxJournal
```

—Berthold Höllmann

Dave Taylor replies: *I find it awesome that we’re still talking about Fortran. Now, what about Algol-68?*

Readers’ Choice Awards 2012

I would really like to see LibreOffice as a contender for next year. It should have had its own slot for the 2012 awards. It has been around and doing great on its own for some time now. Thanks for your consideration. As you noted in the article, many even wrote in LibreOffice.

—Fran Parker

I agree. It’s funny, the difficult part every year is coming up with the correct options!—Ed.

Digital LJ and Dropbox Sync

I love the digital version of the magazine! But, I would like to suggest you make the digital versions available via Dropbox sync, much like O’Reilly and Pragmatic Bookshelf do with their digital books. I find myself logging in to the LJ site and downloading the PDF versions for safe-keeping into a Dropbox folder (I already get LJ on my iPad mini via Newsstand). It would be awesome if this was automatic with Dropbox sync!

Keep up the great work!

—Terry Dunlap

That’s an interesting idea, Terry. I do the

same thing, actually. I'm not sure if it is easy with our current distribution model, but I like the idea. We'll look into it!—Ed.

January 2013 Work the Shell

I've been a Linux user and reader of *Linux Journal* since the beginning, and I always find something valuable in every issue of your magazine. One of my favourite columns is Work the Shell, because I'm always looking for neat ways to process files and data in different ways. Often—no offense, Dave—it is the letters from readers that provide that aha moment!

This time (regarding Dave Taylor's column the January 2013 issue), I have a comment of my own. When attempting to shuffle a deck of cards, why not just use one of the well-known shuffle algorithms? Then, there is no need to worry about picking a card for the second (or third) time on reaching the end of the deck. Here is a piece of code for the Fisher-Yates algorithm, courtesy of Wikipedia:

```
#!/bin/bash

for ((i=0;i<52;i++))
do
deck[$i]=$i
```

```
done
for ((i=51; i>0; i-- ))
do
rand=$(( $RANDOM % (i+1) ))
tmp=${deck[$i]} deck[$i]=${deck[$rand]}
deck[$rand]=$tmp
done
echo ${deck[@]}
```

I'm sure this can be improved, but you get the idea.

—**dik harris**

Dave Taylor replies: *Thanks for your note and sample code. It's ironic that when I was in college studying for my degree in computer science I complained about having to re-invent the wheel when Knuth's The Art of Computer Programming was on my shelf and had tons of great algorithms. Zoom forward, and now I'm re-inventing the wheel in every column rather than looking for optimal results.*

But here's the thing: while the code you sent me might be a good "shuffle", it's completely obfuscated, and since my primary goal with the Work the Shell column is to explain and explore the concepts underlying good script writing, using a 12-line block of complex code seems counter-productive.

I'll simply suggest we publish your letter and my response and see how the rest of the reader community feels about the subject.

RPi

I've been thinking of getting a Raspberry Pi to make a digital video-capture device for our wood-turning chapter (BAW). We currently use an old tape video camera. Then we have to transfer from tape to a PC, edit the video and produce it. We have two small cameras attached to tripods and can switch back and forth between them to show the demo on a 46" flatscreen so everyone watching the demo, up to 40 people, can see it better. But, when making a video, we have only the one camera! We sure would like to improve the process. What do you think?

—Rex

***Shawn Powers replies:** I like the idea of using a little Pi device, but I worry it might not have the processing horsepower to process and store video. You might consider a couple decent USB Webcams with long cables connected to a single beefy computer. Either solution sounds more fun than video tape, however, so I think you're headed in the right direction!*

WiFi Analyzer

I have a picture of Tux alongside the Android robot on the outside of my cubicle. The going joke in the office is that I can analyze the Wi-Fi with my phone, but my co-engineers cannot because they all have iOS. Another good app is WiFi Map Maker by Dave McKellar. It really helped me figure out which neighbor is flooding my house with his Wi-Fi signal. Thanks for the article and good work.

—Roman

***Shawn Powers replies:** I love that app too! I've never had a good enough GPS signal indoors to use it very well. Your scenario sounds perfect. I'm glad it worked for you.*

Memorable Password Generator

After reading this XKCD comic about developing easy-to-remember, hard-to-crack passwords: <http://xkcd.com/936>, I stumbled upon this page for manually generating a strong password with six-sided die: <http://world.std.com/~reinhold/diceware.html>.

Now, rolling a bunch of die to select a password is rather cumbersome (I'm not a table-top RPG geek), so I developed a script that will select any number of words (by default, five) and

select a pseudo-random list of words from the Diceware word list (see the article linked above for the list):

```
ppgen.sh:
#!/bin/bash

# Copyright © 2012, Trey Blancher
# Licensed under a Creative Commons Attribution-ShareAlike
# 3.0 Unported license.
#
# Inspired by XKCD
# http://xkcd.com/936/
#
# adapted from Diceware's Passphrase table-top generator
# http://world.std.com/~reinhold/diceware.html

while getopts "f:n:" opt; do
    case $opt in
        f) # Process file argument
            #echo "File argument passed: $OPTARG" >&2
            WORDLIST=$OPTARG
            ;;

        n) # Process number argument
            #echo "Number argument passed: $OPTARG" >&2
            NUM=$OPTARG
            ;;

        \?) echo "Invalid option: -$OPTARG" >&2
            echo "usage: $0 [-f <word list>] [-n <number
            of words in passphrase>]" >&2
            exit 1
    esac
done
```

```
;;
esac
done

if [ -z $WORDLIST ]; then
    # Default word list comes from Diceware (see above link)
    WORDLIST="/usr/share/diceware/diceware.wordlist.asc"
fi

if [ -z $NUM ]; then
    NUM=5
fi

#echo "$0 -f $WORDLIST -n $NUM" >&2

# Algorithm uses six-sided die
D=6

# Initialize RANDOM variable
RANDOM=$RANDOM

for ((i=1;i<=$NUM;i++));
do
    for d in {1..5} # five digits per word in master list
    do
        WORD+=$((($RANDOM % $D)+1)
    done
    grep $WORD $WORDLIST | awk '{print $2}' | awk '{ printf "%s ",
$0
}'
    WORD=
done
echo
```

This script lets you choose how many words you want in your generated password and uses some tips on shell programming from Dave Taylor. There probably are better ways to do what I've done here, but this works for me. I use it any time I need a password I can remember (without writing it down or storing it in a password manager).

—Trey

Shawn Powers replies: *I love Randall Monroe. When I saw his comic (days after writing my column, of course), it made me think how silly passwords are in general. Your script sounds like a good idea to me. Thanks for sharing.*

Platform-Agnostic Backup

I just read the January 2013 issue of *LJ* and noticed the letter about a backup system for Linux. Crashplan looks interesting. I haven't tried it yet, but I plan to play around with its free offerings at a minimum. The trial looks like a good way to test the full experience too. I run both Mac and Linux computers, and I like that all major OSes are supported.

—Nathan

Shawn Powers replies: *I happen to agree with you. In fact, I wrote a piece about it in this issue's Upfront section. I use it everywhere, and I really appreciate the generous free-feature offering.*

Another Vote for Crashplan

It works on Solaris, Linux, Mac and Windows. I use it on Solaris and Windows machines, and it has been completely set-and-forget for me for a number of years now. I do the occasional restore to make sure that it is working as expected.

Oh, Crashplan also has an Android application that is restore-only. It's useful for getting files onto phones and tablets.

—Gary Schmidt

Shawn Powers replies: *As with my response to Nathan, I completely agree. Crashplan is awesome.*

Mint+Google?

Has Linux Mint sold its soul to Google? See <http://forum.linuxmint.com/viewtopic.php?f=157&t=108859>.

—B.r. Mintman

I can't speak to Linux Mint's procedures and goals. I can only give my take on using Mint in the past: I thought it was very user-friendly and configured well out of the box, but I didn't care for it much myself. I don't think I'm the intended target, however, so that's okay.

I know Linux Mint always has had a custom Google search in the Firefox browser as a way to raise some capital. It's possible there is more Google interaction, but I honestly don't know. The great thing about Linux is that if you don't like the philosophy and/or actions of one company, there are many other distros from which to choose!—Ed.

Elliptic Curve and Putty

Regarding the "Elliptic Curve Cryptography" article in the January 2013 issue: Putty unfortunately doesn't support ecc/ecdsa keys, but it's on the wish list.

—Wes

Could You Review Slax Linux?

Could you please take the time to review Slax Linux? I have never seen this version of Linux reviewed by your magazine in the past or present and


would love to see what you think of it. Here is the URL: <http://www.slax.org>.
—Christina Cross


I've covered Slax in a video review before. If there are cool new features or significant changes, we might take a look at it. I do recall liking the look of Slax, and I appreciated its underlying Slackware roots.—Ed.

System on Module

- Atmel ARM9 400 MHZ Fanless Processor
- Up to 128 MB of DDR2 SDRAM
- Up to 1GB of NAND Flash
- Up to 8 MB Serial Data Flash
- 6 Serial Ports, 2 I2C and 2 SPI ports
- 2 Full Speed USB 2.0 Host ports
- 1 Full Speed USB 2.0 Device port
- CAN 2.0 B Controller, I2S Audio Port
- 10/100 BaseT Fast Ethernet with PHY
- Access to Processor Bus
- 5 Channels of 10-Bit A/D & 32 GPIO Lines
- SD/MMC Flash Card Interface
- System Reset, Real Time Clock
- Timers/Counters, PWM controller
- Small, 144 pin SODIMM form factor (2.66" x 1.50")

Wide Temperature
SoM-9x25



Designed and manufactured in the USA the SoM-9x25 uses the same small SODIMM form-factor utilized by other EMAC SoM modules and is the ideal processor engine for your next design. All of the ARM processor core is included on this tiny board including: Flash, Memory, Serial Ports, Ethernet, SPI, I2C, I2S Audio, CAN 2.0B, PWMs, Timer/Counters, A/D, Digital I/O lines, Clock/Calendar, and more. The SoM-9x25 is designed to plug into a custom or off-the-shelf carrier board containing all the connectors and any additional I/O components that may be required. The System on Module approach provides the flexibility of a fully customized product at a greatly reduced cost. Quantity 1 price begins at \$180.

<http://www.emacinc.com/som/som9x25.htm>

Since 1985
OVER
28
YEARS OF
SINGLE BOARD
SOLUTIONS

EMAC, inc.

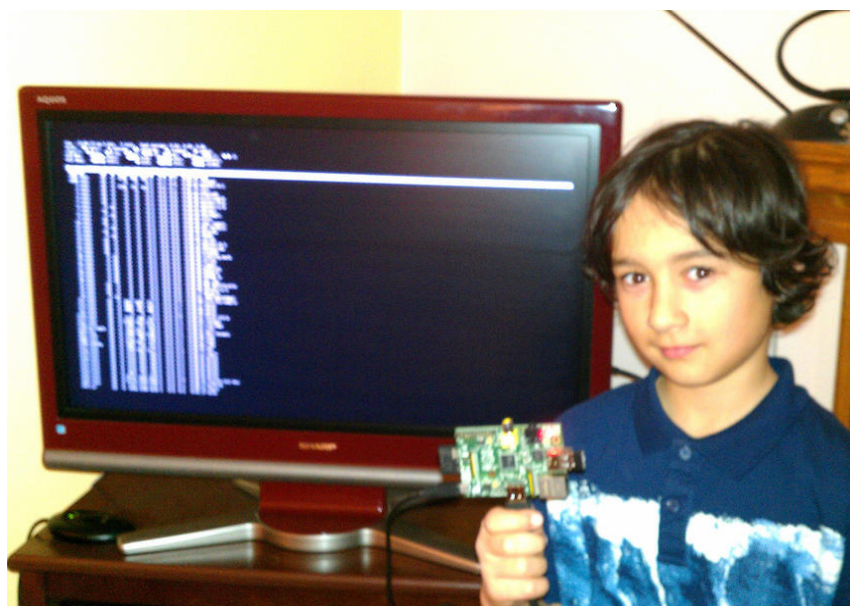
EQUIPMENT MONITOR AND CONTROL

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

Photo of the Month

My wife bought me a Raspberry Pi for my birthday, and I thought you might get a kick out of this picture of my son holding it on its first boot. It's connected to the flatscreen, with top running in the login shell.

—Michael Soulier



First Boot of a Pi

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

PHOTO OF THE MONTH

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

LINUX JOURNAL

At Your Service

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

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

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

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

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

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

1&1 DEDICATED SERVERS

1&1

YEARS

25

1&1 is celebrating its 25th anniversary.

Over the past 25 years 1&1 has grown to become one of the world's leading web hosts. Today, with 12 million customer contracts and 5000 employees 1&1 provides superior web hosting and server solutions to support your business. In celebration here is a gift from us to you.



Our data centers offer top security, Cisco firewall protection and maximum uptime.

- ✓ Unlimited traffic with no extra cost
- ✓ Parallels Plesk® Panel 11 for unlimited domains
- ✓ Exclusive to 1&1: Optional SUSE Linux Enterprise Server
- ✓ Mobile Server Monitoring for Android/iOS devices



1and1.com

Save \$360

Server XL 6

AMD Hexa-Core

6 Cores x 2,8 GHz
(3.3 GHz Turbo Core)

16 GB RAM DDR3 ECC

1,000 GB (2 x 1,000 SATA)

Software RAID 1

Free choice of CentOS, Debian, Ubuntu, or openSUSE.

Unlimited Traffic (100 Mbit/s)

~~\$129.99~~ **\$99.99** per month*

**OFFER ENDS
02/28/13**



* Offer valid for a limited time only. First year save \$360 off regular price of Dedicated Server XL 6. Other terms and conditions may apply. Visit www.1and1.com for full promotional offer details. Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners. © 2013 1&1 Internet. All rights reserved.

Intel, the Intel logo, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and/or other countries.

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Sasha Levin wrote a new **hash table** implementation for the kernel. Typically in a big, decentralized software project, developers aren't necessarily aware of the cool stuff being implemented in the various nooks and crannies of the code, and there's a tendency for lots of alternative implementations to crop up everywhere. Hash tables are a good candidate for that. Sasha's patch was intended to provide a generic hash table that all kernel developers can use.

Something like this can never find users unless it's really fast. Sasha's implementation uses a variety of speed-enhancing optimizations, including doing as much as possible at compile time via macros. But, macros have their own little syntactic idiosyncrasies that can require a little extra care during use. **Mathieu Desnoyers** and **Tejun Heo** pointed out some improvements and bugs to Sasha's code, addressing some of those problems.

Darrick J. Wong took the trouble to create a nearly three-hour

source animation of Linux kernel development history, going all the way back to the beginning: <http://www.youtube.com/watch?v=pOSqctHH9vY>.

It's lovely to watch. It's also interesting because gource relies on having a computer-readable log of a project's development history, while the early history of Linux is more or less a mystery. **Linus Torvalds** accepted vast numbers of patches in the 1990s, but he just used the UNIX diff and patch commands to apply them. The only real records of the patches were e-mail logs and Usenet logs, but these don't give clear indications of which patches actually were accepted into the kernel, and when. It was only with the arrival of **BitKeeper**, and later of **git**, that a detailed record of Linux's history began to be compiled.

There have been some efforts to re-create some of that early history into its own git tree, but this is nowhere near as detailed as the kind of historical records being created these days. The beginning of Darrick's

video, therefore, just shows Linus doing everything for a while and then being joined by other developers, presumably once the historical record became more accurate.

After six and a half years of maintaining the **KVM** code (the Kernel Virtual Machine), **Avi Kivity** has stepped down. His co-maintainer **Marcelo Tosatti**, formerly also the Linux 2.4 maintainer, was left briefly as the sole KVM maintainer, until **Gleb Natapov** came in to replace Avi.

Everyone should use KVM. It's an easy, secure way to boot up a new computer without having to go buy one. You can play around with the `/etc` directory and other strange internals in ways you might have been afraid to before. You can try out different Linux distributions, or maybe create your own, and you also can experiment with creating clusters of many interoperating systems. If anything goes wrong, just turn off the VM and start fresh. It's very cool.

—**ZACK BROWN**

Powerful: Rhino



Rhino M4700/M6700

- Dell Precision M4700/M6700 w/ Core i7 Quad (8 core)
- 15.6"-17.3" FHD LED w/ X@1920x1080
- NVidia Quadro K5000M
- 750 GB - 1 TB hard drive
- Up to 32 GB RAM (1866 MHz)
- DVD±RW or Blu-ray
- 802.11a/b/g/n
- Starts at \$1190
- E6230, E6330, E6430, E6530 also available

- High performance NVidia 3-D on an FHD RGB/LED
- High performance Core i7 Quad CPUs, 32 GB RAM
- Ultimate configurability — choose your laptop's features
- One year Linux tech support — phone and email
- Three year manufacturer's on-site warranty
- Choice of pre-installed Linux distribution:



Tablet: Raven



Raven X230/X230 Tablet

- ThinkPad X230/X230 tablet by Lenovo
- 12.1" HD LED w/ X@1366x768
- 2.6-2.9 GHz Core i7
- Up to 16 GB RAM
- 750 GB hard drive / 180 GB SSD
- Pen/finger input to screen, rotation
- Starts at \$2050
- T430, T530, W530 also available

Rugged: Tarantula



Tarantula CF-31

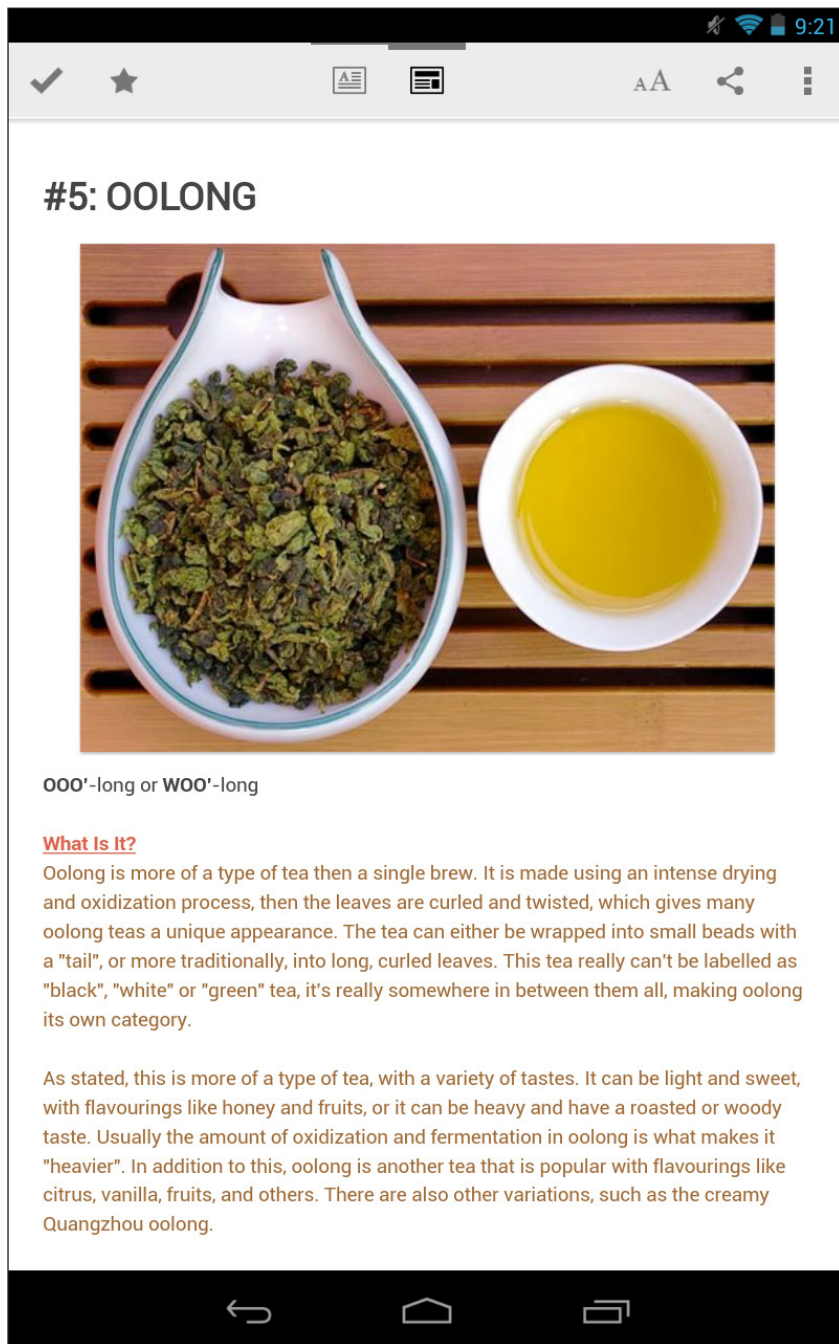
- Panasonic Toughbook CF-31
- Fully rugged MIL-SPEC-810G tested: drops, dust, moisture & more
- 13.1" XGA TouchScreen
- 2.4-2.53 GHz Core i5
- Up to 8 GB RAM
- 320-750 GB hard drive / 512 GB SSD
- CF-19, CF-52, CF-H2 also available

EmperorLinux
...where Linux & laptops converge

www.EmperorLinux.com
1-888-651-6686



Android Candy: Pocket



#5: OOLONG



OOO'-long or WOO'-long

What Is It?

Oolong is more of a type of tea than a single brew. It is made using an intense drying and oxidization process, then the leaves are curled and twisted, which gives many oolong teas a unique appearance. The tea can either be wrapped into small beads with a "tail", or more traditionally, into long, curled leaves. This tea really can't be labelled as "black", "white" or "green" tea, it's really somewhere in between them all, making oolong its own category.

As stated, this is more of a type of tea, with a variety of tastes. It can be light and sweet, with flavourings like honey and fruits, or it can be heavy and have a roasted or woody taste. Usually the amount of oxidization and fermentation in oolong is what makes it "heavier". In addition to this, oolong is another tea that is popular with flavourings like citrus, vanilla, fruits, and others. There are also other variations, such as the creamy Quanzhou oolong.

Most people are familiar with Instapaper and Read It Later. Those types of services are great for tagging Web articles for later reading, and in the case of Read It Later (now called "Pocket"), they do a wonderful job of copying articles off-line for reading when the Internet isn't available.

