GlusterFS: A Dependable Distributed File System

Waf: An Excellent Build Automation Tool

₹120

Volume: 05 | Issue: 04 | Pages: 100 | January 2017

OpenSource ForYou

THE COMPLETE MAGAZINE ON OPEN SOURCE

An EFY GROUP Publication

Go The DevOps Way With DOCKER

Docker Tools That Developers Will Find Useful

LXD: The Pure Container Hypervisor

Deploying A Jekyll Blog In Docker

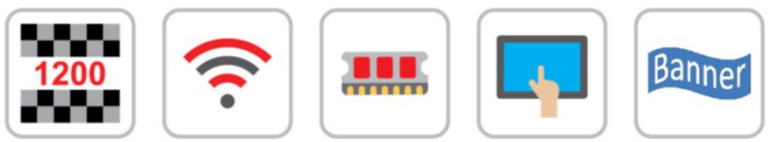
An Interview With **Nithya Ruff,** Director, Western Digital Open Source Office

A Case Study: **At Airtel,** Open Source Improved Flexibility, Agility And Security

MAKE A STRONGER IMPRESSION WITH COLOR.

THE NEW **TASKalfa MFPs** WITH BRILLIANT COLOR OUTPUT.

KYOCERA proudly presents its cutting-edge print technology with the new range of TASKalfa MFPs.



Fine 1200 resolution | Wireless printing option | 4 GB RAM | 9 inch color touch panel display | 1.2 m banner printing

Printers I A4 MFP I A3 MFP I COLOR MFP

Check out our latest 2016 range at our official website.

North : Delhi - 011-47340777, Gurgaon - 0124-4671000, Lucknow - 0522-2620261, 0522-2618261
South : Bangalore - 080-43403535, Chennai - 044-49012424, Ernakulam - 0484-4015842, Hyderabad - 040-40212343/44/45, Vijayawada - 0891-2737204
West : Ahmedabad - 079-40066864/65, Mumbai - 022-61299292, Pune - 020-66472709, Nagpur - 0712-2446377
East : Guwahati - 0361-2454512, Kolkata - 033-4019 6200, Ranchi - 0651-2241517

Supported by a Strong Dealer Network in over 200 locations across India

TASKalfa ECOSYS®

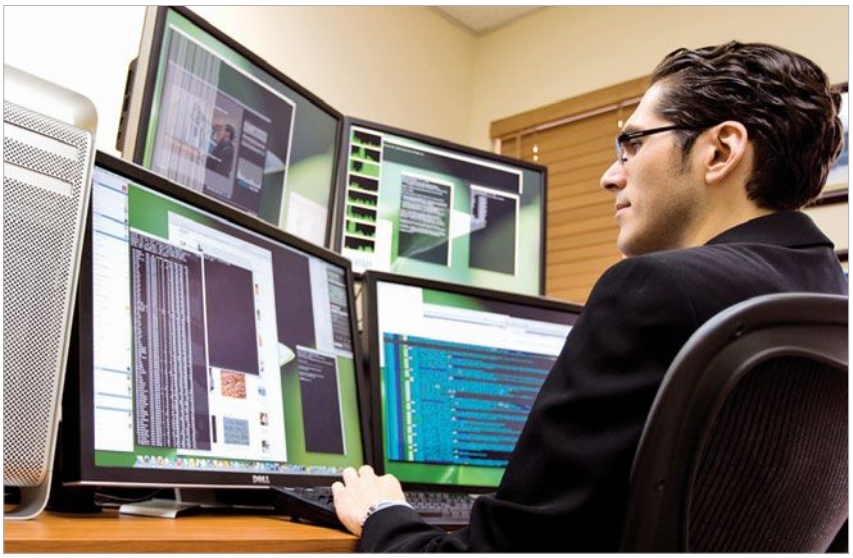
KYOCERA Document Solutions India Private Limited | Toll Free No.: 1800-103-7172
Website: www.kyoceradocumentsolutions.co.in | Email: ContactUs@did.kyocera.com

KYOCERA Document Solutions Inc. Japan
www.kyoceradocumentsolutions.com

Disclaimer: The product specification and image may be different from actual and is subject to change at the sole discretion of the Kyocera management.

KYOCERA
Document Solutions

Contents



70

Waf: An Excellent Build Automation Tool

Nithya Ruff, *director, Western Digital Open Source Office*

"India has become a great supporter of open source" | 18

C. Ramanan, *director of cloud networking – India*

"Cloud networking business is definitely growing in India" | 56

Open Gurus

Columns

DVD OF THE MONTH

Try out this distro for its stability.

- openSUSE Leap 42.2

EDITOR
RAHUL CHOPRA

EDITORIAL, SUBSCRIPTIONS & ADVERTISING
DELHI (HQ)
D-87/1, Okhla Industrial Area, Phase I, New Delhi 110020
Ph: (011) 26810602, 26810603; Fax: 26817563
E-mail: info@efy.in

MISSING ISSUES
E-mail: support@efy.in

BACK ISSUES
Kits 'n' Spares
New Delhi 110020
Ph: (011) 26371661, 26371662
E-mail: info@kitsnspares.com

NEWSSTAND DISTRIBUTION
Ph: 011-40596600
E-mail: efycirc@efy.in

ADVERTISEMENTS
MUMBAI
Ph: (022) 24950047, 24928520
E-mail: efymum@efy.in

BENGALURU
Ph: (080) 25260394, 25260023
E-mail: efyblr@efy.in

PUNE
Ph: 08800295610/ 09870682995
E-mail: efypune@efy.in

GUJARAT
Ph: (079) 61344948
E-mail: efyahd@efy.in

CHINA
Power Pioneer Group Inc.
Ph: (86 755) 83729797, (86) 13923802595
E-mail: powerpioneer@efy.in

JAPAN
Tandem Inc., Ph: 81-3-3541-4166
E-mail: tandem@efy.in

SINGAPORE
Publicitas Singapore Pte Ltd
Ph: +65-6836 2272
E-mail: publicitas@efy.in

TAIWAN
J.K. Media, Ph: 886-2-87726780 ext. 10
E-mail: jkmedia@efy.in

UNITED STATES
E & Tech Media
Ph: +1 860 536 6677
E-mail: veroniquelamarque@gmail.com

Printed, published and owned by Ramesh Chopra. Printed at Tara Art Printers Pvt Ltd, A-46,47, Sec-5, Noida, on 28th of the previous month, and published from D-87/1, Okhla Industrial Area, Phase I, New Delhi 110020.

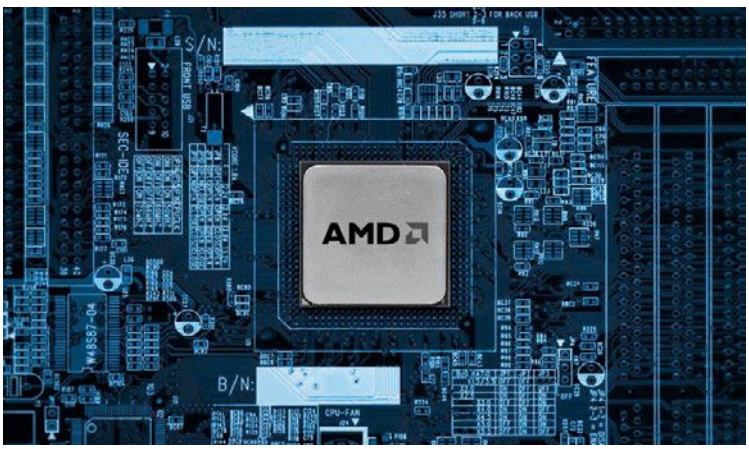
SUBSCRIPTION RATES

Year	Newstand Price (₹)	You Pay (₹)	Overseas
Five	7200	4320	—
Three	4320	3030	—
One	1440	1150	US$ 120

Kindly add ₹ 50/- for outside Delhi cheques.
Please send payments only in favour of **EFY Enterprises Pvt Ltd.**
Non-receipt of copies may be reported to **support@efy.in**—do mention your subscription number.

AMD brings FreeSync and Radeon Pro driver to Linux

AMD has added intrinsic support for FreeSync and brought the Radeon Pro driver to Linux. The new updates are supposed to improve the gaming and virtual reality (VR) experience on the open source platform.



Notably, this is the first time FreeSync has been made available on Linux. It improves the rendering experience of videos and games by reducing image lag and stutter, and helps in direct communication with the display.

In addition to the latest support, AMD can create high-quality VR content with its new Radeon Pro Software Linux driver. The driver is optimised to make the best of Radeon Pro GPUs on Polaris architecture. This will help professionals working on CAD/CAM and engineering applications to use Linux more efficiently.

AMD is trying to establish itself in this important niche market of high-performance GPUs. Most of these high-performance use cases have so far been supported on Windows. However, the latest updates from AMD will help in developing the Linux environment as a multimedia platform. The open source community has also changed its approach towards Linux. Multiple options like SteamOS and Vulkan API are designed to improve the gaming experience on Linux platforms.

The new updates by AMD are supported on Red Hat Enterprise Linux, CentOS and Ubuntu. Additionally, the Radeon Pro Software Linux driver works on platforms like the Vulkan API as well.

Red Hat's OpenShift now on Google Cloud

Red Hat has expanded OpenShift by announcing its dedicated presence on Google Cloud Platform. The open source Platform-as-a-Service (PaaS) was previously offering cloud support on Amazon Web Services.

With OpenShift Dedicated on Google Cloud Platform, enterprise customers can build, launch and manage their applications right on Google's cloud solution. The new offering also enables users to speed adoption of Docker containers and Kubernetes, and brings advanced cloud-native application patterns.

"By expanding OpenShift Dedicated to now include Google Cloud Platform, our customers' apps that run

Bluetooth 5 is here, promising to drive IoT adoption

The Bluetooth Special Interest Group (SIG) has just announced the fifth version of its wireless protocol. Bluetooth Low Energy (BLE) technology is likely to gain considerably from the new upgrade.

Bluetooth 5 is touted to be twice as fast for low-power applications than the preceding version. The speed of the latest Bluetooth standard could be as fast as 2Mbps. The new technology supposedly covers four times broader range, which could be an entire house or at least a floor of a building.

Mark Powell, executive director of the Bluetooth SIG, believes that Bluetooth 5 opens up new possibilities for IoT devices. "With the launch of Bluetooth 5, we continue to evolve to meet the needs of IoT developers and consumers while staying true to what Bluetooth is at its core—the global wireless standard for simple and secure connectivity," he said.

The concepts of smart homes and connected home automation products are still in the nascent stage. But home automation with Bluetooth 5 is predicted to become a lot easier. It would also enhance smartphones of the present age. However, it will take two to three years for all mobile devices to feature Bluetooth 5.

BLE devices have already outnumbered the Bluetooth enabled products. The fifth-generation technology is expected to boost this growth further and drive the adoption of BLE devices even higher. The new speed and range for BLE will help enterprises use BLE-powered beacons for location. A high rise in the adoption of BLE products is expected to continue to help users broadcast data.

on OpenShift Dedicated can also take advantage of Google Cloud services, furthering our vision for a truly portable platform across multiple cloud and infrastructure platforms," said Ashesh Badani, vice president and general manager, OpenShift, Red Hat, in a statement.

Red Hat has confirmed that OpenShift Dedicated is built on the same code base as the original Red Hat OpenShift Container Platform. This enables customers to move their workloads across public and private clouds with ease. Moreover, the container platform itself is based on Red Hat Enterprise Linux.

The prime benefit of opting for OpenShift Dedicated is its integration with Google Cloud services including Google Cloud PubSub, Big Query and Cloud Big Table. The platform can also now work across all six regions where Google Cloud is available. Additionally, there are administrative and security controls to enable a customisable and secure experience on cloud environments using VPN and Google Virtual Private Cloud functions.

"Customers consistently tell us that they are looking for open solutions that will enable them to use both public and private infrastructure; they want choice. Red Hat has been an essential partner in making Kubernetes available to enterprises looking for the support and flexibility to manage a hybrid cloud," said Nan Boden, head of technology partners, Google Cloud.

Red Hat is set to support customers migrate container workloads, adopt continuous delivery and DevOps processes and build microservices-based apps on Google Cloud using OpenShift Dedicated. Further, the open source solutions provider will offer technical consultations to encourage customers to opt for the latest development.

Interestingly, the new announcement comes days after Red Hat announced comprehensive JBoss Middleware support for OpenShift.

Enterprise customers can access OpenShift Dedicated on Google Cloud Platform using a valid Red Hat OpenShift Container Platform subscription.

SUSE acquires HPE's OpenStack and Cloud Foundry offerings

SUSE has announced the acquisition of OpenStack and Cloud Foundry offerings from HPE. The deal is aimed at bringing HPE's talent as well as technology assets to develop SUSE Cloud Foundry Platform-as-a-Service (PaaS) for the enterprise market.

As per the terms of the agreement, HPE will formally name SUSE as its preferred open source partner for Linux, OpenStack and Cloud Foundry solutions. The Nuremberg-based company is additionally set to become a platinum member of the Cloud Foundry Foundation board to enhance its engagement with the community.

"The driving force behind this acquisition is SUSE's commitment to providing open source software-defined infrastructure technologies that deliver enterprise value for our customers and partners. This also demonstrates how we are building our business through a combination of organic growth and technology acquisition," said Nils Brauckmann, chief executive officer, SUSE.

SUSE is certainly aiming to give stiff competition to Red Hat through the latest acquisition. However, the deal will also help HPE, which recently decided

WordPress to favour only SSL hosting, starting 2017

As secure browsing is becoming crucial for netizens, the Web world has started looking for some protection. In this ongoing race, WordPress is all set to protect the Web by favouring hosting accounts using Secure Sockets Layer (SSL).

To give a boost to the websites supporting HTTP over SSL (HTTPS), WordPress has announced that it will only promote hosting partners that, by default, provide an SSL certificate to their clients. This development will begin early in 2017. WordPress plans to persuade a large number of Web masters to choose an HTTPS connection over HTTP for their Web properties.

The WordPress team also plans to bring out features such as API authentication exclusively for SSL-supported websites. This would limit some advanced features on WordPress websites that are based on HTTP and are yet to opt for an SSL certificate.

"2017 is going to be the year that we will see features in WordPress which require hosts to have HTTPS available," wrote Matt Mullenweg, one of the creators of WordPress, in a blog post.

Apart from WordPress, companies like Google and Facebook are also supporting SSL to make the Web a safer place for all. In August, Google announced that HTTPS would become a ranking signal for its search engine. Notable results through the tweak are likely to be 'very lightweight, though significant' in order to influence Web masters to receive organic traffic steadily. Although opting for HTTPS is the need of the hour for Web masters, implementing an SSL certificate on websites is still quite a difficult task. WordPress suggests using projects like Let's Encrypt to enable the security layer in a fast and free manner.

The WordPress developers are additionally improving JavaScript support and overall performance in PHP7. These changes will improve the blogging experience over time.

Apple's Swift 3.1 to improve the programming experience on Linux

Apple has announced the next version of its open source programming language, Swift 3.1, which will include some hidden but worthy improvements to the Linux platform.

The Swift programming language has already been a breakthrough introduction for developers as well as a fruitful move by Apple to reach out to the open source community. But to influence Linux users, its upcoming version will include certain positive tweaks. Apple is also considering the source compatibility with Swift 3.0 as the top priority for developers.

"It is a strong goal that the vast majority of sources that built with the Swift 3.0 compiler continue to build with the Swift 3.1 compiler. The exception will be bug fixes to the compiler that cause it to reject code that should never have been accepted in the first place," said Ted Kremenek, senior manager for languages and runtimes, Apple, in a blog post.

In addition to the improvements on Linux, Apple has improved the Swift package manager, compiler and standard library within Switch 3.1. There are also quite a few major additions to upgrade the development experience.

Apple has not yet declared any definite release schedule. However, the next Swift update is expected to be rolled out some time between March to June in 2017. The Cupertino giant is planning to deploy considerable efforts for Switch 4.0 that will be out a few months after the Swift 3.1 release.

to merge its software business with SUSE's parent company Micro Focus, in order to concentrate on next-generation hybrid cloud solutions.

"We are evolving our investment strategy to focus on developing the next generation of hybrid cloud solutions, which combines HPE technology with a broad ecosystem of open source and partner technologies that support traditional and cloud-native applications," said Ric Lewis, senior vice president and general manager of the software-defined and cloud group, HPE.

"SUSE's acquisition of HPE talent and technology assets would add more muscle to the OpenStack Cloud by SUSE," Kalyan Muppaneni, founder and CEO, Pi Datacenters, told *Open Source For You*. "We believe that this will help enhance SUSE OpenStack's capabilities and deliver even more superior enterprise cloud solutions to our customers," he added.

This major step taken by HPE does not mean that the enterprise-centric company is leaving the field of PaaS and Infrastructure-as-a-Service (IaaS) or exiting the OpenStack and Cloud Foundry market. In fact, HPE is apparently aiming to improve its Helion OpenStack and Stackato solutions through SUSE's experience. "By partnering with SUSE, HPE will continue to provide high-quality OpenStack and Cloud Foundry PaaS solutions that are simple to deploy into customer's multi-cloud environments," stated Lewis.

In September 2016, HPE had announced that SUSE was going to be its preferred Linux partner. That development was made public in the midst of the merger between Micro Focus and HPE's software business segment.

Market analysts believe that the fresh deal is a 'real win-win' for SUSE and HPE and their clients worldwide. "This expanded partnership and transfer of technology assets have the potential to be a real win-win for SUSE and HPE, as well as customers of both companies," said Ashish Nadkarni, programme director — computing platforms, IDC.

Qualcomm develops 10nm server processor with Linux support

Qualcomm has initiated what it claims is the next level of data centre computing with the launch of its Centriq 2400 series. This new chip is supposedly the world's first 10-nanometre server processor and supports the Linux platform out-of-the-box.

San Diego-based Qualcomm has designed its Centriq family 'to reshape the future' of data centres. As the first in the advanced line-up, the Qualcomm Centriq 2400 series comes with the 10nm FinFET process technology. It also includes a Falkor CPU and a custom ARMv8-compliant core to enhance performance along with a major reduction in power consumption.

"The Qualcomm Centriq 2400 series processors will drive high performance, power-efficient ARM-based servers from concept to reality," said

Anand Chandrasekher, senior vice president, Qualcomm Technologies, in a statement.

While these processors are yet to be available in the server market, Qualcomm has started its commercial sampling. The company even conducted a live demonstration of the new chip running Linux-based Apache Spark and Hadoop.

Qualcomm is attempting to steal the march over Intel by developing the Centriq family. The top-end Intel chip supports 24 cores, whereas Qualcomm's latest offering comes with up to 48 cores. Also, this is the first time an ARM architecture is moving towards servers. This will encourage the market to leverage a powerful performance alongside open source software, as ARM-based chips are not likely to support the applications designed for x86 architecture.

The performance results will be out following the commercial availability of the Qualcomm Centriq 2400, which is expected to begin in the second half of 2017.

Microsoft R Server 9.0 enhances machine learning efforts

Microsoft has released R Server 9.0 to bring machine learning close to servers. The new update includes some cutting-edge algorithms within an all-new MicrosoftML package to make server technologies smarter than ever before.

Microsoft R Server 9.0 allows a combination of the latest algorithms with the conventional open source CRAN R packages and preloaded parallel external memory algorithms like the RevoScaleR package. There are offerings like a fast linear learner, GPU-accelerated Deep Neural Networks with convolutions and fast random forest.

Alongside the algorithms, the updated R Server improves some existing operationalisation capabilities. There is an option to convert existing R models and scripts into Web services using a single-line code on IDE such as R Tools for Visual Studio, RStudio or Jupyter Notebooks. Also, the R Server simplifies the application integration experience from the Swagger standard and brings cross-platform support to train models.

Microsoft has also introduced support for Spark 2.0, an upgrade from Spark 1.6. This change offers new methodologies for handling streaming data and includes an advanced memory management sub-system to improve performance.

"With this latest release, you have access to a powerful tool, one that supports popular operating systems and a variety of data sources, helping you to create sophisticated analytics models and deploy them in the real world, efficiently and at scale," said Nagesh Pabbisetty, partner director of programme management, Microsoft, in a statement.

Microsoft R Server 9.0 can be deployed on Ubuntu in addition to other platforms. This means the new R Server can be a part of SUSE and Red Hat

CoreOS gets transformed into Container Linux; enables self-driving Kubernetes

Security-centric CoreOS has made a couple of major announcements. The open source platform has been transformed into Container Linux and added self-driving Kubernetes support.

The CoreOS team has designed self-driving Kubernetes to let organisations enjoy an up-to-date experience with the latest patches and features. This new development offers auto updates alongside patches, upgrades and vulnerability responses.

"With the self-driving Kubernetes platform, users get the most advanced version of Kubernetes with the comfort of knowing that security updates can be patched seamlessly," said Alex Polvi, CEO of CoreOS, in a blog post. The self-driving Kubernetes is available as an option within CoreOS Tectonic that works on up to 10 nodes.

In addition to the latest Kubernetes development, CoreOS has differentiated between the open source distribution and its own brand by launching Container Linux. This platform comes along with a new logo. "Over the years, CoreOS (the brand) has grown to represent not just a product but the leadership and expertise we provide to our customers and in the open community," Polvi stated.

The original CoreOS platform has over one million unique instances on a monthly basis. The distribution was initially developed to secure the Internet. However, it is now streamlined for containers and delivers an upgraded platform with the auto-update capability.

Linux as well as major Hadoop distributions such as Cloudera, Hortonworks and MapR. Also, there are data sources for Apache Hive and Parquet.

The prime aim behind the release of R Server 9.0 is not only to favour machine learning extensively but also to boost Microsoft's presence in the server computing world. Interestingly, the Redmond based company is opting for open source technologies to expand its server offerings.

Browsix brings UNIX to Web browsers

To let developers build applications without installing any bulky software, Browsix has emerged as a one-stop solution. The framework offers the essence of UNIX to enable Web app development on any compatible system, irrespective of a particular build or version.

Browsix comes as a JavaScript-only framework to provide developers with extensive JavaScript runtimes for C, C++, Go and Node.js. There is also a POSIX-like shell that eases app development.

Initially developed as a research project by the PLASMA lab at the University of Massachusetts, Browsix has the capabilities to convert a client-server application to run completely in a browser. It has a TypeScript kernel behind the scenes, alongside the runtimes for open source languages such as C, C++ and Go.

"The core of Browsix is a kernel that controls access to shared UNIX services, which include the shared file system, pipes, sockets, and task structures, live inside the kernel, which run in the main browser thread," the Massachusetts University team comprising Boby Powers, John Vilk and Emery D. Berger, wrote in a detailed research paper.

Browsix enables developers to integrate a MemeGenerator-like meme server into the browser with no code modifications. Additionally, there is a need to modify the existing HTML code to load and initialise the Browsix JavaScript library.

Developers can convert their existing Web applications for Browsix using a process that compiles the code to JavaScript, staging the files required by the application to place in the in-browser file system, and then adding set-up code to the core HTML file.

The Browsix code can be accessed directly from a GitHub repository. It also includes the necessary licence files and documentation.

Slack ties up with Google to enhance the cloud-integrated experience

Slack, a popular team collaboration tool, has announced its strategic partnership with Google. The new development is aimed at offering enterprises and organisations 'deep integration' to enhance the existing cloud-integrated experience.

As a result of the partnership, Slack users will now get intrinsic Google Drive integration. Google has been building a special bot for Slack that provides access to Google Docs, Sheets and Slides as well as enabling comments and requests right from email notifications. Further, IT heads can connect specific spaces for teams to store files on Google Drive, called Team Drives, with Slack channels.

These Team Drives will receive automatic backups when the connected Slack channels get new files from users.

In addition to the Google Drive integration, Google developers are closely working with the Slack team to bring Doc previews into the collaboration tool. This means your Slack account will provide you Google Docs access to let you collaboratively create documents, spreadsheets and presentations.

"Google and Slack share the same vision for the future of work—that smart software could bring teams together and make all of their work and conversations seamlessly available in one place. The result is that teams can move faster and more efficiently," said Nan Boden, head of global technology partners, Google Cloud.

The Google team is working to enable admins to provision Slack for their entire organisation directly from the G Suite admin console. This will help in reducing user errors and ensure that accounts are provisioned.

Apart from the latest additions for teams, Google is actively working on scaling up Slack. Open source Google Cloud technologies are expected to improve the tool to take on Microsoft Teams and the recently launched Facebook Workspace.

"As users of Google Cloud ourselves, we are thrilled about this partnership and how it will simplify our working lives," the Slack team wrote in a blog post.

The changes brought about by the Slack-Google partnership will be rolled out to end users in the first half of 2017.

Android Things is the next big thing from Google

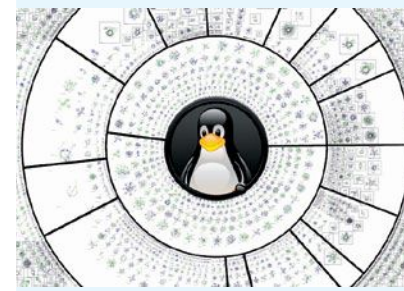
After playing a successful round in the smartphones arena, Google has now entered the ring of connected devices with its latest development, Android Things. This new platform is specifically designed to meet the growing demands of Internet of Things (IoT) using Android.

"Now, any Android developer can quickly build a smart device using Android APIs and Google services while staying highly secure with updates direct from Google," said Wayne Piekarski, developer advocate for IoT, in a blog post.

Focusing on IoT is important for Google. But instead of bringing out a completely distinct model, the Android maker has preferred to use its existing

Linux kernel gets fix for serious DoS vulnerability

The Linux community has received a patch for a security hole that could cause a denial of service (DoS) attack. Linux distributions that are affected by the flaw include the recent versions of Debian, Fedora, Red Hat Enterprise Linux and Ubuntu.



Security researcher Philip Pettersson spotted the vulnerability, designated CVE-2016-8655, within the *packet_set_ring* function of the Linux kernel. Pettersson described a race condition that exploits a local user through *AF_PACKET* sockets with *CAP_NET_RAW* in the network namespace.

"A local unprivileged attacker could use this to cause a denial of service (system crash) or run arbitrary code with administrative privileges," read the brief description.

Apart from the DoS vulnerability spotted by Pettersson, researchers have found CVE-2016-6480 and CVE-2016-6828. Both the flaws exist within the kernel code and can crash the system by a local attacker.

Patches for all the three vulnerabilities have started rolling out for major Linux distributions. It is recommended that users download the latest versions to avoid any severe attacks.

open source platform. “We incorporated the feedback from Project Brillo to include familiar tools such as Android Studio, the Android Software Development Kit (SDK), Google Play Services and Google Cloud Platform,” stated Piekarski.

Google announced Project Brillo as its initial IoT platform at the I/O developer conference in 2015. This will now be a part of Android Things.

On the hardware front, Google has partnered with some manufacturers for innovations such as Intel Edison, NXP Pico and Raspberry Pi 3. This strategic move will enable these device makers to integrate Android Things support into their hardware.

Alongside the latest Android platform for IoT devices, Google has announced an update to its Weave platform. The new release is aimed “to make it easier for all types of devices to connect to the cloud” and enhance existing services such as Google Assistant.

It is worth noting that Google’s Weave is already a part of devices under the Philips Hue and Samsung SmartThings line-up. Thus, it is powering not just the hardware that has Android support but many smart devices like light bulbs, smart plugs and switches.

Google is also set to merge Weave with Nest Weave to give developers a unified way to enter the world of IoT.

Developers can access documentation and code samples for Android Things and the updated Weave directly from their respective websites. Moreover, a specific IoT developers’ community is available on Google+ to offer real-time support to those who need it.

Google Keyboard now becomes Gboard on Android

Google has replaced the default keyboard app on Android with its Gboard. Initially available on iOS, the virtual keyboard is designed to support hardware apart from Nexus and Pixel devices.

Taking on SwiftKey, Gboard delivers an advanced typing experience on the open source Android platform.

It supports GIFs as well as Google Search integration. Additionally, there is an option to search for new emojis.

There is a dedicated G button on the keyboard that gives you direct access to Google Search. For large-screen devices, the keyboard has a one-handed mode. You can also use the built-in emoji options to express your feelings through some attractive emoticons.

Besides this, the basics of the original Google Keyboard remain in Gboard. There are features like auto-predict and glide typing to ease texting on the virtual keyboard.

Google is yet to enable separate Gboard access for developers. Meanwhile, you can use it on your Android devices by downloading an APK file. The keyboard is expected to reach the Google Play store soon.

Kubernetes v1.5 adds support for Windows server containers

Kubernetes has received a new update. The version 1.5 comes with native support for Windows server containers and is designed to run a distributed database.

The Google team behind Kubernetes is aiming to help .NET developers with the latest release. This is the reason Kubernetes 1.5 enables initial support for Windows Server 2016 nodes and scheduling Windows server containers. This support is likely to get better over time, as developers are expected to give feedback on areas for improvement.

The support for Windows containers is big news for the open source world after Amazon Web Services entered the container management market with Blox. Amazon’s offering might face stiff competition from Kubernetes.

Apart from the Windows container support, the updated Kubernetes has Kubefed that comes as a new command-line tool to help you manage multi-cluster federations at once. You can also use StatefulSet to manage pods in a smarter way.

StatefulSet beta allows persistent identity and per-instance storage workloads to be created, scaled, deleted and repaired right on Kubernetes. Also, it eases the deployment of stateful services and provides tutorial examples.

The new Kubernetes also comes preloaded with the first version of the Container Runtime Interface (CRI) API to enable pluggable container runtimes. The platform also includes a beta version of the Node conformance test.

You can download Kubernetes 1.5 on your system directly from its online repository on GitHub. It is also available through get.k8s.io.

For more news, visit *www.opensourceforu.com*

CODE SPORT

Sandya Mannarswamy

In this month's column, we discuss the topic of robotic process automation.

Happy New Year to all our readers! As we start a new year, this month's column also is about a new topic. One of our readers had requested me to cover robotic process automation (RPA). Hence, this month's column provides a brief overview of this interesting and important trend in information technology. We will also examine how natural language processing and machine learning play a role in RPA.

What is robotic process automation?

Robotic process automation (RPA) is defined by the Institute for Robotic Process Automation (IRPA) as 'the application of technology allowing employees in a company to configure computer software or a 'robot' to capture and interpret existing applications for processing a transaction, manipulating data, triggering responses and communicating with other digital systems.' In other words, it involves automating certain tasks or processes that manipulate data. While the definition of RPA as defined by IRPA is very broad and uses the word 'robot', real life RPA has nothing to do with robots. Though the term 'robotic process automation' conjures up visions of an army of robots doing some human tasks, such as moving things or carrying out some other such physical labour, RPA does not have anything to do with robots themselves. It is a form of automation where intelligent software processes take over certain tasks that have typically been performed by human beings.

The readers may now be wondering how this is different from normal software processes involved in desktop automation, which typically do many tasks such as generating bar charts from data, or performing accounting calculations in Excel spreadsheets, typically performed by defining custom macros. Well, the difference comes from the fact that such simple automation scripts fail when there is a decision to be made and when there are a set of complicated steps to follow in order to achieve a task. Typically, desktop automation is limited to defining functions or macros that operate within an application. Let us consider an example from the financial services, namely, the accounts payable process. This is a critical function in all organisations. It includes reconciling the statements submitted by the vendors against the statements of the internal buying departments of an organisation. An organisation can buy goods and services from outside vendors. The purchase process is typically carried out by means of purchase orders. When the goods and services are received, the buying departments create the 'received' reports. The vendors or sellers of these goods and services then submit invoices for payment by the organisation. The reconciliation is carried out by a three-way process between the vendor invoices, received reports and the purchase orders.

While this appears to be a routine task, mismatches between these three data sources can require human intervention and decision making. A typical mismatch could be where the vendor shows the invoice as unpaid whereas the internal department system shows the invoice as paid. The accounts payable process then needs to identify the root cause of the mismatch, and so carries out one of the following three actions. If it finds that the invoice has already been paid, it needs to send the payment details to the vendor. If it has been processed and marked as not paid, it needs to provide the reasons for non-payment to the vendor. Or if the invoice has not been received, it should send a notification to the vendor to send a duplicate

invoice. This is a repetitive task and needs decision-making based on the type of mismatch found. Hence, this task is typically done by humans.

Given the repetitiveness of the task and the need for decision-making, this is an ideal task for RPA. The RPA tool, when deployed, can isolate the mismatches and take the required action. Instead of having humans perform this repetitive task, using a robotic process automation tool will improve productivity by over 90 per cent, while achieving the highest accuracy. This is just one simple example of an RPA tool. Many such instances exist in the organisational workflow. Today, many of the business processes that were outsourced are getting transformed as a result of RPA. Business processes can either be front-end operations dealing with customers in call centres, or they can be back-end processes such as accounts payable, checking the derivatives trades, insurance claims processing, etc.

Automation of front-end operations typically involves chat bots or conversational agents. These automation processes usually require natural language processing or natural language understanding. Back-end operations can be either based on structured data or unstructured data. Structured data operations typically include data operations on various databases. Unstructured data operations typically include processing of documents, images and multi-modal data. Now, let us take a brief diversion to understand how natural language processing (NLP) can aid in RPA.

Natural language processing and RPA

According to industry experts, RPA is likely to progress through three stages of innovation. The first stage consists of dealing with static rules and structured data, with tasks such as data entry and validation getting automated. Stage 2 of RPA consists of some form of natural language processing capabilities and performing tasks such as content analytics and process automation. Stage 3 consists of advanced natural language processing and cognitive capabilities—this deals with RPA systems capable of human-like decision-making. Figure 1 shows the evolution of RPA capabilities through these three stages*.

While existing RPA solutions work on the 'click and automate' mode when dealing with structured data, with regard to unstructured text data, RPA has not been exploited fully for intelligent automation. Existing RPA solutions are 'action based' – they understand the actions being performed on the GUI or on the back-end enterprise systems, be it data entered in a specific field in a form or a table retrieved from a back-end database. Existing RPA scripts do not interpret the content of the textual data that arises in the workflow. Knowledge from contextual text artefacts is not used either in process discovery or in exception management.

Moving RPA systems from the 'Repetitive Do' cycle to a 'Think – Judgement-based on context' cycle requires incorporating content analytics of textual artefacts that are present in the business process workflow. For instance, the 'Repetitive Do' cycle automation that handles structured data and static rules is better suited for automating tasks such as claim processing, account data reconciliation or data consolidation and validation. However, in scenarios where contextual knowledge and judgement are required, such as in service desk incident resolution, complaint resolution and management, eligibility processing—all of which involve unstructured text and considerable exceptions—existing RPA solutions do not fare well, due to lack of intelligent text processing.

Let us illustrate this with a simple use-case—the business workflow for a customer service desk ticket resolution system for an ecommerce retailer back-end. The system's tickets can be raised through customer chats/emails and can either be responded to in real-time by agents via chat, or offline over emails. For each problem reported, the information is hidden in unstructured text. Each complaint needs to be identified for its respective category – such as missing item, delayed delivery, cancelling of purchase, change of address of the customer, etc. For each specific problem, there is a sequence of actions involved. All problem tickets have the initial sequence steps of:

a. Authenticating the customer.
b. Processing order details and retrieving them.
c. Verifying the customer address, which is part of the back-end compliance mechanism.

After these steps, the problem resolution consists

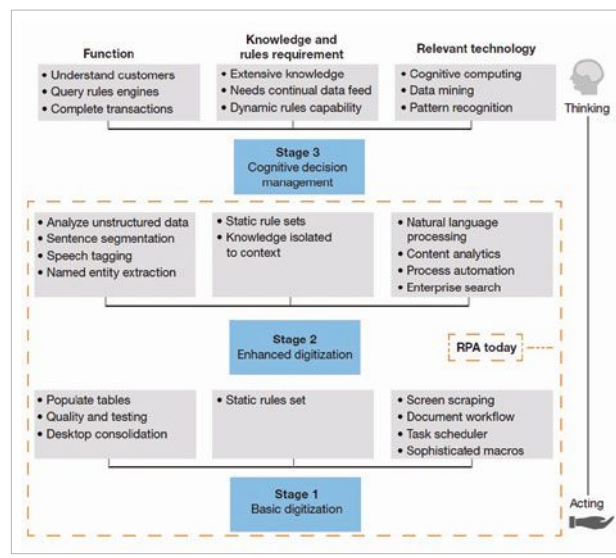


Figure 1: RPA capabilities in three stages

*As per a research report from Forrester Research on the state of RPA

of a sequence of actions that can be different for different problems.

d. Identify the problem category.
e. Find the appropriate resolution.
f. Identify the action sequence to resolve the problem.
g. Record the action sequence for resolution.

While steps (a) to (c) are ideal for traditional RPA since they typically deal with structured data entered through forms, steps (d) to (g) require dealing with unstructured textual content, understanding contextual knowledge, as well as content analytics and judgement. NLP modules aid in problem categorisation based on unstructured text, knowledge mining of back-office problem manuals to identify the resolution, parsing previous interactions to mine the action sequence in terms of extracting the verbs/subject/object from the text, and outputting the necessary action sequence. Existing RPA solutions can help automate steps (a) to (c), but typically fail when automating steps (d) to (e). However, by employing natural language processing techniques, RPA tool suites can be enhanced to automate steps (d) to (e).

A similar problem emerges in insurance claim validation. Typically, insurance claims contain unstructured data like supporting documents that need to be analysed in case of any potential violations. Data also needs to be extracted from multiple systems and validated against each other. Given that it is possible to synthesise a set of rules to cover the validation process, this is a good candidate for RPA.

So far, we have been talking about existing RPA capabilities in terms of being able to handle static rule-based tasks which are repetitive and whose decision-making can be condensed into a form of *if-then* statements. How can RPA deal with more complex decision-making scenarios? How can it learn to handle exceptions in a workflow like humans do? Who are the leading vendors in RPA? We will cover these topics in next month's column.

If you have any favourite programming questions/software topics that you would like to discuss on this forum, please send them to me, along with your solutions and feedback, at *sandyasm_AT_yahoo_DOT_com*. Till we meet again next month, here's wishing all our readers a wonderful and productive new year ahead! END

By: Sandya Mannarswamy

The author is an expert in systems software and is currently working as a research scientist in Xerox India Research Centre. Her interests include compilers, programming languages, file systems and natural language processing. If you are preparing for systems software interviews, you may find it useful to visit Sandya's LinkedIn group Computer Science Interview Training India at *http://www.linkedin.com/groups?home=&gid=2339182*

NEW PRODUCTS

Price: ₹ 4,999

A portable, wireless speaker from Syska

Syska, the manufacturer of innovative solutions for mobile phones and computing accessories as well as a range of LED products, has strengthened its audio portfolio with the launch of its latest Ipx4 wireless speaker called Blade.

The speaker has a sturdy and robust build, including a splash-resistant feature protecting it from damage.

The device is enhanced with an innovative 360 degree rotation feature, allowing its powerful sound to travel in any particular direction. The slim and compact speaker is portable and offers clear, rich sound across all frequencies with dual 5W speaker drives. It also features dual external passive radiators to accentuate the bass response, along with a 2000mAh rechargeable battery that supports up to six hours of non-stop playtime and five hours of talk time.

The speaker is capable of connecting seamlessly with any Bluetooth enabled device, supports FM radio, microSD card (32GB) and playback.

Available in vibrant shades of blue, green, orange, red and yellow, the IPX4 speaker can be purchased online and at retail stores.

Address: Syska Accessories, Syska House, Plot No. 89, 90, 91, Lane No. 4, Serial No 232/1+2, Sakore Nagar, Off Airport Road, Lohegaon, Pune – 411014; **Ph:** 1800-102-8787

Track-and-talk enabled smartwatch for kids

Price: ₹ 4,799

Mobile handset manufacturer, Alcatel, has launched its GPS enabled smartwatch, MoveTime, in India. The watch allows kids to communicate with pre-approved contacts.