Using Pocket's Chrome extension, as I browse the Web during the day and find articles I'd like to read later, I add them to my Pocket queue with a single click. This works perfectly for me, because although I find lots of things to read during the work day, I don't have time to read them. Now, I simply can open the Pocket app on my Nexus 7 and catch up on my reading as if it were an e-book. As you can see, articles are formatted for easy reading,

and because they sync off-line, reading doesn't require Internet access.

Pocket is a great system, a really nice Android app and provides an effective way of staying out of trouble at work! Check it out at the Google Play store, or visit the Web site: <http://getpocket.com>.

—SHAWN POWERS



innovation
enterprise

Big Data Innovation Summit San Francisco 11 & 12 April, 2013

Join 1000+ of the industry's top minds at the world's largest executive led Big Data summit.

Explore big data technology across 6 functional and technical tracks:

- Big Data Innovation
- Big Data in Healthcare
- Big Data in Finance
- Big Data in Government
- Women in Big Data & Tech
- Hadoop Innovation



Attendee Registration

Robert Shanley
+1 415 992 7605
rshanley@theigroup.com

analytics.theigroup.com/bigdata-sanfrancisco

The State of Drupal

We asked some Drupal community members and leaders from all over the world about the current state of Drupal and what they are most looking forward to in Drupal 8. We got a variety of opinions from everyone from core developers to end users. Here's what they had to say.

Ryan Szrama: The current state of Drupal is frenzied. Everyone has a lot of irons in the fire (as always), and core devs are working madly to get awesome new features into Drupal 8. I think everyone feels not just the challenge but the opportunity to make Drupal better, the ecosystem larger and their businesses more successful.

I'm most excited to have fago's Entity API updated and included in Drupal core. A robust entity system will make building and maintaining contributed modules like Drupal Commerce much easier than it's been on Drupal 7, and we have both the will and code in place now to make this shine in Drupal 8. Great work all around.

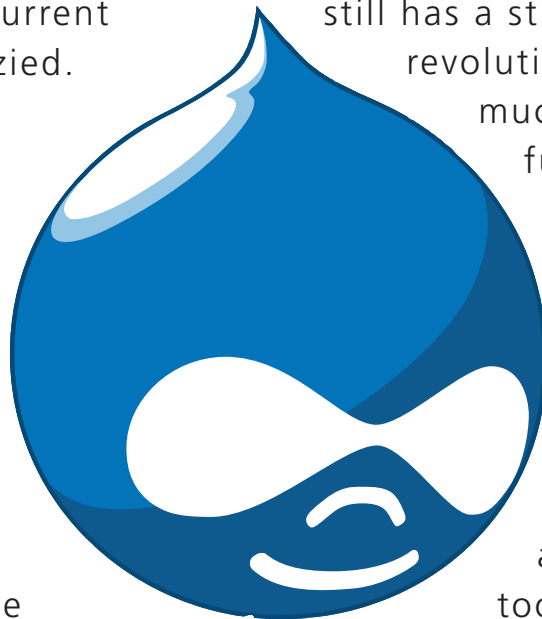
Pol Dell'Aiera: Drupal 8 will be the bomb.

Bert Boerland: Web services/WSCCI is *by far* the best thing happening to Drupal and even the Web. Web services first!

Larry Garfield: Current state: the sands are still very shifty. Drupal 8 still has a strong potential to be revolutionary and kick so much ass it's not even

funny. The trick is if we can close the deal and not leave ourselves in a fugly transitional state. That is still a real risk that keeps me up at night.

Most excited about? There's too much to limit myself. Mostly I think the formalization of APIs that's happening in many places. The detangling of hook_menu, EntityNG, the conversion of more and more systems to cleanly injected (and therefore swappable) systems, etc. The architectural quality of Drupal 8 is steadily increasing, and once we wrap our heads around that transition it should be so much more powerful.



Look at the distinction between a Content Management System, a Content Management Framework and a Web Publishing Tool. Really, really think about it. Then listen to Karen McGrane talk.

Drupal is so close to something so awesome and I don't think we even realize it. Stay tuned.

Diana Montalion Dupuis: In general, I am concerned about the schism between the needs/desires of the development community and the needs/desires of people who pay us to solve their business problems with Drupal. I'm excited about many aspects, and as always, beyond grateful to the dedicated geniuses of the community. I also fear that we are doing something (from the "What problem do I solve?" perspective) different enough from previous versions/visions that we are taking a sharp left turn and actually making something else.

I feel the enthusiasm, Larry's especially. I agree that transition management is a make or break challenge. A steady stream of information that connects our current marketshares' investment in Drupal, and the Drupal shops' investment in team skills, with the changes coming will ease that transition. "Drupal will now be widely applicable to XYZ and thus,

extend our Web presence into ABC functional areas. So, train your team!" (In layman's terms.)

Karoly Negyesi: Disclaimer: I am paid to work on some aspects of Drupal 8 core. Regardless, this my opinion.

Current state: the core developers are often screaming. Sometimes from joy, sometimes from frustration.

The new APIs are superbly powerful. At the same time, it'll take some effort to learn them. Parallel to this, there are concerns about the developer experience, but they were raised before the code freeze even—this is good. Previously, similar concerns only occurred much later in the cycle, and so they couldn't be addressed. No matter how much will we enhance the DX of the new APIs, the "surface" between the old and new APIs will be rough. See Larry's worries above about a transitional state. I consider that inevitable—just the degree is the question.

I love what is happening to the UI. However, I always did. I am just a PHP guy.

Jason Smith: I'm very interested in seeing where the services/WSCCI and configuration management stuff goes. Those are killer features.

I'm a little concerned about the

barrier to entry being raised high enough due to architecture to discourage new folks. Drupal's pulling toward enterprise, which is a good thing. But, my concern is that it may neglect that roadmap to mastery that makes Drupal a fantastic platform for open-source devs.

Barrett Smith: I'm most excited for the configuration management changes slated for D8. Improving the way settings are managed will go a long way toward securing Drupal's place in enterprise-class, multi-tier environments.

Bronius Motekaitis: I love that the Drupal community has grown so large, that Drupal-powered projects keep increasing in size and complexity, and that core and the community are working diligently to keep up. I also like that user experience and accessibility continue to receive a lot of attention. While this is very exciting, it is disconcerting, because sweeping changes necessitate climbing up that steep, personal learning curve. By Drupal 8, I believe we in the community will have gotten much better about documenting core and contrib, migrations and upgrades, and build recipes.

As code developers and application builders already marinated in Drupal, we love the

increases in power and flexibility, but it is important to retain the community of "simple, one-off builders" and continue to pique the interest of prospects: spending time on documentation, UX/design and appearance are all important in these regards.

David Stagg: The biggest issue with Drupal 7 (and as a consequence, Drupal 8) is that it's coming too quickly. While the core is being developed at such a rapid pace (that's good!), a lot of modules and other libraries that worked exceptionally well in Drupal 6 aren't being ported over quickly enough to Drupal 7 or 8 (that's bad!).

Events are a perfect example. Drupal 6 has a robust set of modules and features that allows us to make an intensive event-based site; in Drupal 7, almost none of these are available. For those of us intently focused on the front end without the capabilities to create the modules ourselves, this can be tough. The best part of open source is the community, but when the development moves too fast, the dust that settles oftentimes forfeits a complete package. And after all, Drupal isn't complete without the community.

Donna Benjamin: More than one core developer has admitted they've

not actually built sites with D7. Our commitment to the future means we abandon the past, and sometimes it seems the upgrade path is an afterthought.

But I'm optimistic about the future—I think D8 has extraordinary potential. The work that's been done on HTML5, Multilingual, Mobile, Config Management, Web Services and Authoring Experience is really quite phenomenal. It will be a game-changer.

I just hope it's still the kind of system that provides the kinds of solutions my clients and

communities require. But I can't deny I am worried its enterprise target puts it out of reach for small sites, for small orgs with small budgets, and volunteer teams. Many people learn Drupal in those environments. At the moment, we don't have an alternative formal training and certification pathway to replace that pipeline.

As for the community itself, keep an eye on AsiaPac: India, China, South East Asia and Australasia. We're just not on the radar for most EuroMericans.

—**KATHERINE DRUCKMAN**

Are you everywhere?

DOOR3 offers the right blend of strategy, design & development.

We build hardworking **web and mobile software** for hardworking businesses.

We've been busy.

See why at www.door3.com.

DOOR3



Design Your Own Rocket

A lot of the software packages I've covered in recent articles have been focused strictly on doing computations on your machine, separate from the real world. So in this article, I explore how to use your computer to design something you can build and use in the real world: your own model rocket. Let's take a look at the OpenRocket utility and see how it can help you design your own rockets. OpenRocket even can run simulations on your designs to show how they should behave in flight.

Most distributions should include a package for OpenRocket. For example, in Ubuntu, you would install it with `apt-get install openrocket`. It is actually a Java program, so you always can download the jar file directly from the Web site (<http://openrocket.sourceforge.net>). To run it, you need to have a reasonably up-to-date Java VM installed as well.

When you first start it up, you are presented with an empty screen, ready to begin designing your first rocket. A project window pops up, allowing you to enter details like the design name, your name and design notes. You can build your rocket from a series of components.

You probably will want to start with the nose cone by clicking on it from the "Add new component" window. It then will appear in the bottom section, beginning your design.

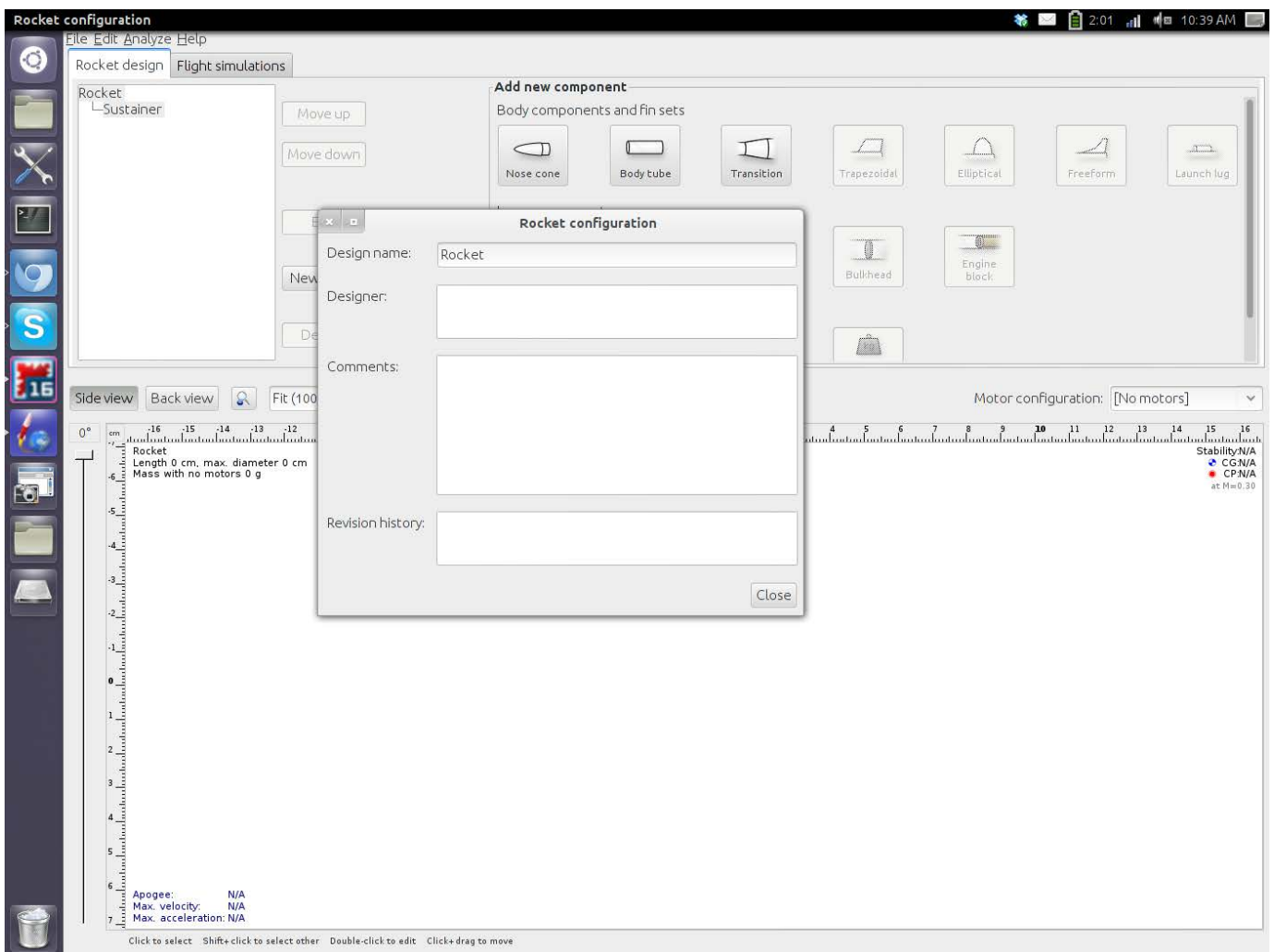
You will notice that OpenRocket already begins to make calculations based on your design. You will see a small blue circle that denotes the center of gravity of your rocket. The center of gravity is the point through which all of the mass acts. There also is a small red dot that marks the center of pressure. This is the point through which all of the atmospheric forces act. OpenRocket calculates these values as you make changes to your design.

You can edit almost all of the parameters for each component. There are two ways to edit these component parameters. You can double-click on the component of interest in the design window to access the edit window. There also is a list of the component layers in the top half. You can highlight the component of interest here and then click on the Edit button.

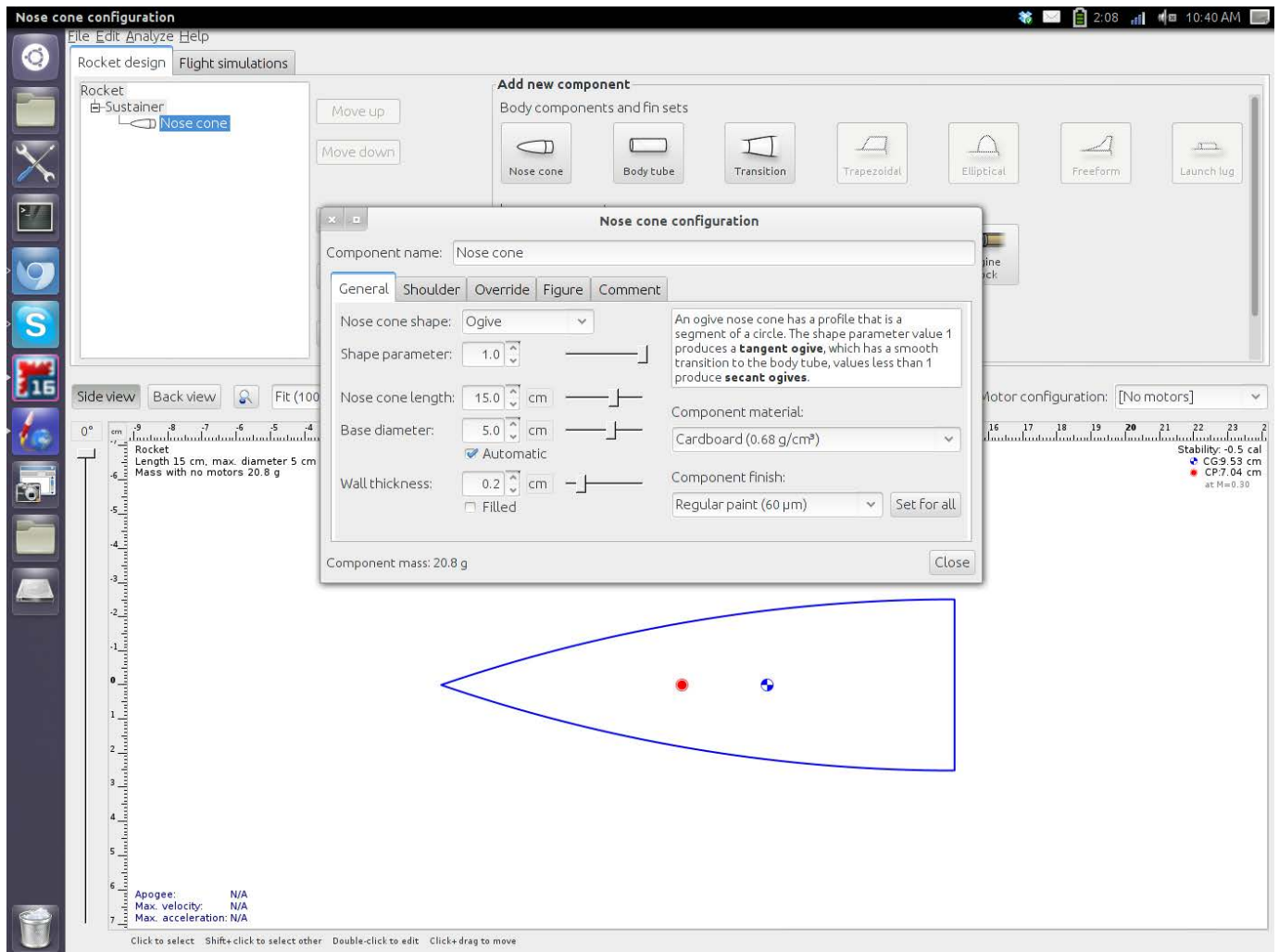
What you can change will depend on which component you are trying to edit. The first component to design probably is the nose cone.

You can select what kind of profile your nose cone should have, such as conical or ellipsoid. You can set the length and base diameter, as well as the wall thickness. But, OpenRocket goes even further. You can select the kind of material from which your nose cone should be made. The different types of materials have different densities, which changes the weight of your component.

Several different material presets are available, but you also can go ahead and define your own custom material type. This custom material either can be used just for one design, or you can add it permanently to your materials database if it is something you will be using over and over again. You even can set the type of finish your rocket will have. This finish can be



When first starting OpenRocket, you get a new project dialog where you can enter details of your rocket.



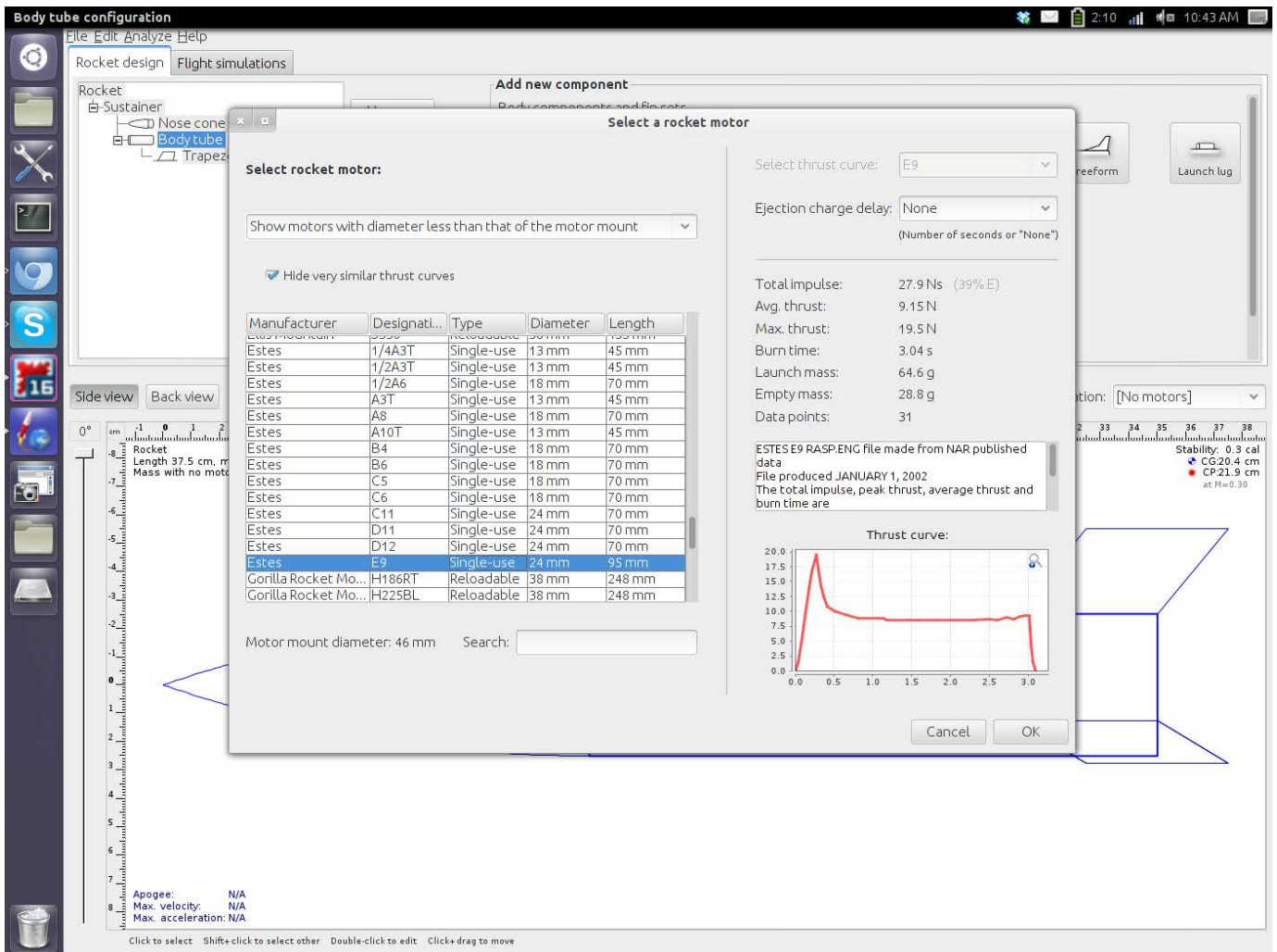
You can edit many details of the different components by double-clicking the component of interest.

different for each component, or you can apply a common finish to your entire rocket.

The next part of your rocket is the actual body tube. If your rocket is going to have a single stage, you will need only a single body tube. For multiple stages, you will need a separate body tube for each stage. Opening the edit window for the body tube allows you to change

the tube length, diameter and wall thickness. You also can change the material from which the body tube is made.

Because body tubes usually contain a rocket motor, you can set the motor that you want to use as well. To do so, open the edit window and click on the motor tab. You can click the Select Motor button and choose from a large selection



The rocket engine database has lots of information on many different brands.

of commercially available motors. This list of motors includes technical information, such as the thrust curve and the run times, as well as physical characteristics like the length, diameter and weight.

As with the other sections of the application, you can define your own custom entries. This means even full-on DIY model-rocket enthusiasts who build their own

motors can use OpenRocket to design their rockets.

The component list includes other items like parachutes, shock cords, connectors and blocks, so you can design your rocket as a complete model.

The final stage is to add tail fins to your rocket. As with the other components, there are well designed presets available that will

[UPFRONT]

satisfy most design needs. You can edit some parameters to customize the fins to some degree. If that's not enough, there is a freehand option where you can design your fins from scratch.

OpenRocket is not only a design program. You also can take your model rocket and run an analysis on it to see how it will behave in flight. The analysis section looks at individual components and shows

how they affect the stability, drag and roll characteristics of your rocket. You can set parameters like wind direction, angle of attack and speed, and calculate how it will behave in flight. This is great functionality, but OpenRocket goes even further. It can take your original design and try to optimize it for the best flight characteristics. You can optimize based on altitude, velocity or some other combination

The screenshot shows the OpenRocket software interface. A 'Component analysis' dialog box is open, displaying various parameters and a table of component properties. The table is as follows:

Component	CG / cm	Mass / g	CP / cm	C _{Na}
Nose cone	9.53	20.8	7.04	2
Body tube	25	41	25	0
Trapezoidal fin set	33.8	2.75	32.5	2.8
Total	25.3	129	21.9	4.8

Below the table, the reference length is 5 cm and the reference area is 19.6 cm². The dialog also shows a 3D model of the rocket with a red dot indicating the center of gravity. The background shows the main software interface with a rocket design tree and various toolbars.

You can run an analysis on the different components of your rocket.

of characteristics.

Once you have a final design, you can run it through simulations. The simulator can apply different conditions on your rocket, like applying crosswinds or taking Coriolis effects into account, and it shows how your model rocket should behave. OpenRocket uses JFreeChart to plot your rocket's behavior based on the simulation results.

You even can add your own

code to the simulation to add extra effects. One way to do this is using Python and jPype to use Python code in the simulations. Additionally, you have the ability to write and use your own expressions in the simulator. This allows you to customize the simulator to a great degree without having to add external code. Then, you can see the limits of what your model should be able to handle and how

Parameters to optimize:

Parameter	Current	Minimum	Maximum
Nose cone: Length	30 cm	7.5 cm	30 cm

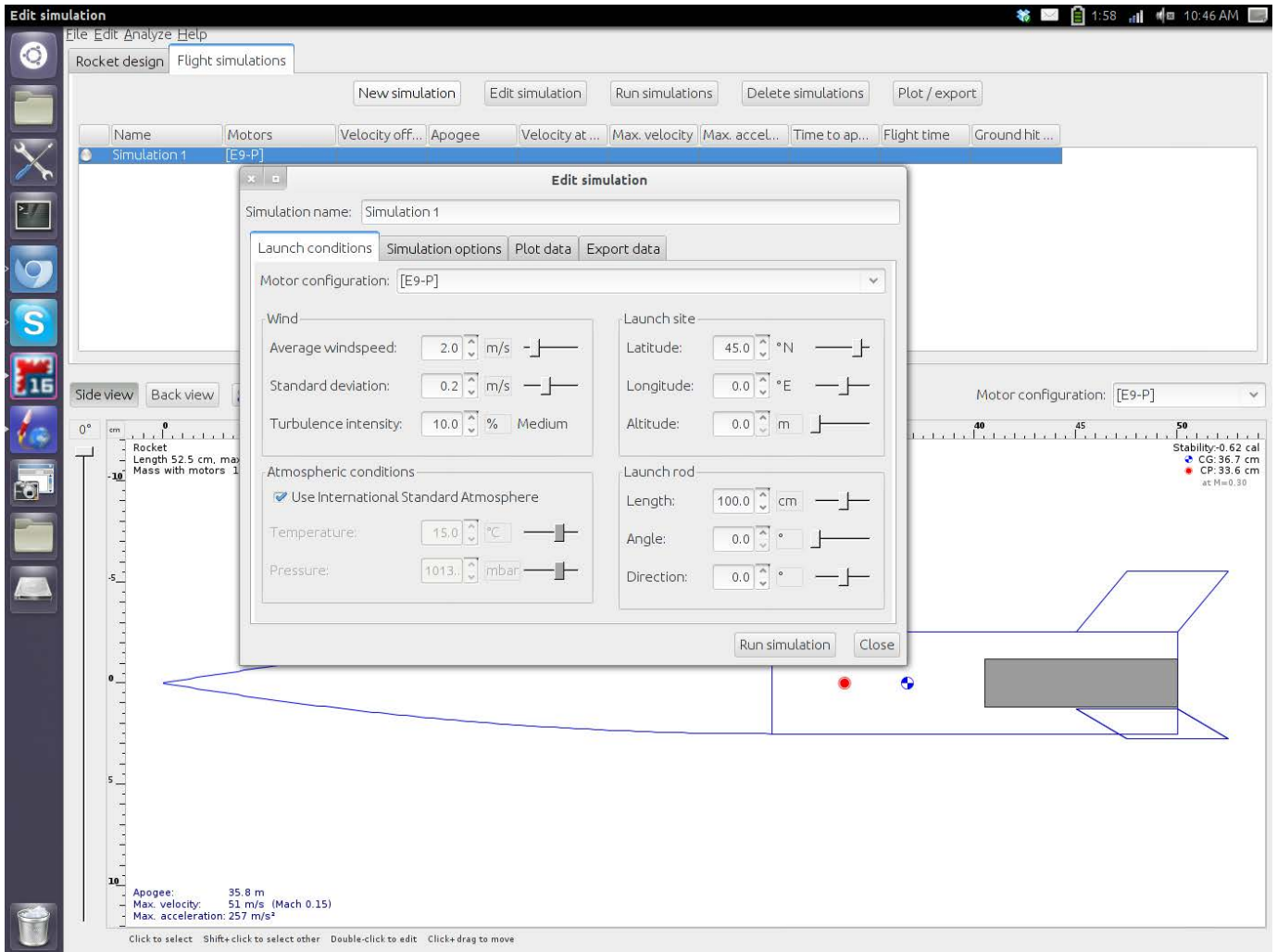
Optimization options:
 Optimize simulation: Basic simulation [E9-P]
 Optimized value: Total flight time
 Optimization goal: Maximize value

Required stability:
 Minimum stability: 1.0 cal
 Maximum stability: 5.0 cal

Available parameters:
 - Sustainer
 - Nose cone
 - Shape parameter
 - Thickness
 - Body tube
 - Optimization parameters
 - Length
 - Motor ignition delay
 - Motor overhang
 - Thickness
 - Trapezoidal fin set
 - Cant angle

Simulation Results:
 Apogee: 35.8 m
 Max. velocity: 51 m/s (Mach 0.15)
 Max. acceleration: 257 m/s²

You can optimize your rocket to maximize certain parameters.



You can run full simulations to see how your rocket will behave in different conditions.

high it will go under different conditions. This is really great in helping you decide when the weather is going to be too rough for your design.

I have covered only the features available in the most minimal way. If you are into building and flying model rockets, your time definitely will be well spent poking around all

of OpenRocket's available features. The Rocketry Forum hosts a forum where you can ask for further help. And once you have gained some experience, you can give back by helping other new rocket enthusiasts. Go ahead and design your fleet, and get yourself out into the wild frontier of space!

—JOEY BERNARD

Shell Game

Many of the cool things in *Linux Journal* require the use of the command line. For us Linux users, that's generally not a big deal, because we have a terminal window readily available. Some of the time, however, it's helpful to have a shell account on an Internet host somewhere.

If your Web-hosting service provides shell access, you might be able to use it for rudimentary command-line procedures. (In fact, Dreamhost in particular allows SSH tunneling through its servers for clients.) If you want to use particular programs like `screen` or `irssi` though, it will require something a little more robust.

Some free shell services are available (like <http://www.geekshells.org>), but they often are very restrictive, and it can be challenging to get an account with them. Thankfully, if you don't mind spending a few dollars a month, shell accounts are fairly common and relatively inexpensive. The Eggdrop folks have compiled a great list here: <http://www.egghelp.org/shells.htm>.