This track-and-talk smartwatch enables parents to stay connected with their children, while controlling who they communicate with. It weighs less than 40g, and comes with the Nucleus operating system. It sports an OLED 2.4cm (0.95 inch) white colour display and is compatible with both iOS and Android platforms.

The device has a 370mAh battery pack with talk time up to two hours, with a full charge taking 1.5 hours. The battery also has a standby time of up to four days. The watch features an SOS button for sending distress signals to parents during panic situations.

The Alcatel MoveTime is available via online stores.

Address: Alcatel India, Plot No. 25, Electronics City, Sector 18, Gurugram, Haryana; **Ph:** 0124-4159999

Portable amplifier from FiiO launched in India

Price: ₹ 8,999

Chinese audio products and accessories manufacturer, FiiO, has launched a new audio amplifier, the A5, in India. Weighing about 168g, the device has a sandblasted finish. The amplifier can be considered an upgrade from the earlier-launched E12 series, but with added features.

The device continues to use FiiO E12A's MUSES02 and LME49600 opamp combination.

It uses high-precision metal film resistors to offer a total harmonic distortion rating of almost 0 per cent. With an improved signal-to-noise ratio of 115db, the FiiO A5 has a low resistance power management system with dual ±11V supplied power, much higher than the E12A's ±7.4V supplied power.

The device offers an output of almost 800mW at 32Ω. With a battery backup of 880mAh, the company claims that the device has a playtime of about 13 hours on a single charge. It is also designed with an LED indicator, making it easy to determine when the device is about to run out of battery or when it is fully charged.

The FiiO A5 is available via eBay and the company's website.

Address: FiiO Electronics Technology Co. Ltd. 2/F, F Building, Hougang Industrial Zone, Shigang Village, Huangshi West Road, Baiyun District, Guangzhou City, 510430, China.

A power-packed, pocket-friendly smartphone from Lenovo

Lenovo has unveiled its latest smartphone, the K6, in India. The powerful and pocket-friendly K6 comes with a 12.7cm (5 inch) display and an all-metal body that makes it sturdy yet attractive.

This light and compact smartphone has a fingerprint scanner at the back along with dual speaker vents.

The highlight of the device is its huge 4000mAh battery, which can last two days on one full charge, with ease. The K6 runs on Android Marshmallow and is powered by a 1.4GHz octa-core Qualcomm Snapdragon 430 processor with Adreno 505GPU.

It has 3GB of RAM and 32GB internal memory expandable up to 128GB via microSD card.

The device supports dual nano SIM card slots and 4G VoLTE. It features a 13 megapixel rear camera and an 8 megapixel front camera. The connectivity options of the smartphone include Wi-Fi, GPS, Bluetooth, USB OTG, FM, 3G and 4G. Sensors on the phone include a compass, magnetometer and a proximity sensor.

The Lenovo K6 is available in dark grey, gold and silver, via online and retail stores.

Address: Lenovo India Pvt Ltd, Vatika Business Park, 1st Floor, Badshah Pur Road, Sector 49, Sohna Road, Gurugram – 122001

New Hyve smartphone is powered by a deca-core processor

Smartphone and accessories manufacturer, Hyve Mobility, has recently launched the Hyve Prym, the company's flagship deca-core smartphone. Offering pure Android 6.0 Marshmallow and a range of useful features, the smartphone has a 14.4cm (5.70 inch) touchscreen display with a resolution of 1080 x 1920 pixels and an all-metal body. It is powered by the 2.3GHz MediaTek Helio X20 SoC, the first mobile processor with the tri-cluster CPU architecture and 10 processing cores (deca-core).

The device comes with 4GB of RAM and 32GB internal memory expandable up to 200GB via microSD card, and a battery backup of 3500mAh offering USB-C with fast charge. The dual-SIM smartphone has a 13 megapixel f/2 rear camera with dual flash, HDR photography and an 8 megapixel front camera with flash.

The connectivity options of the device include Wi-Fi, GPS, Bluetooth, FM, 3G and 4G. Sensors on the phone include a compass, magnetometer, proximity sensor and accelerometer.

The Hyve Pryme is available in champagne colour via Amazon.

Address: 4GM Hyve Mobility Pvt Ltd, M-67, First Floor, Greater Kailash 2, New Delhi – 110048; **Ph:** 1800-121-4983

The prices, features and specifications are based on information provided to us, or as available on various websites and portals. OSFY cannot vouch for their accuracy.

Compiled by: Aashima Sharma

“INDIA HAS BECOME A GREAT SUPPORTER OF OPEN SOURCE”

Nithya Ruff, director, Western Digital Open Source Office

Open source is influencing the IT industry around the globe. Not just the software leaders but also the big hardware players of the industry are closely watching the growth of open source technologies. India is emerging as one of the top regions for open source, for many corporates as well as the community. ***Jagmeet Singh*** of ***OSFY*** interacted with ***Nithya Ruff, director, Western Digital Open Source Office***, to understand where India is on the radar of the global IT giant. Ruff recently joined the ***Linux Foundation*** as an ***at-large director*** to bring additional focus on gender equality and diversity to the open source world among other things. Edited excerpts of this exclusive conversation...

Q What were the prime challenges you faced while adopting open source at Western Digital?

A large number of companies moving to open source face the challenges of education, creating awareness and a compliance policy that strikes the right balance. We are no different, and have grown over the past three years to be an active contributor and visible supporter of open source in the community.

Q How are you planning to enable open source within Western Digital?

Open source, like standards, is considered an essential and strategic part of how we innovate at Western Digital. It is connected to our technology and business strategy as a company. Going forward, we plan to continue to consume and contribute to key projects, speak to and sponsor developers and be a part of projects and communities that we work in. We will also collaborate with other companies and projects to develop areas that are important to the industry and to us. Certainly, you will see us being more visible in our support for open source innovation.

Q Has it become vital for an IT solutions provider like Western Digital to choose the open source path, today?

Open source innovation is in every major area of our business, starting from handheld devices to mobiles and embedded storage, to the data centre and the cloud. We held an Investor Day in December 2016, and encouraged customers to view our latest strategies on our investor microsite.

Being a leading storage solutions provider, our customers often ask for some standards-based solutions. Open source has been creating standards and leading innovation for some time now. We recognise and leverage it in our product and technology plans. It is one aspect of an innovation plan.

Q Is it necessary to have strong community support for the success of an open source offering?

It is always good to have strong community support for an open source project. Companies like Western Digital work hard to build awareness and support for their open source offerings. Sometimes, a project may be small and niche -- appealing to only a few people. In such cases, it needs to serve only that small group. But in all other cases, a project needs to have broad and strong support from the community, in order to succeed. Some of the metrics we look at are the number of

downloads, users, developer contributions and issues reported and pending, as well as the level of engagement, to gauge if the support from the community is strong.

Q How can IT companies ensure active participation from various communities for their open source solutions?

Interaction between the community and company is critical. We need each other. One of the best ways to build community support is to make it easy for contributors and users to collaborate and contribute, and have their voices heard. Companies can communicate the project at meetups and events, and share the many ways people can contribute to, download the software, and be involved.

Also, companies and project leaders need to be responsive to issues and messages in order to create an inclusive community. Having an open and transparent way to collaborate through a good governance model is indeed the key.

Q Where do you see India in terms of open source contributions?

I have not tracked contributions from India closely, but have observed a growing interest and presence from many Indian companies and developers in open source projects that I am a part of.

India has become a great supporter of open source. From students working as interns to developers in multinational and Indian companies, a large part of the country is leveraging open source. The government also has a more open policy regarding the use of open source.

> "A track record of working and contributing to open source is vital when choosing the right talent."

Q Do global IT concerns like Western Digital and SanDisk look at hiring developers from India?

Yes, absolutely. The industry needs smart and talented people. We have an office in India, and have many qualified and talented developers that work for us there and even in the US.

Q What qualities do you as an open source head look for in open source developers?

A track record of working and contributing to open source is vital when choosing the right talent. Also, the ability to work with the community and understand the norms of contributing, the variety of licences, the quality of work when involved collaboratively with others in a distributed and remote project—these are some of the key points to consider.

Most important is a passion for doing open source work, which involves a different way of developing. It is important to understand this in order to be effective in a community-run project.

Q Do you think there is a huge gender gap in the tech sphere across the globe?

Yes, unfortunately, there is a gender gap. There are many factors contributing to this. In my opinion, the factors include support at home to learn tech, media images of who is in tech, how school classes are structured, and finally, the culture of the tech world. All of these reinforce some gender stereotypes that tech is male dominated and suited only for men.

However, women nowadays are moving into a new and more visible role, and I encourage that. When I talk to successful women in the tech industry, they have some things in common -- a very supportive home with a parent or spouse who is in tech or encouraged them to get into tech. Early exposure to tech in gaming or other technical areas is also key. A teacher or champion who supported them in engineering school and helped them stay in engineering can also be advantageous. It also takes persistence and strength for a woman to be in tech today. I have done it—and so can other women. Based in Silicon Valley, I see many successful women; so I encourage all of them to take the challenge.

Q How can we build a women-friendly environment in the technology world?

The environment needs to change to include more women and newcomers into tech. There is definitely more awareness today, and things are slowly changing. But some things still need to change.

What we need are mentors and the exposure to tech careers and skills for girls, right at the school level. Many companies are doing this, and we need more working professionals who spend time in schools, sharing and educating girls and boys on a career in tech and about the skills needed. Similarly, teachers who know how to teach coding and computer science in schools need to engage with girl students. Schools like Harvey Mudd, which create and craft curriculum to welcome women into tech and make the on-ramp into tech easier, also need to be encouraged for further initiatives.

To bring women into engineering, tech companies also need to make certain cultural changes in their recruiting, retention and development processes. Often, activities and culture are geared around men who have been the dominant group. From work hours to informal socialisation to how communication happens, companies should examine how this needs to change to make it easier for women to join and stay. Further, managers need to be trained to be inclusive and demonstrate as leaders how to include and support all the people in their teams. Lastly, CEOs and leaders of companies need to make this a business imperative and ensure that all levels of the organisation believe and support inclusion. I am proud to say that Western Digital does all this.

Q Do you have any plans to leverage your leadership at the Linux Foundation to enable greater participation of women in the growth of open source?

Yes, this is one of my key focus areas at the Linux Foundation—to advise, advocate and support programming around welcoming more women and minorities into open source.

Q How is the Linux Foundation making open source developments easy for developers and solutions providers?

The Linux Foundation's biggest impact is to create training, events and education around how to work with open source for developers and companies. It also houses and funds critical projects in the industry and provides tools, best practices and education around open source compliance. All of this makes it easier for developers to create projects that are valuable.

Q What are the advantages of joining the Linux Foundation for companies like Western Digital?

The Linux Foundation has become a central place for collaborative development in the industry, and companies that do work with open source have benefited by becoming members. This non-profit organisation helps companies like Western Digital become strategic and skilled at consuming, contributing, collaborating and becoming compliant in open source. This is a key external engagement that all companies need to learn to manage as they become disrupted by digitisation and open source.

Q On a concluding note, how do you see open source evolving in the next five years?

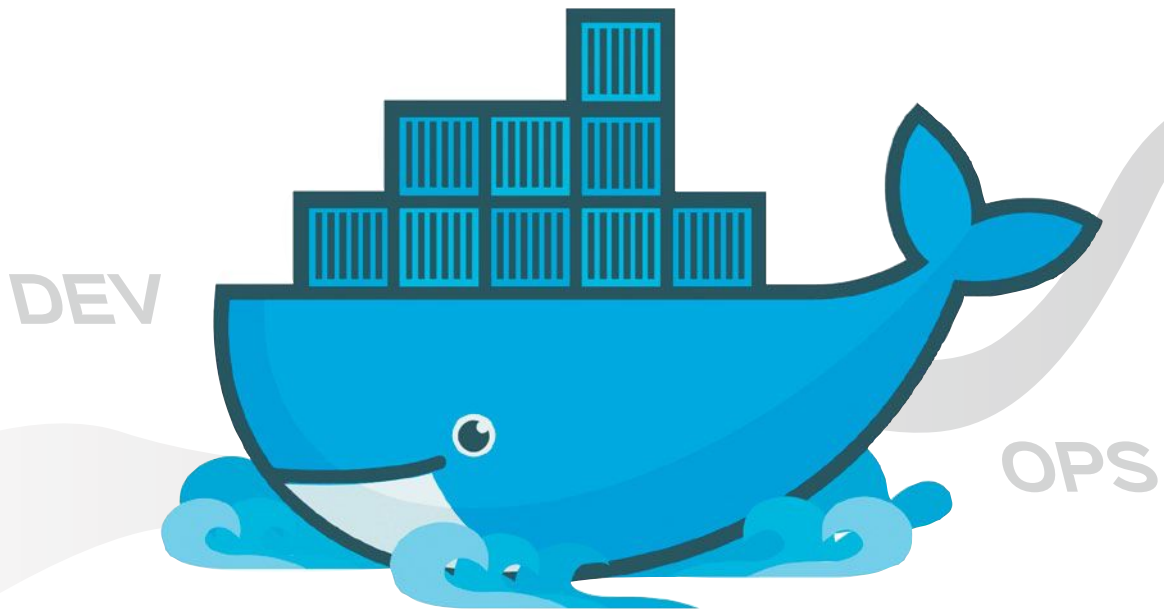
As more industries and products are touched by open source, we will see a lot of new people and industries enter the community. We need to make it welcoming and easy to get involved.

There will be a need for collaboration across open source projects, so that individual projects do not work as islands.

Open source tools will become easier to use and will be taught in schools. Best practices and consistent practice of how to manage open source development will reduce fears and uncertainty about open source across the industry. Moreover, open source methods inside companies will be increasingly used as the way forward to develop software. END

Go the DevOps Way with Docker

DevOps is a term that's emerged from 'development and operations'. It refers to the collaboration, communication and integration between software developer and IT operations teams, leading to faster production of software and services. This tutorial starts off using the official MySQL image with a minimal Dockerfile, and then moves to Docker Compose to set up a development environment for Rails.



Of late, Docker has clearly emerged as a winner in the DevOps world. It is not a panacea for all problems, though, and has a few issues, the biggest being the learning curve. Docker is fairly new while the technology it uses, i.e., Linux containers, has been around for well over 10 years. Google is a pioneer in the development of Linux containers, while Docker has come up with a more user-friendly approach to using containers.

Let's consider an example of a Rails application that uses MySQL and Redis. If you are running DevOps for your startup, one of your roles would be to bring on board new developers and help them set up services on their machines. You will probably use some automation tool like Vagrant to get this done. If you are doing it manually via *apt-get* or Brew, then this is the best time to start using Docker. Dockerising a service simply means adding a `Dockerfile` to your project, just like a Vagrantfile. In this tutorial, we will start with using the official MySQL image with a minimal Dockerfile, and later move to Docker Compose to set up a development environment for Rails. This article assumes readers have a basic knowledge of the Linux command line.

You must have Docker and Docker Compose installed on your machine before starting, for which the official documentation can be followed (Docker: *https://docs.docker.com/engine/installation*; Docker-compose: *https://docs.docker.com/compose/install*).

Once you have Docker installed, you can verify it by running *docker* without any arguments. You can verify your installation as shown below:

```
<code: shell>
$ docker -v
Docker version 1.12.3

$ docker-compose -v
docker-compose version 1.8.0-rc2, build c72c966
```

Docker requires sudo to run. However, we will not use Docker with sudo, because we wish to avoid repetition and maintain simplicity. You can copy-paste the commands shown below to avoid using sudo. (Logout → Login for group changes to take effect.)

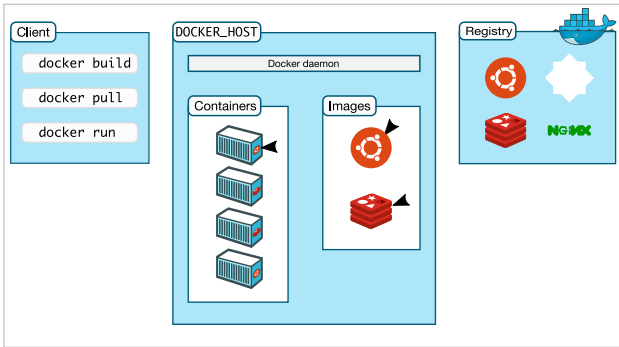


Figure 1: Docker architecture [*https://docs.docker.com/engine/ article-img/architecture.svg]*

```
<code: shell>
$sudo gpasswd -a ${USER} docker
```

There are three major terminologies to understand when working with Docker.

1. *Dockerfile:* This is a text document, which contains a set of commands that are required to build an image.
2. *Image:* This is a compressed archive that is built from executing the commands in your Dockerfile.
3. *Container:* This is a running instance of an image. You can create any number of containers from an image.

Before we start, let's set up our environment. We will be doing all the work inside *~/dock*.

```
<shell>
$ mkdir -p ~/dock/mysql
$ cd ~/dock
# demo rails application
$ git clone https://gitlab.com/techmaniack/rails-mrd-demo.git
~/dock/rails-rmd-demo
```

One of the major benefits of using Docker is that most of the popular open source tools are Dockerised by their maintainers and are available for anybody to use. To know what a Dockerfile is and what it does, we will use a MySQL image as a starting point and build our own image locally.

```
<code: shell>
$ cd ~/dock/mysql
$ touch ~/dock/Dockerfile
```

Open the Dockerfile in the editor of your choice and copy the following code inside it:

```
<code Dockerfile>
#Use official mysql as a starting point
FROM mysql
#Set password for root user
ENV MYSQL_ROOT_PASSWORD=root
```

Now, we are all set to build our first image, which we will name '*mysql_local*'. To build an image from Dockerfile, we have to use 'docker build' as shown below:

```
<code: shell>
$ docker build -t mysql_local .
Sending build context to Docker daemon 2.048 kB
Step 1: FROM mysql
latest: Pulling from library/mysql

386a066cd84a: Pull complete
827c8d62b332: Pull complete
de135f87677c: Pull complete
05822f26ca6e: Pull complete
63ddbddf6165: Pull complete
15fe0fbc587e: Pull complete
93e74acdb291: Pull complete
11c2df82e984: Pull complete
d42a9e6a85c8: Pull complete
e7fb2f3afd87: Pull complete
30724006a583: Pull complete
Digest: sha256:89cc6ff6a7ac9916c3384e864fb04b8ee9415b572f872a
2a4cf5b909dbbca81b
Status: Downloaded newer image for mysql:latest
 ---> d9124e6c552f
Successfully built d9124e6c552f
```

You will see a bunch of 'Pull complete' messages, which is fine because we asked Docker to build 'FROM mysql' in our Dockerfile. Now the MySQL image is stored on your local machine, which means that when you try to build a new image of 'FROM mysql', it won't pull it from the Internet. The very last line says 'Successfully built d9124e6c552f' , where the ID is of the newly created image, i.e., *mysql_local*. You can list all the images on your machines with 'docker images' as shown below:

```
$ docker images
REPOSITORY    TAG       IMAGE ID      CREATED          SIZE
mysql_local   latest    d61327195850  1 minutes ago    383.4 MB
\mysql        latest    d9124e6c552f     4 days ago    383.4 MB
```

As you can see, we now have two images on our machine.

1. *mysql:* The base image.
2. *mysql_local:* The image we created with the custom root password.

Let's go ahead and set up a container, and connect to the MySQL prompt just to verify that things are working.

```
<code: shell>
$ docker run -d -p 3306:3306 mysql_local

# Check running containers
$ docker ps
CONTAINER ID   IMAGE    COMMAND     CREATED          STATUS
```

```
PORTS     NAMES
003d069f511b  mysql_local "docker-entrypoint.sh"   2 minutes
ago       Up 2 minutes        0.0.0.0:3306->3306/tcp
hopeful_torvalds

$ mysql -h127.0.0.1 -u root -proot
Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.

mysql>
mysql> exit
```

That's it! You have created an image for MySQL and started a container running MySQL server on port 3306. All we needed were two lines of code in our Dockerfile, which can now be used across operating systems with the same commands to achieve the same output. There are a few more commands that we need to know before we move forward.

Note: If you get an error while running the container, it must be because you already have MySQL running locally. Stop your local MySQL by using `*sudo service mysql stop*'.

```
<code: shell>
# Stopping a container. The 'id' is taken from 'docker ps'
$ docker stop 003d069f511b
# List all containers (running and stopped)
$ docker ps -a
# Delete a container
$ docker rm <id>
#Delete an image
$ docker rmi IMAGE
#Delete ALL containers. (Use with caution)
$ docker rm -f $(docker ps -a -q)
#Delete ALL images (Use with caution. It deletes every image
on you machine!)
$ docker rmi -f $(docker images -q)
```

You can play around with this set-up as much as you wish. Try modifying the Dockerfile to build new images with different configurations. Consider achieving this with what you have just learned.

1. Run two MySQL containers on different ports (3306 and 3307).
2. Create an image which allows a blank root password.
3. Create an image with a database named 'Books' and a user with the name 'Librarian'.

Moving on to our Rails application, let's create a Dockerfile and *docker-compose.yml* inside the application directory. Just like MySQL, you can also get base images for Ruby, and we will use 'Ruby:2.3.1' in our Dockerfile to build the image for the app. The *yml* file is the template for our environment, and it will be used to spin off dependent services in separate containers, saving us from the overhead of linking them. 'Compose' is a tool for defining and running multi-container Docker applications. With Compose, you create and start all the services from your configuration using a single command.

```
<code: shell>
$ cd ~/dock/rails-mrd-demo
$ touch Dockerfile docker-compose.yml

<code Dockerfile>
#Using ruby 2.3.1 as our base image
FROM ruby:2.3.1
#
# Installing dependencies. You can execute any linux
command using RUN
#
RUN apt-get update -qq && apt-get install -y build-
essential libpq-dev nodejs
#
#Prepare directory where the application will be copied
inside the image
RUN mkdir /myapp
#
#Change working directory (cd /myapp)
WORKDIR /myapp
#
# Copy over Gemfile and Gemfile.lock. These files have all the
ruby needs defined.
ADD Gemfile /myapp/Gemfile
ADD Gemfile.lock /myapp/Gemfile.lock
#
# Execute bundle install
RUN bundle install
#
# Copy contents from current directory to /myapp.
ADD . /myapp
```

The Dockerfile syntax is minimal and simple to understand. In this file, we started from a base Ruby image, installed OS packages, copied files, executed commands like *bundle install* and finally copied all our code inside the image. As discussed earlier, we have the option of using 'docker build', but we will use 'docker-compose build' instead. It will read the *yml* and create three containers for us, which all talk to each other by service names. This means that your *web* container can communicate with the database with *mysql* as an endpoint and the same is true for Redis. Here is the *docker-compose.yml:*

```
<code docker-compose.yml>
version: '2'
services:
  mysql:
    image: mysql
```

```
    environment:
      - MYSQL_ROOT_PASSWORD=root
  redis:
    image: redis
  web:
    build: .
    command: bundle exec rails s -p 3000 -b '0.0.0.0'
    volumes:
      - .:/myapp
    ports:
      - "3000:3000"
    depends_on:
      - mysql
      - redis
```

In the *web* section, you can see 'build'. which says,'build a container from the current working directory'; for MySQL and Redis, we need not build an image locally as we are using publicly available images. Build your environment with the 'docker-compose build' command:

```
<code:shell>
$ docker-compose build
docker-compose build
redis uses an image, skipping
mysql uses an image, skipping
Building web
Step 1: FROM ruby:2.3.1
2.3.1: Pulling from library/ruby
386a066cd84a: Already exists
75ea84187083: Pull complete
.
.
.
Step 8 : ADD . /app
 ---> 3a51a5637b94
Removing intermediate container 2f1b024e5daa
Successfully built 3a51a5637b94
```

The output makes it clear that it skips building an image for Redis and MySQL, and proceeds with *web*. As you can see, it starts with Step 1 and ends at Step 8. This is because we have specified eight commands in the Dockerfile. You will need to build whenever there is a change made to Gemfile or Dockerfile. Coming to the final step, let's run our Dockerised Rails development environment:

```
<code:shell>
$ docker-compose up -d
Pulling redis (redis:latest)...
latest: Pulling from library/redis
386a066cd84a: Already exists
769149e3a45c: Pull complete
1f43b3c0854a: Pull complete
70e928127ad8: Pull complete
9ad9c0058d76: Pull complete
bc845722f255: Pull complete
105d1e8cd76a: Pull complete
Digest: sha256:c2ce5403bddabd407c0c63f67566fcab6facef90877de58
f05587cdf244a28a8
Status: Downloaded newer image for redis:latest
Creating railsredismysqldocker_redis_1
Creating railsredismysqldocker_mysql_1
            Creating railsredismysqldocker_web_1
```

You have your application running and you can verify this by visiting *http://localhost:3000* in your browser. You might see a database error and that is because we need to set up a database for our application. Open a new terminal and run the following code:

```
<code:shell>
$ docker-compose run web rake db:create
Created database 'rails-redis-mysql-docker_development'
```

Here are a few commands that you might need while working:

```
<code:shell>
# Stop all running containers
$ docker-compose stop

# Destroy all containers, images, networks. You will have to
run rake db:create again if you do this.
$ docker-compose down

# Check logs
$ docker-compose logs web

# Validate docker-compose.yml. If there are no errors it will
display the contents of yaml.
$ docker-compose config
```

That's it for this session. If by now you have doubts like 'Why not add *rake db:create* to Dockerfile?' or 'Where is my data actually stored?' then you are thinking along the right lines. END

References

[1] *https://docs.docker.com/compose/overview/*
[2] *https://docs.docker.com/engine/getstarted/*
[3] *https://docs.docker.com/engine/tutorials/dockervolumes/*
[4] Rails demo project with Docker files: *https://gitlab.com/techmaniack/rails-mrd-demo/tree/docker*

By: Abdul Karim Memon

The author has been a Linux user for the past seven years and has also helped Ola Cabs run DevOps. He is a FOSS evangelist and prefers Emacs as his IDE. He can be reached at *abdulkarim@reversehack.in*.

OpenShift: The Fast and Friendly Platform-as-a-Service

OpenShift is a next-generation container based, secure and scalable platform for rapid application development. Let's first understand the way it works, and then use it to build a simple application.

OpenShift is a Platform-as-a-Service by Red Hat, which provides a quicker way to develop, deploy, monitor and scale applications in a secure and on-demand cloud. OpenShift has been built with only one goal, which is "To quickly build an application and make your clients happy." By using OpenShift, developers can focus on the core logic of an application rather than configuring and maintaining it.

OpenShift became famous after v2.0. Currently, it runs on v3.0, which is the next-gen PaaS that uses Docker containers and Kubernetes. In the next section, we will look more closely at all these terms. OpenShift v3.0 is different from OpenShift v2.0, so we will not go deep into v2.0 but focus more on v3.0.

Let's study some of the basic terms before trying to understand the OpenShift stack.

Virtual machines vs containers: Virtual machines contain all the binaries, libraries and application code along with the guest OS, which will be in GBs.

Containers contain the application and related dependencies. They share the same kernel with other containers, and every container runs as an isolated process in user space on the host OS.

By comparing virtual machines and containers, we can get more clarity on how containers differ from virtual machines.

Docker: A Docker container is a piece of code in a file system which contains everything needed to run it, such as libraries, application code and system tools. Basically, it provides everything that can be installed on a server. The advantage of using Docker containers is that they can run anywhere, in any cloud, and with any infrastructure.

Kubernetes: This is a tool/system that is used for automating containerised applications, and its functions include deployment, scaling, load balancing and management.

OpenShift stack

Figure 1 demonstrates the new OpenShift stack for v3.0.

- OpenShift v3 is built on RHEL 7, which has more capabilities compared to RHEL 6. Also, it is a part of the Atomic project, which was launched to enhance Linux host capabilities and optimised container based application environments.
- Red Hat is contributing to a project called libcontainer, which provides standard APIs for defining containers like network interfaces, namespaces, cGroups, etc. So OpenShift uses Docker libcontainer, which is lightweight and provides isolation through OpenShift gear, which is the main focus of developers. OpenShift v3 cartridge enables customers to leverage any application component packaged as a Docker image, which also enables users to share and access container images globally (in the market place).
- xPaaS services in OpenShift provide capabilities like JBOSS portfolio, SQL or NoSQL databases, cache, log managements, monitoring tools, message brokers and many more functionalities.

- Any application in OpenShift contains multiple gears (each gear has memory, bandwidth, CPU and disk allocated to it). We can either use single or multiple gears to create applications. Sometimes, one application is part of one gear and the other application is part of another gear, so to manage them we need an orchestrator. OpenShift broker manages orchestration and scheduling activities. In OpenShift v3 we leverage the advantage of Kubernetes, which is used along with an OpenShift broker to handle large scale cluster based applications.

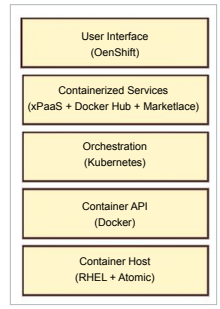


Figure 1: OpenShift v3.0 stack

- The OpenShift dashboard is user-friendly, and any kind of application development in Java, PHP, Python, Node.js or other supported languages is much easier and quicker than in other PaaS service providers.
- OpenShift has wide community support as Red Hat is actively participating in many open source projects, and OpenShift is the combined effort of all those communities.

OpenShift v3.0 architecture

Figure 2 demonstrates the overall architecture of OpenShift v3.0.

As we have seen in Figure 1, OpenShift v3.0 follows the layered approach that leverages the benefits of Docker and Kubernetes. OpenShift v3.0 focuses more on how we can combine different services and compose an application easily. For example, in v3.0 we install Python, push code and then add MySQL.

However, OpenShift v2.0 provides configuration flexibility after composing all components. In v2.0, the application is used as a separate object, which is removed in v3.0 to provide flexibility in terms of combining various services— like different containers can reuse the same databases and expose them directly to the edge of the network.

OpenShift provides the functionalities listed below:

- Code management, build management and deployment
- Application management (elasticity, scalability and load balancing)
- Managing images in scale systems
- Team and user management for any organisation

The OpenShift architecture is made up of different small components, which run on top of the Kubernetes cluster where data about all the objects is stored in *etcd* (it's a master state—other components will look for this state and if any changes occur, then those components will change themselves into the desired state). It is also configured for high availability and deployed with 2n+1 peer services. REST APIs are used for communication between core objects. The controller reads those APIs and makes changes accordingly, to different objects. For example, when the

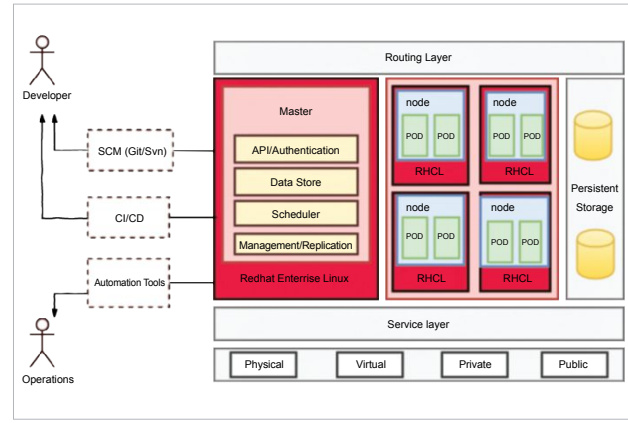


Figure 2: OpenShift v3.0 architecture

request comes for build from the user side, a build object is created. The build controller then sees this object and runs a process to perform that build. Once the build is completed, the controller updates the status via the REST API to the build object, and the user can see that the build is over.

OpenShift follows the controller pattern, which provides flexibility and extensibility. We can define the behaviour of the controller whenever any object is created. Basically, it's business logic—it takes the controller and can also leverage the benefits of APIs to do any kind of automation and optimisation using scripting languages. We can also schedule cron jobs for any specific administrative tasks.

The controller should be in sync with the system and users, which happens due to even streams that push data from *etcd* to REST API and controller, so that the controller can quickly perform the required actions. In a failure scenario, the controller should resync all the data and resume the tasks accordingly.

OpenShift uses OAuth tokens and SSL certification for authorisation. A client typically makes API calls using a Web console or client program, and uses OAuth tokens for all the communications. Infrastructure (nodes) validation is done by client certificates generated by the systems that own the identities. Components inside containers use tokens to connect the APIs.

The OpenShift policy engine handles the authorisation part by creating different policies for a set of users and groups. Whenever any service account or user tries to perform an action, the engine checks for its identity before approving it.

Every container associates with the service account in the cluster. We can associate secrets (this is a service that handles all the passwords) with the service account, and automatically deliver these into the container. This provides the infrastructure the flexibility to manage secrets and perform different actions like pushing images, and also allows the application code to benefit from the secrets.

OpenShift flavours

OpenShift currently comes in three flavours, as indicated in the table below.

OpenShift Online	OpenShift Dedicated	OpenShift Container Platform (OpenShift Enterprise)
Multi-tenant	Single-tenant	Single/Multi-tenant
Cloud-based container platform	Cloud-based container platform	Container platform software
Managed by Red Hat	Managed by Red Hat	Managed by customers. They can customise it according to their requirements.
		OpenShift Container is divided into two parts — Local and Lab. Local can be used by developers for testing and Lab can be used by the ops/testing teams for POC/pre-production use cases.

Pricing

There are three plans available for OpenShift (see Table 1).

1. *Bronze Plan:* Free + pay as per use + private SSL support, team management, no application idling.
2. *Silver Plan:* Pay as per use + phone/ticket based support for US$ 20/month + 5GB extra storage for every gear.
3. *Developer Preview Plan:* Free plan available for preview without any extra support.

* All plans include three small basic gears for free.

** In the Silver plan, each gear includes 6GB of space.

** Bronze and Silver plan users can add up to 30GB of additional storage to individual gears for US$ 1.00/GB/month.

In India, we can adopt free plans, as these are accessible worldwide, but Silver and Bronze plans are available only in North America and Europe.

For more details, visit the OpenShift pricing page which covers all the details mentioned above.

Deploy an application using OpenShift Online

Let's use the developer preview plan to deploy the application using OpenShift.

Table 1

Basic gears	***CPU***	***Memory***	***Storage***	***Region***	***Usage***
Small	1x	512MB	1GB**	US only	US$ 0.02/ hour*
Production gears	***CPU***	***Memory***	***Storage***	***Region***	***Usage***
Small.*highcpu*	2x	512MB	1GB**	US and Europe	US$ 0.025/ hour
Medium	2x	1GB	1GB**	US and Europe	US$ 0.05/ hour
Large	4x	2GB	1GB**	US and Europe	US$ 0.10/ hour

Source: *https://www.openshift.com/pricing/index.html*

A long time back, I created an account on OpenShift v2.0, which had a free quota available for more than a year. After August 1, 2016, the OpenShift Community is not accepting new registrations for OpenShift v2.0. They strongly recommend that new users start with OpenShift v3.0.

The OpenShift v3.0 developer preview plan has only a 30-day trial period and once it expires, all your data is deleted; so if you are using a preview plan, then please migrate the data before the trial period ends. As I have already created an account in OpenShift v2.0 long back, my application is still running, but once OpenShift v3.0 matures, all OpenShift v2.0 users will have to migrate their applications to OpenShift v3.0.

For the OpenShift v3.0 developer preview plan too, you need to wait for approval, as currently very limited resources are available. Based on the availability, you will be sent a mail and you will get access for 30 days.

Here is the step-by-step approach to creating an app in OpenShift Online.

Go to *https://www.openshift.com/* and sign up. You will be asked for your GitHub account. If you don't have one, create it first and then log in with this account.

Now you will be asked to authorise the application. Please do so.

Once this is done, check whether your request has been approved. I am getting messages saying that currently all resources have been allocated so I have to wait for my turn.

As I have already created an account in OpenShift v2.0, we will use it to see how we can leverage the benefits of OpenShift. Users who have already created an account before August 1, 2016, can follow the procedure shown below and deploy an application.

Go to *https://www.openshift.com/* and then to *My Account*, and select the *OpenShift Web Console* in the Openshift v2.0 platform or *New Gen Web Console* in v3.0.

You will see the application page, where all the running apps are listed. I have already created apps for testing.

Click on *Add application*, and you can see a list of cartridges (managed runtime for applications) supported by OpenShift v2.0.

Based on the application's requirements, select the appropriate cartridges. Currently, I have chosen WordPress, as I want to create Web applications with it.

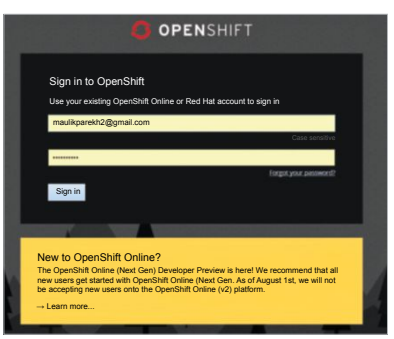


Figure 3: Login with OpenShift v2.0

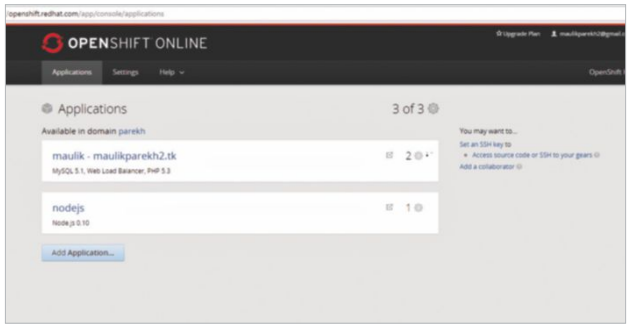


Figure 4: Applications in OpenShift v2.0

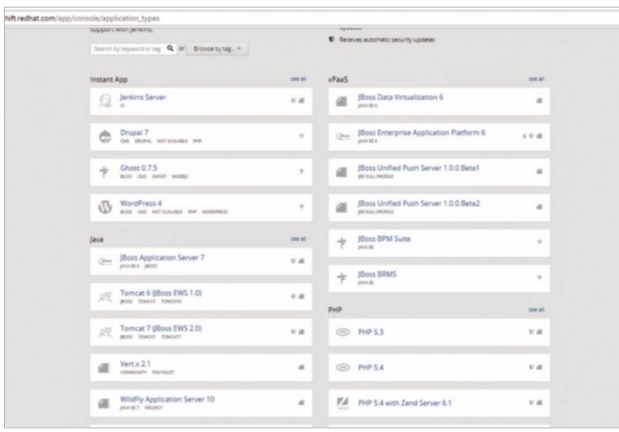


Figure 5: List of cartridges supported by OpenShift

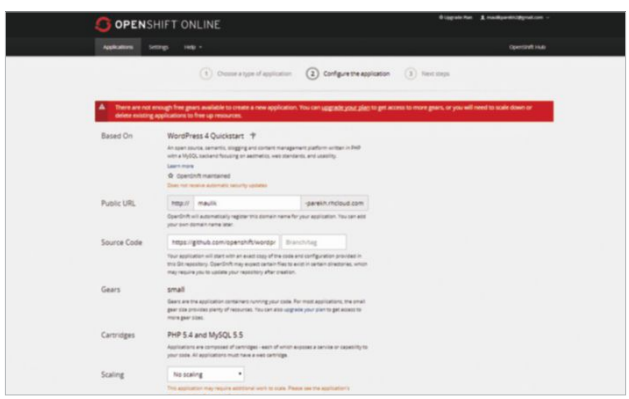


Figure 6: The WordPress application page

Provide the domain name for the public URL. OpenShift will automatically register this domain name as a public URL. Leave all other options as default, and click on *Create application*.

Once the application is created, it will give you two options to change its code—either now or later. Choose *Later* and continue.