```

Terminal - spowers@kermit: ~
File Edit View Terminal Go Help
spowers@kermit ~ $ apt-get moo
      (__)
     (oo)
  /-----\
 / |       | \
*  ^-----^
   ~       ~
...."Have you mooed today?"...
spowers@kermit ~ $

```

Figure 1. Sometimes, you just need a shell.

Of course, if you want to have a full-blown server on the Internet, it's hard to beat a colocated Raspberry Pi server like the one Kyle Rankin talked about last month. However you manage it, it's hard to be a geek without access to a terminal!

—SHAWN POWERS

Non-Linux FOSS

For some reason, single-site Web applications like the kind created with Prism seem to have fallen out of style. I've always been particularly fond of Chrome-based applications, and yet, there doesn't seem to be a way to create a single-site Web application properly on OS X using Chrome. Thankfully, even with modern releases of Chrome, it's certainly possible and still very useful.

Two different scripts are available and freely downloadable for creating single-site Chrome apps on OS X. The first is an AppleScript file created by Mait Vilbiks, available at <http://snar.co/chromeapp1>. The other is a Bash script that does the same identical thing, but it works on the command line. It's available at <http://snar.co/chromeapp2>. (Note: both files are public domain and can be modified if desired.)

If you have ever wished Google Calendar was a standalone application, or if you just want to have dock icons for your various Web activities, either of these scripts work wonders. Even better, if you want to delete the application, simply drag it to the trash. Chrome apps work great as standalone applications in OS X, and now you can try them yourself!—**SHAWN POWERS**

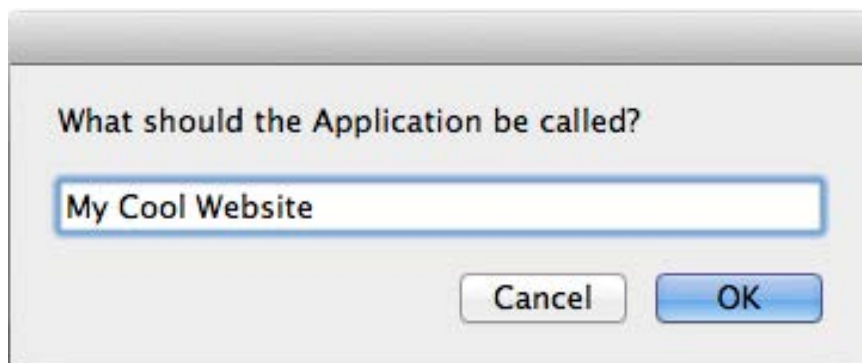


Figure 1. The AppleScript version walks you through the process.

They Said It

If this is coffee, please bring me some tea; but if this is tea, please bring me some coffee.

—*Abraham Lincoln*

Almost all my middle-aged and elderly acquaintances, including me, feel about 25, unless we haven't had our coffee, in which case we feel 107.

—*Martha Beck*

Do Lipton employees take coffee breaks?

—*Steven Wright*

I have measured out my life with coffee spoons.—*T. S. Eliot*

To me, the smell of fresh-made coffee is one of the greatest inventions.

—*Hugh Jackman*

I never drink coffee at lunch. I find it keeps me awake for the afternoon.

—*Ronald Reagan*

20
13

ATTEND FLOURISH!
INSPIRING TALKS,
ILLUMINATING WORKSHOPS

OPEN SOURCE, OPEN FUTURE.

MARCH

1-2

UIC FORUM
725 W ROOSEVELT RD.
(MC 126)
CHICAGO, IL



FLOURISH!

A CONFERENCE PROMOTING
THE ADOPTION OF FREE,
LIBRE, OPEN SOURCE
SOFTWARE IN THE MIDWEST.

sponsored by:

github
SOCIAL CODING

GET UPDATES AT [HTTP://FLOURISHCONF.COM/](http://flourishconf.com/)

Web Development Poll

We polled our on-line readers about their Web development preferences and got some surprising and not-so-surprising results. Perhaps the biggest surprise is that Drupal just slightly edged out WordPress. WordPress traditionally has been the most popular of the two among our readers, with Joomla a close third to Drupal's second place. This time, Drupal beat out WordPress by a hair with Joomla a distant third. Chrome's popularity over Firefox was a surprise, but as someone who tests in everything but prefers Chrome, I can't say I'm shocked. Our readers' love of vim and JQuery is not a surprise, nor is the popularity of Python and PHP.

Thanks, as always, for participating in our on-line polls. The full results follow.

1) What is your favorite browser for testing?

- Chrome: 56%
- Firefox: 37%
- Other: 3%
- Internet Explorer: 2%
- Safari: 2%

2) What is your favorite Web programming language?

- PHP: 37%
- Python: 27%
- Java: 20%
- Perl: 8%
- Ruby: 8%

3) What is the best CMS to develop for?

- Drupal: 31%
- WordPress: 30%
- Other: 18%
- Joomla: 11%
- Plone: 4%
- DotNetNuke: 2%
- Typo3: 2%
- ModX: 1%
- Moveable Type: 1%

4) What is your favorite editor?

- vim: 30%
- Other: 18%
- GEdit: 14%
- JEdit: 11%
- Kate: 6%
- Komodo Edit: 4%
- TextMate: 3%

5) What is your preferred development framework?

- Django: 30%
- Other: 26%
- Ruby on Rails: 16%
- Zend: 9%
- CakePHP: 7%
- CodeIgniter: 7%
- Symfony: 5%

6) Which JavaScript libraries do you use most frequently (multiple answers permitted)?

- JQuery: 74%
- Node.js: 20%
- Other: 18%
- Backbone.js: 13%
- Dojo: 11%
- Ext JS: 11%
- MooTools: 11%
- YUI Library: 10%

7) What is your favorite hosting company?

- Amazon: 33%
- Other: 28%
- GoDaddy: 11%
- Linode: 9%
- Dreamhost: 7%
- Rackspace: 7%
- 1&1: 5%

8) Which do you prefer to use?

- A text editor: 37%
- An IDE: 16%
- Both: 47%

9) What is your preferred revision control system?

- Git: 63%
- Subversion: 18%
- CVS: 8%
- Mercurial: 8%
- Other: 3%

10) Do you currently practice continuous integration?

- No: 59%
- Yes: 41%

11) Which SQL database do you work with most frequently?

- MySQL: 67%
- PostgreSQL: 17%
- SQLite: 8%
- Other: 5%
- MariaDB: 3%

12) Which NoSQL database do you work with most frequently?

- MongoDB: 51%
- Other: 18%
- Redis: 11%
- Cassandra: 10%
- CouchDB: 10%

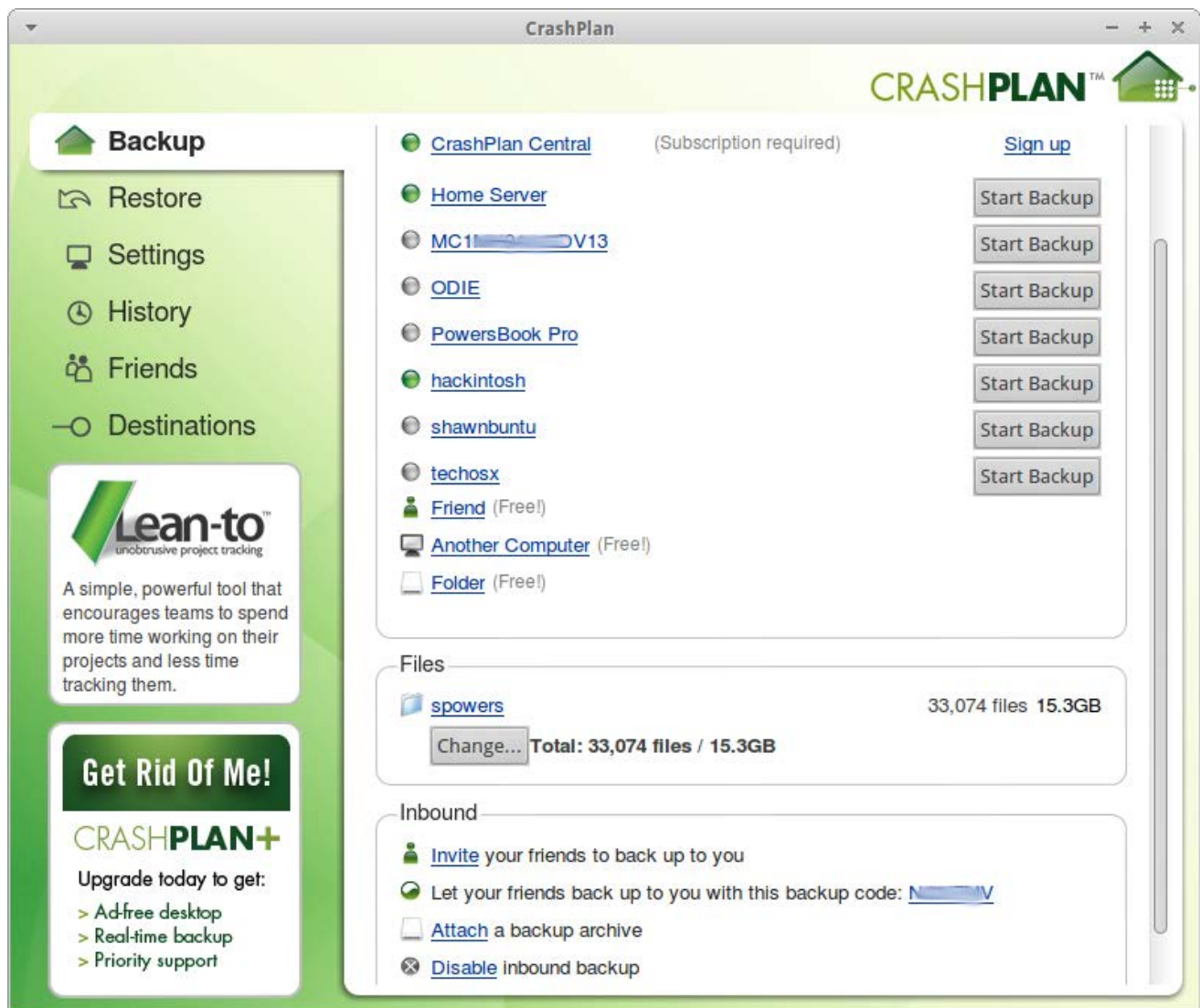
—KATHERINE DRUCKMAN

Crashplan, the Only Reason I Install Java



I'm the sort of person who doesn't like to install Java. I actually don't like to install Flash either, but it's still tough to survive browsing the Internet without Flash installed. There is one program that makes me break my own rules, however, and that's Crashplan.

For years, I've been singing the praises of BackupPC, and for servers, I still think it's the best thing going. The problem with BackupPC, however, is in order for it to work reliably, your workstations need to be on all the time. This is especially



difficult with laptops.

Crashplan is an incredibly powerful backup utility that allows local or offsite backup, and the company offers cloud-based storage for reasonable rates. Normally, I wouldn't be so excited by a paid service and a non-open-source software package, even if it does offer a Linux-native client. The folks at Code 42, however, have given away the ability to swap storage with friends as an alternative to their paid-cloud-based service. If you have a computer at work, and a computer at home, you can back them up over the

Internet to each other completely free!

As I already mentioned, I normally don't like Java-based programs like Crashplan, but its functionality is so great, I don't mind breaking my own rules. More than that, I give a lot of credit to Code 42 for not only making a native Linux client, but also for giving away incredible functionality for free. If you're not backing up your data, be sure to consider Crashplan at <http://www.crashplan.com>. Its price, feature set and generous non-paid features make it this month's Editors' Choice!—**SHAWN POWERS**

LINUX JOURNAL

on your
Android device

Download app now in
the **Android Marketplace**



www.linuxjournal.com/android

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



REUVEN M.
LERNER

Watir

Test your Web applications by controlling real browsers from Ruby.

I recently was visiting a new client when one of the staff developers asked me to help debug his automated test suite. I asked what test frameworks he used, and he responded by saying, “Cucumber and Watir”.

Now, I’ve long known about Cucumber, and I even wrote about it several years ago in this column. It’s an advanced acceptance-testing framework that’s especially popular among Rubyists who promote behavior-driven development (BDD), an offshoot of test-driven development (TDD), which looks at a system’s functionality from the end user’s perspective. Cucumber allows you to describe your tests in plain English, which makes the test specifications easy to understand, even for nontechnical users.

But Watir? I knew that Watir (Web Application Testing In Ruby, pronounced “water”) is a BSD-licensed framework that lets you interact with a Web browser from within a Ruby program, allowing

you to test with a real browser (like the macro-based, cross-platform Selenium test tool) but with Ruby commands (like the popular Capybara test tool). I had last looked at Watir more than four years ago, at which time it had a fatal flaw: it worked only on Windows systems and only with Internet Explorer.

But it seems that in the past few years, Watir has grown and expanded, with a suite of software projects and products that are increasingly flexible and impressive in their scope, handling many different types of browsers, operating systems and conditions.

In this article, I take a look at Watir and how you can use it to test your Web applications more easily. My client’s test suite, implemented with Watir, really impressed me, and it showed how with a bit of cleverness, you can take advantage of Watir’s platform-independence to ensure that your application works on different browsers and operating systems with a minimum of hassle—and without

having to worry if the JavaScript emulation package you're using will reflect the version that's in a user's browser accurately.

Installing Watir

One of the confusing things about Watir is that it's the name of one particular open-source project, as well as a set of related projects that have the "watir" name in them. Each project has slightly different capabilities and dependencies, and is maintained by a different person or group.

Moreover, although documentation for Watir certainly exists, it's not nearly as centralized, organized or easy to follow as many other open-source projects that have reached this maturity level. Understanding how to use Watir frequently requires that you understand the general world of Web browsers and automatic testing frameworks in order to work with it, and that you figure out which version of Watir will do what you want.

For example, as I mentioned previously, Watir originally worked only on Windows and IE. This remains true for the "watir" system itself, also known as "watir-classic".

You can use Watir on non-Windows machines by using Webdriver—part of Selenium, an Apache-licensed

Web application testing framework. Just in case you missed that part, let me repeat it: Watir's cross-platform capabilities are made possible by Webdriver, which is developed by the maintainers of the "competing" Selenium test automation system.

(Wait, did I say that Webdriver is an open-source product? It is, but it's also a draft specification API from the W3C, the reference implementation of which is, you guessed it, the Webdriver software in the Selenium project.)

Because Selenium is widely used and well maintained, and because the Selenium team wants to have connections from many languages and to many browsers, it's generally safe to assume that you can control just about any Web browser programmatically using Webdriver. The Watir community has taken advantage of this, creating the `watir-webdriver` gem—basically, connecting the Watir API to the Webdriver back end. Of course, there are exceptions; I recently found that `watir-webdriver` did not yet support the just-released Firefox 19. A quick check on Freenode IRC's #watir channel confirmed that Webdriver doesn't yet support the most-recent Firefox version.

The bottom line is that for virtually any work you'll want to do on a Linux

system, you'll need to install the watir-webdriver gem:

```
sudo gem install watir-webdriver -V
```

Once you have installed watir-webdriver, you can work with Chrome and Firefox. Actually, that's not entirely true. You'll also need to install chromedriver, a program that lets you control Chrome via Webdriver (see Resources for the download URL).

Watir Basics

Once you have installed Watir, things should be much easier and more exciting. Fire up an interactive Ruby shell, either using IRB or the more modern Pry. If you're running Ruby 1.8, you first need to load the Ruby gems package:

```
pry(main)> require 'rubygems'
```

And in all cases, you then need to load the watir library:

```
pry(main)> require 'watir'
```

Now I'm going to create an instance of Watir::Browser, the class that represents a Web browser, which I can control via Ruby. I do this by passing the "new" constructor method a

parameter indicating which kind of browser I want. For example, I can create a Chrome browser:

```
pry(main)> browser = Watir::Browser.new :chrome
```

In my case, this takes a little while to execute, as the browser binary starts up and then opens a communication channel, via Webdriver and Watir, with my Ruby instance. I often like to use the "inspect" method on a Ruby object to examine it. Here's what I get when I do this with my browser:

```
pry(main)> print browser.inspect
#<Watir::Browser:0x1b4e74e97c5b6930 url="about:blank"
  ▶title="about:blank">
```

Not surprisingly, my browser instance has both a current URL and a title, neither of which is particularly exciting. So, let's point the browser to somewhere that's a bit more interesting:

```
pry(main)> browser.goto 'http://linuxjournal.com'
```

Within a few moments, I not only get control back at my interactive Ruby prompt, but the Web browser also has gone to the *LJ* home page. I can ask the browser for its title with the "title" method:

```
pry(main)> browser.title  
=> "Linux Journal | The Original Magazine of the Linux Community"
```

I similarly can invoke the “url” method:

```
pry(main)> browser.url  
=> "http://www.linuxjournal.com/"
```

Notice that although I told the browser to go to `http://linuxjournal.com`, it was redirected to `http://www.linuxjournal.com`, and the browser’s current URL reflects this.

At this point, I can use the “html” method to get the HTML from the current browser window or the “text” method to retrieve a version of the current browser, stripped of HTML tags. I also can go to the previous page (with the “back” method) or reload the current page (with the “refresh” method).

Now, let’s say that I want to retrieve all of the headlines from the *LJ* site. From a quick inspection of the site, I can see that each headline is in an “h2” tag. I can ask Watir to retrieve all of the “h2” tags in a collection, from which I then can display the first headline’s text. For example:

```
pry(main)> browser.h2s[2].text  
=> "Kyle Rankin to Keynote SCALE 11x"
```

But why stop there? I can retrieve and display all of the current headlines:

```
pry(main)> browser.h2s.each {|h| puts h.text}
```

I’m not going to use up all of my column with a list of headlines, but you can be sure that this does print all of the headlines on the site. Actually, it does a little more than that. The above code prints all of the h2 tags anywhere in the site, which includes a bit more than that. Upon closer inspection, I don’t really want all of the h2 tags on this page, but rather all of the h2 tags that are within the div whose name is “content-area”.

So, what I need to do is tell Watir to grab the “content-area” div and then retrieve all of the h2s contained within it. To grab the content-area div, I use the “div” method, which tags a hash describing the attributes of the div I want:

```
browser.div(id: 'content-area')
```

That returns the div, which is a good start. But I want the h2s within the div, so I can just say:

```
browser.div(id: 'content-area').h2s
```

Yes, the “h2s” method that I used

Watir provides a number of tricks that let you find and work with Web pages.

before, when executed within the context of a div, restricts the search to that div, rather than the entire browser window. So, I can display all of the latest headlines with:

```
browser.div(id: 'content-area').h2s.each {|h| puts h.text}
```

And sure enough, that works very nicely. Watir provides methods for you to retrieve many different types of elements from within the full browser context, or within a more restrictive context. Thus, you can find a single div with the singular “div” method or a number of them matching (optional) criteria with the plural “divs” method. You can retrieve the first h2, or the first h2 to match optional criteria, with the “h2” method, or all of the h2s that match optional criteria with the plural “h2s” method. The same is true for “span”, “h3”, “p” and even “a” tags. The singular method retrieves the first item to match your stated criteria, or if multiple elements matched, the first from that list.

Watir provides a number of tricks that let you find and work with Web pages. For example, I can search for all paragraphs whose text is the

word “Linux”:

```
pry(main)> browser.ps(text: 'Linux').count
```

Not surprisingly, this returns a value of 0, because no paragraph consists solely of the word “Linux”. However, if I pass a regexp object rather than a string, I’ll get back the “p” elements that contain the word “Linux” anywhere inside them:

```
pry(main)> browser.ps(text: /Linux/).count  
=> 4
```

As you can see, one of the ways in which I check that my criteria are working is by using the “count” method. If I’m working with a single Watir element, I also can use the “flash” method to highlight the element briefly on the screen, in the browser. For example:

```
pry(main)> browser.h2(text: 'Help Us Feed You Pi!').flash
```

Although this makes the element extremely obvious in the browser for a few seconds, you do need to be looking at the browser, and have it scrolled to the appropriate element,

to see it. You also can loop through a number of matches and flash each of them in turn:

```
pry(main)> browser.ps(text: /Linux/).each {|p| p.flash}
```

Interaction

Of course, Watir can do much more than retrieve information from the browser. For example, let's assume that I want to execute a search on the *LJ* site. Looking at the home page, I see a text field and a search button next to it. If I enter text in that text field and click on the search button, I should see some results.

By using the "view source" feature in my browser, I see that the search field has a class of "gsc-input". So, I focus on that element, which removes the grayed-out "hint" text:

```
pry(main)> browser.input(class:'gsc-input').focus
```

Now, I type into the element:

```
pry(main)> browser.input(class:'gsc-input').send_keys('Reuven')
```

Finally, I submit the search form:

```
pry(main)> browser.input(class:'gsc-search-button').click
```

Sure enough, the browser submits the form, and I get a list of ego-boosting results from the *LJ* site.

Notice that I used the "send_keys" method when typing into the form element. There are other ways to modify the text. For example, I can retrieve the text of the text field with the "text" method:

```
pry(main)> browser.input(class:'gsc-input').text
```

If I use the "text_field" method, I also can set the value of the text in the search field before clicking on it:

```
pry(main)> browser.text_field(class:'gsc-input').set 'Reuven'
```

```
pry(main)> browser.input(value:'Search').click
```

One of the best things about Watir is that you're not working inside an emulation package, but rather an actual browser. This means that you can use and test things that require JavaScript. For example, if I go to the AirBNB.com home page, I see a huge photo behind the simple "Where do you want to go?" form. That photo changes every 30 seconds or so, but I can force it to move forward or backward by clicking on the < and > signs. Is there a way for me to move the image forward or backward without clicking on these buttons with the mouse?

The answer is yes, but first I have to find the elements that will take the click. Opening the HTML source,

I found that the < and > buttons were defined as <i> elements inside a div whose class was "arrows". I was able to grab that div with:

```
browser.div(class: 'arrows')
```

I then wanted to get at the <i> elements inside of the div. Using the "elements" method on the div, I grabbed the left-arrow:

```
browser.div(class: 'arrows').elements.first
```

Of course, I double-checked that I had grabbed the correct element with the "flash" method:

```
browser.div(class: 'arrows').elements.first.flash
```

Once establishing that it was the right item, I clicked on it:

```
browser.div(class: 'arrows').elements.first.click
```

Sure enough, that did it—the front page of AirBNB.com switched images right away, rather than waiting for the timeout to occur. In this way, you can test JavaScript and Ajax events, as they take place in an actual browser.

Conclusions

The good news is that Watir is a great tool for testing Web applications. I've

been using it only a short while, and I'm already delighted with the sorts of sophisticated tests I can run. The fact that I can test different browsers automatically, work against actual sites and know that I'm using a real-world environment, rather than one designed for testing, is a big help. Watir has come a long way from its Windows-and-IE-only roots, and the developers deserve a great deal of credit.

That said, there are a number of issues with Watir, ranging from the difficult-to-follow installation instructions, to the confusingly large number of related (and unrelated) Ruby gems, to the API, which is well written, but poorly documented. It's frustrating, for example, that I can retrieve the contents of a text area using either the "input" or "text_field" methods, but that only text_field allows me to change the current text value.

Integrating Watir into existing test facilities, such as RSpec and Cucumber, appears to be possible, but the documentation for doing so (like much in the Watir world) is incomplete, somewhat out of date and spread across a number of sites.

Should you use Watir? Based on what I've seen, the answer is yes, although you should be prepared to

spend some time reading blog posts, downloading a number of versions of Watir and trying to understand why things that should work don't. Once you're past the initial installation and learning curve, working with Watir is pleasant and straightforward, with an API that is second to none. I've started to incorporate Watir into my own test

procedures, and I encourage you to consider doing the same on your own projects. ■

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

Resources

For the latest versions of Watir and documentation, go to <http://watir.com>.

There is now a Watir book that I downloaded and read called *Homebrewer's Guide to Watir* by Zeljko Filipin, an active Watir community member. I'm sorry to say that the book isn't well written and doesn't include very many examples or considerations beyond simple documentation of what Watir can do. To his credit, the author is contributing all proceeds from the book to the Watir project.

A second book, *Cucumber and Cheese*, by Jeff Morgan, is also available for \$14.99, but I didn't have a chance to read or review it. Links to both books are at <http://watir.com/book>.

There is a Watir podcast in which prominent Watir users are interviewed about their test techniques, available from <http://watirpodcast.com> and distributed via a variety of networks, including iTunes. It is hosted by author and community member Zeljko Filipin.

A variety of Watir-related projects, such as watir-grid, are available for download as Ruby gems. You can find a large number of Watir-related Ruby gems by searching for "watir" at <http://rubygems.org>. Other gems and resources are available from <http://watir.com>, which has a large number of pointers to other sites and software.

You can download Chromedriver from Google Code at <http://code.google.com/p/chromedriver>. From the downloads tab, choose the driver for either Linux32 or Linux64, according to your computer's needs. After downloading Chromedriver, install it in a directory on your PATH, such as /usr/local/bin, or in your own personal ~/bin directory.



DAVE TAYLOR

Cribbage: Calculating Hand Value

This month, Dave talks combinatorics and factorials. If you're a math geek, you'll love the discussion. Oh, and there's a little bit of shell script coding too.

The last few months, we've been building a complex shell script to play elements of the game of *Cribbage*, demonstrating a variety of concepts and techniques as we proceed. That's all good, and last month, the script expanded to include a "shuffle" capability and the ability to deal out six cards, a typical two-player starting hand.

Cribbage tip: in a three-player game, each player gets five cards, and the extra is dealt directly into the "crib"—the extra hand counted after play by the dealer.

The challenge we're going to tackle this month is figuring out the best four cards to keep of those initial six, based on potential point values of the different hands.

Ultimately, a fifth, "cut card", is turned up that everyone shares (somewhat akin to the shared cards in *Texas Hold 'em*), but smart *Cribbage* players seek to maximize the four they hold and celebrate when the fifth adds to the point value of the hand, rather than count on that cut card to make the hand.

So what's valuable in a *Cribbage* hand? Sequential runs of three or more cards, pairs or better of the same rank, having all the cards in your hand match suit (and possibly the cut card), combinations of card ranks that add up to 15, and having the jack of the same suit as is on the cut card.

Cards of the same rank—pairs or better—are tricky to count because

the game works on combinations. So 5H, 5D is worth two points “for the pair”, but 5H, 5D, 5S is worth six points, because it combines to three pairs: 5H+5D, 5H+5S and 5D+5S. Four of a kind? A sweet 12 points and a very rare combination indeed.

Critical to know: aces always are worth 1 when counting up fifteens, and all face cards are worth 10 points. So A+4+K is a fifteen, just as 9+6 is, though for pairs, it’s the same rank, not just pairs of face cards. In other words, J+K is not a pair.

Where counting gets tricky is those combinations of fifteens. A great hand, for example, is 6S, 7H, 8D, 8S, which offers two points for 7H+8D, two points for 7H+8S, three points for 6S+7H+8D, another three points for 6S+7H+8S and a final two points for the pair 8D+8S—very nice. If the cut card is another 7 or, even better, another 8, you can see where there are a lot of points in that potential hand!

The challenge, of course, is doing all of this programmatically. When we left off the program last month, it could deal six cards to us and sort them in increasing rank order, which is helpful, particularly in recognizing runs.

Calculating All Four-Card Combinations

The first problem we have is actually earlier than anything to do with card values. It’s to calculate all possible four-card combinations out of a six-card hand. It’s combinatorics—something I really enjoyed back when I took that class in college for my CS degree.

What we’re looking at is actually factorial math. So “six choose four” is really $6! / 4!(6! - 4!)$.

$$\frac{n!}{m!(n - m)!}$$

Figure 1. Equation for “Six Choose Four”

A factorial number is calculated as the multiple of all decreasing numbers, so $4! = 4 \times 3 \times 2 \times 1$, and $6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1$. Of course, the “x 1” part is pointless, so we can skip it. $4! = 24$ and $6! = 720$, which means that we have 15 card combinations to check for points. (If we had only five cards dealt

because it was a three- or four-person *Cribbage* game, we'd have only five combinations to evaluate instead.)

The question is, how do we figure out all 15 of those combinations easily? I mean, I could hard-code all 15 of them—it's only four digits per subset, after all—but that seems less interesting than trying to solve the more-general mathematical equation. And yes, I can see myself getting sidetracked as I proceed, but that's part of the fun of programming when you're not on a deadline, right?

Um, never mind. Let's just take as a given that for six-choose-four, there are these specific combinations and no more: {1,2,3,4}, {1,2,3,5}, {1,2,3,6}, {1,2,4,5}, {1,2,4,6}, {1,2,5,6}, {1,3,4,5}, {1,3,4,6}, {1,3,5,6}, {1,4,5,6}, {2,3,4,5}, {2,3,4,6}, {2,3,5,6}, {2,4,5,6} and {3,4,5,6}.

If we're working with five-choose-four, the combinations are {a,b,c,d}, {a,b,c,e}, {a,b,d,e}, {a,c,d,e} and {b,c,d,e}.

The easy way to code this is as an array of arrays. Let's do it this way:

```
# six choose four
sixfour[0]="0 1 2 3"; sixfour[1]="0 1 2 4"
sixfour[2]="0 1 2 5"; sixfour[3]="0 1 3 4"
sixfour[4]="0 1 3 5"; sixfour[5]="0 1 4 5"
```

```
sixfour[6]="0 2 3 4"; sixfour[7]="0 2 3 5"
sixfour[8]="0 2 4 5"; sixfour[9]="0 3 4 5"
sixfour[10]="1 2 3 4"; sixfour[11]="1 2 3 5"
sixfour[12]="1 2 4 5"; sixfour[13]="1 3 4 5"
sixfour[14]="2 3 4 5"
```

If you're paying close attention to the script we've been creating, you know that our hand is indexed starting at 0, so the card values are 0–5 instead. That's easily fixed by shifting every value down one in the predefined arrays.

Now it's just a matter of interpreting the different `sixfour` array values as a set of four cards to select, and here's how I do it:

```
for subhand in {0..14}
do
    /bin/echo -n "Subhand ${subhand}:"
    for thecard in ${sixfour[$subhand]}
    do
        showcard ${hand[$thecard]}
        /bin/echo -n " $showcardvalue"
    done
    echo ""
done
```

The `echo` statements (remember `-n` prevents a carriage return being appended to the output) aren't important to the final code, but for this interim solution, you'll see how it works by me just tucking it into the

end of the program, just after the code that deals, sorts and shows the full six-card hand:

Hand: AS, 5C, 7H, 8H, 9D, JS.

Subhand 0: AS 5C 7H 8H

Subhand 1: AS 5C 7H 9D

Subhand 2: AS 5C 7H JS

Subhand 3: AS 5C 8H 9D

Subhand 4: AS 5C 8H JS

Subhand 5: AS 5C 9D JS

Subhand 6: AS 7H 8H 9D

Subhand 7: AS 7H 8H JS

Subhand 8: AS 7H 9D JS

Subhand 9: AS 8H 9D JS

Subhand 10: 5C 7H 8H 9D

Subhand 11: 5C 7H 8H JS

Subhand 12: 5C 7H 9D JS

Subhand 13: 5C 8H 9D JS

Subhand 14: 7H 8H 9D JS

That's a lot of four-card hands, but at least we've got them. Next month, we'll actually start writing the code to evaluate hand values. Stay tuned! ■

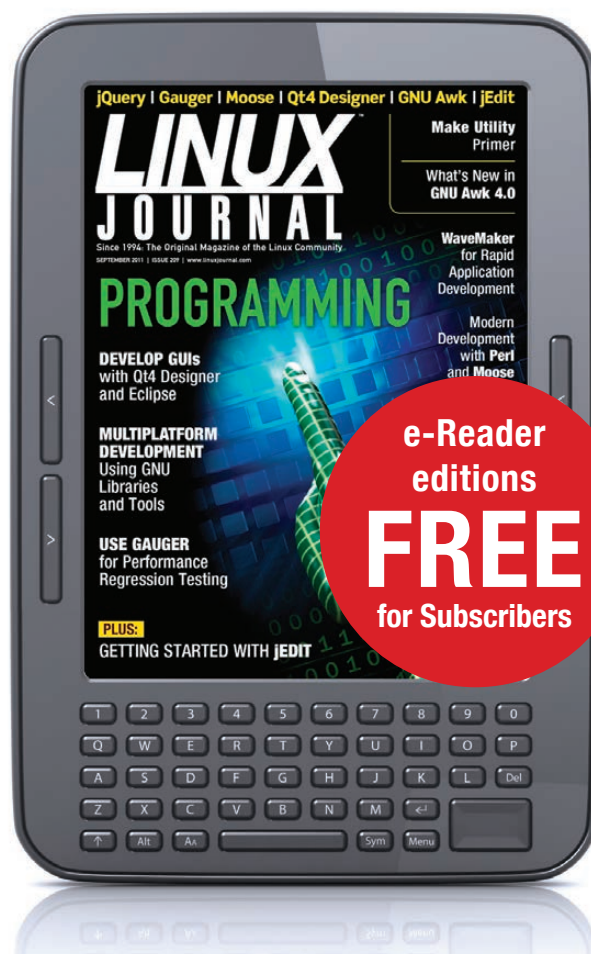
Dave Taylor has been hacking shell scripts for more than 30 years. Really. He's the author of the popular *Wicked Cool Shell Scripts* and can be found on Twitter as @DaveTaylor and more generally at <http://www.DaveTaylorOnline.com>.

LINUX JOURNAL

on your
e-Reader

Customized
Kindle and Nook
editions
now available

LEARN MORE





KYLE RANKIN

Temper Pi

Kyle returns to his quest to control a beer fridge with ever cheaper and smaller Linux devices. Today's candidate is a Raspberry Pi.

It was inevitable. Back when the Raspberry Pi was announced, I knew I eventually would use one to power a beer fridge. If you have been following my column through the years, you know that three years ago (see my Hack and / column titled "Temper Temper" in the August 2010 issue, <http://www.linuxjournal.com/article/10809>), I set up a temperature controller for my beer fermenting fridge with an X10 serial controller to control the power to the fridge and a heating pad, an inexpensive TEMPer USB thermometer to take the fridge temperature, and a simple Perl script.

As described in the original article, everything was connected to a spare Debian laptop I had lying around, and the setup worked great. Every minute, my Perl script would launch, take the temperature and control the power to a heating pad at the bottom of the fridge or the fridge itself, depending on whether it needed to be warmer

or cooler. A few months later (see my December 2010 Hack and / column "Working on My Temper", <http://www.linuxjournal.com/article/10904>), I decided that the laptop was overkill for this use case, so I replaced it with a low-power Pogoplug NAS device that was modified to boot Plugbox Linux, an Arch derivative. The Pogoplug has been powering my beer fridge reliably ever since.

That brings us to today. I just happened to have a spare insulated cabinet on the other side of the garage that I wanted to use as excess fermentation capacity. Unfortunately, the cabinet is too far away from the fridge for me to use the Pogoplug with an additional temperature probe, so I had to work out a different solution. I happened to have a spare Raspberry Pi lying around and realized it would be perfect for the job. All I needed to do was buy a new TEMPer USB probe and copy over my Perl script. The only big change I'd need was to have the

The problem with cheap electronics is that sometimes the internals change without your knowing.

script ssh back in to the Pogoplug so it could control the X10 devices (I have only one X10 serial adapter).

Prepare the Raspberry Pi

I didn't really need anything fancy for this setup. In fact, Arduino fans who read this probably would say that even the Raspberry Pi is overkill for such a simple project. I decided to use the standard Raspbian "wheezy"

Debian distribution. This procedure has been documented many times before, so I won't document it here. Because I was using a Debian-based release, I figured I even could follow the same steps from my original "Temper Temper" column.

New TEMPer Thermometers

The problem with cheap electronics is that sometimes the internals change

LINUX JOURNAL

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



linuxjournal.com/ios



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

without your knowing. Apparently, there are different USB thermometers all under the TEMPer name with the same packaging and overall look. Although I'm sure they all work with their included Windows software, it turns out that they need completely different software under Linux. Wouldn't you know it, the second TEMPer probe I bought turned out to be a different revision, so it requires a completely different set of software.

You can tell which TEMPer thermometer you have with `dmesg` and `lsusb`. If the `dmesg` output looks like this:

```
input: PCsensor Temper as
/devices/platform/orion-ehci.0/usb1/1-1/1-1.1/1-1.1:1.0/input/input0
generic-usb 0003:1130:660C.0001: input: USB HID v1.10 Keyboard
↳ [ PCsensor Temper]
on usb-orion-ehci.0-1.1/input0
usb 1-1.3: new high speed USB device using orion-ehci and address 5
input: PCsensor Temper as
/devices/platform/orion-ehci.0/usb1/1-1/1-1.1/1-1.1:1.1/input/input1
generic-usb 0003:1130:660C.0002: input: USB HID v1.10 Device
↳ [ PCsensor Temper] on usb-orion-ehci.0-1.1/input1
```

and you see something like this in `lsusb`:

```
$ lsusb
Bus 001 Device 051: ID 1130:660c Tenx Technology, Inc.
```

you have the older version of the TEMPer probe, and you can follow the steps from

my original "Temper Temper" column.

If instead `dmesg` says this:

```
[ 3.213110] usb 1-1.3: new low-speed USB device number
↳4 using dwc_otg
[ 3.339127] usb 1-1.3: New USB device found,
↳idVendor=0c45, idProduct=7401
[ 3.355218] usb 1-1.3: New USB device strings:
↳Mfr=1, Product=2, SerialNumber=0
[ 3.37771] usb 1-1.3: Product: TEMPerV1.2
[ 3.392684] usb 1-1.3: Manufacturer: RDing
[ 3.420037] input: RDing TEMPerV1.2 as
↳/devices/platform/bcm2708_usb/usb1/1-1/1-1.3/1-1.3:1.0/input/input0
[ 3.436838] generic-usb 0003:0C45:7401.0001: input:
↳USB HID v1.10 Keyboard
[RDing TEMPerV1.2] on usb-bcm2708_usb-1.3/input0
[ 3.465103] generic-usb 0003:0C45:7401.0002: hiddev0:
↳USB HID v1.10 Device
[RDing TEMPerV1.2] on usb-bcm2708_usb-1.3/input1
```

and `lsusb` says:

```
$ lsusb
Bus 001 Device 005: ID 0c45:7401 Microdia
```

then congratulations, you have the new TEMPer probe and will have to use completely different software.

Temper.py

The original software project to control these new-style TEMPer probes was at <http://www.isp-sl.com/pcsensor-1.0.0.tgz>, but right before I started writing this article, I got

word from Philipp Adelt that he had updated that project recently to work with Python. The updated project is hosted on Github at <https://github.com/padelt/pcsensor-temper>, and the Python version is at <https://github.com/padelt/temper-python> and has additional features, such as multiple probe management and SNMP support. So to get started with this project, I first needed to install git and then install a few Python libraries to provide USB support:

```
$ sudo apt-get install git python-usb
```

(Note that the project page also tells you to install the python-setuptools package and the snmp-passpersist Python library, but as I'm not planning to use SNMP, I skipped that step.)

With git installed, I pulled down the latest release of temper-python:

```
$ git clone git://github.com/padelt/temper-python.git
Cloning into 'temper-python'...
remote: Counting objects: 17, done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 17 (delta 4), reused 15 (delta 2)
Receiving objects: 100% (17/17), 19.07 KiB, done.
Resolving deltas: 100% (4/4), done.
```

The main Python program can be found under temper-python/

src/temper.py, and the project also includes a sample udev rule you can copy to /etc/udev/rules.d if you want access to the TEMPer probe by a user other than root. In my case, I was fine with root-only access, so I left the udev rules alone.

If you install the python-usb libraries correctly, and you have the supported TEMPer device, you should see output like the following:

```
$ sudo ./temper-python/src/temper.py
Found 1 devices
Device #0: 17.6°C 63.7°F
```

Once you see this, you know the temperature probe is working. I don't like running system programs like this within a home directory, so I decided to copy it up to /usr/local/sbin:

```
$ sudo cp ./temper-python/src/temper.py /usr/local/sbin/
```

Now in my case, I wanted to act on this temperature output, and I realized that my old temper.pl wrapper script wasn't going to cut it. Although I certainly could just modify it a bit to work with the new output, I figured a Perl script that called a Python script was just asking for too much hate mail. Instead, I decided to write a new simple wrapper script in bash called /usr/local/sbin/temper:

```
#!/bin/bash

TEMP_MIN="55"
TEMP_MAX="65"

LOGFILE='/var/log/temper.log'
TIME=`date +%b %d %T`
TEMPERATURE=`/usr/local/sbin/temper.py 2>/dev/null |
↳tail -n1 | cut -f4 -d ' ' | sed 's/.F$//`

if [[ $TEMPERATURE == "" ]]; then
    echo ERROR
    exit 1
fi

# B6 = peltier cooler B7 = heater

if [[ $TEMPERATURE < $(($TEMP_MIN - 1)) ]]; then
    ssh pogoplug "/usr/local/bin/br --port /dev/ttyUSB0 B7 ON"
    ssh pogoplug "/usr/local/bin/br --port /dev/ttyUSB0 B6 OFF"
    echo -e "$TIME\t$TEMPERATURE\tHON" >> $LOGFILE
elif [[ $TEMPERATURE < $TEMP_MIN ]]; then
    ssh pogoplug "/usr/local/bin/br --port /dev/ttyUSB0 B6 OFF"
    ssh pogoplug "/usr/local/bin/br --port /dev/ttyUSB0 B7 OFF"
    echo -e "$TIME\t$TEMPERATURE\tOFF" >> $LOGFILE
elif [[ $TEMPERATURE > $TEMP_MAX ]]; then
    ssh pogoplug "/usr/local/bin/br --port /dev/ttyUSB0 B6 ON"
    ssh pogoplug "/usr/local/bin/br --port /dev/ttyUSB0 B7 OFF"
    echo -e "$TIME\t$TEMPERATURE\tCON" >> $LOGFILE
else
    echo -e "$TIME\t$TEMPERATURE\t" >> $LOGFILE
fi
```

Although the logic of this script is similar to my old `temper.pl`, I just call

the `temper.py` script and use some pipes to pull out the temperature data I need. In this case, I also have to `ssh` back to my machine named “pogoplug” to run `/usr/local/bin/br` (the bottlerocket software that controls my X10 devices). This means I need to run `ssh-keygen` as the root user and then run `ssh-copy-id` to copy my public key to the pogoplug host. If I had wanted to replace my existing Pogoplug with a Raspberry Pi, I could just `apt-get install bottlerocket`, connect a USB-to-serial adapter with my X10 serial controller and run the `br` commands directly.

The final step is set up a cronjob. For that, I just create a file in called `/etc/cron.d/temper` with these contents:

```
* * * * * root /usr/local/sbin/temper 2>/dev/null
```

With that file in place, every minute, my script will probe the temperature and control the power to extra X10 appliances I have in place via the Pogoplug. It seems like I keep replacing these temperature probe systems with simpler and cheaper Linux machines. I wonder what’s next? ■

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.

Why Join USENIX?

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

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

Your membership dollars go towards programs including:

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

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

Join us at www.usenix.org



usenix

THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

OPEN
ACCESS



SHAWN POWERS

Dynamic DNS— an Object Lesson in Problem Solving

Got a computer problem? Learn to wield digital duct tape like a pro!

The other day in the *Linux Journal* IRC room (#linuxjournal on Freenode), I was whining to the channel about no-ip.com deleting my account without warning. My home IP address hadn't changed in a couple months, and because there was no update, it appeared abandoned. The problem was, although the IP address hadn't changed, I was using the Dynamic DNS domain name to connect to my house. When the account was deleted, the domain name wouldn't resolve, and I couldn't connect to my house anymore.

The folks in the IRC channel were very helpful in recommending alternate Dynamic DNS hosting services, and although there are still a few free options out there, I was frustrated by relying on someone else to manage my DNS services. So, like any nerd, I tried to figure out a way to host a Dynamic DNS service on my

own. I thought it would be a simple `apt-get install` on my colocated server, but it turns out there's not a simple Dynamic DNS server package out there—at least, not one I could find. So, again like any particularly nerdy nerd, I decided to roll my own. It's not elegant, it's not pretty, and it's really just a bunch of cheap hacks. But, it's a great example of solving a problem using existing tools, so in this article, I explain the process.

First, it's important to point out a few things. The purpose of this article is not really to explain the best way to make a self-hosted Dynamic DNS system. In fact, there probably are a dozen better ways to do the same thing. That's sort of the point. The more familiar you are with Linux tools, the more resourceful you can be when it comes to problem solving. I go through the steps I took, and

hopefully most readers can take my method and improve on it several times over. That type of collaboration is what makes the Open Source community so great! Let's get started.

My Particular Toolbox

We all have a slightly different bag of tricks. I'm in the "extremely lucky" category, thanks to Kyle Rankin tipping me off about the free Raspberry Pi colocation service. A full-blown Linux box with a static IP on the Internet is truly the sonic screwdriver when it comes to these sorts of things, but perhaps someone else's solution won't require a complete server.

Along with the Raspberry Pi server, I have a Linux server at home, a home router and a handful of domains I own. I also have accounts on several Web hosting servers, and accounts with cloud-based storage like Dropbox, Google Drive and a handful of others.

First Problem—What's My IP?

The beauty of Dynamic DNS hosting is that regardless of what your home IP address is, the same domain name will resolve even if it changes. Granted, my home IP address hadn't changed for months, but I still used the DNS name to access it. Therefore, when my account at no-ip.com was deleted, I had

no way to tell what my home IP was.

There are plenty of sites on the Internet that will return your IP address. Because I just had my Dynamic DNS account deleted, I really didn't want to depend on a free on-line service for detecting my IP address. Thankfully, this piece of the puzzle was simple. A few lines of PHP hosted on any Web host that supports PHP is all it takes. For example, my IP detection script is hosted at <http://snar.co/ip> (my personal domain—feel free to use it). See it in action in Figure 1. It contains nothing more than this:

```
<?php
// Save the IP to a variable
$ip_address = $_SERVER['REMOTE_ADDR'];

// To display the IP:
echo $ip_address;
?>
```

The frustrating part is that although I had a way to detect my IP address at

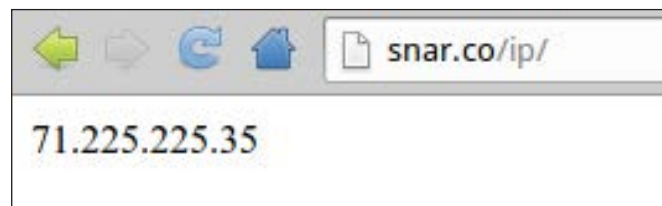


Figure 1. Unlike many “what is my IP” services, my script returns nothing but an IP address.

home, I wasn't actually at home, so I had a catch-22 situation. Thankfully, my wife was home. I texted her and asked her to visit my snar.co address and let me know the IP address she got back. Once I had that IP address, I could connect to my home server and set up some automation.

Let's Never Do This Again

It's not that I don't love texting my wife, it's just that hoping someone is home to check an IP address is not the best way to roll your own DNS. There's also the possibility that whatever Dynamic DNS solution I dream up might fail, and I want to make sure I always can figure out my home IP address.

There are a couple different ways I considered for making my IP address always accessible. The simplest was to set up a cron job to upload regularly the results of my PHP script to my Raspberry Pi or to one of my Web hosts. In order to do that, I'd simply have to set up SSH keys so my home server could upload a file without any interactive authentication. This is, in fact, what I recommend. As it happens, however, I'm lazy. What I actually did was set up a cron job that copied my IP address into a text file inside my Dropbox folder. It's not a better solution than scp-ing, but the

end result is the same. Here's what my cron job looks like:

```
1 * * * * /usr/bin/wget -r --quiet -O ~/Dropbox/Public/IP.txt  
↳ 'http://snar.co/ip'
```

It basically updates my Dropbox folder every hour with my current home IP address. Because Dropbox syncs onto every device and computer I own, it's always readily accessible.

But That's Still Not DNS

Now things get a little complicated. Because I have a full-blown server and a handful of domains to use, it would make sense to set up BIND and serve out a subdomain. BIND does have the ability to change a host entry with a remote update command. It requires setting up encryption keys, and of course, the BIND daemon has to be configured properly. Remember when I said I was lazy? It's still true. Because all I wanted to do was serve up a single domain name for my home IP address, I opted for something simpler.

DNSMasq is a very simple daemon that runs on my Linux-based home routers. It handles both DHCP services and DNS resolution. In both cases, the services are very stripped down and simplistic, but if all you need is simple DNS resolution, it doesn't get simpler than DNSMasq. It will look at

the server's `/etc/hosts` file and serve out those entries when queried. All I had to do was get my home IP address into my server's `/etc/hosts` file, and regularly send DNSMasq the HUP signal to reload its files. A simplistic DNS server was the final piece to the puzzle. Next came implementation.

Putting It All Together

The first step was to create a DNS entry that I could update with DNSMasq. This is simpler than most people realize. I just added an NS record pointing to my Linux server. So basically, I have an entry that looks like this:

```
home.mydomain.org. IN NS server.mydomain.org.
```

That means, "when resolving `home.mydomain.org`, or any subdomain of it, ask `server.mydomain.org` for the address." This is exactly what I want, because then any time I (or anyone else) tries to access `home.mydomain.org`, it will ask my server to resolve the name. The only thing left to do is to get my server, which is running DNSMasq, to respond with the proper IP address. That means a couple more cron jobs.

Remember my cool little `IP.txt` file I keep in my Dropbox? Well, in order to hack together the `/etc/hosts` file on my server, I had to modify my PHP script a little. In order to create an

output compatible with `/etc/hosts`, I changed it to:

```
<?php
// Save the IP to a variable
$ip_address = $_SERVER['REMOTE_ADDR'];

// To display the IP:
echo $ip_address;
echo " home.mydomain.org";
?>
```

Note the space before "home". Now the file in my Public Dropbox folder is a properly formatted `/etc/hosts` line. In order to combine that with my original `hosts` file, I created a folder `/etc/hosts.d/` on my server, and copied `/etc/hosts` to `/etc/hosts.d/00-original`.

Still with me? The last step is to run the following script on the server. I do this script every hour, so if my IP address changes, it should take at the most an hour before it's corrected. Here is the server script:

```
#!/bin/bash
/usr/bin/wget -r --quiet -O /etc/hosts.d/home
  ↳ 'https://dl.dropbox.com/xxx/IP.txt'
cat /etc/hosts.d/* > /etc/hosts
killall -SIGHUP dnsmasq
```

The first line retrieves the current IP address stored in my Dropbox Public folder. The second line creates a new

/etc/hosts file by concatenating all the files in /etc/hosts.d/. Then finally, I send the SIGHUP signal to dnsmasq, so it will reload the /etc/hosts file.

Final Thoughts

The thing I really like about this example as a way to demonstrate problem solving is that there are so many different ways to accomplish the same results. My solution is far from the best. Off the top of my head:

- I could have the script on my home server check for a change in IP address rather than just constantly updating. If there was a change, it could start the update process on my remote server rather than updating the hosts file every hour whether it needs it or not.
- Depending on what DNS hosting company you use, it's probably possible to change an address with a remote command. It's also possible there are free DNS servers out there directly supported by a client like ddclient.
- Because my solution requires a remote Linux server with a static IP, it makes my specific solution inaccessible to many people. That just means you need to think harder

in order to dream up a solution!

Where to Go from Here

It seems apropos to make my disclaimer again: the process I just explained is not the most efficient way to solve the problem of changing IP addresses. My methods are crude, my scripts are simplistic, and I haven't included any error correcting whatsoever. (What happens if I can't download the file from Dropbox? Will my script fail? Probably!) The purpose of this article is to make you think. Linux gives us tools that are powerful, flexible and above all else, useful. Sometimes you need to create a little digital duct tape and solve a problem on the fly.








What if you can't seem to come up with a solution to your particular problem? That's where the Linux community really shines. Stop in at the #linuxjournal channel, or attend a local LUG meeting. Folks there are much like me and are eager to help solve problems. Everyone loves a puzzle, and when you get to solve it with Linux? Awesome! ■

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.



LinuxFest Northwest

Bellingham, WA
April 27th & 28th

- 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! 

Hosted By

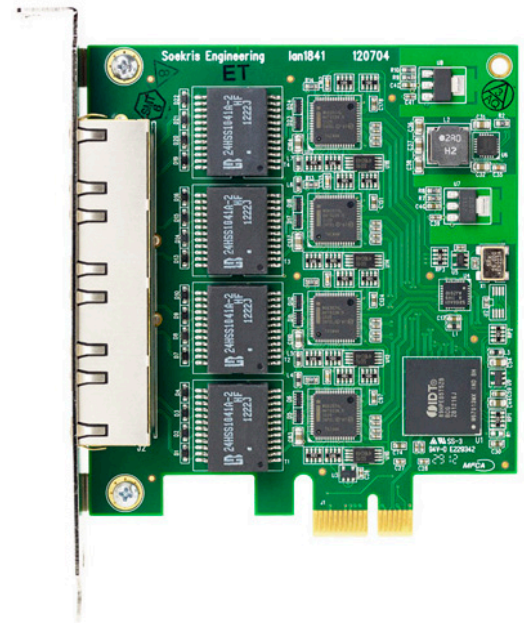


linuxfestnorthwest.org

Soekris Engineering, Inc.'s lan1841

The US-manufactured lan1841 from Soekris Engineering, Inc., is a short standard-profile PCI Express board featuring four independent Intel 82574L Gigabit Ethernet controllers for systems where the highest number of ports is required. The lan1841 is designed to work either with the Soekris net6501 for communications applications or as a standalone PCI Express LAN card. Together with the Soekris' net6501 series featuring a 1U 19" rackmount case, the net6501 PCI-Express riser and two lan1841s, users can configure a reliable, low-power rackmount unit with 12 ports. Alternatively, those looking for a four-port PCI Express Gigabit LAN card at a reasonable price for use with other hardware will find their needs met as well. The lan1841 features a three-year warranty.

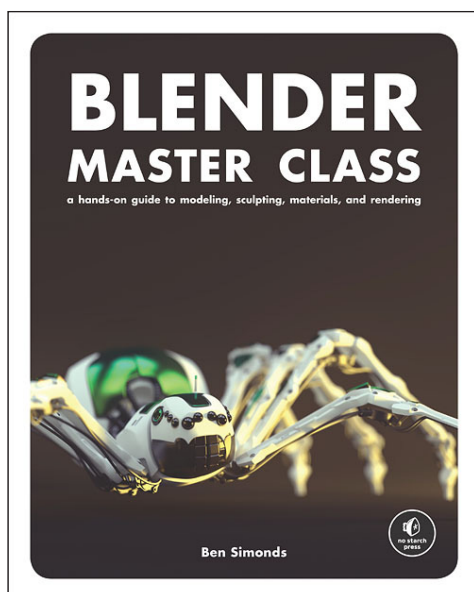
<http://soekris.com>



OpenERP 7.0

The essence of what's novel in the OpenERP 7.0 open-source business suite boils down to the concept of "Integrated Apps", which the company says "bring together the best of the ERP and the standalone app world". The idea is that as customers proceed to install applications, the new ones will integrate automatically with existing ones. Other aspects of the complete product redesign include a full-featured library of open-source business apps and out-of-the-box use with no prior configuration, all driven using "a stunning user interface". OpenERP 7.0 comes with new modules including touchscreen POS, contract management, fleet management apps, LinkedIn and Google docs integration, social collaboration between apps as an alternative to traditional e-mail, innovative kanban views to manage sales pipeline and projects, and an integrated customer portal, among others. The portal includes hundreds of modules developed by the community. OpenERP 7.0. is available either as an on-site deployment or via OpenERP's SaaS platform.

<http://www.openerp.com>



Ben Simonds' *Blender Master Class* (No Starch Press)

The miracle that is open-source has given us stunningly powerful free software like the Blender 3-D graphics tool. Harness the power of Blender with the new book *Blender Master Class: A Hands-On Guide to Modeling, Sculpting, Materials, and Rendering* by 3-D artist and animator Ben Simonds. Simonds' goal with *Master Class*, which is filled with eye-grabbing full-color artwork, is to inspire and equip new users to create their own masterpieces. Simonds walks

readers through three complete projects: a muscle-bound gargoyle, a futuristic robotic spider and ancient temple ruins overgrown with jungle. In the process, readers master the Blender interface and learn how to model and sculpt amazing models. Other topics include working with reference and concept art in Blender and GIMP, utilizing textures from GIMP, mastering the art of creating materials and lighting for scenes and rendering and compositing images to create striking images.

<http://www.nostarch.com>

AcousticSheep's Bluetooth SleepPhones Wireless

Counting sheep to get to sleep has never been so passé with the release of AcousticSheep LLC's Bluetooth SleepPhones Wireless, the company's latest addition to its line of doctor-designed miniature headphones integrated into comfortable, washable sports-style headbands. AcousticSheep assures users of SleepPhones that they will "fall asleep faster and drug-free" while using the product. Although AcousticSheep already offers similar wired products without hard earbuds not only for sleeping but also for exercising, this new product eliminates cords entirely by playing back media from any Bluetooth-enabled device. SleepPhones Wireless was selected as an Innovations Award Honoree from the Consumer Electronics Association in advance of CES 2013.

<http://www.sleepphones.com>



Velocity Micro's Cruz D610 and Cruz Q610

Hardware-maker Velocity Micro's design philosophy is to provide products that are "aesthetically pleasing, blazingly fast, competitively priced and designed and built in the USA". Although Velocity has yet to add "powered by Linux" to the list, the company recently released the Cruz D610 and Cruz Q610, affordable Android 4.1 tablets powered by dual-core and quad-core ARM processors, respectively. Both devices feature a 10" capacitive touchscreen, camera (D610 front-facing, Q610 front- and rear-facing), Google Play, Flash, custom software suite and access to Kindle for Android and the Amazon app store. The company says that these new tablets achieve this year's goal to blend the enthusiast level and the mainstream to create products with mass appeal.

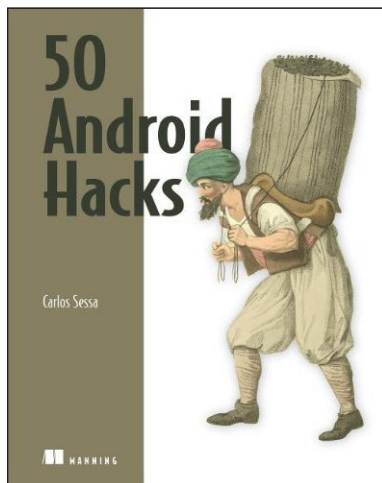
<http://www.velocitymicro.com>



Red Hat Enterprise Linux

The tried-and-true Red Hat Enterprise Linux (RHEL) open-source enterprise platform recently was upgraded to version 6.4. The new release, which continues in RHEL's tradition of delivering technology that is tested and stable, includes updates to the platform's existing feature set and provides new functionality in the areas of identity management, filesystem, virtualization, storage and productivity tools. Identity management is improved via SSSD enhancements that strengthen the interoperability experience with Microsoft Active Directory. Filesystem enhancements from pNFS deliver performance improvements with the addition of Direct I/O for faster data access. Virtualization is more robust, thanks to the inclusion of the Microsoft Hyper-V Linux drivers and other improvements. Storage management is now easier thanks to new system log features that simplify the process of locating specific devices. Productivity-tool upgrades include interoperability improvements with Microsoft Exchange, calendar support in Evolution and support for newer Wacom tablets.

<http://www.redhat.com>



Carlos Sessa's *Android Hacks* (Manning)

News from the DIY Android desk is the availability of Carlos Sessa's *Android Hacks*, a new Manning Publications book of 50 short and simple programming hacks that are destined to "save you time, stretch your skills and maybe even make you smile". The 50 "little gems" are organized into categories such as Layout, Animation, Patterns and UX, and each covers just a few pages,

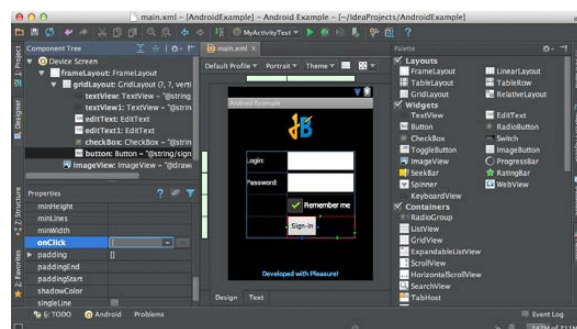
including annotated source code. Some examples of the hacks include creating a wizard form using a Gallery, using Scala inside Android, adding an MP3 to the Media ContentProvider and batching database operations. Most hacks work with Android 2.x and greater.

<http://manning.com/sessa>

JetBrains IntelliJ IDEA

The headlining new feature in IntelliJ IDEA 12, the latest volume of JetBrains' Java IDE for Web, enterprise and mobile development, is a brand-new, much faster compiler. JetBrains says that it pulled out all the stops, rebuilding the compiler from the ground up and moving it to a separate process. The result is that projects can be compiled automatically in the background on every change a developer makes, so that it can run it almost instantly any time. IntelliJ IDEA also embraces the latest versions of Java, including its previews and new features that every developer is eager to try. With v.12, JDK 8 already can be enjoyed with code assistance for new syntax, such as lambda expressions, method references and default methods. Also in the new edition, support for Android is expanded with a new Android UI designer. Aesthetically speaking, IntelliJ IDEA 12 explores the "darker side" of productive coding by unveiling a stylish new dark look and feel, dubbed "Darcula". JetBrains says that Darcula makes the interface clearer and more functional, minimizing distraction so users can focus more on the code and less on the IDE.

<http://www.jetbrains.com/idea>



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.

Things a Front-End Developer Should Know about Drupal



Experienced with front-end development but new to Drupal? Good with Drupal configuration but shy about theming? Read on to find out how to get started with custom themes, working with templates, JavaScript and CSS, considerations for mobile and tips for performance. Plus, learn what's next in Drupal 8.

ALEXANDER CASTILLO

Drupal allows you to set up a site in no time and lets you focus on what really matters—content. With Drupal, front-end developers can eliminate most of the boilerplate code and jump right in to styling (otherwise known as “theming”) fairly quickly. Drupal provides you with the base, markup and most of the needed functionality right out of the box.

This article, written by a seasoned Drupal front-end developer, intends to walk you through the must-know Drupal front-end development techniques and best practices, specifically as related to the latest stable version, Drupal 7. This article covers the basics on how to create a custom theme from scratch and how to enhance it in different ways with JavaScript and Media Queries for mobile support. It explains the administration interface and shows ways to design pages using modules—for example, Panels. Finally, it covers performance issues and offers pointers on what to expect in the next version, Drupal 8.

The “Theme” Layer

There are two ways of theming a Drupal site. One, use a base theme that provides most layout and

functionality needs in combination with a sub-theme for custom styles. The main benefits of using a contributed base theme include 1) theme settings in the administrator interface, 2) multi-column layout options, 3) predefined print style sheets, 4) contextual classes added to the markup and 5) cross-browser compatibility fixes.

Or, option two, you can create a theme from scratch and control, in detail, everything in the theme layer and add features as needed. I personally like having total control of what is in my theme; therefore, I always create a theme from scratch. I think every project deserves this treatment, so, keeping that in mind, let’s explore a creating custom theme and compare it with a base theme, so you can decide your preferred approach.

Creating a Custom Theme

Creating Drupal themes from scratch is easier than it sounds. What do you need? A folder and an .info file will get you started:

- Create a folder in /sites/all/themes.
- Name the folder with the name of the theme. The name should be all lowercase with

Drupal's API allows for customization of any of the page's areas in a very modular way.

no spaces—for example, /sites/all/themes/cooltheme.

- Create a new document with your preferred text editor or IDE.
- Add some base settings to this document (see below).
- Save the document inside the newly created theme folder.
- Name the document the same as the folder plus the .info extension—for example, cooltheme.info.

Here are the .info base settings:

```
name = cooltheme
description = Description of the theme.
core = 7.x
engine = phptemplate
regions[left] = Left sidebar
regions[content] = Content
regions[header] = Header
regions[footer] = Footer
```

Once the theme has these components, enable it from the administration interface. On the

Drupal toolbar, click on appearance (URL: /admin/appearance). The new custom theme should be listed on this page; enable it, and set it as the default.

TIP: Explore other contributed themes. See how they do it!

Working with Templates

Drupal's API allows for customization of any of the page's areas in a very modular way. Customize the page wrapper (html.tpl.php), the page itself (page.tpl.php), regions within the page (region.tpl.php), blocks within regions (block.tpl.php), nodes (node.tpl.php) and fields within nodes (field.tpl.php). Add these files to the theme folder if you want to override the templates. You can get these templates from core modules like node, blocks and field. Simply copy and paste the templates inside the theme's folder, and let the overriding begin!

Note: make sure to clear Drupal's cache when adding a new template or editing the .info file. On the Drupal toolbar, click on Configuration→Performance→Clear

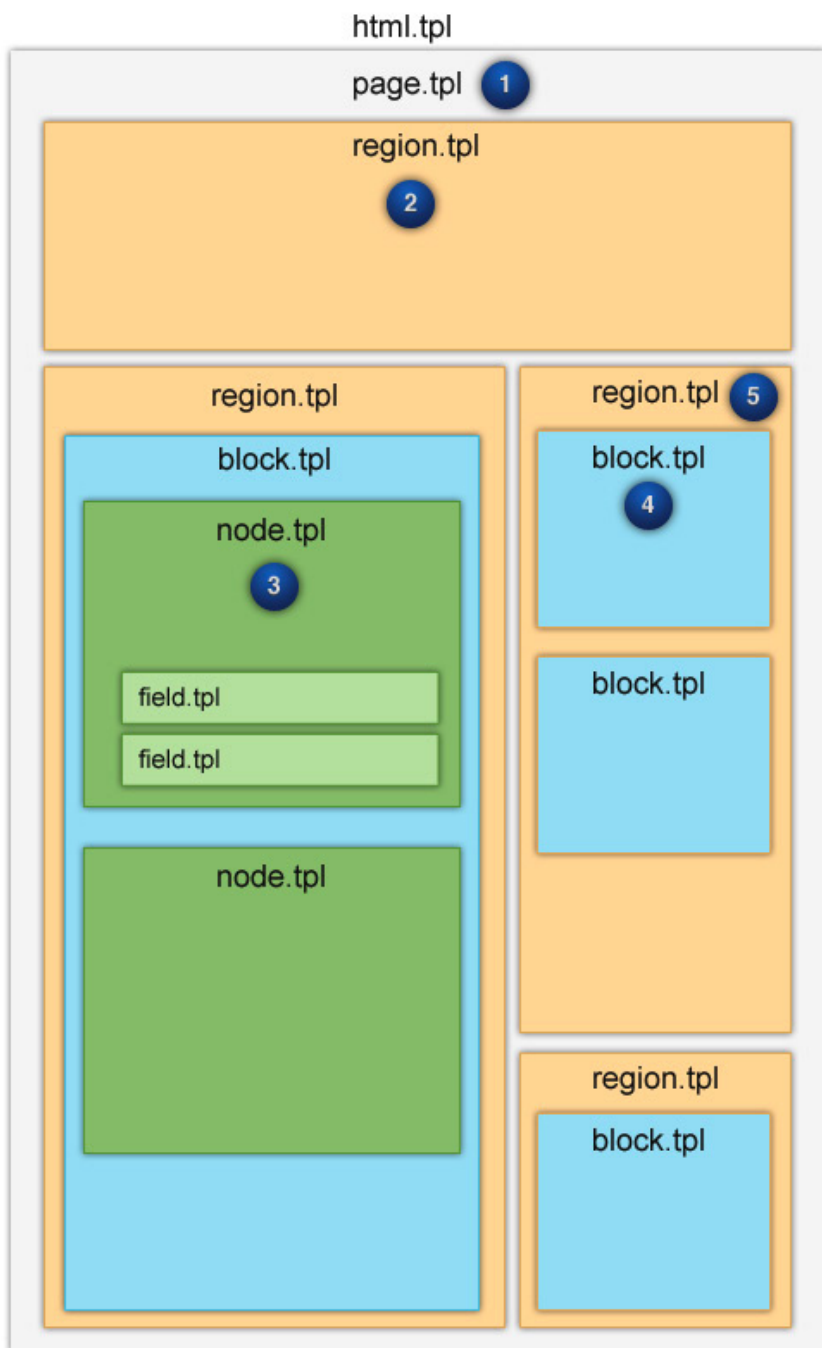


Figure 1. Drupal's File Template System Hierarchy and Priority

all caches (URL: admin/config/development/performance).

Are you connecting the dots already? As shown in Figure 1, there are many regions defined in the .info file. See what content displays in the regions by going to the block page in the administration interface. On the Drupal toolbar, click on Structure→Blocks (URL: /admin/structure/blocks).

Working with CSS

To get started styling your site, create a folder called css inside the theme's folder. In there, you can add all the stylesheets and then reference them from the .info file by adding the following line(s):

```
stylesheets[all][] = css/style.css
stylesheets[all][] = css/layout.css
stylesheets[all][] = css/etc.css
```

TIPS: You can download the Drupal 7 cheat sheet from http://batayneh.me/sites/default/files/Drupal7-theming-cheatsheet_0.pdf.

Core modules can be found in the /modules/ folder in the root.

Do you ever need to do a “one-second” CSS change remotely? Before when doing that you needed to download or pull the code, make the edit, and upload or push, right? This incredible module called Live CSS (http://drupal.org/project/live_css) allows you to edit CSS or LESS live within the page, and save the changes back to the file instantly. This tool is very handy when developing a Drupal site. There’s no need to switch back from the text editor or IDE to the browser and refresh the page. All of your edits happen right away.

Working with JavaScript

JavaScript adds behavior and interaction to the theme. Popular JavaScript libraries like jQuery and jQuery UI already come with Drupal. To add custom JavaScript code, first create a folder called `js` inside the theme’s folder. Then, simply add JavaScript files to the `js` folder and reference them in the theme `.info` file by adding the following:

```
scripts[] = js/main.js
scripts[] = js/etc.js
```

You also can use Drupal’s PHP function `drupal_add_js` in the `template.php` file as follows:

```
<?php
function example_preprocess_html(&$variables) {
    $options = array(
        'group' => JS_THEME,
    );
    drupal_add_js(drupal_get_path('theme', 'example'). '/script.js',
        $options);
}
?>
```

Once the JavaScript file is referenced in the theme, start adding behavior and interaction to the theme. Best practice is to wrap the JavaScript code inside a jQuery closure function—something like this:

```
(function($) {
    // Javascript code
})(jQuery);
```

But wait, doesn’t Drupal’s API provide support for JavaScript? Yes, it actually provides specific JavaScript functions through jQuery. So, you can pass information from the PHP code and attach Drupal-specific behaviors with something like this:

```
(function($) {
    Drupal.behaviors.customToggler = {
        attach: function(context, settings){
            $(".toggler", context).once('custom-toggler').click(function(){
                $(this).next().slideToggle("slow");
            });
            return false;
        }
    };
});
```

```
    }).next().hide();  
  };  
};  
) (jQuery);
```

If you're using modules that include other JavaScript libraries, such as MooTools or YUI, use jQuery's `$.noConflict()` function to avoid confusion over the famous dollar-sign alias (`$`).

To practice progressive enhancement like some of us do, explore Modernizr. Adding Modernizr to your project is as easy as installing the Modernizr module from drupal.org (<http://drupal.org/project/modernizr>). Modernizr is a JavaScript library that detects HTML5 and CSS3 features in the user's browser.

Mobile Support

When optimizing a Drupal Web site for mobile, it is possible to create a sub-theme with all mobile features. This allows control of the experience of the mobile site vs. the desktop site. Drupal offers a few things to make the job easier. For starters, modules like ThemeKey will allow a switch to the mobile sub-theme when the user is accessing the Web site from a mobile device.

Drupal is "technology-agnostic" regarding the front-end technology, so making a responsive Web site in

Drupal is mostly the same as with any other platform. Define the media queries in the CSS file, or simply add them to the `.info` file as follows:

```
stylesheets[all and (max-device-width: 480px)][] = mobile.css
```

Other Drupal modules like Adaptive Images will play nicely with Drupal's Image Styles module to create, cache and deliver device-appropriate re-scaled versions of the site's embedded images automatically.

Working with the Drupal Administration Interface

Now that you have a custom theme in place, let's look at what the Drupal administration interface offers to front-end developers. Repetitive development tasks like applying custom filters to images, embedding custom fonts and defining custom layouts have never been easier than they are with Drupal!

Image Styles

The image styles module comes with Drupal out the box. With this module, you can create image instances of the original image and apply effects to those instances dynamically—no more Photoshop batch scripts; let Drupal do it for you, on the Web.

Effects include:

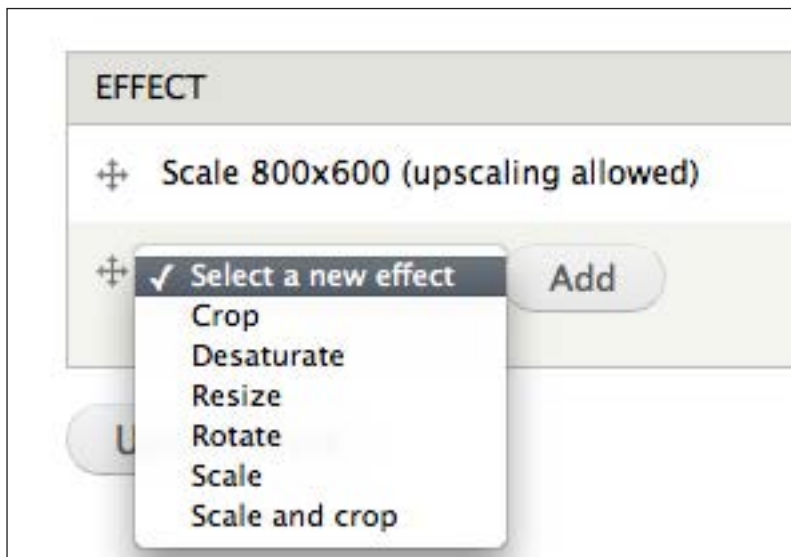


Figure 2. Drupal Core's Image Styles Module Administration Screen for Presets

Configuration→Image styles (URL: admin/config/media/image-styles).

@font-the-face Module

This module provides an administrative interface for browsing and applying Web fonts embedded via CSS from a variety of sources and font providers, such as the following:

- Crop
- Desaturate
- Resize
- Rotate
- Scale
- Scale and crop
- Adobe Edge Web Fonts
- Font Squirrel
- Fontdeck
- Fonts.com
- Google Fonts
- Typekit.com

Other effects, such as watermarking, overlays, brighten/darken, rounded corners and many more, are available via the ImageCache Actions contributed module (http://drupal.org/project/imagecache_actions).

From the Drupal toolbar, click on

Here's the link: <http://drupal.org/project/fonttheface>. It also is possible to import local fonts in all Web formats: EOT, TTF, WOFF and SVG.

Icon Fonts Module

What about the idea of managing the icon fonts from the administration interface? With this module (<http://drupal.org/project/iconfonts>),

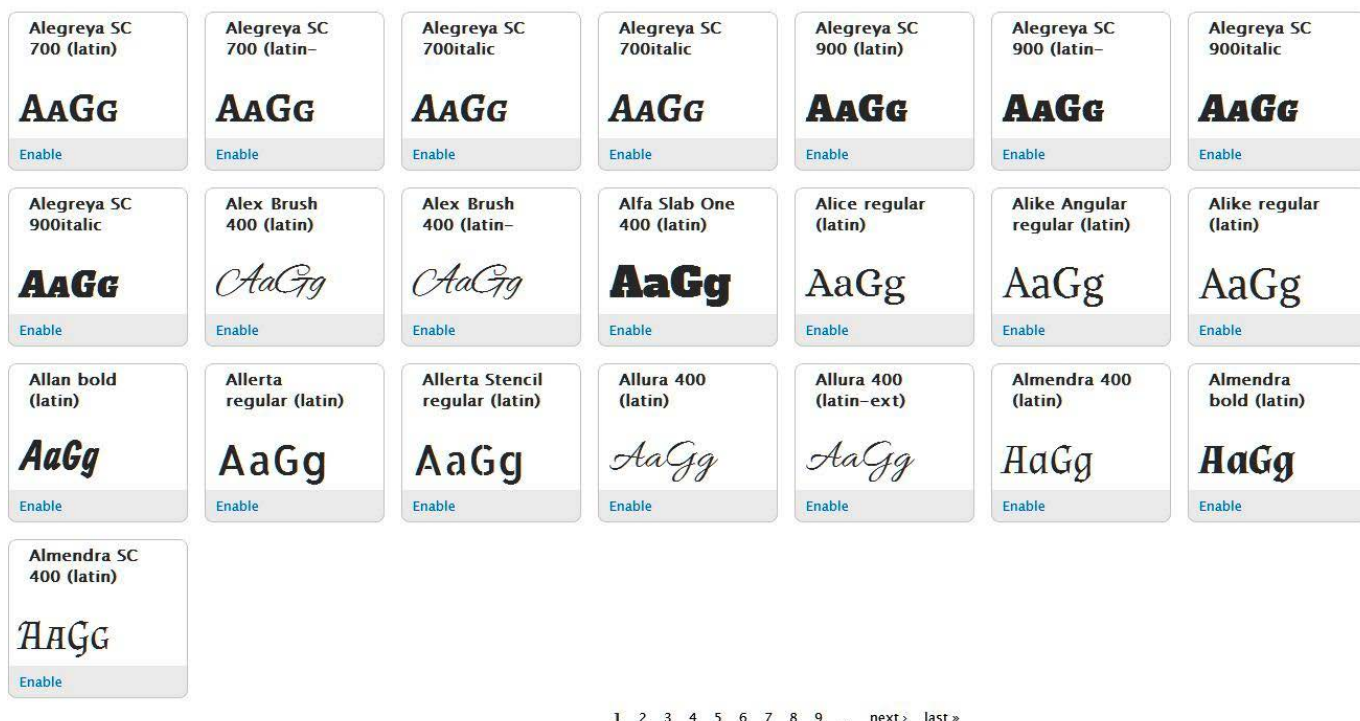


Figure 3. Google Fonts Drupal Module Administration Screen for Fonts

you can do so. Install this module to get started with the font icon at IconMoon (<http://icomoon.io>) and bring it to Drupal in no time.

Google Fonts Module

Those familiar with Google Fonts know how convenient it is to load the custom fonts in a site with a couple lines of code. Well, with Drupal, there is no need to use those lines of code any more. Install the Google Fonts module, enable the desired fonts, map it to the CSS selectors from the Drupal administration interface and enjoy: http://drupal.org/project/google_fonts.

Panels Module

If you like the idea of being able to create customized page layouts and manage the content with a nice-and-neat drag-and-drop interface, the Panels module is for you. Explore it at <http://drupal.org/project/panels>.

With Panels, you can do the following:

- Create customizable and re-usable layout for pages and blocks.
- Insert the previously created content in any region.
- Insert and create re-usable custom content in any region.

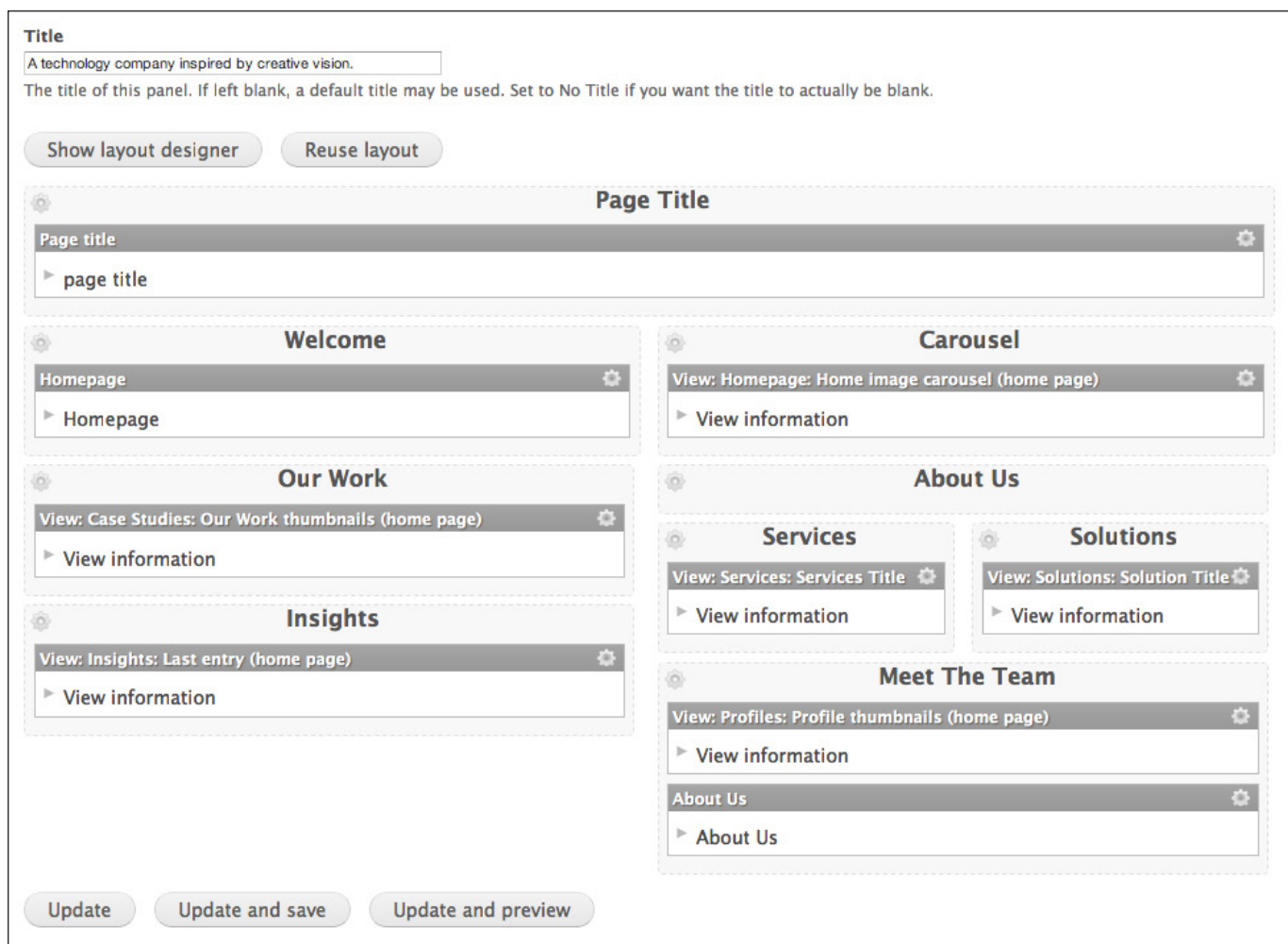


Figure 4. Panels Drupal Module Administration Screen for Content

- Customize the node pages and node forms at the field level.
- Add conditional rules to hide or show the panes.
- Customize the user pages and user forms at the field level.
- Switch panes based on context—for example: user role, is mobile and so on.
- Edit and modify the order of the panes at all times with drag and drop.
- Add CSS classes and IDs.
- Cache your panes individually.

CACHING

Cache pages for anonymous users

Cache blocks

Minimum cache lifetime

Cached pages will not be re-created until at least this much time has elapsed.

Expiration of cached pages

The maximum time an external cache can use an old version of a page.

BANDWIDTH OPTIMIZATION

External resources can be optimized automatically, which can reduce both the size and number of requests made to your website.

Aggregate and compress CSS files.

Aggregate JavaScript files.

jQuery compression level

Production (minified)

Development (uncompressed)

jQuery and jQuery UI CDN

None

Google

Microsoft

Use jQuery and jQuery UI from a CDN. If the CDN is not available the local version of jQuery and jQuery UI will be used.

Figure 5. Drupal Core’s Performance Administration Screen for Caching, File Aggregation, jQuery and jQuery UI CDN

- Export the configuration as PHP code.

Performance

Now that the theme is ready and configurations on the administration interface are complete, there are a few things you can do to optimize

the site’s performance. Let’s start with Drupal core’s performance settings.

From the Drupal toolbar, click on Configuration→Performance (URL: admin/config/development/performance).

On this screen, activate caching for anonymous users and blocks, CSS and

JavaScript aggregation and jQuery and jQuery UI file compression and CDN options. By enabling caching and aggregation alone, the site performance increases significantly.

Furthermore, modules like Varnish, Boost and Memcache will offer more robust caching solutions. And, it is recommended to use a CDN in combination with the CDN module when high-performance is in order.

Note: for performance testing and analysis, visit: <http://yslow.org>.

The Future of the Drupal Front End

How is Drupal moving forward, and in what areas will it benefit front-end development? There are three major initiatives for Drupal 8 that front-end developers should be very excited about: Design, HTML5 and Mobile.

The Drupal community has been improving the user experience constantly, and for version 8, they plan to take it another step further. As for HTML5 and mobile, the goal is to make HTML5 the default output and leverage it to make Drupal a fully mobile platform. Other unofficial initiatives like “Spark” may bring a responsive layout builder to Drupal 8. Stay tuned!

So, what have you learned? Although there are many benefits to using an out-of-the-box theme, it's worth it to explore creating a

custom theme to have control of everything in the theme layer and features as needed. Take advantage of template overrides to craft markup, add custom CSS classes, and add behavior and interaction to pages with JavaScript. For mobile, you can create a sub-theme for mobile support or just simply use CSS3 Media Queries in support of responsive design. Before importing a third-party library in the theme, check whether a Drupal integration is available. Most likely a module already exists! Leverage design modules like image styles, custom fonts and panels. For performance, be sure to enable caching and aggregation, explore contributed caching modules and think about a CDN. Finally, get excited about the future. Drupal 8 will make everything even better for cross-device optimization. If you have any questions, please let me know: alex.castillo@door3.com. ■

Alexander Castillo is the Manager of Front-End Development, a solutions architect and an essential part of the Drupal team at DOOR3. He has been building Web sites and Web applications since age 15 and is both a user-experience designer and developer. His languages and technologies include: HTML5, CSS3, JavaScript, jQuery, AngularJS, EmberJS, Adobe Design Suite, PHP and Drupal. He also specializes in cross-browser, cross-platform compatibility, responsive design and mobile Web development.



10th Annual

2013 HIGH PERFORMANCE COMPUTING LINUX FOR WALL STREET Show and Conference

APRIL 8, 2013 (Monday)

ROOSEVELT HOTEL, NYC
Madison Ave and 45th St, next to Grand Central Station

Register for
the Free Show

Big Data, Cloud, Data Centers, Networks, Low Latency, Cost Savings, Speed, Virtualization for Wall Street. The Largest Show of HPC in 2013.

See new products in operation. Network with friends. Efficient one-day program.

April 8, 2013: 10th Annual High Performance Computing Linux covering high speed, low latency, reducing costs and total cost of ownership. The explosion of data mandates new technology to harness the power of financial information.

Register for the free show today. Save time. Free badge will be mailed to you if you register online,

Register for the full conference program and save \$100. Conference only \$295 before deadline. Conference includes general sessions, concurrent sessions, industry luncheon, exhibits, handouts and special Sponsor programs. See program online.

Don't have time for the conference?
Attend the free show from 8-4 pm.



Jean Staten Healy
Director, WW Cross-IBM Linux & Open Virtualization, IBM



Mike Day, Chief Virtualization Architect for IBM's Open Systems Group, IBM



Dino Vitale
Director
Morgan Stanley Cross Technology Services



David Rukshin
SVP
DE Shaw



Jeffrey M. Birnbaum
Founder & CEO
60East Technologies, Inc.



Nan Boden
CEO
Myricom, Inc.,



Dave Malik
Director, Advanced Services,
Cisco



Paul Jameson
Global Senior Director,
Financial Services,
Cisco



Tom Steinert-Threlkeld
Editorial Director,
Money Mgmt Group,
Source Media



Emile Werr
VP, Head of Enterprise Architecture,
NYSE Euronext



Adam Sussman
Partner, Director of Research,
TABB Group

2013 Sponsors



ARISTA



LSI
Storage. Networking. Accelerated.



DataDirect
NETWORKS

Myricom

HPCI
INTELS

HPC
in the Cloud

LINUX
MAGAZINE

SECURITIES
TECHNOLOGY
MONITOR

datanami
REGULATORY ANALYTICS • RISK REPORTS

LINUX
JOURNAL

Integration
developer news | Architecting the
Intelligent Enterprise

information
management
How Your Business Works

TRADERS
MAGAZINE

OMG
WE SET THE STANDARD

GREEN
computing
REPORT

WSTA
Wall Street Technology
Association

Show Hours:
Mon, Apr 8 8:00-4:00
Conference Hours: 8:30-4:50

Show Management:
Flagg Management Inc
353 Lexington Ave, NY10016
(212) 286 0333
flaggmgt@msn.com

Visit: www.flaggmgt.com/linux





SPEED UP YOUR DRUPAL DEVELOPMENT

Improve and monitor site speed

Monitor, manage and improve the end-to-end performance
of your web apps with one simple tool.

Using **SALT STACK** and **VAGRANT** for **DRUPAL** **DEVELOPMENT**

With Vagrant and Salt Stack, you can have identical development environments and share them across local, development, stage and production using small text configuration files.

BEN HOSMER

What if, just like Bill Murray in *Groundhog Day*, you could wake up to a fresh and identical development environment completely free of yesterday's experiments and mistakes? Vagrant lets you do exactly that.

Or, what if, like Jake Epping in Stephen King's *11/22/63*, you could make changes and script the past without fear, play around with some new Drupal modules, and quickly reset everything just by leaving and then walking back down the stairs of the pantry again?

Would you like to automate the creation and installation of a pristine Drupal environment, instead of manually installing Apache, PHP and all of the needed libraries?

Recently, I read a post from Treehouse Agency titled, "End 'Works on My Machine' Surprises with Vagrant" (<http://treehouseagency.com/blog/steven-merrill/2011/11/02/end-works-my-machine-surprises-vagrant>), which is about using Vagrant and Puppet for Drupal development. You may have seen it as well on Drupal Planet (<http://planet.drupal.org>) and wondered about the benefits described there.

This is a great technique that outlines using various tools to build a consistent development environment

quickly that is repeatable and shareable with everyone on your team.

Linux Journal also recently featured an article that introduced Vagrant (<http://www.linuxjournal.com/content/introducing-vagrant>). After you're finished here, I urge you to go read that one too, because it offers more general information about Vagrant itself.

Salt Stack (<http://saltstack.org>) is a tool similar to Puppet, but it's powered by Python, instead of Ruby. If you are interested in standalone Salt installation and use, check out my November 2012 *LJ* article on that topic (<http://www.linuxjournal.com/content/getting-started-salt-stack-other-configuration-management-system-built-python>), where I introduce Salt Stack and show how to install it to control one or thousands of other machines. I won't go in depth into Salt's installation here, because Salty-Vagrant takes care of all of that for you.

I've been using Salt, Vagrant (<http://vagrantup.com>) and a Vagrant gem called Salty-Vagrant (<https://github.com/saltstack/salty-vagrant>) for Drupal development quite a bit lately. I've found that having a standard configuration that mirrors my development, testing, staging and

production environments has streamlined my workflow and helped prevent a lot of the unknowns between different stages of the Drupal development life cycle. I've been able to minimize a lot of the errors and headaches that come with integrating multiple software stacks and speed up my workflow too.

I also enjoy being able to try new things quickly and easily without spending hours reconfiguring and re-installing the entire software stack needed for Drupal development.

I then can share these configurations quickly and easily with other developers on my team and know that we are each working with the same versions of software already installed. In this article, I describe how you can do that too.

With Salt Stack, you store your configurations in text files, similar to Puppet. The difference is that the configurations rely on YAML (<http://yaml.org>) templates instead of Ruby code. This makes them easy for humans to read, even if they aren't Python coders.

These configurations can be used to launch development machines, as well as testing, staging and production machines—and all of them are identical and easily transportable. I use Git to manage my team's configurations. This

brings all of the great aspects of version control to my server configurations too.

With Vagrant, you easily can delete a development environment and then re-create it in about as much time as it takes for the server to boot and install the packages and software you specify—kind of like a quick undo or reset button.

Getting Started

You need to download and install a few things to get all of these pieces to work together. Each component has very good documentation on-line for installation and use, so I won't duplicate the instructions here. Instead, follow the links for each piece of software and check the projects' sites for the most up-to-date information.

This article focuses on using an Ubuntu-based 64-bit VirtualBox, but Salty-Vagrant with Salt Stack supports BSD, Red Hat, OS X and Windows virtual machines as well.

The beauty of using Vagrant with Salty-Vagrant is that the configuration files, also known as SLS files, transfer exactly to production servers, as well as anywhere else you might use them. The benefit of creating one template and then re-using it again and again can help you direct more energy

toward Drupal development, instead of troubleshooting server-software installation too.

You need to do this installation and configuration only once. After that, it is as simple as `vagrant up!`

Install VirtualBox

VirtualBox allows you to run a guest operating system inside your own host operating system. It is similar to dual-booting, except you still can use your main operating system and have access to additional running operating systems as well. For example, you can run an Ubuntu server inside your Windows, Mac or Linux host operating system and still access your e-mail and text editor from your native host operating system.

Follow the instructions at <http://virtualbox.org> to install VirtualBox. Versions are available for most major operating systems. A major portion of popular Linux distributions also have packages available through their respective package managers. This is the most effective and efficient way to install the majority of tools I describe here.

Install Vagrant

Vagrant is a Ruby gem that allows you to create and destroy virtual machines quickly, and then reset

them almost instantly.

Gems are like Drupal modules, but for the Ruby programming language. These plugins let you extend Ruby's functionality and add additional capabilities. You should ensure that you have a recent version of Ruby installed. I recommend using your package manager if you don't have it, but a lot of distributions include Ruby already, so you likely won't need to install it. You easily can find out whether you have Ruby installed by typing `ruby --version` from your terminal.

Install Salty-Vagrant

Salty-Vagrant is another Ruby gem that brings the functionality and automated configuration tools of Salt Stack to Vagrant. Salty-Vagrant acts as a bridge between the Salt Stack configuration management system and Vagrant/VirtualBox. The project's page has installation instructions that are updated on a regular basis. It would be a good idea to refer to these instructions just in case something has changed after this article is published. If you know a little Ruby or shell scripting, feel free to join in this open-source project and help contribute back. You'll find the development community to be friendly and welcoming.

Get the Base Box

Get the base box from <http://vagrantbox.es>. You'll need a box in which to create and run your environment. Boxes are what Vagrant uses for your virtual machines, and they are simply virtual machines that have been configured specifically with Vagrant support. You can create your own from scratch, but it is much quicker and easier just to download a pre-made one that has all of the necessary Vagrant configurations already set up. I used the `precise64` box. This is an Ubuntu 12.04 64-bit Linux server with the necessary Vagrant tools and guest additions already in place. You're downloading an entire operating system, so be patient, because it might take a little while. It usually is best to download your box and then save it locally somewhere so you can use it again.

If you're using a 32-bit host operating system, you should instead download the 32-bit version.

Import the Box

Hopefully by this point, you have everything you need. Let's use the command line to interact with Vagrant and import your new box.

To import your box, use:

```
vagrant box add precise64 /path/to/your/downloaded/box
```

Relative path names don't work well here, so if you downloaded your box to your home directory, you need to type out the full path—for example, `~/bhosmer/precise64.box` should be typed out like this: `/home/bhosmer/precise64.box`.

The `vagrant box add` command creates a copy of your box and stores it so that Vagrant can access it. This box is used to create the instantiation of the particular box you happen to be using. The nice part is that you always can press the "reset" button by using `vagrant destroy` to delete anything you've done and set it back to its original state before you started. I go into that in more detail a little later.

Launch the Box and Verify the Guest Additions

Now, launch your box just to make sure that the VirtualBox guest additions are up to date. If they aren't, Salty-Vagrant may not work properly.

Where you store your Vagrant configurations is entirely up to you. I like to keep mine in my home directory under a `vagrant` folder, just so I can keep track of the different environments I have.

Create a new directory called `salty-vagrant` and `cd` into that directory.

Now use `vagrant init` to initialize

a Vagrantfile there. This is a standard Vagrant configuration file that is well documented. You won't need to change any of the configurations, except for line 10:

```
config.vm.box = "base"
```

If you followed the quick-start instructions from <http://www.vagrantup.com>, Vagrant specifies your default boxes as *base*. Change this to the name of the box you imported, *precise64*:

```
config.vm.box = "precise64"
```

Notice that I left the `.box` extension off.

Now that your Vagrantfile points to the correct box, use `vagrant up` to start your virtual machine.

If you get a warning that your guest additions are out of date, and it is quite likely that you will if you just installed VirtualBox, proceed to the next step. If you didn't, you can skip to the section "Add a Salty-Vagrant Vagrantfile".

Updating the Guest Additions

If you do need to update the guest additions installed on your box, in this section, I explain how to do it on the Ubuntu-based box you just

downloaded. This will be one of the few times that your changes will be permanent within a box. And this time, you want them to be. The guest additions allow your host operating system to interact with the guest operating system. This includes sharing files and network resources between them. You store your configuration files on your host operating system, and then they are copied to the guest operating system when you start it up each time, so it is important that the shared-folder functionality works.

After Vagrant finishes booting the guest operating system, you'll need to connect to this box using SSH. Use the Vagrant-supplied command: `vagrant ssh`.

You then will be logged in to the guest operating system as the "vagrant" user.

Updating the guest additions probably is the trickiest and most-involved task. If you run into any problems, lots of resources are available on-line. Most distributions mention a number of solutions in their respective forums, and the VirtualBox Manual (<https://www.virtualbox.org/manual/ch04.html>) also has some detailed information. I'm going to show you how to download and

update the virtualbox guest additions from virtualbox.org, and update them on the precise64 box that you just downloaded as an example.

First, you need some packages to build and compile the Linux kernel. Use Ubuntu's package manager, apt, to install them:

```
sudo apt-get install dkms linux-headers-3.2.0-23-generic
```

If you're using a non-Debian machine, search for the proper names in your particular package manager and use the header version that matches your particular kernel.

With kernel support now in place, browse to **<http://download.virtualbox.org/virtualbox>** and locate the folder that matches the version of VirtualBox that you have installed. Generally, you can find what version you have through the VirtualBox graphical interface or the warning message generated by Vagrant when you booted your virtual machine. Now, locate the `VBoxGuestAdditions_x.x.x.iso` that also matches the version of Virtualbox that you have installed. Replace the x in the filename with the specific version for your version of VirtualBox.

You can copy the URL to this file and paste it into the terminal window

of your guest Ubuntu system that you had previously SSH'd into. Use `wget` to download the ISO:

```
wget http://download.virtualbox.org/virtualbox/x.x.x/  
↳VBoxGuestAdditions_x.x.x.iso
```

Again, replace the x in the filename with the version that matches your VirtualBox installation.

You now should have an `.iso` file in your home directory.

After it's downloaded, mount it so that you can access the files contained in it by first creating a temporary directory in which to mount this `.iso`. Within your home directory, use this command:

```
mkdir -p tmp/vbox
```

Now, mount the `.iso` into that folder:

```
sudo mount -o loop VBoxGuestAdditions_x.x.x.iso tmp/vbox/
```

Within the `.iso`, you'll find a `VBoxLinuxAdditions.run` script. Copy this to your home directory:

```
cp VBoxLinuxAdditions.run ~
```

Now, make it executable by adding the execute flag:

```
chmod +x VBoxLinuxAdditions.run
```

And finally, run the installation script:

```
sudo ./VBoxLinuxAdditions.run
```

If you get a warning about failing to install the window system drivers, you safely can ignore it. The box you are installing these guest additions on is a server and doesn't have a window manager installed. You also can use the `--nox11` flag like this when you install the updated guest additions:

```
sudo ./VBoxLinuxAdditions.run --nox11
```

The last step in this installation is to reboot the machine to ensure that the guest additions are up to date with:

```
sudo reboot
```

After the box restarts, log back in with:

```
vagrant ssh
```

Now you can unmount the `.iso` using `sudo umount /mnt`, if it isn't already, and delete the `VBoxLinuxAdditions.run` and the `.iso` file to save space. You also optionally can do some housekeeping by deleting the `.bash_history` file.

Package the Updated Box

Now that the guest additions are updated, you should repackage a new box so the guest additions match each time you turn on this Vagrant box. This keeps you from needing to update them every time. As long as you don't change the version of VirtualBox that you are using, you won't need to update your guest additions. If you do, you'll need to update them again.

Using the `vagrant package` command, export your new box. This will shut down your machine and export a `package.box` file in the current directory.

Rename this to something more descriptive, like `myprecise64.box`.

Now add this new box to Vagrant:

```
vagrant box add myprecise64 myprecise64.box
```

The names you give your boxes are arbitrary and entirely up to you. Feel free to use a description that makes sense to you.

You're done with the original box now, so destroy it with `vagrant destroy`. You can remove it entirely from Vagrant with `vagrant box remove precise64`.

Add your new `myprecise64` box, or whatever you happened to call it, like you did previously with the base box you downloaded with `vagrant box add`.

Make sure to change the name in

your Vagrantfile to match the new box, and test it with `vagrant up`.

Congratulations! You now have an updated Ubuntu 12.04 VirtualBox suitable for Salt Stack and Salty-Vagrant automation.

Install the Vagrant Salty-Vagrant Gem

The final piece of software you'll need to install is the Salty-Vagrant gem. It allows the automatic installation of Salt Stack on your guest virtual machines and uses Salt Stack to initiate your configuration and installation of software on them as well.

Back on your host machine (that is, the one that is not your virtual machine), use the Vagrant gem command `vagrant gem install vagrant-salt` to install the Salty-Vagrant gem.

Add a Salty-Vagrant-Specific Vagrantfile

Now you need to add a slightly customized Vagrantfile to configure some Salt-specific parameters, share your state tree and automatically install Drupal and the Linux Apache MySQL stack after your VirtualBox is started. If you added a Vagrantfile previously, you can delete it now. You're going to add a smaller, more customized one that is available from the Salty-Vagrant project page at <https://github.com/saltstack/salty-vagrant>. You simply can copy

and paste this into a new Vagrantfile.

Locate the line `config.vm.box = "precise64"` and change "precise64" to the name of your box. If you've been following along, earlier in the tutorial, I exported my box as "myprecise64".

I also like to add guest/host network communications with this line to my Vagrantfile too:

```
config.vm.network :hostonly, "192.168.33.19"
```

You can use any available IP address for your internal VirtualBox network. Keep in mind, this is separate from your own network and is created by VirtualBox. This allows you to access your Drupal site from your own operating system's Web browser using the IP address that you specified. You can choose another IP address if you want.

Create Your Minion Configuration File

In the same directory as your Vagrantfile, create a salt directory, srv directory and lamp-drupal directory like this:

```
mkdir -p salt/srv/lamp-drupal
```

Create a new minion.conf file in your salt directory that contains this line:

```
file_client: local
```

This file instructs Salt that when the machine is booted to install all of the needed packages for running a Drupal Web server. I've even included a Drush installation.

This instructs Salt not to search for a master server, but instead to use the local file structure for its configuration information. This is what you'll see referred to in the Salt Stack documentation as a *masterless* configuration.

Locate the line in your Vagrantfile labeled `salt.minion_conf` and uncomment it by removing the `#`. This tells Vagrant where your minion configuration is located. The path is relative to your Vagrantfile.

Get the State Files

My Drupal SLS file is available in the salt-states repository (<https://github.com/saltstack/salt-states/tree/master/small/lamp-drupal>). You can use this to install all of the needed packages in Ubuntu for Drupal development. Download that file now and place it in your `/srv/salt/lamp-drupal` directory. You'll also need to create a `/srv/salt/top.sls` file that looks like this:

```
base:
  '*':
    - lamp-drupal
```

For more details about the top file, see my previously mentioned article about installing Salt Stack.

Add the State File to Your Shared Folder

Within the `salt/srv/lamp-drupal` directory, copy the `init.sls` file from the salt-states repository.

This file instructs Salt that when the machine is booted to install all of the needed packages for running a Drupal Web server. I've even included a Drush installation. If you examine the file, you'll find that it is fairly understandable already. I won't go into much more detail here, but feel free to take a look at the `init.sls` file to see what actually is being installed.

Now edit your Vagrantfile to reflect the location of this state file by editing the line:

```
config.vm.share_folder "salt_file_root", "/srv",
  ↳"/path/to/salt_file_root"
```

to:

```
config.vm.share_folder "salt_file_root", "/srv/salt", "salt/srv"
```

Vagrant Up!

With your minion configuration file in place and your state file ready to go, this last step will start your machine automatically, install Salt and install all of the necessary components needed for a Drupal development server.

Ready? Use `vagrant up` to start your machine.

You can watch the output and responses as Vagrant, Salty-Vagrant and Salt Stack automatically download all of the software you specified and configure it for you.

After the server is started, simply SSH to your virtual server using `vagrant ssh` again. You'll find a `/var/www` folder ready to receive your Drupal installation.

This SLS file is reusable across multiple environments, and I haven't configured it to change the permissions of the `/var/www` folder. You'll notice this folder is owned by `root`, just like the Ubuntu package manager sets it.

The Drush directory located within the `vagrant` user's home directory is also owned by `root`. You'll get an error with Drush if you don't change the ownership of that folder to the user that is initiating the Drush command.

Conclusion

By combining four very powerful tools:

Vagrant, Salty-Vagrant, VirtualBox and Salt, you can speed up your Drupal development considerably. Maintaining small development environments and then scripting their configuration helps create a win for everyone involved by adding consistency across all your developers' platforms. Maintaining small, text-based configuration files can save money on storage costs too. You and your developers also can save time transferring new configurations to remote servers.

You can try out new configurations and install additional libraries if you need or want to, without fear of breaking your local or production systems.

Each of the tools I have outlined are available for no cost and are open-source projects. You'll find the Salt development community to be one of the friendliest and most welcoming communities around the Open Source world.

Each project is updated on a regular basis, and it would be a good idea to review the documentation, as well as changes to the software, if you encounter any problems. ■

Ben Hosmer is a Drupal Developer and DevOp with RadiantBlue Technologies. He has made contributions to Drupal as well as Salt Stack. He is the maintainer of the Classified Drupal module, and he enjoys teaching Drupal.



PERCONA
LIVE

My SQL Conference & Expo 2013

April 22-25 • Santa Clara, CA

92+ My SQL experts in 15 tutorials
and 112 breakout sessions



www.percona.com/live

DART

*A NEW WEB
PROGRAMMING
EXPERIENCE*

**BREAKING AWAY
FROM JAVASCRIPT
FOR A FRESH PERSPECTIVE
ON WEB APP DESIGN.**

JAMES SLOCUM

JavaScript has had a long-standing monopoly on client-side Web programming. It has a tremendously large user base, and countless libraries have been written in it. Surely it is the perfect language with no flaws at all! Unfortunately, that is simply not the case. JavaScript is not without its problems, and there exists a large number of libraries and “trans-pilers” that attempt to work around JavaScript’s more quirky behaviors. JQuery, Coffeescript, Objective-J and RubyJS are examples of how people have been trying to make JavaScript better. However, a new contender is throwing its hat into the ring in a big way.

In comes Google Dart, a new JavaScript replacement language. Dart is a ground-up re-imagining of what JavaScript should be. It requires its own runtime environment (which Google makes available for free under the three-clause BSD license) and has its own syntax and libraries. Dart is an object-orientated language with a heavy Java-like feel, but it maintains many of the loved JavaScript paradigms like first-class functions.

So, why have I chosen to use Dart? Good question! I have chosen Dart because it is a clean break from JavaScript, and it has the object-orientated programming style I have

come to know and love. Because I have a Java background, the learning curve didn’t seem as steep with Dart as it does with JavaScript. Also, because Dart is so new, it gives me a chance to become an early adopter of the language and watch its evolution.

Installing Dart

Before you can program with Dart, you need to grab a copy from <http://dartlang.org>. I chose to install only the SDK; however, there is an option to grab the full Dart integrated development environment with the SDK included. Eclipse users will feel right at home with the IDE, because it is based on Eclipse components.

To install the SDK, I just unzipped the files and copied the whole directory to \$HOME/bin. Then I modified my path variable to look in the folder I created:

```
PATH=$PATH:$HOME/bin/dart-sdk/bin
```

Now I can run `dart`, `dart2js` and `dartdoc` from anywhere.

Language Features

The core Dart language is pretty straightforward. The basic data types available are `var` (stores any object), `num` (stores any number type), `int`, `double`, `String`, `bool`, `List`

(arrays) and Map (associative array). All of these data types are declared in the dart:core library. dart:core is always available and does not need to be imported. Functions also can be considered a data type, because Dart treats them as first-class objects. You can assign functions to variables and pass them as parameters to other functions or write anonymous functions in-line.

For flow control, you have the “usual” if, else if, else, for, while and do-while loops, break, continue, switch and assert. Exceptions are handled through try-catch blocks.

Dart has a lot of support for object-oriented programming. Classes are defined with the class keyword. Every object is an instance of some class, and all classes descend from the Object type. Dart allows only for single inheritance. The extends keyword is used to inherit from a parent class other than Object. Abstract classes can be used to define an interface with some default implementation. They cannot be instantiated directly, but can make use of factory constructors to create the appearance of direct instantiation. Abstract classes are defined with the abstract modifier in front of the class declaration.

Standard Library

Dart ships with an impressive standard library. I use a few of the libraries and classes in my examples here, so it will be helpful to have an idea of what they can do ahead of time. I can't cover all of them, but I cover the ones I personally find most useful.

As I said earlier, dart:core defines all of the core data types that are available. That's not all though! dart:core also contains the regular expression classes that are an invaluable addition to any standard library. Dart uses the same regular expression syntax as JavaScript.

dart:io provides classes that let your program reach out to the world. The File and Directory classes can be used to interact with the local filesystem. The File class also will allow you to open input and output streams on the specific file. If you want to write cross-platform code and allow users to specify the path of a file native to their operating system, the Path class provides a really nice Path.fromNative(String path) constructor that will convert Windows and UNIX paths to their Dart counterparts. Also included in this library are the Socket and ServerSocket classes that can be used for traditional network communications. The HttpServer class allows you to program

a Web server quickly. This is great if you want to add a custom rest API to your application. `dart:io` can be imported and used only on server-side applications, so don't try to use it in your browser apps!

`dart:html` contains all of the classes necessary to interact with a client browser's document object model. This library is required to write any client-side code that runs in a browser. The library defines two static methods: `Element query(String selector)` and `List<Element> queryAll(String selector)`. These methods allow you to grab HTML5 elements from the browser's DOM using cascading-stylesheet selectors. (I show an example of this later.)

`dart:math`, `dart:json` and `dart:crypto` provide helpers that are hard to live without. `dart:math` provides all of the static math methods that programmers have come to expect. `dart:json` provides the JSON helper class. It has only three static methods: `parse(String json)`, which returns a `Map` containing the parsed document; `String stringify(Object object)` and `void printOn(Object object, StringBuffer output)` can be used to serialize an object into JSON. Any object can be made serializable by implementing a `toJson()` method. `dart:crypto` has helpers for performing `md5`, `sha1`

and `sha256` hashing. There also is a `CryptoUtils` class with methods for converting bytes to hex and bytes to base64.

Server-Side Programming

Let's jump into Dart by looking at some server-side programming:

```
import 'dart:io';

void main(){
    String fileName = './test.txt';
    File file = new File(fileName);

    var out = file.openOutputStream();
    out.writeString("Hello, Dart!\n");
    out.close();
}
```

Does it look pretty familiar? It's not much different from a Java program at this point. You start by importing the `dart:io` library. This gives you access to the `File` and `OutputStream` classes. Next, you declare a main method. Just like Java and C, `main` acts as the entry point for all programs. The `File` object is used to hold a reference to a single file on the filesystem. In order to write to this file, you open the file's output stream. This method will create the file if it does not exist or clears the contents of the file if it does. It returns an `OutputStream` object that

then can be used to send data to the file. You write a single string and close the OutputStream.

To run this program, save it to a file called first.dart. Then use the Dart runtime environment provided

Listing 1. wunder.dart

```
import 'dart:io';
import 'dart:uri';
import 'dart:json';

void main(){
  List jsonData = [];
  String apiKey = "";
  String zipcode = "";

  //Read the user supplied data form the options
  //object
  try {
    apiKey = new Options().arguments[0];
    zipcode = new Options().arguments[1];
  } on RangeError {
    print("Please supply an API key and zipcode!");
    print("dart wunder.dart <apiKey> <zipCode>");
    exit(1);
  }

  //Build the URI we are going to request data from
  Uri uri = new Uri("http://api.wunderground.com/"
    "api/${apiKey}/conditions/q/${zipcode}.json");

  HttpClient client = new HttpClient();
  HttpClientConnection connection =
    client.getUrl(uri);
  connection.onResponse =
    (HttpClientResponse response) {

    //Our client has a response, open an input
    //stream to read it
    InputStream stream = response.inputStream;
    stream.onData = () {

      //The input stream has data to read,
      //read it and add it to our list
      jsonData.addAll(stream.read());
    };
  };

  stream.onClosed = () {

    //response and print the location and temp.
    try {
      Map jsonDocument =
        JSON.parse(new String.fromCharCodes(jsonData));
      if (jsonDocument["response"].containsKey("error")){
        throw jsonDocument["response"]["error"]["description"];
      }
      String temp =
        jsonDocument["current_observation"]["temperature_string"];
      String location = jsonDocument["current_observation"]
        ["display_location"]["full"];

      print('The temperature for $location is $temp');
    } catch(e) {
      print("Error: $e");
      exit(2);
    }
  };

  //Register the error handler for the InputStream
  stream.onError = () {
    print("Stream Failure!");
    exit(3);
  };
};

//Register the error handler for the HttpClientConnection
connection.onError = (e){
  print("Http error (check api key)");
  print("$e");
  exit(4);
};
}
```

with the SDK:

```
$ dart first.dart
```

When your program is done, you should see a file called `test.txt` in the same directory. Open it, and you will see your text.

What's interesting about Dart is that all I/O is event-based. This is much in the same style as Node.js. Every time you call a method that performs I/O, it is added to the event queue and the method returns immediately. Almost every single I/O method takes in a callback function or returns a Future object. The reason for this design choice is for scalability. Because Dart runs your code in a single thread, non-blocking asynchronous I/O calls are the only way to allow the software to scale to hundreds or even thousands of users.

In Listing 1, you can see this evented I/O style put to work with the `HttpClientConnection` object returned by the `HttpClient.getUrl(Uri url)` method. This object is working in the background waiting for a response from the HTTP server. In order to know when the response has been received, you must register an `onResponse(HttpClientResponse response)` callback method. I

created an anonymous method to handle this. Also notice that toward the bottom of the program, I register an `onError()` callback as well. Don't worry; all of the callbacks are registered before the HTTP transaction begins.

Once the `onResponse()` callback is executed, I pull the `InputStream` object out of the response to begin reading data. I register the `onData()`, `onClosed()` and `onError()` callbacks to handle the different states the `InputStream` can be in. `onData()` simply reads bytes off the stream and appends them to the `jsonData` list object. `onData()` is guaranteed to be called as long as there is data to read. Once the stream has hit "end of file", `onClosed()` is executed. At this point, I know all of the data from the `HttpRequest` has been transferred and read, so I can use the JSON helper class to parse the response into a Map object and print the final result to the user. This is where the program actually exits from if everything was successful. If there was an error in the `InputStream`, then the `onError()` callback would have been called, and the program would have exited from there.

To run this program, call it with the Dart runtime environment. You will need to register for an API

key from Weather Underground (<http://www.wunderground.com/weather/api>). Don't worry; it's completely free. Once you have your API key, you can check the current temperature for any US zip code:

```
$ dart wunder.dart ec7...93b 10001
```

```
The temperature for New York, NY is 57.2 F (14.0 C)
```

Client-Side Dart

Now that you've seen what Dart can do on the server side, let's take a look at what it really was designed for, the client side. Dart excels at programming large-scale browser applications. Unfortunately, space constraints prevent me from showing a truly large application. Instead, I cover a not-so-large but very cool application using the HTML5 Canvas object. Let's use Dart for finger painting.

In our simple finger-painting application, there will be buttons for each color that is available to users, as well as buttons to increment and decrement the thickness of their strokes. What good is painting a masterpiece if you can't save it and share it with the world? So, let's make a save button that will convert the canvas to a PNG image.

First, let's take a look at the markup for this project. In Listing 3, you can see there is an HTML5 Web page that contains a canvas element

called `draw-surface`. This is where the work of art will be made. Below the canvas are the control buttons that allow users to select colors and stroke width, and the save button. The last part of the document is the most interesting part. There are two script elements. The first is a script tag with the type attribute set to "application/dart". This script type is currently recognized only by a fork of Chromium called Dartium (<http://www.dartlang.org/dartium>). The second is a JavaScript bootstrap file that is required to start the Dart VM in Dartium. It also has a special second function that I talk about later.

Now let's take a look at the application itself. At the top of Listing 2, I start the program by importing `dart:html`. All client-side applications must import this library to have access to the DOM. Next, I create a class called `DrawSurface` that will act as a container class for the canvas object. The constructor takes in a `CanvasElement` and grabs its 2-D-rendering context. It also registers all of the callbacks to handle mouse movements on the draw surface. When the user presses down on the mouse, somewhere on the draw surface canvas I begin a draw path.

As the user moves the mouse around with the button pressed down, I add line segments to the drawing. When the user releases the mouse or

Listing 2. fingerprint.dart

```
library fingerprint;

import 'dart:html';

class DrawSurface {
  String _color = "black";
  int _lineThickness = 1;
  CanvasElement _canvas;
  bool _drawing = false;
  var _context;

  DrawSurface(CanvasElement canvas) {
    _canvas = canvas;
    _context = _canvas.context2d;
    _canvas.on.mouseDown.add((Event e) => _onMouseDown(e));
    _canvas.on.mouseUp.add((Event e) => _onMouseUp(e));
    _canvas.on.mouseMove.add((Event e) => _onMouseMove(e));
    _canvas.on.mouseOut.add((Event e) => _onMouseUp(e));
  }

  set lineThickness(int lineThickness) {
    _lineThickness = lineThickness;
    _context.lineWidth = _lineThickness;
  }

  set color(String color) {
    _color = color;
    _context.fillStyle = _color;
    _context.strokeStyle = _color;
  }

  int get lineThickness => _lineThickness;

  int get color => _color;

  void incrementLineThickness(int amount){
    _lineThickness += amount;
    _context.lineWidth = _lineThickness;
  }

  String getPNGImageUrl() {
    return _canvas.toDataURL('image/png');
  }

  _onMouseDown(Event e){
    _context.beginPath();
    _context.moveTo(e.offsetX, e.offsetY);
    _drawing = true;
  }

  _onMouseUp(Event e){
    _context.closePath();
    _drawing = false;
  }

  _onMouseMove(Event e){
    if (_drawing == true){
      _drawOnCanvas(e.offsetX, e.offsetY);
    }
  }

  _drawOnCanvas(int x, int y){
    _context.lineTo(x, y);
    _context.stroke();
  }
}

void main() {
  CanvasElement canvas = query("#draw-surface");
  DrawSurface ds = new DrawSurface(canvas);

  List<Element> buttons = queryAll("#colors input");
  for (Element e in buttons){
    e.on.click.add((Event eve) {
      ds.color = e.id;
    });
  }

  var sizeDisplay = query("#currentsize");
  sizeDisplay.text = ds.lineThickness.toString();

  query("#sizeup").on.click.add((Event e) {
    ds.incrementLineThickness(1);
    sizeDisplay.text = ds.lineThickness.toString();
  });

  query("#sizedown").on.click.add((Event e) {
    ds.incrementLineThickness(-1);
    sizeDisplay.text = ds.lineThickness.toString();
  });

  query("#save").on.click.add((Event e) {
    String url = ds.getPNGImageUrl();
    window.open(url, "save");
  });
}
```

Listing 3. fingerprint.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Finger Paint</title>
    <link rel="stylesheet" href="fingerprint.css" />
  </head>
  <body>
    <h1>Finger Paint</h1>
    <div>
      <canvas id="draw-surface" width="800px" height="600px">
        </canvas>
      </div>
    <div id="colors">
      <input id="white" type="button"></input>
      <input id="red" type="button"></input>
      <input id="black" type="button"></input>
      <input id="blue" type="button"></input>
      <input id="green" type="button"></input>
      <input id="purple" type="button"></input>
      <input id="yellow" type="button"></input>
      <input id="orange" type="button"></input>
      <input id="brown" type="button"></input>
    </div>
    <div>
      <input id="sizeup" type="button" value="Increase width">
      </input>
      <input id="sizedown" type="button" value="Decrease width">
      </input>
      <span id="currentsize"></span>
    </div>
    <div>
      <input id="save" type="button" value="Save"></input>
    </div>
    <script type="application/dart" src="fingerprint.dart">
    </script>
    <script type="application/javascript"
      src="http://dart.googlecode.com/svn/trunk/dart/client/dart.js">
    </script>
  </body>
</html>
```


Listing 4. fingerprint.css

```
#draw-surface {
    border-style: solid;
    border-width: 2px;
}
#white {
    background-color: white;
    width: 30px;
}
#red {
    background-color: red;
    width: 30px;
}
#black {
    background-color: black;
    width: 30px;
}
#blue {
    background-color: blue;
    width: 30px;
}
#green {
    background-color: green;
    width: 30px;
}
#purple {
    background-color: purple;
    width: 30px;
}
#yellow {
    background-color: yellow;
    width: 30px;
}
#orange {
    background-color: orange;
    width: 30px;
}
#brown {
    background-color: brown;
    width: 30px;
}
```

moves out of the canvas element, I close the drawing path.

I implemented getters and setters for the color and lineThickness attributes. In the setter methods, I make sure to update the rendering context on any change. I also add two methods `incrementLineThickness(int amount)` that will allow the user to adjust the lineThickness by some amount, instead of just setting it, and `getPNGImageUrl()` to expose the canvas element's `toDataURL()` method. This method will allow the save button to function.

In main, I use the static `query(String selector)` function to get the canvas element by its ID. The query function takes any CSS selector and returns an `Element` object that matches it. If there is more than one element that you want to access on a page, you can use the `queryAll(String selector)` function that will return a `List<Element>` object. I use this function to gather up all of the color buttons at once and register `onClick()` events to set the current color to its respective ID value.

Finally, I register the callbacks for the size-up and size-down buttons that change the thickness of the line by 1. I also register the callback for the save button to grab the PNG data URL from the canvas and open it in a new window. The user then can save

the image by right-clicking on it and choosing "save image as".

Running the Application

If you have chosen to download and install the full Dart editor package, you already have Dartium installed. If, like me, you chose to install only the SDK,

you need to grab a copy of Dartium from <http://www.dartlang.org/dartium>. To install it, I just unzipped the file and created a symlink in my \$HOME/bin directory to the chrome program that I extracted:

```
$ ln -s /path/to/unzipped/folder/chrome dartium
```

NOTE: Dartium is an experimental browser, so it should be used only for developing Dart applications locally. Don't use it as your normal browser! There might be security exploits or stability issues that have not been discovered yet.

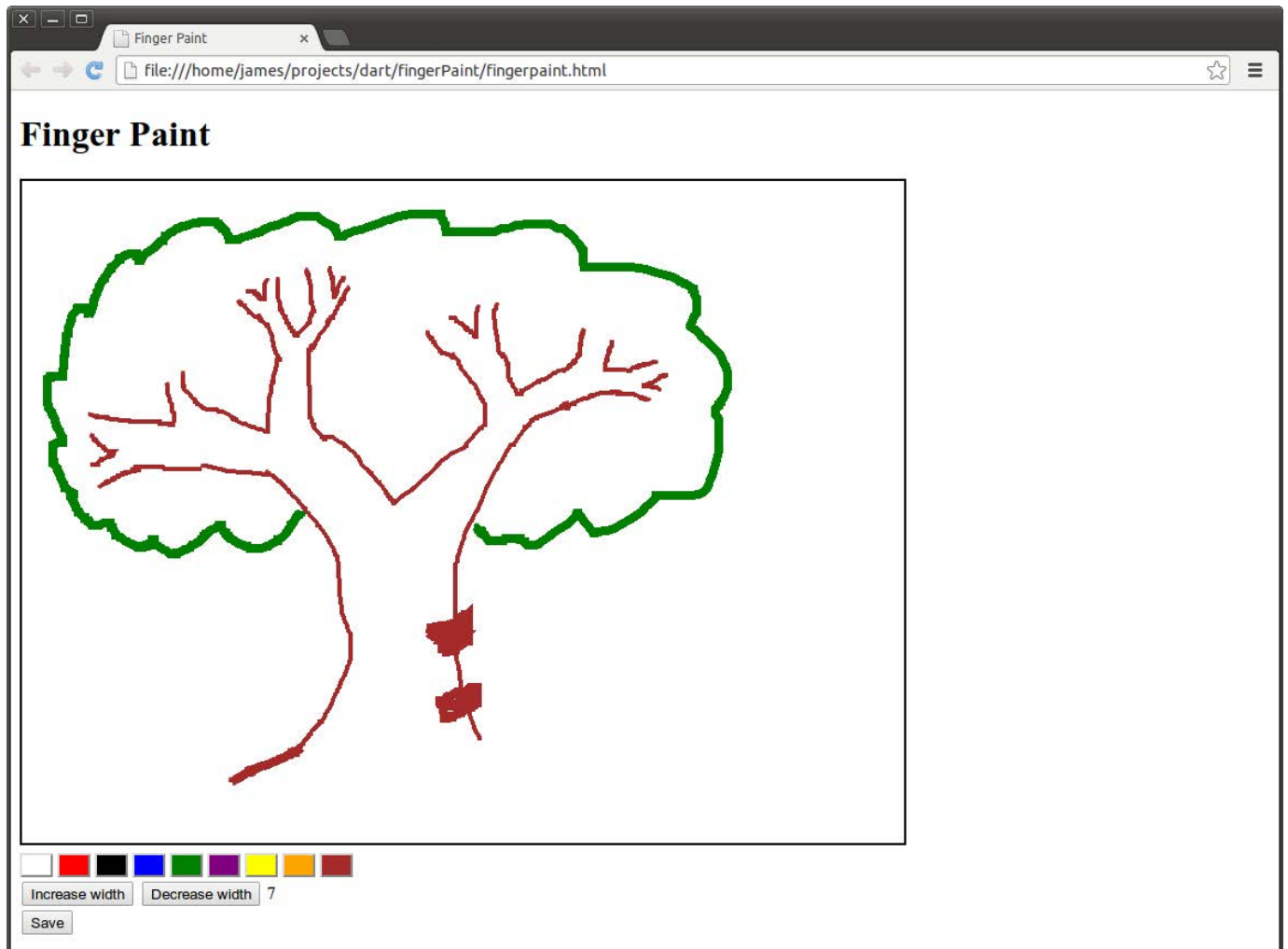


Figure 1. The finger-paint application running on Dartium, a special fork of Chromium.