Now the application is ready with MySQL 5.5 and PHP 5.4. The application's URL in my case is *maulik-parekh@rhcloud.com*.

Click on this URL and you can see that the WordPress installation has started. Enter the application's details and click on *Install WordPress*.

Figure 7: Change the code of the application

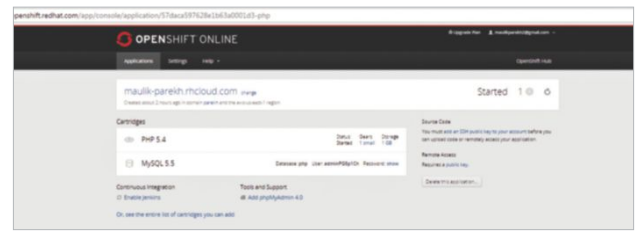


Figure 8: Application deployed successfully

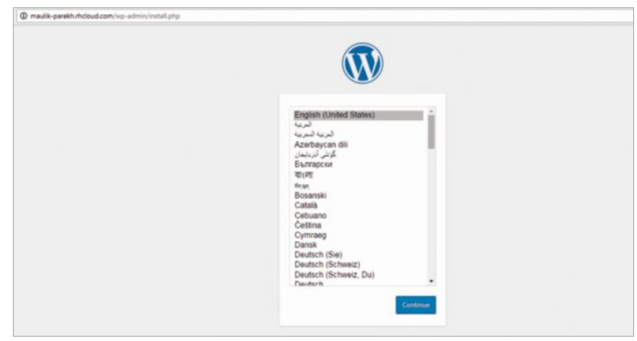


Figure 9: WordPress installation started

You will see an *Installation successful* message, so click on *Login* and start using WordPress. You can choose any theme and activate it. Modify your contents accordingly. So this is how we can deploy a cloud based application on OpenShift v2.0 using WordPress.

WordPress is easy to learn and people who don't know how to create a website on it can search the Internet—there is a lot of material available on the topic.

In later articles, we will look at how we can deploy Java, Python and Node.js based applications using OpenShift. Since, for v3, the free trial is limited to 30 days (which was not the case with OpenShift v2.0), you could also pay to enjoy the benefits of a next-gen container based PaaS. END

References

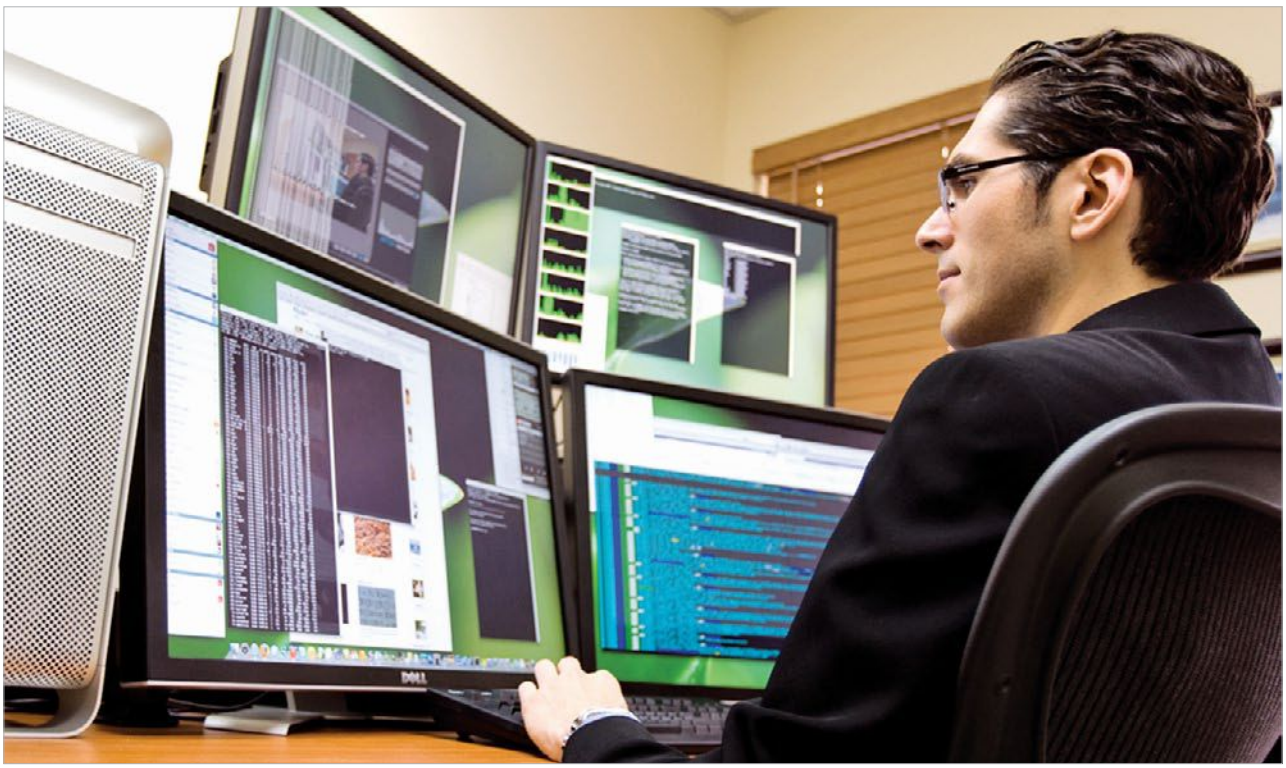
[1] *https://www.openshift.com*
[2] *https://en.wikipedia.org/wiki/OpenShift*

By: Maulik Parekh

The author has an M. Tech degree in cloud computing from VIT University, Chennai. He can be reached at *maulikparekh2@gmail.com*. Website: *https://www.linkedin.com/in/maulikparekh2*.

Linux System Administration: Managing Users and Groups

This is an article that takes the reader back to the basics of Linux. It covers the various aspects of users and groups in Linux, like adding or removing them, giving them passwords, etc—all from a systems administrator's point of view.



Linux is a multi-user operating system, which means that more than one user can use Linux at the same time. Linux provides a beautiful mechanism to manage users in a system. One of the most important roles of a system administrator is to manage the users and groups in a system. All the commands used in this article are explained using the CentOS Linux distro.

Linux user

A user or account of a system is uniquely identified by a numerical number called the UID (unique identification number). There are two types of users – the root or super user and normal users. A root or super user can access all the files, while the normal user has limited access to files. A super user can add, delete and modify a user account. The full account information is stored in the */etc/passwd* file and a hash password is stored in the file */etc/shadow*. Some operations on a user account are discussed below.

Creating a user with a default setting: A user can be added by running the *useradd* command at the command prompt. After creating the user, set a password using the *passwd* utility, as follows:

```
[root@localhost bhargab]# useradd anirban
[root@localhost bhargab]# passwd anirban
Changing password for user anirban.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

The system automatically assigns a UID, creates the home directory (*/home/<username>*) and sets the default shell to */bin/bash*. The *useradd* command creates a user private group whenever a new user is added to the system and names the group after the user.

Specifying a user's full name when creating a user: A systems administrator can use the *–c* option with *useradd* to specify the user's full name, as shown below:

```
[root@localhost bhargab]# useradd -c "Anirban Choudhury" anirban
```

Creating a user with the UID: You can create a user with a custom UID with the *–u* option, as follows:

```
[root@localhost bhargab]# useradd -u 1036 anirban
```

Creating a user with non-default home directory: A non-default home directory can be set by executing the following command:

```
[root@localhost bhargab]# useradd -d /home/test anirban
```

Adding a user to a primary group and supplementary group: A systems administrator can specify a primary group and a supplementary one by specifying the *–g* and *–G* option, respectively.

```
[root@localhost bhargab]# useradd -g "head" -G "faculty"
anirban
```

Locking and unlocking a user: A super user can lock and unlock a user account. To lock an account, one needs to invoke *passwd* with the *-l* option.

```
[root@localhost bhargab]# passwd -l anirban
Locking password for user anirban.
passwd: Success
```

The *–u* option with *passwd* unlock an account, as shown below:

```
[root@localhost bhargab]# passwd -u anirban
Unlocking password for user anirban.
passwd: Success
```

Changing a user name: The *–l* option with the *usermod* command changes the login (user) name, as shown below:

```
[root@localhost bhargab]# usermod -l "nishant" anirban
```

Removing a user: Combining *userdel* with the *–r* option drop a user and the home directory associated with that user, as shown below:

```
[root@localhost bhargab]# userdel -r nishant
```

Linux group

Linux group is a mechanism to organise a collection of users. Like the user ID, each group is also associated with a unique ID called the GID (group ID). There are two types of groups – a primary group and a supplementary group. Each user is a member of a primary group and of zero or 'more than zero' supplementary groups. The group information is stored in */etc/group* and the respective passwords are stored in the */etc/gshadow* file. Some operations such as creating, deleting and modifying on a group are discussed below.

Creating a group with default settings: To add a new group with default settings, run the *groupadd* command as a root user, as shown below:

```
[root@localhost bhargab]# groupadd employee
```

If you wish to add a password, then type *gpasswd* with the group name, as follow:

```
[root@localhost bhargab]# gpasswd employee
Changing the password for group employee
New Password:
Re-enter new password:
```

Creating a group with a specified GID: To explicitly specify the GID of a group, execute the *groupadd* command with the *–g* option, as follow:

```
[root@localhost bhargab]# groupadd -g 1200 manager
```

Removing group password: To remove a group password, run *gpasswd –r* with the relevant group name, as follow:

```
[root@localhost bhargab]# gpasswd -r employee
```

Changing the group's name: To change the *group's* name, run the *groupmod* command with the *-n* option as a super user, as shown below:

```
[root@localhost bhargab]# groupmod -n hrmanager employee
```

Changing the group's GID: To change the GID of a group, run the *groupmod* command with *–g*, as follow:

```
[root@localhost bhargab]# groupmod -g 1050 manager
```

Deleting a group: Before deleting a primary group, delete the users of that primary group. To delete a group, run the *groupdel* command with the group name, as shown below:

```
[root@localhost bhargab]# groupdel employee
```

If you wish to know much more about the user and group management, you can refer to the Red Hat system administration and manual page of each command. END

By: Bhargab Choudhury

The author is interested in information security and systems administration. He can be reached at *bhargabchoudhury24@gmail.com*.

Puppet: The Popular Choice for IT Automation

Puppet is used to configure UNIX-like computer systems as well as Windows systems. It uses its own declarative language or a domain specific language. Let's get to know this tool a bit better.

Puppet is an IT configuration management tool and key software in the IT automation arena. In Puppet, you can define your infrastructure configurations and application configurations. It gives you good visibility of what has gone wrong and, by reporting all tasks, will help you to record all the changes. By using Puppet, systems administrators can reduce the time spent on repetitive tasks, ensuring consistent configuration across their infrastructure.

Puppet is an open source tool developed by Puppet Labs. It can be used on Linux (most flavours), Solaris, AIX and Windows platforms.

Puppet comes in two versions—the open source community version and the enterprise paid-for version. Even on a paid version, you get a one-month trial with a learning VM from Puppet Labs.

Before installing Puppet, let's compare it with another similar tool called Chef on the affordability parameter.

1. Chef uses Ruby, which is a difficult language to learn for beginners, while Puppet uses DSL (domain specific language), which is optimised for the task of describing resources.
2. Puppet has a larger installed base, as compared to Chef.
3. Puppet has a large developer community.
4. It supports more platforms than Chef.
5. Puppet is declarative while Chef is not.

Puppet being a declarative language means that *you* are writing the code, and you decide what the status of your machine should be after running the code and what not to do. The program will decide what to do to achieve the target state of the server.

We will now discover how to install Puppet and use it. For testing purposes, you can either download the learning VM from the Puppet Labs site, or you can download Puppet and install it on a machine.

Prerequisites for installing Puppet

1. Ensure the server and client names are fully resolvable.
2. Configure the NTP service and ensure that it is running.
3. Download the Puppet package (*wget https://github.com/puppetlabs/puppet*).
4. Install the Puppet package (create a repo with the folder and run *yum install puppet*).
5. Install Apache (*yum install httpd*) using the following command:

```
#Install puppetmaster-passenger
```

6. Install the package passenger (*yum install puppetmaster-passenger*).
7. Start Puppet (*service puppetserver start*).
8. Make Puppet start when the server boots (*chkconfig on puppetserver*), or download the learning VM from Puppet Labs.
9. Install the Puppet agent on all client servers.

Puppet uses SSL certificates to authenticate communication between master and agent nodes. The Puppet

INDIA ELECTRONICS WEEK

March 2-4, 2017. BIEC. Bengaluru

6 co-located shows.

Electronics ForYou expo
MAKE. BUY. SELL. INVEST.

IoT show.in

LEDASIA.in

electronics Rocks

TEST & Measurement India

raksha India

www.indiaelectronicsweek.com

An EFY GROUP EVENT

For Those Who Value Technology

Partners of India Electronics Week 2017

CSA Group | DRIVE TECHNOLOGIES | ANSYS | messung erfi intelligence works | maxim integrated

emst Marketing Pvt. Ltd. | PHYTEC | HumiSeal | cirkit ELECTRO-COMPONENTS PVT. LTD. | MOUSER ELECTRONICS

Ocean Since 1990 | MetroQ | technosphere labs | GE GALA ELECTRONICS Dealers in Industrial Electronic Components | SEMIKART www.semikart.com

For details, call us on +91-11-40596600 or write to growmybiz@efy.in

master acts as a certificate authority, and must generate its own certificates which are used to sign agent certificate requests for authentication.

When you run the Puppet agent the first time, it will generate an SSL certificate and send a signing request to the Puppet master. The client can communicate and be controlled by the Puppet master server only after the agent's certificate is signed. All the agent certificates must be signed by the master server.

Before creating a certificate, we have to edit the */etc/puppet/puppet.conf* file and append the following lines in the [main] section:

```
certname = puppet
dns_alt_names = puppet.master.com
```

To generate a certificate on the master, use the command shown below:

```
#puppet master  --no-daemonize
```

To list all unsigned certificate requests, use the following command:

```
#puppet cert list
```

Now the master is ready. Let's now install agents on client servers. After you download the Puppet agent, install it as follows:

```
#yum install puppet
```

To install the Puppet agent, use the following command:

```
#vi /etc/default/puppet
   modify  START=yes
```

To add the master server entry in the client, use the command given below:

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

Delete the [master] section. Add the following line in the [agent] section:

```
Server = puppet.master.com
```

Start the Puppet agent, as follows:

```
#service puppet start
```

Once you start the agent, it will create and send the certificate to the master server. Once the certificate is authenticated, the client can communicate with the master. So log in to the master server to authenticate the client certification.

To list the certificates received, run the following commands from the master server:

```
#puppet cert list   This will list the Agent FQDN name
#puppet cert sign <FQDN>
```

In case you have many certificates from multiple clients, you can use 'all' instead of <*FQDN*> to certify all clients. After signing, the client can be controlled by the master server. In any case, if you want to remove the controls from the master, you can delete the certificate by using the following code:

```
#puppet cert clean <FQDN>
```

Repeat the above steps to set up the agent for all client servers and once done, we are good to go—to test Puppet.

How Puppet works

Once you have installed the agent, Puppet collects information about each client by using Facter.

If you run the *facter* command in an agent, you will get an output similar to the one shown below.

```
#facter
<output>
login as: root
root@192.168.0.102's password:
Last login: Wed Sep  7 14:25:50 2016
aio_agent_version => 1.5.2
augeas => {
  version => "1.4.0"
}
disks => {
  sda => {
    model => "VBOX HARDDISK",
    size => "20.00 GiB",
    size_bytes => 21474836480,
    vendor => "ATA"
  }
}
dmi => {
  bios => {
    release_date => "12/01/2006",
<Truncated>
```

By default, the Puppet agent connects with the master server every 30 minutes and shares the status information with the master. In parallel, it will pull the data about the desired status of defined services from the master and take appropriate action to meet the desired status.

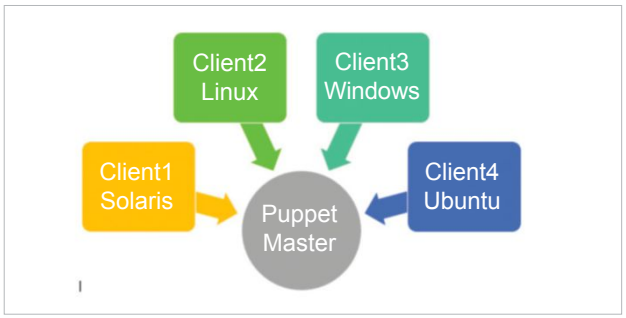


Figure 1: Puppet's communication with client servers

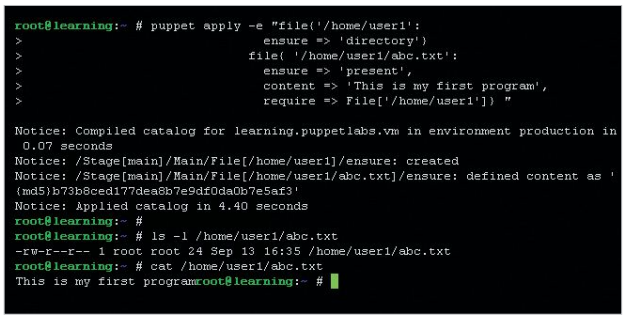


Figure 2: Example of a Puppet program

If you want to run it manually, you can run the command given below from the client server.