Once it is installed, you can run this application from the command line with the command:

```
$ dartium fingerprint.html
```

Running on Other Browsers

It's okay if you don't have Dartium.

Remember that bootstrap script line in `fingerprint.html`? Aside from starting the Dart VM in Dartium, it also will fall back to a JavaScript application if Dart is not supported. The JavaScript application must have the same name as the Dart application with the extension `.dart.js`. The Dart SDK comes with a nifty

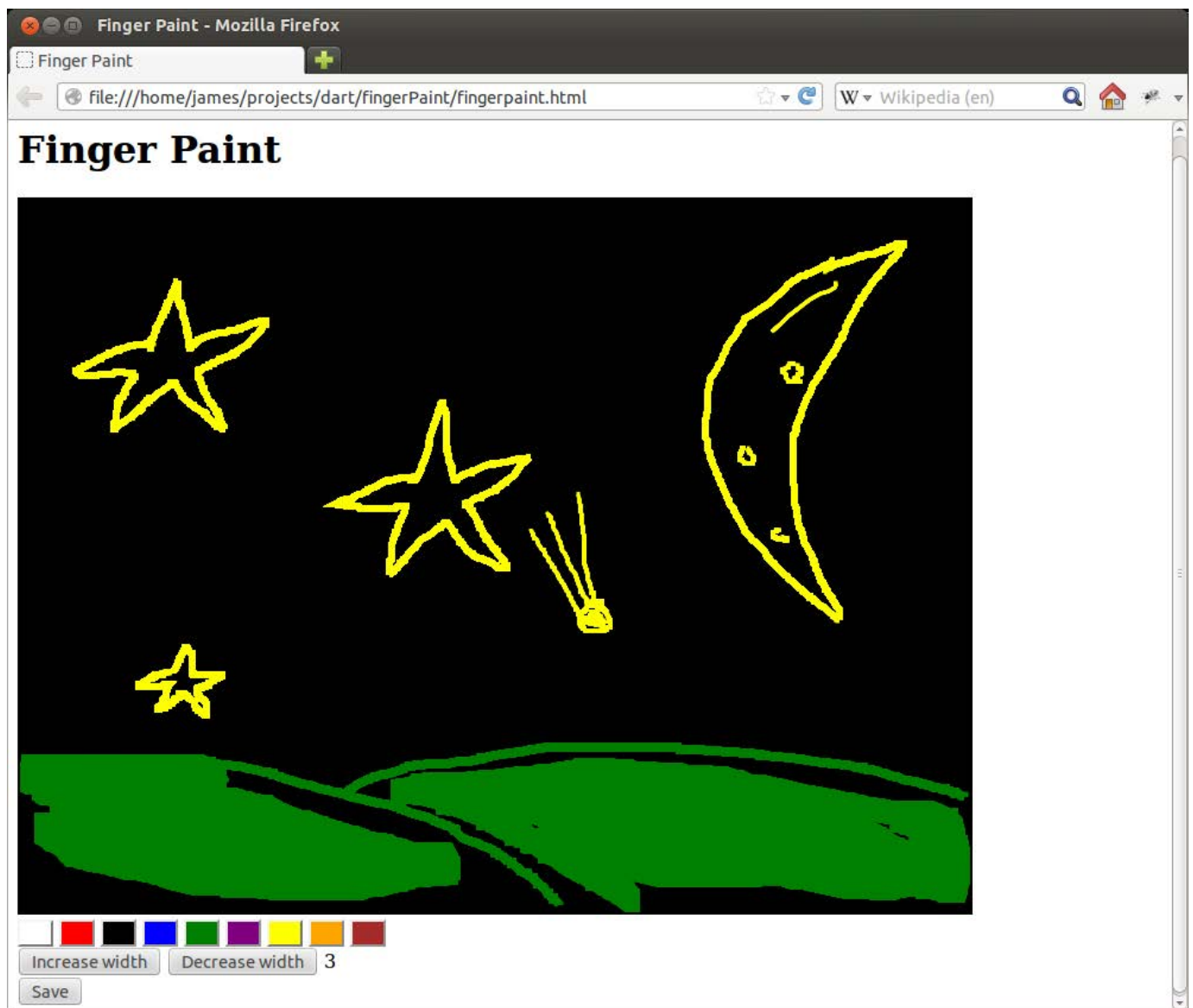


Figure 2. After being converted to JavaScript, the finger-paint application can now run on any browser. Here it is running on Firefox.

program called `dart2js` that will convert a Dart browser application into a JavaScript application for use in any browser. To convert this application, you can run `dart2js` on `fingerpaint.dart`:

```
$ dart2js -ofingerpaint.dart.js fingerpaint.dart
```

When this is done, you will see several new files, including `fingerpaint.dart.js`.

Now the application will work in any browser that can handle JavaScript. I personally recommend using Dartium for application development and then converting the application to JavaScript for release.

State of Dart

I would love to tell you that the community has welcomed Dart with open arms, but that's simply not the case. The people in charge are afraid of Dart becoming the next VBScript and hurting the open Web. So far,

Microsoft, Mozilla and Apple have rejected the idea of embedding a Dart runtime into their browsers. As Dart matures and gains popularity, I hope to see this stance reverse, but for now, `dart2js` is the only way to get Dart projects on-line for all to use.

Conclusion

Dart is a fantastic language that presents an entirely new approach to writing large-scale, client-side, object-oriented applications. I have enjoyed working with it, and I hope you will too. The potential of this language is limitless, and I hope to see widespread adoption of it in the future. ■

James Slocum is a software developer for Telvue Corporation, where he designs and programs digital media decoders and delivery systems. In his spare time, he enjoys learning new programming languages and snowboarding. Feel free to contact him at jms44@njit.edu, and check out his blog at <http://www.jamesslocum.com>.

Resources

Dart Home Page: <http://www.dartlang.org>

Dart API Reference: http://api.dartlang.org/docs/bleeding_edge/index.html

Dartium: <http://www.dartlang.org/dartium>

Dart Source Code and Bug Tracking: <http://code.google.com/p/dart>

HTML5 for Publishers: <http://shop.oreilly.com/product/0636920022473.do>

Big Data gets real at Big Data TechCon!

The **HOW-TO** conference for Big Data and IT Professionals



Discover how to master Big Data from real-world practitioners – instructors who work in the trenches and can teach you from real-world experience!

Come to Big Data TechCon to learn the best ways to:

- Collect, sort and store massive quantities of structured and unstructured data
- Process real-time data pouring into your organization
- Master Big Data tools and technologies like Hadoop, Map/Reduce, NoSQL databases, and more
- Learn HOW TO integrate data-collection technologies with analysis and business-analysis tools to produce the kind of workable information and reports your organization needs
- Understand HOW TO leverage Big Data to help your organization today

Over 50
how-to
practical classes
and workshops
to choose
from!



BigData
TECHCON

April 8-10, 2013

Boston, MA

www.BigDataTechCon.com

Register Early and SAVE!

A BZ Media Event

Big Data TechCon™ is a trademark of BZ Media LLC.

Speed Up Your Web Site with Varnish

Want your Web site to respond faster and scale better? Learn how to use Varnish to make it happen. [PABLO GRAZIANO](#)

Varnish is a program that can greatly speed up a Web site while reducing the load on the Web server. According to Varnish's official site, Varnish is a "Web application accelerator also known as a caching HTTP reverse proxy".

When you think about what a Web server does at a high level, it receives HTTP requests and returns HTTP responses. In a perfect world, the server would return a response immediately without having to do any real work. In the real world, however, the server may have to do quite a bit of work before returning a response to the client. Let's first look at how a typical Web server handles this, and then see what Varnish does to improve the situation.

Although every server is different, a typical Web server will go through a potentially long sequence of steps to service each request it receives. It may start by spawning a new process to handle the request. Then, it may have to load script files from disk, launch an interpreter process to interpret and compile those files into bytecode and then execute that bytecode. Executing the code may result in additional work, such as performing expensive database queries and retrieving more files from disk. Multiply this by hundreds or thousands of requests, and you can see how the server quickly can become overloaded, draining system resources trying to fulfill requests. To make matters worse, many of

the requests are repeats of recent requests, but the server may not have a way to remember the responses, so it's sentenced to repeating the same painful process from the beginning for each request it encounters.

Things are a little different with Varnish in place. For starters, the request is received by Varnish instead of the Web server. Varnish then will look at what's being requested and forward the request to the Web server (known as a back end to Varnish). The back-end server does its regular work and returns a response to Varnish, which in turn gives the response to the client that sent the original request.

If that's all Varnish did, it wouldn't be much help. What gives us the performance gains is that Varnish can store responses from the back end in its cache for future use. Varnish quickly can serve the next response directly from its cache without placing any needless load on the back-end server. The result is that the load on the back end is reduced significantly, response times improve, and more requests can be served per second. One of the things that makes Varnish so fast is that it keeps its cache completely in memory instead of on disk. This and other optimizations allow Varnish to process requests at blinding speeds. However, because memory typically is

more limited than disk, you have to size your Varnish cache properly and take measures not to cache duplicate objects that would waste valuable space.

Let's install Varnish. I'm going to explain how to install it from source, but you can install it using your distribution's package manager. The latest version of Varnish is 3.0.3, and that's the version I work with here. Be aware that the 2.x versions of Varnish have some subtle differences in the configuration syntax that could trip you up. Take a look at the Varnish upgrade page on the Web site for a full list of the changes between versions 2.x and 3.x (<https://www.varnish-cache.org/docs/3.0/installation/upgrade.html>).

Missing dependencies is one of the most common installation problems. Check the Varnish installation page for the full list of build dependencies (<https://www.varnish-cache.org/docs/3.0/installation/install.html#compiling-varnish-from-source>).

Run the following commands as root to download and install the latest version of Varnish:

```
cd /var/tmp
wget http://repo.varnish-cache.org/source/varnish-3.0.3.tar.gz
tar xzf varnish-3.0.3.tar.gz
cd varnish-3.0.3
sh autogen.sh
sh configure
```

```
make
make test
make install
```

Varnish is now installed under the `/usr/local` directory. The full path to the main binary is `/usr/local/sbin/varnishd`, and the default configuration file is `/usr/local/etc/varnish/default.vcl`.

You can start Varnish by running the `varnishd` binary. Before you can do that though, you have to tell Varnish which back-end server it's caching for. Let's specify the back end in the `default.vcl` file. Edit the `default.vcl` file as shown below, substituting the values for those of your Web server:

```
backend default {
    .host = "127.0.0.1";
    .port = "80";
}
```

Now you can start Varnish with this command:

```
/usr/local/sbin/varnishd -f /usr/local/etc/varnish/default.vcl
➔-a :6081 -P /var/run/varnish.pid -s malloc,256m
```

This will run `varnishd` as a `dæmon` and return you to the command prompt. One thing worth pointing out is that `varnishd` will launch two processes. The first is the manager process, and the second is the child worker process. If the child process dies for whatever reason, the manager process will spawn a new process.

If you installed Varnish from your package manager, it may be running already. In that case, you can stop it first, then use the command above to start it manually. Otherwise, the options it was started with may differ from those in this example. A quick way to see if Varnish is running and

Varnishd Startup Options

The `-f` option tells Varnish where your configuration file lives.

The `-a` option is the `address:port` that Varnish will listen on for incoming HTTP requests from clients.

The `-P` option is the path to the PID file, which will make it easier to stop Varnish in a few moments.

The `-s` option configures where the cache is kept. In this case, we're using a 256MB memory-resident cache.

what options it was given is with the `pgrep` command:

```
/usr/bin/pgrep -lf varnish
```

Varnish now will relay any requests it receives to the back end you specified, possibly cache the response,

and deliver the response back to the client. Let's submit some simple GET requests and see what Varnish does. First, run these two commands on separate terminals:

```
/usr/local/bin/varnishlog
/usr/local/bin/varnishstat
```

The screenshot shows a terminal window titled "pablo@tar: ~". The terminal output is as follows:

```
root@falcon:~# GET -Used http://localhost:6081/cgi-bin/test
GET http://localhost:6081/cgi-bin/test
User-Agent: lwp-request/5.834 libwww-perl/5.836

200 OK
Cache-Control: private;
Connection: close
Date: Tue, 04 Dec 2012 02:22:11 GMT
Via: 1.1 varnish
Age: 0
Server: Apache
Vary: Accept-Encoding
Content-Type: text/html
Client-Date: Tue, 04 Dec 2012 02:22:11 GMT
Client-Peer: 127.0.0.1:6081
Client-Response-Num: 1
Client-Transfer-Encoding: chunked
X-Varnish: 910098091

root@falcon:~#
```

The terminal window also shows a status bar at the bottom: "[2] 0:bash* 4:cli- "falcon" 18:22 03-Dec-12".

Figure 1. Varnish Response Headers

The following GET command is part of the Perl www library (libwww-perl). I use it so you can see the response headers you get back from Varnish. If you don't have libwww-perl, you could use Firefox with the Live HTTP Headers extension or another tool of your choice:

```
GET -Used http://localhost:6081/
```

The options given to the GET command aren't important here. The important thing is that the URL points to the port on which varnishd is listening. There are three response headers that were added by Varnish. They are X-Varnish, Via and Age. These headers are useful once you know what they are. The X-Varnish header will be followed by either one or two numbers. The single-number version means the response was not in Varnish's cache (miss), and the number shown is the ID Varnish assigned to the request. If two numbers are shown, it means Varnish found a response in its cache (hit). The first is the ID of the request, and the second is the ID of the request from which the cached response was populated. The Via header just shows that the request went through a proxy. The Age header tells you how long the response has been cached by Varnish,

in seconds. The first response will have an Age of 0, and subsequent hits will have an incrementing Age value. If subsequent responses to the same page don't increment the Age header, that means Varnish is not caching the response.

Now let's look at the varnishstat command launched earlier. You should see something similar to Figure 2.

The important lines are cache_hit and cache_miss. cache_hits won't be shown if you haven't had any hits yet. As more requests come in, the counters are updated to reflect hits and misses.

Next, let's look at the varnishlog command launched earlier (Figure 3).

This shows you fairly verbose details of the requests and responses that have gone through Varnish. The documentation on the Varnish Web site explains the log output as follows:

The first column is an arbitrary number, it defines the request. Lines with the same number are part of the same HTTP transaction. The second column is the tag of the log message. All log entries are tagged with a tag indicating what sort of activity is being logged. Tags starting with Rx indicate Varnish is receiving data and Tx indicates sending data. The third

column tell us whether this is data coming or going to the client (c) or to/from the back end (b). The forth column is the data being logged.

varnishlog has various filtering options to help you find what you're looking for. I recommend

playing around and getting comfortable with varnishlog, because it will really help you debug Varnish. Read the varnishlog(1) man page for all the details. Next are some simple examples of how to filter with varnishlog.

To view communication between

```

pablo@tar: ~
File Edit View Terminal Help
0+02:16:24
Hitrate ratio:      1      1      1
Hitrate avg:      1.0000  1.0000  1.0000

 3212021      0.00      392.48 client_conn - Client connections accepted
 3212101      0.00      392.49 client_req - Client requests received
 3212094      0.00      392.48 cache_hit - Cache hits
    7      0.00      0.00 cache_miss - Cache misses
    7      0.00      0.00 backend_conn - Backend conn. success
    6      0.00      0.00 backend_toolate - Backend conn. was closed
    7      0.00      0.00 backend_recycle - Backend conn. recycles
    7      0.00      0.00 fetch_length - Fetch with Length
   57      .      . n_sess_mem - N struct sess_mem
    9      .      . n_objectcore - N struct objectcore
   10      .      . n_objecthead - N struct objecthead
   27      .      . n_waitinglist - N struct waitinglist
    1      .      . n_vbc - N struct vbc
   10      .      . n_wrk - N worker threads
   23      0.00      0.00 n_wrk_create - N worker threads created
  336      0.00      0.04 n_wrk_queued - N queued work requests
    1      .      . n_backend - N backends
    7      .      . n_expired - N expired objects
   135      .      . n_lru_moved - N LRU moved objects
3191145      0.00      389.92 n_objwrite - Objects sent with write
3212199      0.00      392.50 s_sess - Total Sessions
3212101      0.00      392.49 s_req - Total Requests
    7      0.00      0.00 s_fetch - Total fetch
716129277      0.00      87503.58 s_hdrbytes - Total header bytes
167045352      0.00      20411.21 s_bodybytes - Total body bytes
 3212199      0.00      392.50 sess_closed - Session Closed
115641582      0.00      14130.20 shm_records - SHM records
12854289      0.00      1570.66 shm_writes - SHM writes
 752150      0.00      91.90 shm_cont - SHM MTX contention
    34      0.00      0.00 shm_cycles - SHM cycles through buffer
    7      0.00      0.00 backend_req - Backend requests made
    1      0.00      0.00 n_vcl - N vcl total
    1      0.00      0.00 n_vcl_avail - N vcl available
    1      .      . n_ban - N total active bans
    1      .      . n_ban_gone - N total gone bans

[2] 0:varnishstat* 4:cli- "falcon" 18:25 03-Dec-12

```

Figure 2.
varnishstat
Command

```

pablo@tar: ~
File Edit View Terminal Help
13 RxResponse b OK
13 RxHeader b Date: Tue, 04 Dec 2012 02:28:56 GMT
13 RxHeader b Server: Apache
13 RxHeader b Cache-Control: private;
13 RxHeader b Vary: Accept-Encoding
13 RxHeader b Content-Encoding: gzip
13 RxHeader b Content-Length: 63
13 RxHeader b Content-Type: text/html
13 Fetch_Body b 4(length) cls 0 mklen 1
13 Length b 63
13 BackendReuse b default
11 SessionOpen c 127.0.0.1 53321 :6081
11 ReqStart c 127.0.0.1 53321 1863511864
11 RxRequest c GET
11 RxURL c /cgi-bin/test
11 RxProtocol c HTTP/1.1
11 RxHeader c TE: deflate,gzip;q=0.3
11 RxHeader c Connection: TE, close
11 RxHeader c Host: localhost:6081
11 RxHeader c User-Agent: lwp-request/5.834 libwww-perl/5.836
11 VCL_call c recv lookup
11 VCL_call c hash
11 Hash c /cgi-bin/test
11 Hash c localhost:6081
11 VCL_return c hash
11 VCL_call c miss fetch
11 Backend c 13 default default
11 TTL c 1863511864 RFC 120 -1 -1 1354588137 0 1354588136 0 0
11 VCL_call c fetch deliver
11 ObjProtocol c HTTP/1.1
11 ObjResponse c OK
11 ObjHeader c Date: Tue, 04 Dec 2012 02:28:56 GMT
11 ObjHeader c Server: Apache
11 ObjHeader c Cache-Control: private;
11 ObjHeader c Vary: Accept-Encoding
11 ObjHeader c Content-Encoding: gzip
11 ObjHeader c Content-Type: text/html
11 Gzip c u F - 63 52 80 80 440
11 VCL_call c deliver deliver
11 TxProtocol c HTTP/1.1
11 TxStatus c 200
11 TxResponse c OK
11 TxHeader c Server: Apache
11 TxHeader c Cache-Control: private;
11 TxHeader c Vary: Accept-Encoding
11 TxHeader c Content-Type: text/html
11 TxHeader c Transfer-Encoding: chunked
11 TxHeader c Date: Tue, 04 Dec 2012 02:28:56 GMT
11 TxHeader c X-Varnish: 1863511864
11 TxHeader c Age: 0
11 TxHeader c Via: 1.1 varnish
11 TxHeader c Connection: close
11 Gzip c U D - 63 52 80 80 440
11 Length c 52
11 ReqEnd c 1863511864 1354588136.742137194 1354588136.744198799 0.000128508 0.002001524
0.000060081
11 SessionClose c Connection: close
11 StatSess c 127.0.0.1 53321 0 1 1 0 0 1 241 52
[2] 0:bash- 1:varnishlog* 4:cli "falcon" 18:28 03-Dec-12

```

Figure 3. varnishlog Command

Varnish and the client (omitting the back end):

```
/usr/local/bin/varnishlog -c
```

To view communication between Varnish and the back end (omitting the client):

```
/usr/local/bin/varnishlog -b
```

To view the headers received by Varnish (both the client's request headers and the back end's response headers):

```
/usr/local/bin/varnishlog -i RxHeader
```

Same thing, but limited to just the client's request headers:

```
/usr/local/bin/varnishlog -c -i RxHeader
```

Same thing, but limited to just the back end's response headers:

```
/usr/local/bin/varnishlog -b -i RxHeader
```

To write all log messages to the `/var/log/varnish.log` file and daemonize:

```
/usr/local/bin/varnishlog -Dw /var/log/varnish.log
```

To read and display all log messages from the `/var/log/varnish.log` file:

```
/usr/local/bin/varnishlog -r /var/log/varnish.log
```

The last two examples demonstrate storing your Varnish log to disk. Varnish keeps a circular log in memory in order to stay fast, but that means old log entries are lost unless saved to disk. The last two examples above demonstrate how to save all log messages to a file for later review.

If you wanted to stop Varnish, you could do so with this command:

```
kill `cat /var/run/varnish.pid`
```

This will send the TERM signal to the process whose PID is stored in the `/var/run/varnish.pid` file. Because this is the `varnishd` manager process, Varnish will shut down.

Now that you know how to start and stop Varnish, and examine cache hits and misses, the natural question to ask is what does Varnish cache, and for how long?

Varnish is conservative with what it will cache by default, but you can change most of these defaults. It will consider only caching GET and HEAD requests. It won't cache a request with either a Cookie or Authorization header. It won't cache a response with either a Set-Cookie or Vary header. One thing Varnish looks at is the Cache-Control header. This header

is optional, and it may be present in the Request or the Response. It may contain a list of one or more semicolon-separated directives. This header is meant to apply caching restrictions. However, Varnish won't alter its caching behavior based on the Cache-Control header, with the exception of the max-age directive. This directive looks like `Cache-Control: max-age=n`, where `n` is a number. If Varnish receives the max-age directive in the back end's response, it will use that value to set the cached response's expiration (TTL), in seconds. Otherwise, Varnish will set the cached response's TTL expiration to the value of its `default_ttl` parameter, which defaults to 120 seconds.

You'll likely want to change what Varnish caches and how long it's cached for—this is called your caching

policy. You express your caching policy in the `default.vcl` file by writing VCL. VCL stands for Varnish Configuration Language, which is like a very simple scripting language specific to Varnish. VCL is fully explained in the `vcl(7)` man page, and I recommend reading it.

Before changing `default.vcl`, let's think about the process Varnish goes through to fulfill an HTTP request. I call this the request/response cycle, and it all starts when Varnish receives a request. Varnish will parse the HTTP request and store the details in an object known to Varnish simply as `req`. Now Varnish has a decision to make based entirely on the `req` object—should it check its cache for a match or just forward the request to the back end without caching the response? If it decides to bypass its cache, the only thing left to do is forward the request to the back end

NOTE: Varnish has configuration parameters with sensible defaults. For example, the `default_ttl` parameter defaults to 120 seconds. Configuration parameters are fully explained in the `varnishd(1)` man page. You may want to change some of the default parameter values. One way to do that is to launch `varnishd` by using the `-p` option. This has the downside of having to stop and restart Varnish, which will flush the cache. A better way of changing parameters is by using what Varnish calls the management interface. The management interface is available only if `varnishd` was started with the `-T` option. It specifies on what port the management interface should listen. You can connect to the management interface with the `varnishadm` command. Once connected, you can query parameters and change their values without having to restart Varnish.

To learn more, read the man pages for `varnishd`, `varnishadm` and `varnish-cli`.

and then forward the response back to the client. However, if it decides to check its cache, things get more interesting. This is called a cache lookup, and the result will either be a hit or a miss. A hit means that Varnish has a response in its cache for the client. A miss means that Varnish doesn't have a cached response to send, so the only logical thing to do is send the request to the back end and then cache the response it gives before sending it back to the client.

Now that you have an idea of Varnish's request/response cycle, let's talk about how to implement your caching policy by changing the decisions Varnish makes in the process. Varnish has a set of subroutines that carry out the process described above. Each of these subroutines performs a different part of the process, and the return value from the subroutine is how you tell Varnish what to do next. In addition to setting the return values, you can inspect and make changes to various objects within the subroutines. These objects represent things like the request and the response. Each subroutine has a default behavior that can be seen in `default.vcl`. You can redefine these subroutines to get Varnish to behave how you want.

The first subroutine to look at is

called `vcl_recv`. This gets executed after receiving the full client request, which is available in the `req` object. Here you can inspect and make changes to the original request via the `req` object. You can use the value of `req` to decide how to proceed. The return value is how you tell Varnish what to do. I'll put the return values in parentheses as they are explained. Here you can tell Varnish to bypass the cache and send the back end's response back to the client (`pass`). You also can tell Varnish to check its cache for a match (`lookup`).

Next is the `vcl_pass` subroutine. If you returned `pass` in `vcl_recv`, this is

Varnish Subroutines

The Varnish subroutines have default definitions, which are shown in `default.vcl`. Just because you redefine one of these subroutines doesn't mean the default definition will not execute. In particular, if you redefine one of the subroutines but don't return a value, Varnish will proceed to execute the default subroutine. All the default Varnish subroutines return a value, so it makes sense that Varnish uses them as a fallback.

where you'll be just before sending the request to the back end. You can tell Varnish to continue as planned (pass) or to restart the cycle at the `vcl_recv` subroutine (restart).

The `vcl_miss` and `vcl_hit` subroutines are executed depending on whether Varnish found a suitable response in the cache. From `vcl_miss`, your main options are to get a response from the back-end server and cache it (fetch) or to get a response from the back end and not cache it (pass). `vcl_hit` is where you'll be if Varnish successfully finds a matching response in its cache. From `vcl_hit`, you have the cached response available to you in the `obj` object. You can tell Varnish to send the cached response to the client (deliver) or have Varnish ignore the cached response and return a fresh response from the back end (pass).

The `vcl_fetch` subroutine is where you'll be after getting a fresh response from the back end. The response will be available to you in the `beresp` object. You either can tell Varnish to

continue as planned (deliver) or to start over (restart).

From `vcl_deliver`, you can finish the request/response cycle by delivering the response to the client and possibly caching it as well (deliver), or you can start over (restart).

As previously stated, you express your caching policy within the subroutines in `default.vcl`. The return values tell Varnish what to do next. You can base your return values on many things, including the values held in the request (`req`) and response (`resp`) objects mentioned earlier. In addition to `req` and `resp`, there also is a client object representing the client, a server object and a `beresp` object representing the back end's response. It's important to realize that not all objects are available in all subroutines. It's also important to return one of the allowed return values from subroutines. One of the hardest things to remember when starting out with Varnish is which objects are available in which

Table 1. This table shows which objects are available in each of the subroutines.

	client	server	req	bereq	beresp	resp	obj
vcl_recv	X	X	X				
vcl_pass	X	X	X	X			
vcl_miss	X	X	X	X			
vcl_hit	X	X	X				X
vcl_fetch	X	X	X	X	X		
vcl_deliver	X	X	X			X	

subroutines, and what the legal return values are. To make it easier, I've created a couple reference tables. They will help you get up to speed quickly by not having to memorize everything up front or dig through the documentation every time you make a change.

TIP: Be sure to read the full explanation of VCL, available subroutines, return values and objects in the `vcl(7)` man page.

Let's put it all together by looking at some examples.

Normalizing the request's Host header:

```
sub vcl_recv {
    if (req.http.host ~ "^www.example.com") {
        set req.http.host = "example.com";
    }
}
```

Notice you access the request's host header by using `req.http.host`. You

have full access to all of the request's headers by putting the header name after `req.http`. The `~` operator is the match operator. That is followed by a regular expression. If you match, you then use the `set` keyword and the assignment operator (`=`) to normalize the hostname to simply "example.com". A really good reason to normalize the hostname is to keep Varnish from caching duplicate responses. Varnish looks at the hostname and the URL to determine if there's a match, so the hostnames should be normalized if possible.