```
#puppet agent -test
```

So now, we have confirmed that Puppet is running fine and the master server is able to communicate with the client. Next, we will consider a few important files and folders.

The configuration file for open source Puppet is */etc/puppet/puppet.conf*.

This *puppet.conf* file is divided into three parts, which are:

Main: Global configuration settings should be placed here.

Agent/user: This is to apply configurations to client servers.

Master: These are configurations of the master server.

To list all the configurations by the command line, use the following code:

```
#puppet config print all
/var/log/puppet ==> contains puppet logs on both master and
client servers
/var/lib/puppet ==> contains Puppet operational data
/var/lib/puppet/ssl ==> contains SSL certificate
/etc/puppet/modules and /usr/share/puppet/modules ==> The
default directories where modules are searched in master
server
```

Let us create a simple file in */home/user1/abc.txt* which contains "this is my first program" using Puppet.

In the screenshot shown in Figure 2, the *ensure* command checks the availability of the file/folder and, in case it is absent, it will create the respective file or folder. So it makes your job easy, because some apps may fail due to the absence of log files.

The directory attribute ensures */home/user1* is a folder and is available. The *ensure* file attribute ensures the availability of a file; otherwise, it will create a file with the mentioned content.

By default, Puppet has many resource types like files, packages, services, file systems, etc. With reference to Figure 2, the file is the resource type and */home/user1/abc.txt* is the attribute. The power of Puppet is that we can create our own resource types.

To view the resource types available, use the following code:

```
#puppet resource -types
computer
cron
docker_compose
docker_network
exec
file
file_line
group
host
interface
maillist
mount
nagios_command
<Truncated>
```

Puppet also has modules like pluggable libraries in applications. These modules are groups of manifests and data such as facts, files and templates. Modules are designed with different components per system. Readymade manifests are available in different scopes such as run instance, image management, repository management and service management.

As an example, the directory base for the Apache module is:

```
<MODULE_PATH>httpd/
```

A directory in this Apache module that serves static files is:

```
<MODULE_PATH>/httpd/files
```

A static configuration file for *httpd* is shown below:

```
<MODULE_PATH>/httpd/files/myhttpd.conf
AccessFileName .acl
```

A directory to hold your manifests in the module is shown below:

```
<MODULE_PATH>/httpd/manifests/
```

A complete solution manifest is:

```
<MODULE_PATH>/httpd/manifests/init.pp
class httpd{
  include httpd::install
  include httpd::config
  include httpd::service
}
```

A manifest that just installs *httpd* is given below:

```
<MODULE_PATH>/httpd/manifests/install.pp
class httpd::install {  package {'httpd': ensure =>
'installed'}
}
```

A manifest that just configures *httpd* is given below:

```
<MODULE_PATH>/httpd/manifests/config.pp
class httpd::config{  file {'/etc/httpd/conf.d/httpd.conf':
    ensure => 'present',
    source => 'puppet:///modules/httpd/myhttpd.conf'
  }
}
```

A manifest that just handles *httpd* service is as follows:

```
<MODULE_PATH>/httpd/manifests/service.pp
class httpd::service{
  service{'httpd': ensure => 'running' }
}
```

Now, using it, we give the following command:

```
$ puppet apply --modulepath=<MODULE_PATH>  -e "include
httpd"
```

This will custom-configure, install and start the *httpd* service.

Apart from modules, we can also use classes and definitions in Puppet. Please refer to Figure 3 for the example for classes. Figure 4 gives the example of a definition with class.

Now, let's look at how to assign the resources and collections to client servers.

Puppet uses this file as a configuration file to determine which action should go to which client server. On client1 we can run the class *httpd* which contains

```
root@learning:~ # cat apa.install
class apache {
        package { 'httpd':
           ensure => 'present',
                }
        service {'httpd':
           ensure => 'running',
           require => Package["httpd"],
           }
}
include apache
root@learning:~ # puppet apply apa.install
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0
 seconds
Notice: Applied catalog in 35.22 seconds
root@learning:~ # rpm -qa | grep -i http
http-parser-devel-2.0-5.20121128gitcd01361.el7.x86_64
nodejs-http-signature-0.10.0-3.el7.noarch
httpd-2.4.6-40.el7.centos.1.x86_64
```

Figure 3: Example of a Puppet class

```
root@learning:~ # cat aa.txt
class testdefine {
    define testdefine ($data) {
      file {"$title":
        ensure  => file,
        content => $data,
      }
    }

    testdefine ('/var/testfile1':
      data => "The name of the file is puppetfile1 and it is created by puppet\n",
    )

    testdefine ('/var/testfile2':
      data => "The name of the file is puppetfile2 and it is created by puppet\n",
    )
    testdefine ('/var/testfile3':
      data => "The name of the file is puppetfile3 and it is created by puppet\n",
  }
}

include testdefine
root@learning:~ # puppet apply aa.txt
```

Figure 4: Example of a definition with class

the *httpd* service and package. On client2, it will run the Samba package and smb service.

The file *clients.pp* will look like what's shown below:

```
node 'client1.test.com' {
include httpd_class
}
node 'client2.test.com' {
include samba_class
}
node default {
    package { "perl":
       ensure => present }
}
```

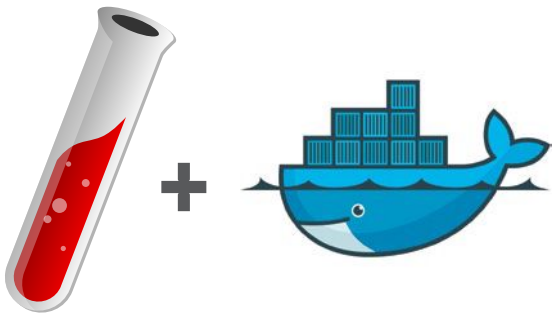
In this article I have taken examples from many sites, especially Puppet Labs. There are many URLs from which we can get more examples of Puppet automation. Puppet is, indeed, a great tool to automate IT tasks. END

By: Ramasamy Panchavarnam

The author is a cloud architect at Sify. He has been in the IT industry since 1997, and is proficient in UNIX and cloud technologies. He is passionate about researching and testing open source tools, and can be reached at *sparcrams@yahoo.co.in*.

Deploying a Jekyll Blog in Docker

Jekyll is a simple, blog-aware, static site generator. It takes a template directory containing raw text files in various formats, runs it through a converter and spits out a complete, ready-to-publish static website. This article integrates Jekyll with Docker.



Jekyll is a blog alternative that generates static HTML from templates that you create. It is a simple, blog-aware, static site generator written on Ruby by Tom Preston Werner, GitHub's co-founder. It takes a template directory (representing the raw form of a website), runs it through Markdown and Liquid converters, and spits out a complete static website.

Why does one need to use Jekyll?

No server-side language or database: This is only good old HTML/CSS/JS. Frankly, I don't want to have anything to do with a database unless I absolutely need to. This also means it's worry-free.

Simpler workflow: One only needs a text editor and Git to update the site or release a blog post. There's no need for a local PHP server or anything. Plus, synchronising the local environment with the one in production takes no more than a single command.

Fewer dependencies: No more *jQuery.paginate* for pagination; Jekyll has a built-in plugin to do it. No more *Prism.js* for syntax highlighting; Jekyll comes with Pygments, a Python based syntax highlighter. Less JS (and especially no more jQuery) means a faster site.

Enough of asking why; let's jump to how to go about things!

Creating the directory structure

The final structure will look something like what's shown below:

```
├── app
│   ├── _drafts
│   ├── _includes
│   ├── _layouts
│   │   ├── default.html
│   │   └── post.html
│   ├── _posts
│   ├── css
│   ├── images
│   ├── index.html
│   ├── js
│   └── robots.txt
├── _config.yml
├── Dockerfile
├── Gemfile
└── web
```

Anything that you put into the app/directory will get copied to the generated site too, so put your css, images, js files here.

Let's have a look at the Jekyll config file *_config.yml:*

```
source: app
destination: web

url: http://blog.your-domain.com
permalink: pretty

encoding: utf-8
```

Fill in your domain in the 'url' parameter.

Next, have a look at the views; first up is *app/_layouts/default.html:*

```
<!DOCTYPE html>
<html lang="en" prefix="og: http://ogp.me/ns#">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="/css/style.
css">

    <meta name="viewport" content="width=device-width,
initial-scale=1">

    <title>{% if page.title %}{{ page.title }} &mdash; {%
endif %}Your Blog</title>
```

```
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
    <meta property="og:url" content="{{ site.url }}{{ page.
url | remove_first:'index.html' }}">
    <meta property="og:site_name" content="blog.your-domain.
com">

    {% if page.title %}
    <meta property="og:title" content="{{ page.title }}">
    {% endif %}

    {% if page.description %}
    <meta name="description" content="{{ page.description
}}">
    <meta name="og:description" content="{{ page.description
}}">
    {% else if page.excerpt %}
    <meta name="description" content="{{ page.excerpt |
strip_html | truncatewords: 25 }}">
    <meta name="og:description" content="{{ page.excerpt |
strip_html | truncatewords: 25 }}">
    {% endif %}

    {% if page.og_image_url %}
    <meta property="og:image" content="{{ page.og_image_url
}}">
    {% else if page.photo_url %}
    <meta property="og:image" content="{{ page.photo_url }}">
    {% endif %}

    {% if page.keywords %}
    <meta name="keywords" content="{{ page.keywords }}" />
    {% endif %}

    {% if page.date %}
    <meta property="og:type" content="article">
    <meta property="article:published_time" content="{{ page.
date | date: "%Y-%m-%d" }}">
    {% endif %}

    <script src="//code.jquery.com/jquery-2.1.1.min.js"></
script>

</head>
<body>

<nav class="navbar navbar-default navbar-static-top">
    <div class="container">
        <!-- Brand and toggle get grouped for better mobile
display-->
    <div class="navbar-header">
            <button type="button" class="navbar-toggle
collapsed" data-toggle="collapse" data-target="#bs-example-
navbar-collapse-1" aria-expanded="false">
                <span class="sr-only">Toggle navigation</
span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="/">Your Blog</a>
        </div>
    </div> <!-- /.container -->
</nav>

<div class="container">

    {{ content }}

</div>

</body>
</html>
```

Most of the other views will be generated from this, which is not at all complicated. You can change it according to your need.

Next up is *app/_layouts/post.html:*

```
---
layout: default
---

<article>
    <header>
        <h1>{{ page.title }}</h1>
        <div class="post-info">
            <div class="author-published">
                by {{ page.author }}
            </div>
            <div class="date-published">
                <time datetime="{{ page.date }}">{{ page.date
| date: '%B %d, %Y' }} </time>
            </div>
        </div>
    </header>

    <div>{{ content }} </div>

</article>
```

Simple stuff, isn't it? Please notice, we are using Jekyll front matter to add some variables that apply only to this template. The front matter is where Jekyll starts to get really cool. Any file that contains a YAML front matter block will be processed by Jekyll as a special file. The front matter must be the first thing in the file, and must take the form of valid YAML set between triple-dashed lines.

Finally, the last template in our simple blog will be the home page. It's in a different location because it's the home page and so it should be at *app/index.html*:

```
---
layout: default
description: Your Beautiful Blog
---

<h1> Recent Posts </h1>
{% for post in site.posts limit:50 %}
<h2> <a href="{{ post.url }}">{{ post.title }}</a> </h2>
<div class="preview">
    {% if post.description %}
    <span class="body">{{ post.description | strip_html |
truncatewords: 300 }}</span>
    {% else %}
    <span class="body">{{ post.excerpt | strip_html }}</span>
    {% endif %}
</div>
{% endfor %}
```

On the home page, we'll just list the 50 most recent articles with an excerpt from each one. You may have noticed that we are again extending from the default layout.

We're nearly ready to generate the site but need one more thing...

Using Gems plugins

Remember we mentioned plugins? We'll configure them now. Create a Gemfile and put this in it, as shown below:

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

group :jekyll_plugins do
    gem 'jekyll', '~>3.0'
    gem 'kramdown'
    gem 'rdiscount'
    gem 'jekyll-sitemap'
    gem 'jekyll-redirect-from'
end
```

We'll install these Gems later when we generate the site. At this point, you might want to write a few words of your first blog post and put it into *app/_posts/*. Let's generate our site, now!

We're going to use a Docker container with Jekyll already installed on it. You can of course install Jekyll locally, but using a Docker container makes it more portable; for example, you might decide to build your blog continuously later on, by containing your tools inside Docker containers. You don't have to install them on every server you want to build the blog on.

Run this to install the Gems and build your shiny new blog, as follows:

```
$ docker run --rm \
        -v "$(pwd):/src" \
        -w /src \
        ruby:2.3 \
        sh -c 'bundle install \
            --path vendor/bundle \
            && exec jekyll build --watch'
```

We've added the *–watch* flag at the end, which is handy when we're making changes and we want to see them reflected immediately on the blog.

Voila! Have a look in the Web/folder—you should see lots of HTML files, which are what Jekyll generated. If you were to FTP that entire Web/folder to a server, you would have a working blog, but we're going to put it into a Docker container.

The Dockerfile

Our container will be super simple. We just need to serve the Web/folder that we just generated. Here's our Dockerfile:

```
FROM nginx
EXPOSE 80
COPY web/ /usr/share/nginx/html
```

That's it! Now, let's build and run the image:

```
$ docker build -t my-shiny-blog .
$ docker run -d \
        -p 80:80 \
        -v "$(pwd)/web:/usr/share/nginx/html" \
        my-shiny-blog
```

Now hit 127.0.0.1:8080 in your browser to see your blog in action!

Notice that we've mounted the source into the container as a volume, which will allow us to see updates in real-time when Jekyll regenerates the site (since Jekyll is still running with '–watch'), without having to rebuild the image again. Before pushing the image to your registry, just remember to rebuild!

That's it! We can of course get fancy and minimise Javascript or compile less to css using Gulp, but we'll leave that as an exercise for the reader. The trick is to put less source code outside of the app/directory and have Gulp place the final versions in the app/directory. That way, Jekyll will copy them over when the site is generated. END

References

[1] *https://jekyllrb.com/*
[2] *https://github.com/jekyll/docker*

By: Srijan Agarwal

The author is a developer at *WikiToLearn,* an open source enthusiast and works with JS and PHP mostly. You can contact him at *www.srijanagarwal.me*.

Docker: A Favourite in the DevOps World

Docker is the world's leading containerisation platform designed to make it easier to create, deploy and run various applications by using containers. Docker containers wrap up a piece of software into a complete file system that contains everything the software needs to run.

Have you ever developed a snippet of code in Java on your local system and then tried to run it on some other desktop with a different environment? You were probably blocked with a configuration error or exception. If the snippet of code works fine on your system but doesn't on the system with a different configuration, then you need to make changes to the latter's configuration in order to make the code run. Sometimes, this gets quite tedious and infuriating.

Containers actually put a full stop to all such problems. They help us run any software reliably when moved from one computing environment to another, whether from a developer's desktop to a test environment or into production, and perhaps even from a physical machine in a data centre to any virtual machine in a public or private cloud. The real need for software containers arises when the supporting environment on which the software is actually developed is not identical to the one where it's implemented. For instance, if you test code using Python 2.7 and then it's made to run on Python 3 in production, something weird happens. There are cases when we rely on the behaviour of a certain version of a library and another updated version gets installed automatically, resulting in the software running on that system behaving unusually. Not only different software, configurations or a library, but even a different network topology or different security policies and storage can prevent software from running successfully.

A container basically consists of an entire runtime environment, which includes an application with all its dependencies, like libraries and other binaries, and even the configuration files needed to run it, all bundled into one package. Hence, by containerising the differences in operating system distributions, the underlying infrastructure is abstracted away. Containers encapsulate all the discrete components of application logic which are provisioned, that too with the minimal resources needed to do their job.

Containers are different from virtual machines

Unlike virtual machines (VMs), containers have no need for embedded operating systems; calls are made for operating

system resources via an Application Programming Interface. By contrast, a server which is running three containerised applications, as with Docker, runs just a single operating system, and each of the containers shares that operating system kernel with the other containers. We need to note that the shared parts of the operating system are read only, while each of the containers has its own way to access the operating system for writing. This means that containers are much more lightweight and use far fewer resources than virtual machines. A container may be only a few megabytes in size, whereas a virtual machine with its own entire operating system may be several gigabytes in size. This is why a single server can host far more containers than virtual machines.

1. Containers are easily packaged and designed to run anywhere. There can be multiple containers deployed in a single VM.
2. Also, it may take several minutes for virtual machines to boot up their operating systems and then begin running the applications they host, whereas containerised applications can be started almost instantly.

Docker: An open source software containerisation platform

Docker is an open source software containerisation platform designed to make it easier to create, deploy and run various applications by using containers. According to the official Web page of Docker, its containers wrap up a piece of software into a complete file system that contains everything it needs to run: runtime, code, system tools or system libraries – anything we can install on a server. This ensures that it will always run the same, irrespective of the environment it is running in.

Docker provides an additional layer of abstraction and even automation of virtualisation at the operating system level on Linux. It actually uses different resource isolation features of the Linux kernel, like cgroups and kernel namespaces, and also a union-capable file system such as OverlayFS to allow independent 'containers' to run within a single Linux instance. This helps in avoiding the overhead of starting and then maintaining virtual machines, which significantly boosts its performance and also reduces the size of the application. The support of the Linux kernel for namespaces isolates an application's view of the operating environment—including process trees, user IDs, network and mounted file systems —while the kernel's cgroups helps in limiting resources, including the memory, CPU, block I/O and network.

As various actions are done to a Docker base image, different union file system layers are created as well as documented, such that each layer fully describes the way to recreate an action. This strategy enables Docker's lightweight images, since only layer updates need to be propagated in this. It implements a high-level API to provide lightweight

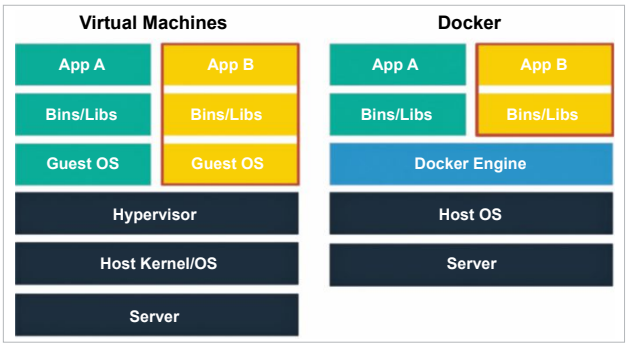


Figure 1: Architecture of virtual machines vs Dockers (Image credits: Google images)

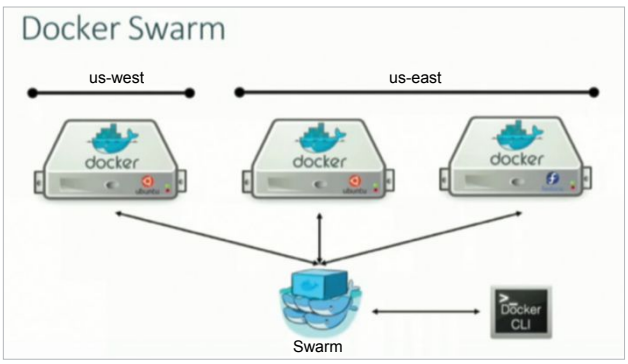


Figure 2: A Docker swarm (Image credits: Google images)

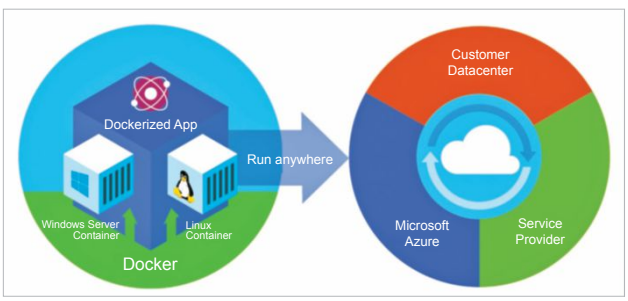


Figure 3: Microsoft Azure support for Docker (Image credits: Google images)

containers that run different processes in isolation. Docker helps enable portability and flexibility wherever the application can run – whether on-premise, the private cloud, the public cloud, bare metal, etc. Docker accesses the different virtualisation features of the Linux kernel either directly, using the libcontainer library available in Docker 0.9, or indirectly via libvirt and LXC (Linux Containers). Since Docker containers are very lightweight, even a single server or a virtual machine can run several containers simultaneously. According to an analysis done in 2016, a typical Docker use case involves running five containers per host, but if we ponder over this, then that many organisations can actually run 10 containers or even more.

Using Docker to create and manage containers may actually simplify the creation of highly distributed systems by allowing multiple worker tasks, applications and other processes to run autonomously on a single physical machine

or even across multiple virtual machines. This allows the deployment of different nodes as soon as the resources become available or even when more nodes are needed. It allows a Platform-as-a-Service (PaaS) style of deployment and the scaling for various systems like Apache Cassandra, Riak or MongoDB. Docker also helps in simplifying the creation and operation of task queues and other such distributed systems. And, most importantly, Docker is open source, which means that anyone can contribute to it and even extend it to meet their own needs in case they need additional features that aren't available.

Differences between Docker and containers

1. There is some confusion when we try to compare containers and Docker. The latter has become quite synonymous with container technology because it has been the most successful at popularising this technology. Docker is an open source software containerisation platform, which helps in creating, deploying, and running various applications by using containers. But, we need to be aware that container technology is not new. It was built into Linux in the form of LXC almost 10 years ago, and similar virtualisation at operating system level has been offered by AIX Workload Partitions, FreeBSD jails and Solaris containers as well.
2. Today, Docker is not the only tool for Linux which helps in software containerisation. There are other alternatives like Rkt also available in the market. Rkt is a command line tool, which is used for running different app containers produced by CoreOS. It is able to handle both Docker containers as well as the ones that comply with its App Container Image specification.
3. One of the most important reasons for launching Rkt is that Docker has become too big and has actually lost its simplicity. Containers help in reliably running any type of software, whereas Docker is now building tools to launch various systems for clustering, cloud servers, and a wide range of other functions like building and running images, uploading or downloading them and eventually even overlay networking—all compiled into a single, monolithic binary running primarily as the root on any server.
4. According to Kelsey Hightower, chief advocate of CoreOS, other container images are intended to be much more secure than Docker images because they are signed by their creators. When we use Docker and pull an App Container image, we can decide if we trust the developer before running it. Rkt can also run Docker images, but they won't always be signed.

Integration of Docker with several other tools

Since Docker provides a platform to deploy an application with its complete package, it's quite obvious that it should have the ability to integrate with other kinds of applications as

Figure 4: Various tools on which Docker can be hosted (Image credits: Google images)

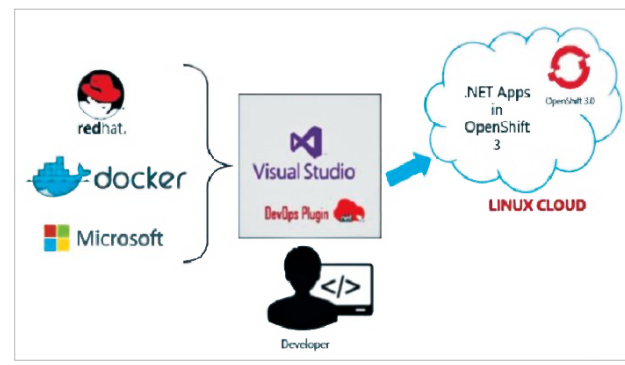


Figure 5: Role of Docker in Visual Studio 2015 (Image credits: Google images)

well, so that the application deployed on it can interact with it, without any blockages. There are several infrastructure tools like Amazon Web Services, Ansible, Chef, CFEngine or Google Cloud platform, which can easily be integrated with Docker.

Docker can also interact with other tools like Jenkins, IBM Bluemix, Microsoft Azure, OpenStack Nova, Salt, Vagrant and VMware vSphere Integrated Containers. The Cloud Foundry Diego integrates Docker into the Cloud Foundry PaaS. The OpenShift PaaS of Red Hat integrates Docker and other related projects (Kubernetes, Project Atomic, Geard and others) since v3 (June 2015). The Apprenda PaaS integrates containers of Docker in version 6.0 of its product.

Stack has complete support for automatically performing builds inside Docker, using the user ID and volume mounts switching to make it mostly seamless. FP Complete provides images for use with Stack that also include other tools, GHC and optionally, have all of the Stackage LTS packages which are pre-installed in the global package database. The main purpose for using Stack/Docker this way is to ensure that all developers build in a consistent environment without any of the team members needing to deal with Docker on their own.

The recent role of Docker in the IT field

1. Docker is a tool that is designed to benefit both systems administrators and developers, making it actually a part of many DevOps (developers + operations) tool chains. For developers, it means that they can simply focus on writing code without worrying about the system that it will be running on. Docker makes it possible to set up a local development environment that is exactly like the live server. It helps to run multiple development environments, that too from the same host having unique software, configurations, operating systems and test projects, on new or different servers. It also gives them an opportunity to get a head start by using one of the thousands of programs already designed to run in Docker as a part of their application. For those in operations, Docker provides flexibility and potentially reduces the number of systems that are needed because of its small footprint and lower overhead.
2. Docker, an open source containerisation platform, isn't just the darling of several Linux powers such as Red Hat and Canonical. Various proprietary software companies such as Microsoft have also adopted it. Red Hat OpenShift 3 is one of the latest technologies that uses Docker Container and Kubernetes. OpenShift 3 has evolved and gained great momentum since its inception in 2015. It uses a Git Repository to store the source code of an application and Docker registry to manage Docker images.
3. Docker has also been used as a tooling mechanism to improve the resource management, process isolation, and security in the virtualisation of any application. This had been made clear with the launch of Project Atomic, a lightweight Linux host specifically designed to run Linux containers. The main focus of this project is to make containers easy to run, update and roll back in an environment that requires fewer resources than a typical Linux host.
4. eBay has been quite focused on incorporating Docker into its continuous integration process so as to standardise the deployment across a distributed network of different servers that run as a single cluster. The company also isolates application dependencies inside containers in order to address the issue of each server having different application dependencies, software versions, and special hardware. This means the host OS need not be the same as the container OS, and eBay's end goal is to have different software and hardware systems running as a single Mesos cluster.
5. CompileBox, a Docker-based sandbox, can run all untrusted code and return the output without any risk to the host on which the software is actually running. During the development of CompileBox, developers considered using Ideone, Chroot jails and traditional virtual machines, but Docker was selected as the best option among all of them. Chroot does not provide the necessary level of security, whereas Ideone can quickly become cost-prohibitive, and virtual machines also take an exceptionally long time to reboot after they are compromised. Hence, Docker was the obvious choice for this application, as malicious code that attempts to destroy the system is limited to the container, and containers can be created and destroyed as quickly as needed.

Advantages of Docker

1. *Rapid application deployment:* Docker requires minimal runtime for any application, reducing the size and allowing quick deployment.
2. *Portability across machines:* An application with all its dependencies can be bundled into a single container, which is independent from the host version of the Linux kernel, deployment model or platform distribution. This container can be simply transferred to another machine that runs Docker, and executed there without any compatibility issues.
3. *Version control and component reuse:* We can track the successive versions of a Docker container, inspect the differences, or even roll back to previous versions. Docker reuses components from the preceding layers, which makes them lightweight.
4. *Sharing:* We can use a remote repository to share our Docker container with others. Red Hat even provides a registry for this purpose, and it is also possible to configure our own private repository.
5. *Lightweight footprint and minimal overhead:* Docker images are usually very small, which helps in rapid delivery and reduces the time to deploy any new application containers.
6. *Simplified maintenance:* Docker also reduces the effort and risk of problems caused due to application dependencies.
7. *Security:* From a security point of view, Docker ensures that applications that are running on containers are completely segregated from each other, granting us complete control over the traffic flow and management. No Docker container can look into processes that are running inside another container. From an architectural standpoint, each container actually gets its own set of resources, ranging from processing to network stacks. END

References

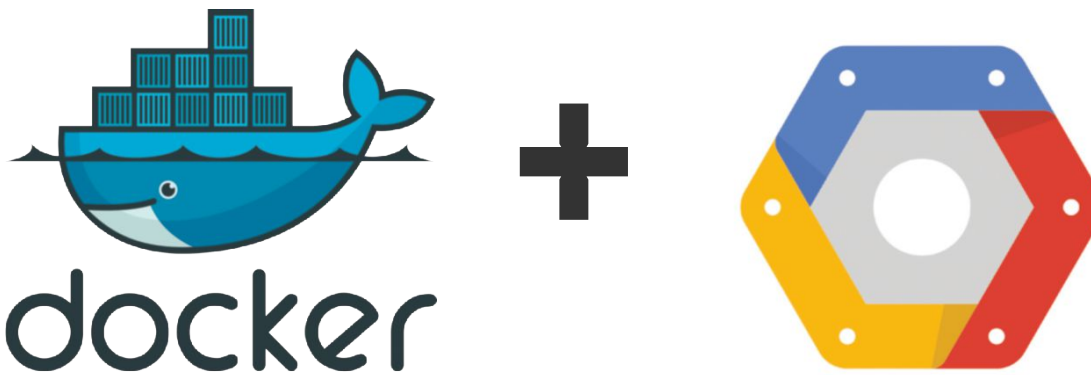
[1] *http://www.wikipedia.org/*
[2] *https://opensource.com/*
[3] *https://www.docker.com/*
[4] *https://dzone.com*

By: Vivek Ratan

The author is a B. Tech in electronics and instrumentation. He is currently working as an automation test engineer at Infosys, Pune, and can be reached at *ratanvivek14@gmail.com*.

Run Docker on the Google Cloud Platform

Google Cloud Platform is one of the leading public cloud infrastructures that allows you to run applications, perform data analysis and various tools. Docker, on the other hand, is an engine that runs container workloads. This article shows the reader how to run Docker workloads on the Google Cloud Platform.



Organisations are moving their workloads to the cloud at a fast pace. At the same time, applications have been shifting towards modularity through microservices and the more efficient running of workloads, whether they are on premise or in the cloud. It is not just virtual machines (VMs) that organisations are provisioning in the cloud, but container runtimes within these VMs for higher efficiency. Docker has been at the forefront of all this, as the engine that runs these container workloads.

The Google Cloud Platform (GCP) provides multiple ways to run Docker workloads in the cloud. The options range from spinning up one's own VMs to fully managed container orchestration platforms (based on Kubernetes) that are offered as options to the customer. This is coupled with powerful APIs and command-line interfaces to provision and manage the infrastructure and applications in the Google Cloud.

Google Cloud Platform (GCP) provides multiple options, depending on how much infrastructure management is desired (for simple deployment of applications, Google can take care of managing the infrastructure for you). Let us go through the options, first in brief and then in detail.

Google Compute Engine: This is the Infrastructure as a Service (IaaS) offering from the Google Cloud Platform. Google provides multiple types of VMs with pricing options, as well as the storage and networking infrastructure that might be needed for setting them up.

Google Container Engine: This is a fully managed offering that provides container orchestration at scale, based on Kubernetes. If you are dealing with Docker images, you can also choose to use the Google Container Registry option to host your images. For those fully committed to Google Cloud, using the Google Container Registry is a good option for faster availability of images.

Google App Engine Flexible: Building on the success of Google's Platform-as-a-Service (PaaS) offering, i.e., App Engine Standard, the Flexible Environment supports multiple other languages including bringing your own stack. The development workflow is based on Docker and eventual deployment is done in the Container Engine offering, behind the scenes, by the provisioning infrastructure.

All of the above services are available via the Google Cloud SDK, which is a set of command line utilities that can be used on a laptop to work with GCP resources. Those who have worked with Amazon Web Services (AWS) and Azure will find it familiar, as Google uses similar client tools to manage its cloud resources.

Google Cloud SDK

Google Cloud SDK is a suite of command-line tools for the platform. These include gcloud, bqutil, gsutil and more

to manage the resources in the cloud, directly from your machine. The installation of this SDK is straightforward and the link is provided in the *References* section.

Once the SDK is installed, you can specifically use the gcloud utility to create and manage VMs in the cloud on which we can have Docker running.

The SDK also ships with multiple client libraries that can be used in popular languages to interact with the cloud platform. In the next section, we will look at how to set up a VM running CoreOS in the Google Cloud, which can then run your Docker loads.

Google Compute Engine

Google Compute Engine is the Infrastructure-as-a-Service (IaaS) offering on GCP. It provides a variety of machine types (CPU, memory), ranging from the basic standard configuration to high-memory instances. It also supports multiple operating system images that can be used to run Docker loads. Specifically, vis-a-vis Docker, it provides custom Google-provided images that come bundled with Docker, rkt and Kubernetes so that you are up and running with your Docker machine as quickly as possible.

One operating system that has been built from the ground up as a container operating system is CoreOS, and Compute Engine supports it while creating the VM. All you need to do is provide a unique VM instance name, the image family (coreos-stable) and image project name (coreos-cloud) while using the gcloud command line utility —and within seconds you have a VM with Docker, rkt and Kubernetes running for you.

The steps shown below provide the usage of the gcloud SDK:

```
$ gcloud compute instances create container-instance1 \
   --image-family coreos-stable \
   --image-project coreos-cloud
```

Once the above command is run, within a minute you will have a VM ready for you, as indicated by the output below:

```
Created [https://www.googleapis.com/compute/v1/projects/
mindstormclouddemo/zones/us-central1-f/instances/container-
instance1].
NAME                      ZONE           MACHINE_TYPE
PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP     STATUS
container-instance1  us-central1-f  n1-standard-1
10.240.0.2        104.155.168.46     RUNNING
```

Now, if we SSH into the container-instance1 IP address, we can see that Docker and rkt are installed and ready for use.

Google Container Engine

Google Container Engine is a container orchestration and cluster manager for running your container workloads. It is based on Kubernetes, the leading orchestration engine for

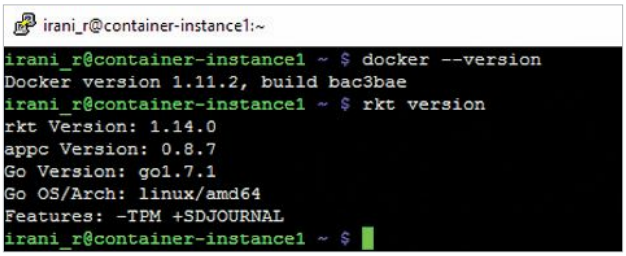


Figure 1: Google Compute Engine instance with Docker and rkt installed

containers, that was developed by Google itself, before the company open sourced the project on GitHub.

Google Container Engine or GKE, as it is called, is the fastest way to set up a fully managed container cluster in the cloud. Coupled with the fact that it provides support for Kubernetes right out of the box, you get scaling, failover and solutions to all the other challenging distributed software problems that have been addressed by Kubernetes.

The next few commands show how easy it is to get a container cluster running in GKE and to deploy your own service to it.

Assuming that you have the gcloud utility installed, creating a container cluster is a breeze. All you need to do is give the following command:

```
$ gcloud container clusters create osfy-cluster
```

It takes a few minutes to create the cluster. By default, a three-node cluster is created with one Kubernetes Master, which is fully managed by GKE itself.

Once the cluster is successfully created, you can view its details, as shown below:

```
Creating cluster osfy-cluster...done.
Created [https://container.googleapis.com/v1/projects/
mindstormclouddemo/zones/us-central1-f/clusters/osfy-
cluster].
kubeconfig entry generated for osfy-cluster.
NAME          ZONE           MASTER_VERSION  MASTER_IP
MACHINE_TYPE   NODE_VERSION  NUM_NODES  STATUS
osfy-cluster  us-central1-f  1.4.6
104.154.239.75  n1-standard-1         1.4.6
3
RUNNING
```

Now, let's create a deployment for nginx as shown below:

```
$ kubectl run nginx --image=nginx --port=80
deployment "nginx" created
```

We can expose this deployment as a service as follows:

```
$kubectl expose deployment nginx --type="LoadBalancer"
service "nginx" exposed
```

Notice that we have exposed the service to the outside world via the LoadBalancer; so this will result in Google Cloud provisioning a load balancer behind the scenes for you.

Now, if we query the services status after a while, we get the following output:

```
$ kubectl get services
NAME         CLUSTER-IP       EXTERNAL-IP      PORT(S)   AGE
kubernetes   10.111.240.1       <none>                   443/
TCP   8m
nginx            10.111.247.193    104.154.168.17      80/
TCP    1m
```

Notice that our nginx service has got an external IP through which the service can be accessed. You might have noticed that GKE, via its inherent fully-managed service, makes the process very easy.

Google Container Registry

If you run your workload on the Google Container Engine, one of the complementary products to consider is the Google Container Registry, which provides a powerful private registry for your Docker images. The Container Registry can also be used independently, and can be integrated with Docker CLI tools. The prefix *gcr.io* is used on images in the Google Container Registry.

Google App Engine Flexible

App Engine has been one of the most popular PaaS offerings available. This is the original standard runtime that supports Java, Python, Go and PHP applications. However, developers demanded flexibility by wanting to use programming languages other than the ones this engine supported, as well as different frameworks, while enjoying the benefits of focusing on their applications and not on managing the infrastructure – which App Engine did well. The App Engine Flexible environment uses Docker at its core in the development environment and behind the scenes deploys it to Google Container Engine. So if you are looking at a flexible PaaS environment based on Google, give App Engine Flexible a try.

Pricing

Google Cloud pricing has been considered by many to be more cost efficient vis-a-vis the other public clouds. Google has introduced pricing innovations like 'Pre-emptible instances', i.e., your instance could be shut down any time, and 'Sustained use' discounts, whereby automatic discounts are applied as you use the VMs for longer periods. This helps in significant savings. There is also a Google Cloud Platform Pricing Calculator, an interactive Web-based tool that you can use to determine your expenditure. END

References

[1] Google Compute Engine (GCE): *https://cloud.google.com/compute/*
[2] Google Container Engine (GKE): *https://cloud.google.com/container-engine/*
[3] Google Container Registry (GCR): *https://cloud.google.com/container-registry/*
[4] App Engine Flexible Runtime (GAE Flexible): *https://cloud.google.com/appengine/docs/flexible/*
[5] gcloud utility: *https://cloud.google.com/sdk/*
[6] Google Cloud Free Trial: *https://console.cloud.google.com/freetrial*
[7] Google Cloud Platform Pricing Calculator: *https://cloud.google.com/products/calculator/*

By: Romin Irani

The author has been working in the software industry for more than 20 years. His passion is to read, write and teach about technology, and make developers successful. He blogs at *www.rominirani.com*.

An Introduction to OpenNebula

OpenNebula is a simple, feature-rich and flexible solution for the management of virtualised data centres. It enables private, public and hybrid clouds. Here are a few facts about this solution.

OpenNebula is an open source cloud middleware solution that manages heterogeneous distributed data centre infrastructures. It is designed to be a simple but feature-rich, production-ready, customisable solution to build and manage enterprise clouds—simple to install, update and operate by the administrators; and simple to use by end users. OpenNebula combines existing virtualisation technologies with advanced features for multi-tenancy, automated provisioning and elasticity. A built-in virtual network manager maps virtual networks to physical networks. Distributions such as Ubuntu and Red Hat Enterprise Linux have already integrated OpenNebula. As you'll learn in this article, you can set up OpenNebula by installing a few packages and performing some cursory configurations. OpenNebula supports Xen, KVM and VMware hypervisors.

The OpenNebula deployment model

An OpenNebula deployment is modelled after the classic cluster architecture. Figure 1 shows the layout of the OpenNebula deployment model.

Master node: A single gateway or front-end machine, sometimes also called the master node, is responsible for queuing, scheduling and submitting jobs to the machines in the cluster. It runs several other OpenNebula services mentioned below:

- Provides an interface to the user to submit virtual machines and monitor their status.
- Manages and monitors all virtual machines running on different nodes in the cluster.
- It hosts the virtual machine repository and also runs a transfer service to manage the transfer of virtual machine images to the concerned worker nodes.
- Provides an easy-to-use mechanism to set up virtual networks in the cloud.
- Finally, the front-end allows you to add new machines to your cluster.

Worker node: The other machines in the cluster, known as 'worker nodes', provide raw computing power for processing the jobs submitted to the cluster. The worker nodes in an OpenNebula cluster are machines that deploy a virtualisation hypervisor, such as VMware, Xen or KVM.

OpenNebula installation

OpenNebula is one of the easiest cloud systems to install and configure. Binary packages exist for several distributions, including Red Hat Enterprise Linux, Ubuntu, Fedora and openSUSE. Distributions like Ubuntu also include packages in their standard repositories, which makes it simple to install OpenNebula.

We will build a private cloud with three Ubuntu 10.04 machines. One machine will serve as the front-end, called the *nebula-cloud-server*. Two machines will be worker nodes deploying KVM. The worker nodes are *open-nebula-wn* and *open-nebula-wn2*.

To install the front-end, you need to install a single package by typing the following command:

```
sudo apt-get install opennebula
```

This package installs the daemon *oned* that manages all OpenNebula services. The installation also creates a new user called *oneadmin*, which OpenNebula retains for its own use. The installation of this package also produces a new pair of SSH RSA keys. OpenNebula uses SSH to communicate with

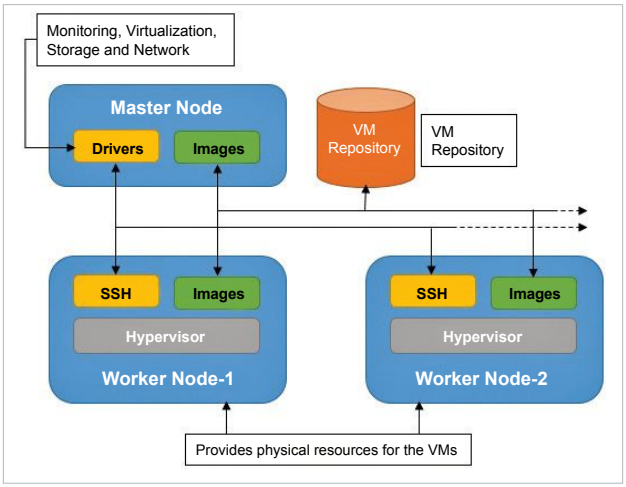


Figure 1: OpenNebula deployment model

other machines in the cluster, so a public key is used to identify the front-end. All worker nodes need to authorise the front-end SSH key.