Here's a snippet from the default `vcl_recv` subroutine:

```
sub vcl_recv {
    if (req.request != "GET" && req.request != "HEAD") {
        return (pass);
    }
    return (lookup);
}
```

That's a snippet of the default

Table 2. This table shows valid return values for each of the subroutines.

	pass	lookup	error	restart	deliver	fetch	pipe	hit_for_pass
vcl_recv	X	X	X				X	
vcl_pass	X		X	X				
vcl_lookup								
vcl_miss	X		X			X		
vcl_hit	X		X	X	X			
vcl_fetch			X	X	X			X
vcl_deliver			X	X	X			

vcl_recv subroutine. You can see that if it's not a GET or HEAD request, Varnish returns pass and won't cache the response. If it is a GET or HEAD request, it looks it up in the cache.

Removing request's Cookies if the URL matches:

```
sub vcl_recv {
    if (req.url ~ "^/images") {
        unset req.http.cookie;
    }
}
```

That's an example from the Varnish Web site. It removes cookies from the request if the URL starts with "/images". This makes sense when you recall that Varnish won't cache a request with a cookie. By removing the cookie, you allow Varnish to cache the response.

Removing response cookies for image files:

```
sub vcl_fetch {
    if (req.url ~ "\.(png|gif|jpg)$") {
        unset beresp.http.set-cookie;
        set beresp.ttl = 1h;
    }
}
```

That's another example from Varnish's Web site. Here you're in the vcl_fetch subroutine, which happens

after fetching a fresh response from the back end. Recall that the response is held in the beresp object. Notice that here you're accessing both the request (req) and the response (beresp). If the request is for an image, you remove the Set-Cookie header set by the server and override the cached response's TTL to one hour. Again, you do this because Varnish won't cache responses with the Set-Cookie header.

Now, let's say you want to add a header to the response called X-Hit. The value should be 1 for a cache hit and 0 for a miss. The easiest way to detect a hit is from within the vcl_hit subroutine. Recall that vcl_hit will be executed only when a cache hit occurs. Ideally, you'd set the response header from within vcl_hit, but looking at Table 1 in this article, you see that neither of the response objects (beresp and resp) are available within vcl_hit. One way around this is to set a temporary header in the request, then later set the response header. Let's take a look at how to solve this.

Adding an X-Hit response header:

```
sub vcl_hit {
    set req.http.tempheader = "1";
}

sub vcl_miss {
```

```

    set req.http.tempheader = "0";
}

sub vcl_deliver {
    set resp.http.X-Hit = "0";
    if (req.http.tempheader) {
        set resp.http.X-Hit = req.http.tempheader;
        unset req.http.tempheader;
    }
}

```

The code in `vcl_hit` and `vcl_miss` is straightforward—set a value in a temporary request header to indicate a cache hit or miss. The interesting bit is in `vcl_deliver`. First, I set a default value for X-Hit to 0, indicating a miss. Next, I detect whether the request's `tempheader` was set, and if so, set the response's X-Hit header to match the temporary header set earlier. I then delete the `tempheader` to keep things tidy, and I'm all done. The reason I chose the `vcl_deliver` subroutine is because the response object that will be sent back to the client (`resp`) is available only within `vcl_deliver`.

Let's explore a similar solution that doesn't work as expected.

Adding an X-Hit response header—the wrong way:

```

sub vcl_hit {
    set req.http.tempheader = "1";
}

```

```

sub vcl_miss {
    set req.http.tempheader = "0";
}

sub vcl_fetch {
    set beresp.http.X-Hit = "0";
    if (req.http.tempheader) {
        set beresp.http.X-Hit = req.http.tempheader;
        unset req.http.tempheader;
    }
}

```

Notice that within `vcl_fetch`, I'm now altering the back end's response (`beresp`), not the final response sent to the client. This code appears to work as expected, but it has a major bug. What happens is that the first request is a miss and fetched from the back end, and that response has X-Hit set to "0" then it's cached. Subsequent requests result in a cache hit and never enter the `vcl_fetch` subroutine. The result is that all cache hits continue having X-Hit set to "0". These are the types of mistakes to look out for when working with Varnish.

The easiest way to avoid these mistakes is to keep those reference tables handy; remember when each subroutine is executed in Varnish's workflow, and always test the results.

Let's look at a simple way to tell Varnish to cache everything for

one hour. This is shown only as an example and isn't recommended for a real server.

Cache all responses for one hour:

```
sub vcl_recv {
    return (lookup);
}

sub vcl_fetch {
    set beresp.ttl = 1h;
    return (deliver);
}
```

Here, I'm overriding two default subroutines with my own. If I hadn't returned "deliver" from `vcl_fetch`, Varnish still would have executed its default `vcl_fetch` subroutine looking for a return value, and this would not have worked as expected.

Once you get Varnish to implement your caching policy, you should run some benchmarks to see if there is any improvement. The benchmarking tool I use here is the Apache benchmark tool, known as `ab`. You can install this tool as part of the Apache Web server or as a separate package—depending on your system's package manager. You can read about the various options available to `ab` in either the man page or at the Apache Web site (<http://httpd.apache.org/docs/2.4/programs/ab.html>).

In the benchmark examples below, I have a stock Apache 2.2 installation listening on port 80, and Varnish listening on port 6081. The page I'm testing is a very basic Perl CGI script I wrote that just outputs a one-liner HTML page. It's important to benchmark the same URL against both the Web server and Varnish so you can make a direct comparison. I run the benchmark from the same machine that Apache and Varnish are running on in order to eliminate the network as a factor. The `ab` options I use are fairly straightforward. Feel free to experiment with different `ab` options and see what happens.

Let's start with 1000 total requests (-n 1000) and a concurrency of 1 (-c 1).

Benchmarking Apache with `ab`:

```
ab -c 1 -n 1000 http://localhost/cgi-bin/test
```

Benchmarking Varnish with `ab`:

```
ab -c 1 -n 1000 http://localhost:6081/cgi-bin/test
```

As you can see, the `ab` command provides a lot of useful output. The metrics I'm looking at here are "Time per request" and "Requests per second" (rps). You can see that Apache came in at just over 1ms per request (780 rps), while Varnish came

```

root@localhost # ab -c 1 -n 1000 http://localhost/cgi-bin/test
This is ApacheBench, Version 2.3 <Revision: 655654 >
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache
Server Hostname:      localhost
Server Port:          80

Document Path:        /cgi-bin/test
Document Length:      52 bytes

Concurrency Level:    1
Time taken for tests:  1.281 seconds
Complete requests:    1000
Failed requests:      0
Write errors:         0
Total transferred:    216060 bytes
HTML transferred:    52000 bytes
Requests per second:  780.90 [#/sec] (mean)
Time per request:     1.281 [ms] (mean)
Time per request:     1.281 [ms] (mean, across all concurrent requests)
Transfer rate:        164.77 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median  max
Connect:     0    0  0.0    0    0
Processing:  1    1  0.0    1    2
Waiting:     1    1  0.0    1    2
Total:       1    1  0.0    1    2

Percentage of the requests served within a certain time (ms)
 50%    1
 66%    1
 75%    1
 80%    1
 90%    1
 95%    1
 98%    1
 99%    1
100%    2 (longest request)
root@localhost #

```

Figure 4. Output from ab Command (Apache)

in at 0.1ms (7336 rps)—nearly ten times faster than Apache. This shows that Varnish is faster, at least based on the current setup and isolated testing. It's a good idea to run ab

with various options to get a feel for performance—particularly by changing the concurrency values and seeing what impact that has on your system.

The goal is to not only improve

```

pablo@tar: ~
File Edit View Terminal Help
root@localhost # ab -c 1 -n 1000 http://localhost:6081/cgi-bin/test
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache
Server Hostname:     localhost
Server Port:         6081

Document Path:       /cgi-bin/test
Document Length:     52 bytes

Concurrency Level:   1
Time taken for tests: 0.136 seconds
Complete requests:   1000
Failed requests:     0
Write errors:        0
Total transferred:   273990 bytes
HTML transferred:    52000 bytes
Requests per second: 7336.22 [#/sec] (mean)
Time per request:    0.136 [ms] (mean)
Time per request:    0.136 [ms] (mean, across all concurrent requests)
Transfer rate:       1962.94 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median  max
Connect:     0    0   0.0      0    0
Processing:  0    0   0.1      0    2
Waiting:     0    0   0.1      0    2
Total:       0    0   0.1      0    2

Percentage of the requests served within a certain time (ms)
 50%    0
 66%    0
 75%    0
 80%    0
 90%    0
 95%    0
 98%    0
 99%    0
100%    2 (longest request)
root@localhost #

```

Figure 5. Output from ab Command (Varnish)

response times, but also to do so with as little impact on system resources as possible. Let's compare how a prolonged traffic surge affects system resources. Two good measures of

system performance are the load average and the %iowait. The load average can be seen with the top utility, and the %iowait can be seen with the iostat command. You're

System Load and %iowait

System load is a measure of how much load is being placed on your CPU(s). As a general rule, you want the number to stay below 1.0 per CPU or core on your system. That means if you have a four-core system as in the machine I'm benchmarking here, you want your system's load to stay below 4.0.

%iowait is a measure of the percentage of CPU time spent waiting on input/output. A high %iowait indicates your system is disk-bound, performing many disk i/o operations causing the system to slow down. For example, if your server had to retrieve 100 files or more for each request, it likely would cause the %iowait time to go up very high indicating that the disk is a bottleneck.

going to want to keep an eye on both top and iostat during the prolonged load test to see how the numbers change. Let's fire up top and iostat, each on separate terminals.

Starting iostat with a two-second update interval:

```
iostat -c 2
```

Starting top:

```
/usr/bin/top
```

Now you're ready to run the benchmark. You want ab to run long enough to see the impact on system performance. This typically means anywhere from one minute to ten minutes. Let's re-run ab with

a lot more total requests and a higher concurrency.

Load testing Apache with ab:

```
ab -c 50 -n 100000 http://localhost/cgi-bin/test
```

Load testing Varnish with ab:

```
ab -c 50 -n 100000 http://localhost:6081/cgi-bin/test
```

First let's compare response times. Although you can't see it in the screenshots, which were taken just before ab finished, Apache came in at 23ms per request (2097 rps), and Varnish clocked in at 4ms per request (12099 rps). The most drastic difference can be seen in the load averages in top. While Apache brought the system load all the way

```

pablo@tar: ~
File Edit View Terminal Help

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           37.16    0.00   61.85    0.00    0.00    1.00

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           37.09    0.00   62.66    0.00    0.00    0.25

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           38.65    0.00   60.60    0.00    0.00    0.75

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           38.00    0.00   61.50    0.00    0.00    0.50

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           36.09    0.00   62.91    0.00    0.00    1.00

top - 16:51:21 up 154 days, 9:09, 1 user, load average: 12.38, 4.33, 1.55
Tasks: 264 total, 34 running, 222 sleeping, 0 stopped, 8 zombie
Cpu(s): 37.6%us, 58.7%sy, 0.0%ni, 0.7%id, 0.0%wa, 0.0%hi, 3.0%si, 0.0%st
Mem: 3103920k total, 2175604k used, 928316k free, 553236k buffers
Swap: 12109808k total, 45240k used, 12064568k free, 1246636k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
  9349 root        20   0   8228 4952 1512  R   21   0.2   0:08.82  ab
  8592 www-data    20   0  21040 4576 1444  S    2   0.1   0:00.52  apache2
 14999 www-data    20   0  21040 4580 1444  S    2   0.1   0:01.51  apache2
 17186 www-data    20   0  21040 4580 1444  S    2   0.1   0:01.48  apache2
 3473  www-data    20   0  21584 5164 1604  S    2   0.2   0:01.98  apache2
 3536  www-data    20   0  21040 4576 1444  S    2   0.1   0:00.26  apache2
 8593  www-data    20   0  21040 4576 1444  S    2   0.1   0:00.50  apache2
 9276  www-data    20   0  21040 4576 1444  S    2   0.1   0:01.07  apache2
11376 www-data    20   0  21040 4636 1496  S    2   0.1   0:00.84  apache2
11377 www-data    20   0  21040 4580 1448  S    2   0.1   0:01.05  apache2

root@falco:~# ab -c 50 -n 100000 http://localhost/cgi-bin/test
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 10000 requests
Completed 20000 requests
Completed 30000 requests
Completed 40000 requests
Completed 50000 requests
Completed 60000 requests
Completed 70000 requests
Completed 80000 requests
Completed 90000 requests

```

Figure 6. System Load Impact of Traffic Surge on Apache

up to 12, Varnish kept the system load near 0 at 0.4. I did have to wait several minutes for the machine's load averages to go back down after the Apache load test before load testing

Varnish. It's also best to run these tests on a non-production system that is mostly idle.

Although everyone's servers and Web sites have different


```

pablo@tar: ~
File Edit View Terminal Help

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           17.37   0.00  32.51   0.00    0.00   50.12

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           19.00   0.00  26.00   0.00    0.00   55.00

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           19.40   0.00  28.86   0.00    0.00   51.74

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           17.12   0.00  30.77   0.00    0.00   52.11

avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           18.36   0.00  26.30   0.00    0.00   55.33

top - 17:27:17 up 154 days, 9:45,  1 user,  load average: 0.44, 0.15, 0.20
Tasks: 177 total,  2 running, 175 sleeping,  0 stopped,  0 zombie
Cpu(s): 18.6%us, 21.1%sy,  0.0%ni, 52.1%id,  0.0%wa,  0.0%hi,  8.3%si,  0.0%st
Mem:   3103920k total, 2190000k used,  913920k free,  556004k buffers
Swap: 12109808k total,  45240k used, 12064568k free, 1247940k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 1413 nobody    20   0 160m  83m  81m  S   123   2.7   5:05.97  varnishd
13665 root       20   0 36352  30m 1516  R    72   1.0   0:52.94  ab
 1221 root       20   0   0    0    0  S    1   0.0   4:06.01  flush-253:2
 7173 root       20   0 3204  744  656  S    0   0.0   0:02.67  iostat
    1 root       20   0 2024  244  196  S    0   0.0   1:08.30  init
    2 root       20   0   0    0    0  S    0   0.0   0:01.38  kthreadd
    3 root       RT   0   0    0    0  S    0   0.0   0:01.08  migration/0
    4 root       20   0   0    0    0  S    0   0.0 182:35.58  ksoftirqd/0
    5 root       RT   0   0    0    0  S    0   0.0   0:00.00  watchdog/0
    6 root       RT   0   0    0    0  S    0   0.0   0:00.21  migration/1

root@falcon:~# ab -c 50 -n 1000000 http://localhost:6081/cgi-bin/test
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100000 requests
Completed 200000 requests
Completed 300000 requests
Completed 400000 requests
Completed 500000 requests
Completed 600000 requests
Completed 700000 requests
Completed 800000 requests
Completed 900000 requests
█

[2] 4:cli*                                     "falcon" 17:27 03-Dec-12

```

Figure 7. System Load Impact of Traffic Surge on Varnish

requirements and configurations, Varnish may be able to improve your site's performance drastically while simultaneously reducing the load on the server. ■

Pablo Graziano has owned and operated a Linux consulting company in the Seattle area for more than ten years. He's also an avid Perl programmer and dabbles in Java.



DOC SEARLS

Android for Independence

Building the next personal revolution on the dominant platform we already have.

At some point in the early 2000s, I got my wife a Nokia phone with a keyboard, so we could text each other. It was a great little phone, not hard to use or understand, but she texted me only once with it, to send the word “no”. Then, in late 2007, not long after the iPhone came out, she told me she wanted one. Why? “Because I can work with it.” So we got her one, and she worked it like a chef with a collection of Wüsthofs. A few months later, I got an iPhone 3G. She immediately schooled me on how to use the thing, and she still knows more about texting and other essential iPhone apps than I do. After the iPhone 4 came out in mid-2010, we traded up to those, and that’s where we’ll stay until our

AT&T contract runs out. The plan after that is to replace them with Androids.

We’ve also been using Androids all along. We got a Nexus One when it came out in 2010, and we’ve gone through a series of HTCs since then. Our teenage son’s phone is an HTC Detail, which is an unlocked version of the HTC Evo Shift 4G. In our now ample experience, the HTC Detail is crap. In fact, every Android phone we’ve had has failed. But none of those failings has cost us faith in the platform. Lately, that faith has been invested in Samsung’s line of smartphones and pads, especially the Note, after envying one over dinner one evening with friends. My wife took one look at it and said, “I can work with that.” This is an ominous sign for Apple.

So is “Samsung vs. Apple: Losing My

Look back up the road of history, and it's clear that Apple's foot has been off the gas ever since Steve Jobs died.

Religion" (<http://www.mediapost.com/publications/article/188716/samsung-vs-apple-losing-my-religion.html#axzz2Gwynhd3W>), a column by Barbara Lippert in *MediaPost* in December 2012. Barbara, who says she belongs to the "cult of Cupertino", heaps praise on Samsung for inspired marketing work at a time when Apple's marketing seems to be going through the motions. But her most damning remark is not about Apple's marketing, but about its flagship device, the iPhone 5. She called it "a bit of a 'meh'".

Let's face it. The iPhone 5 is a stretched iPhone 4s, which is an iPhone 4 with sprinkles. The iPhone 4 came out in mid-2010, so the whole line is pretty old in mobile phone years. There is nothing "gotta have" about the iPhone 5, and there's at least one big deal-killer: Apple's Maps app, which replaces the Google-based one that had come standard with all iPhones and iPads up until iOS 6, which is what the iPhone 5 runs. The Maps app was a huge fail

when it came out in September 2012, and it's not noticeably better (to me, at least) in January 2013, when the app still showed no subways in New York, Boston, Paris or London. Its traffic reporting is limited to a few tiny red dotted lines that are easy to miss and appear only on major roads. Google finally came to Apple's rescue in December with an outstanding Maps app for iPhone (and, passive-aggressively, not for iPad). But, with monarchical self-regard, Apple continued bringing up its broken Maps app for searches off linked addresses in Mail, Calendar and Contact. Just today, the Apple Maps app failed to find my eye doctor's address on a main street in Santa Barbara. So I copied the address from the calendar and pasted it into the Google Maps app, which gave me three alternate ways of getting there, all with degrees of traffic listed.

Look back up the road of history, and it's clear that Apple's foot has been off the gas ever since Steve Jobs died. Nearly every new product since then has been an incremental

advance of an old one. The only notable exception was the iPad Mini, which should have come out a year or two earlier and still lacks the retina screen featured on the current-generation iPad. At this stage, Apple has two advantages that no other tech-gear maker can match: retail stores and customer service. But those advantages are limited mostly to first-world cities and suburbs. Elsewhere, Apple's mobile devices are bling for the rich. And even for those people, Apple's advantages will matter less and less as the company's mobile products fall farther and farther behind the many curves being paved by competing devices out in the open marketplace. This is too bad, because the horizontally driven open world needs to see what verticalizing closed-system makers like Apple are up to. In *The Intention Economy*, I explain the symbiosis between the two:

The smartphone business was invented by Nokia and RIM around the turn of the millennium and then royally disrupted by Apple and Google a few years later. Today, Apple and Google define the smartphone business, together, though not always in direct competition. It is important

to understand how this works, because the two companies' directions are orthogonal: ninety degrees off from each other. And because they do that, the market for both—and for everybody else as well—is huge.

Apple's punch to the smartphone market was vertical. It came up from below like a volcano and went straight toward the sky. With the iPhone, Apple showed how much invention and innovation the old original equipment manufacturer (OEM)-operator marriage had locked out of the smartphone marketplace, completely redefining the smartphone as a pocket computer that also worked as a phone. iPhones were beautiful, easy to use, and open to a zillion applications that were easy for programmers to write and for users to install.

The Google punch was horizontal. It came from the side, spreading its open Android platform toward the far horizons. As a platform, Android supported everything Apple's iOS did—and more, because it was open to anybody,

making it more like geology than a foundation. The old cartels could still build vertical silos on Android, but anybody could build just about anything, anywhere.

So, while Apple shows how high one can pile up features and services inside one big beautiful high-rise of a silo, Google provides a way not only to match or beat Apple's portfolio, but to show how broad and rich the open marketplace can be.

We need innovation in both directions, but we can't see how complementary these vectors are if we cast the companies innovating in both directions as competitors for just one space. So, while it's true that Apple phones compete with Android-based phones straight up, it's also true that Apple makes phones and Google doesn't. And, while it's true that iOS and Android compete for developers, iOS runs only on Apple devices; there is no limit to the number and variety of devices that run on Android. In the larger picture here, Apple and Google are stretching the market in orthogonal directions. The

result is a bigger marketplace for both and for everybody else who depends on smartphones.

I wrote that in late 2011, in faith that Apple would continue to break new ground in the vertical direction. Sadly, it hasn't (or barely). But ground is being broken in all directions horizontally (as well as some vertical ones) on Android.

The direction that interests me most is toward personal independence, both *from* and *to*. We need independence from phone companies, which continue to contain what we imagine can be done with connected mobile devices. And we need independence to do what we please, as free and fully empowered human beings.

Toward both ends on the hardware side, I am encouraged by Ubuntu for Phones (<http://www.ubuntu.com/devices/phone>) and Ubuntu for Android (<http://www.ubuntu.com/devices/android>), even though both are being pitched to corporate developers. I also like seeing custom ROMs for Google's Nexus 7 (<http://reviews.cnet.co.uk/ipad-and-tablets/top-six-custom-roms-for-the-google-nexus-7-50009354>). The reason there's an after-market for

Nexus hardware is that Google designed the line to be open and generative from the start. As the copy boasts, it's "unlocked and contract free" (<http://www.google.com/nexus/4>).

On the software side, I want to see tools that give each of us ways to take control of the Internet of Things and of API-based services. For that, I believe we need to be able to do our own programming, in the simplest and most uncomplicated ways possible. In fact, I see personal programming as the latest in a series of revolutions in which individuals have gained huge advances in power. In each of these following cases, individuals could do far more than could companies and large organizations:

1. Computing, thanks to the PC.
2. Communications, thanks to the Internet.
3. Portable computing and communications, thanks to the smartphone.
4. Programming, with...well, that's what's next. Read on.

We need to be able to program stuff in our own lives and in how we interact

with two other domains—and to do so independently, outside the control of any centralized entity or service:

1. The Internet of Things.
2. The portfolio of API-based services that are what large organizations need to become, whether they like it or not.

It's very early in the future that will grow here, but there are some early efforts that should invite our interest.

The best known (and funded, far as I know) is IFTTT (<https://ifttt.com>), which stands for "if this, then that". It lets you make connections between "channels" that are actually Web services, through those services' APIs. "Each Channel has its own Triggers and Actions", IFTTT explains (<https://ifttt.com/wtf>):

The *this* part of a Recipe is a Trigger. Some example Triggers are "I'm tagged in a photo on Facebook" or "I check in on Foursquare." The *that* part of a Recipe is an Action. Some example Actions are "send me a text message" or "create a status message on Facebook". Pieces of data from a Trigger are

called Ingredients. For example, the Ingredients of an Email Trigger could be: subject, body, attachment, received date and the sender's address. Personal Recipes are a combination of a Trigger and an Action from your active Channels. Personal Recipes look like this: "if Any new photo by (your handle) then Add file from URL to (your handle's) Dropbox."

Less well-known, but no less interesting, is on{x} (<https://www.onx.ms/#!landingPage>) a Microsoft effort targeted (yes) at Android devices. Still in beta, it "lets you control and extend the capabilities of your Android phone using a JavaScript API to remotely program it". It currently requires a Facebook login, which annoys me, but at this early stage, I'll forgive it.

Last (but far from least) is the growing suite of tools, concepts and services that have been rolling out of Kynetx for the last several years. The open-source side is the language KRL and a rules engine (<https://github.com/kre/Kinetic-Rules-Engine>). Phil Windley (<http://www.windley.com>) is the Linus of both, and he describes KRL this way: "KRL is a language that is designed to help programmers build

Advertiser Index

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

ADVERTISER	URL	PAGE #
1&1	http://www.1and1.com	19
ACQUIA	http://www.acquia.com	2
BIG DATA INNOVATION	http://analytics.theigroup.com/bigdata-sanfrancisco	23
BIG DATA TECH CON	http://www.bigdatatechcon.com/boston2013/	109
DOOR3	http://www.door3.com	27
EMAC, INC.	http://www.emacinc.com	17
EMPERORLINUX	http://www.emperorlinux.com	21
FLOURISH CONFERENCE	http://www.flourishconf.com/2013/	37
10TH ANNUAL HPC LINUX FOR WALL STREET CONFERENCE	http://www.flagmgmt.com/linux/	81
IXSYSTEMS, INC.	http://www.ixsystems.com	7
LINUXFEST NORTHWEST	http://www.linuxfestnorthwest.org/	65
NEW RELIC	http://www.newrelic.com	10, 11, 82, 83
NSDI'13	https://www.usenix.org/conference/nsdi13	59
PERCONA LIVE MYSQL CONFERENCE & EXPO 2013	http://www.percona.com/live/mysql-conference-2013/	95
SILICON MECHANICS	http://www.siliconmechanics.com	3

ATTENTION ADVERTISERS

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

and understand distributed, persistent data objects (PDOs) that live in the cloud and interact, primarily, through events. Collections of these PDOs form what I've frequently referred to as a personal cloud" (http://www.windley.com/archives/2012/12/programing_the_cloud_with_persistent_data_objects.shtml). These clouds can be hosted by one's self or at a service. They are, as Phil describes them, operating systems of one's own, with a "core set of services around identity, data and communications, as well as a programming model" (http://www.windley.com/archives/2012/07/a_road_map_for_the_personal_cloud_operating_system.shtml). Early applications of this model have me more excited than I've been in years. My wife too, and she's no techie. Take a look at SquareTag (<http://www.squaretag.com>), for example.

KuppingerCole (<http://www.kuppingercole.com>) has a term I like for the category being built out here: Life Management Platforms (<http://www.kuppingercole.com/report/advisorylifemanagementplatforms7060813412>). Hey, better to manage our own lives in the networked world than to be managed

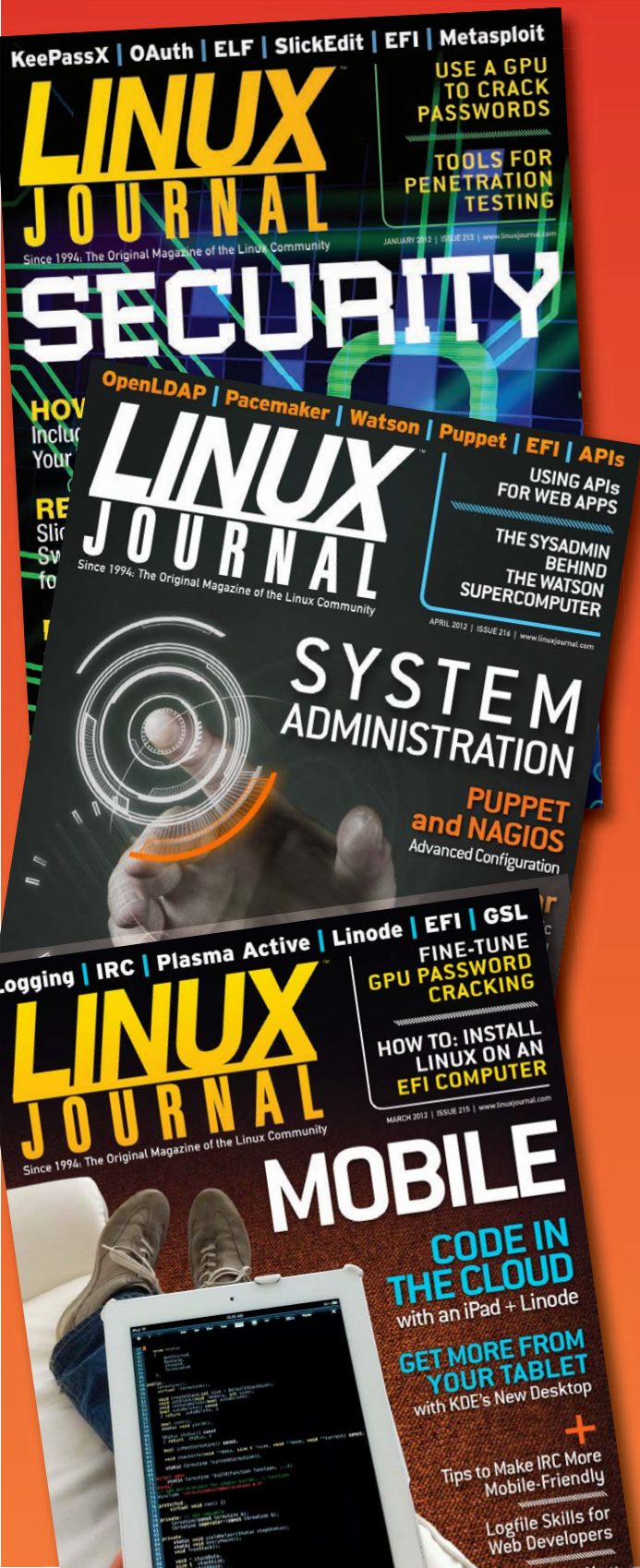
by the likes of Apple, mobile-phone companies or Google (which, let's remember, is an advertising company and takes advantage of its position with Android to watch much of what we're doing).

Given Android's overwhelming and growing success as a smartphone and tablet platform, it's obviously the bulls-eye toward which this kind of development should be targeted. Yes, it's not as open as many of us would like. For example, the latest SDK requires agreement to a nonfree license—though reportedly "to give Google the power to enforce the free software licenses of the components, if a third-party tries to break them" (<http://code.paulk.fr/article0008/what-s-up-with-the-android-sdk>). But, building on a majority OS is a nice place to start, especially when the market we're addressing is everybody.

I'm curious to see if, and how, readers dig some of the software developments I visit above—and, better yet, how nontechnical friends and spouses take to them. Listen for the magic line: "I can work with that." ■

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

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



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

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

SUBSCRIBE NOW AT:
WWW.LINUXJOURNAL.COM/SUBSCRIBE