After the *opennebula* package has been installed, the next step is to add the worker nodes, but before doing that, you need to install the worker node OpenNebula package on the worker node machines. At both worker nodes, you need to install the *opennebula-node* package by typing the following command:

```
sudo apt-get install opennebula-node
```

This package installs the necessary worker node packages and creates the *oneadmin* user. The following command is used to perform any operation related to the hosts:

```
onehost [options] command [parameters]
```

The command *argument* specifies the operation you want to carry out. You can add new machines to your cluster, retrieve monitoring information, enable or disable scheduling of virtual machines to the host, and list the currently active machines.

The *onehost create* command assists with adding new worker nodes to a cluster, as shown below:

```
onehost create hostname im_mad vmm_mad tm_mad
```

The *hostname* argument is the *hostname* or the IP of the machine; the next three parameters relate to the hypervisor and the transfer manager service. OpenNebula uses different drivers to access different hypervisors. These three parameters essentially tell the front-end that the machine you are adding deploys this specific hypervisor, and this mechanism should be used to transfer data to this machine. In this example, I add a host deploying KVM:

```
onehost add open-nebula-wn im_kvm vmm_kvm tm_ssh
```

The final arguments relate to the way virtual machines will be transferred to the node. In this example, SSH is used to transfer virtual machines. Note that the *oneadmin* user on this host needs to be able to SSH to *oneadmin@open-nebula-wn*; so, on this host, run the following command:

```
sudo -u oneadmin ssh open-nebula-wn
```

Then, verify the host's authenticity. On *open-nebula-wn*, run the following code:

```
sudo apt-get install opennebula-node
sudo tee /var/lib/one/.ssh/authorized_keys << EOT
onehost add open-nebula-wn2 im_kvm vmm_kvm tm_ssh
```

On *open-nebula-wn2*, run:

```
sudo apt-get install opennebula-node
sudo tee /var/lib/one/.ssh/authorized_keys << EOT
```

The *onehost create* command will prompt for the front-end's key to the authorised key file, which OpenNebula uses internally. So, you have to go back to the worker nodes and add the SSH key to these nodes.

If everything goes well, you can check the status of the nodes by typing the *onehost list* command, as follows:

```
01 nebula-user@nebula-cloud-server:~$ onehost list
02 HID NAME     RVM   TCPU   FCPU   ACPU    TMEM    FMEM STAT
03 0 open-nebula-wn   0   100    99   100 1068948  921356  on
04 1 open-nebula-wn2  0   100    12   100 1173072 1027776  on
```

OpenNebula configuration file

You can specify the configuration for your OpenNebula set-up in the *oned.conf* file. This file is critical to your OpenNebula deployment. The folks at OpenNebula provide all the possible configurations for existing drivers that ship with the standard OpenNebula distribution. You can comment and uncomment as required for your environment. If, for example, you are interested in using Xen, just uncomment the relevant section in the *oned.conf* file as shown below:

```
01 #IM_MAD = [
02 #    name       = "im_xen",
03 #    executable = "one_im_ssh",
04 #    arguments  = "im_xen/im_xen.conf",
05 #    default    = "im_xen/im_xen.conf" ]
06
07 #VM_MAD = [
08 #    name       = "vmm_xen",
09 #    executable = "one_vmm_xen",
10 #    default    = "vmm_xen/vmm_xen.conf",
11 #    type       = "xen" ]
```

The IM_MAD section relates to the information monitoring driver for status monitoring. The VM_MAD section deals with the hypervisor driver.

After you have added the worker nodes, your private three-node cloud is ready. However, before you can start launching virtual machines, you need to define a virtual network, and then create and launch a virtual machine.

Virtual networks

You can define custom private networks in OpenNebula, and virtual machines can be associated with several of these virtual networks. Different networks can link specific sets of virtual machines together. You could also permit some machines access to the Internet by connecting them to a specific virtual network.

To define your own virtual network, you need to write a template file. Fortunately, the syntax of a virtual network template file is pretty simple. In OpenNebula, you can define a 'fixed' virtual network or a 'ranged' network. A fixed network specifies certain IP addresses against specific virtual machines identified by their MAC addresses. On the other hand, a ranged network is more like a DHCP configuration, where a base network address is specified and all addresses follow that same pattern. A sample fixed network configuration template is shown below.

```
01 NAME = "Private Cloud"
02 TYPE = FIXED
03 BRIDGE = vbr0
04 LEASES = [IP=192.168.0.1, MAC=50:20:20:20:20:20]
Ranged Network Template
01 NAME = "wanNetwork"
02 TYPE = RANGED
03 BRIDGE = vbr0
04 NETWORK_SIZE = C
05 NETWORK_ADDRESS = 192.168.0.0
```

In creating a virtual network, I will use the fixed network template file as specified above.

```
01 nebula-user@nebula-cloud-server:~$ onevnet create nebula.
template
02 nebula-user@nebula-cloud-server:~$ onevnet list
03  NID NAME              TYPE BRIDGE
04    0 Private Cloud    Fixed   eth0
```

Creating a KVM virtual machine

To create a KVM virtual machine, you would start by creating the disk file, as follows:

```
dd if=/dev/zero of= ubuntu_server.img bs=1M count=4096
```

Use the *kvm* command to start the installation from the Ubuntu ISO file:

```
kvm -m 512 -cdrom /isos/ubuntu.iso -boot d ubuntu_server.img
```

After you run this command, a QEMU window should pop up and launch the Ubuntu CD boot screen. Select *Install Ubuntu* and follow the instructions.

Submitting the VM to OpenNebula

After the virtual machine is created, the next step is to submit the VM to OpenNebula. Before doing that, you need to carry out a few tasks. To begin with, you must copy the *ubuntu_server.img* created in the last section to the virtual machine repository location. This location is specified in the *oned.conf* file. The variable is called *VM_DIR* and it is commented by default. You'll need to store all images in */opt/vmImages/* on the front-end and update the configuration file accordingly, keeping in mind that you must restart the daemon after every configuration change.

After you've saved the image to the appropriate directory, you can write a VM description for OpenNebula. VM descriptions are mostly hypervisor-specific; however, some general fields are required as shown below:

```
01   NAME   = kvmVM #specify the name
02     CPU    = 1 # How many CPUs required?
03     MEMORY = 512 # RAM in MB
04     OS     = [
05      KERNAL   = "/boot/vmlinuz-2.6.32-24-generic", # Kernel
to use
06      INITRD   = "/boot/initrd.img-2.6.32-24-generic", #
initrd to use
07        ROOT    = "sda1", #Root partition
08        BOOT = hd #boot form harddisk
09     ]
10     DISK   = [
11       SOURCE   = "/opt/vmImages/ubuntu_server.img",
#location of source image
12        TARGET = "sda1", # mount as partition ]
13     DISK   = [
14        TYPE     = "swap", #swap drive
15        SIZE     = 1024, # size of swap drive
16        TARGET   = "sdb" #mount to partition
17     ]
18     NIC    = [ NETWORK = "Private Cloud" ] #connect to
specified virtual network, several NIC can be specified, is you
want to connect to several VNs
```

Save the VM description file to the *kvmVM.template*. Note that if you forget to remove the comments, which have been inserted to explain the contents of the template, it will produce parsing errors.

To submit this virtual machine, use the *openvm* command, as shown below:

```
onevm [options] subcommand [parameters]
```

The sub-command arguments associated with *openvm*

Continued on Page 52...

LXD: The Pure Container Hypervisor

A hypervisor is computer software, firmware or hardware that can create virtual machines. LXD is bundled with Ubuntu 16.04, and has the advantage of being able to run hundreds of unmodified Linux operating systems on a single server and at incredible speeds.

When Ubuntu 16.04 was launched, one of its most interesting features was software called LXD that was included with it. LXD is somewhat similar to Docker but differs from it in a few aspects.

The base technology used in both Docker and LXD is LXC or Linux Containers, which has been around for a few years. It didn't witness much traction because of the complexity involved in setting up the containers. Both LXD and Docker provide a high level user-friendly interface to the low level commands offered by LXC.

Differences between LXD, Docker and KVM

The primary difference between LXD and Docker is that the latter is meant for application-specific containers. Consider a typical use case of a Web application. You'll need a database server and a Web server. So you will have two Docker containers—one which runs the database and the other running the Web server. The containers do not perform any other function.

In LXD, it is possible to run a complete OS as a container. For this, typically, one requires KVM or Xen and full hardware virtualisation (emulated disk, network cards and CPU). Due to the number of layers in full HVM, the performance is not as good as LXC but security is better. The guest virtual machine is fully isolated from the host machine, which decreases the attack surface. Ubuntu officially supports running RHEL, CentOS, Ubuntu, Debian and Oracle Linux containers in LXD.

Setting up LXD

LXD is best used with the Z File System (ZFS), which has become relatively stable. Ubuntu 16.04 supports ZFS officially. ZFS is not supported as a root file system in an Ubuntu installation—you don't get that option even while installing Ubuntu, but you can always use it as a data storage file system.

The advantage of ZFS is that you get quick CoW (Copy-on-Write) snapshots, clones which are very useful for containers. Let's assume that you have created a container and installed the LAMP stack. Now, you can snapshot it and create as many LAMP containers at no time just by cloning them. The best part is that the clones consume disk space only for what differs from the original data. If you have installed your WordPress application in one of the containers and, let's say, WordPress takes up 10MB, then that particular container will consume only 10MB, unlike a full HVM for which you have to allocate a complete 10-20GB disk image file.

Creating the ZFS pool

Assuming you have a fresh installation of Ubuntu, with a spare disk or partition in the existing disk, we can install the required

ZFS utilities as shown below:

```
apt install zfsutils-linux
```

In my set-up, I have */dev/sda* as the OS drive and */dev/sdb* as another drive, on which I'll be creating the ZFS pool. ZFS is very sensitive to the disk names provided when creating the pool; so to ensure that it picks up the proper disk, we need to go to */dev/disk/by-id* to find the appropriate disk.

```
root@ubuntu:~# ls -lh /dev/disk/by-id
total 0
lrwxrwxrwx 1 root root  9 Dec  4 10:55 scsi-14d5346542020202
07305e3437703544694957d7ced624a7d -> ../../sr0
lrwxrwxrwx 1 root root  9 Dec  4 10:55 scsi-3600224804e58652
893517c552e9d9cc3 -> ../../sdb
lrwxrwxrwx 1 root root  9 Dec  4 10:55 scsi-360022480cf72d7b
58630bab61658ad98 -> ../../sda
lrwxrwxrwx 1 root root 10 Dec  4 10:55 scsi-360022480cf72d7b
58630bab61658ad98-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Dec  4 10:55 scsi-360022480cf72d7b
58630bab61658ad98-part2 -> ../../sda2
lrwxrwxrwx 1 root root 10 Dec  4 10:55 scsi-360022480cf72d7b
58630bab61658ad98-part3 -> ../../sda3
lrwxrwxrwx 1 root root  9 Dec  4 10:55 wwn-
0x600224804e58652893517c552e9d9cc3 -> ../../sdb
lrwxrwxrwx 1 root root  9 Dec  4 10:55 wwn-0x60022480cf72d7b
58630bab61658ad98 -> ../../sda
lrwxrwxrwx 1 root root 10 Dec  4 10:55 wwn-0x60022480cf72d7b
58630bab61658ad98-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Dec  4 10:55 wwn-0x60022480cf72d7b
58630bab61658ad98-part2 -> ../../sda2
lrwxrwxrwx 1 root root 10 Dec  4 10:55 wwn-0x60022480cf72d7b
58630bab61658ad98-part3 -> ../../sda3
```

As seen above in the output, the disk-ID for *sdb* is scsi-3600224804e58652893517c552e9d9cc3. If you use partitions on an existing disk, take the ID numbers from */dev/disk/by-id*. If you use normal names like */dev/sda* or */dev/sdb*, then you may not be able to get back the ZFS pool on reboot, in case the order of disks changes for some reason – which will cause the names of the disks to be changed.

```
root@ubuntu:~# zpool create zdata /dev/disk/by-id/scsi-
3600224804e58652893517c552e9d9cc3

invalid vdev specification
use '-f' to override the following errors:
/dev/disk/by-id/scsi-3600224804e58652893517c552e9d9cc3 does
not contain an EFI label but it may contain partition
```

And there, I got an error while creating the pool. Since I know that the disk doesn't contain anything, I'll simply force it using *–f* and create my pool.

```
root@ubuntu:~# zpool status
  pool: zdata
 state: ONLINE
  scan: none requested
config:

NAME             STATE     READ WRITE CKSUM
zdata            ONLINE    0    0       0
scsi-3600224804e58652893517c552e9d9cc3  ONLINE   0   0     0

errors: No known data errors
```

This is a very basic ZFS pool with just one disk. ZFS directly supports mirroring, striping and RAID5 modes. So it eliminates the need for MDADM or LVM. Documentation of such advanced configurations is easily available online; in particular, FreeBSD's ZFS guide is very easy to understand and apply the information to Ubuntu in spite of it being so different from FreeBSD.

Now we can initialise LXD using the following command:

```
lxd init
```

It will ask you a few questions and offer to even create a ZFS pool, but I have not used that option since I always create my ZFS pools manually, which gives me flexibility on how to set them up.

```
root@ubuntu:~# lxd init

Name of the storage backend to use (dir or zfs) [default=zfs]:
Create a new ZFS pool (yes/no) [default=yes]? no
Name of the existing ZFS pool or dataset: zdata
Would you like LXD to be available over the network (yes/no)
[default=no]?
Do you want to configure the LXD bridge (yes/no) [default=yes]?
Warning: Stopping lxd.service, but it can still be activated
by:
  lxd.socket
LXD has been successfully configured.
```

By default, *lxdbr0*, the interface that is created by this set-up script will do a NAT. In case you don't want to do a NAT, you can either select that option during the init time or run *dpkg-reconfigure lxd* after this to do it again.

Creating your first container

This is pretty simple. First, use the following command:

```
lxc launch ubuntu: first-container
```

In the above command, *launch* is the command name for creating and launching a container; *ubuntu:* is the name

of the image which is to be used to create the container named *1stc*. The name of the image follows this syntax: <repository>:<image name>. So, essentially, *ubuntu* in that command is the name of the image repository. Note that the name of the container must be a valid host name.

```
root@ubuntu:~# lxc launch ubuntu: first-container
Creating first-container
Starting first-container

To get into your container,
root@ubuntu:~# lxc exec first-container /bin/bash
root@first-container:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:b1:5d:bb
          inet addr:10.134.32.8  Bcast:10.134.32.255
Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:feb1:5dbb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2055 (2.0 KB)  TX bytes:1837 (1.8 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

LXD has a lot of other features, all of which cannot be covered in one article. If you'd like to explore LXD in more depth, head on to Google and LXD's official GitHub repository. END

By: Nilesh Govindrajan

The author is a technology and open source buff based in Pune, India. He can be contacted at *https://nileshgr.com* or on *twitter@nileshgr.*

Continued from Page 49...

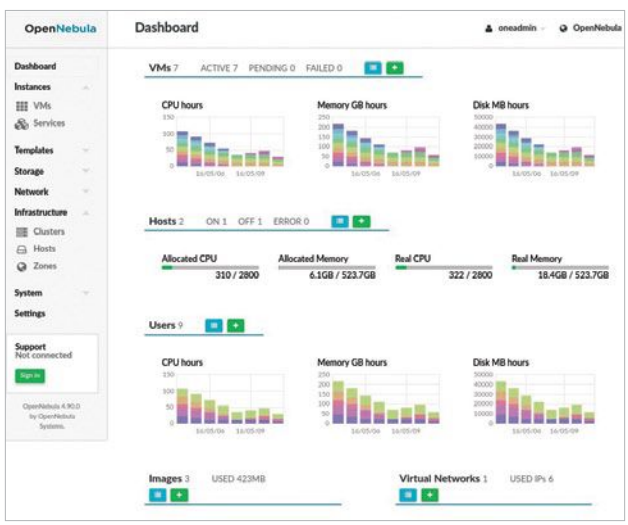


Figure 2: OpenNebula Sunstone dashboard

can be any of the following: *create*, *deploy*, *shutdown*, *livemigrate*, *migrate*, *hold*, *release*, *stop*, *cancel*, *suspend*, *resume*, *delete*, *list*, *show*, *top* and *history*. See the OpenNebula documentation for more on these commands.

The *create* command allows you to submit a new VM to the OpenNebula cluster. This command requires the user to pass the VM template file. Once you submit a VM, you can view its status with *show*. You will also know the ID assigned to the VM. To apply commands like *shutdown*, *stop*, *cancel*, *suspend* and *resume* to a VM, just specify the VM ID.

OpenNebula allows a user to migrate a VM from one host in the cluster to another. With *migrate*, the VM is first stopped before it is redeployed on another machine, whereas *livemigrate* migrates the VM without stopping and redeploying it. To specify a migration, you need to specify both the host ID and the VM ID.

Launching a virtual machine

To launch the virtual machine, type the following commands:

```
01 nebula-user@nebula-cloud-server:~$ onevm create kvmVM.
template
02 nebula-user@nebula-cloud-server:~$ onevm list
03   ID     NAME STAT CPU     MEM        HOSTNAME        TIME
04    0    kvmVM pend   0       0             00    00:00:07
```

The private cloud set-up is complete.

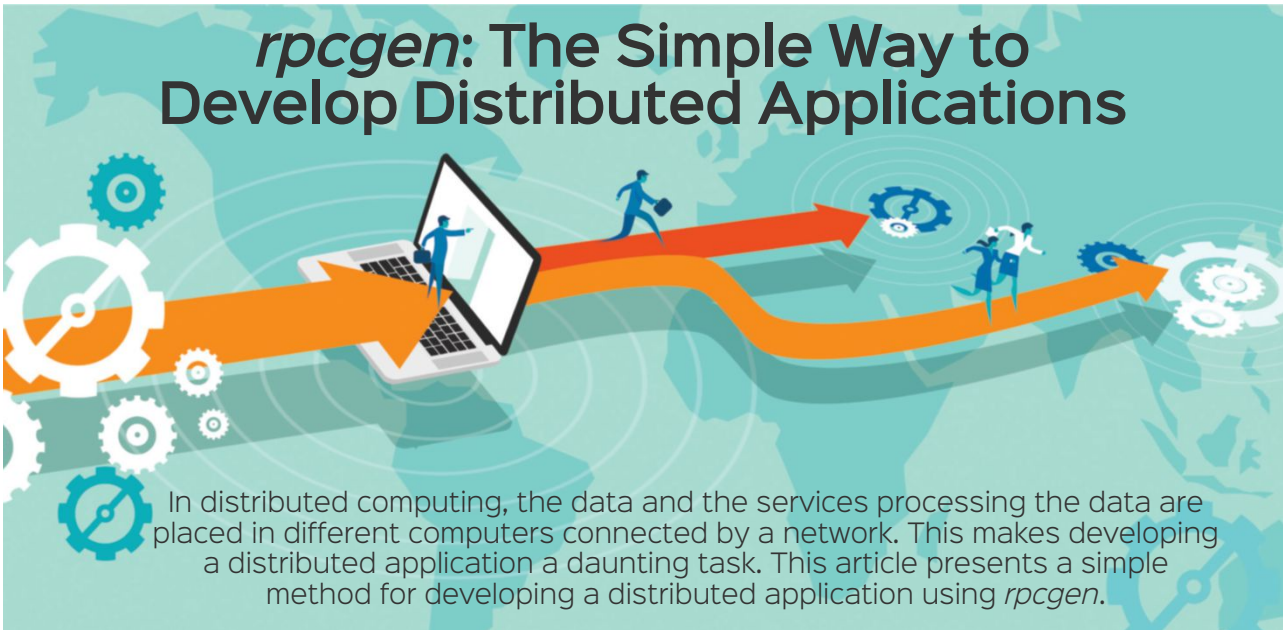
OpenNebula Sunstone

OpenNebula Sunstone is a graphical user interface (GUI), intended for both end users and administrators, which simplifies the typical management operations in private and hybrid cloud infrastructures. OpenNebula Sunstone allows one to easily manage all OpenNebula resources and perform typical operations on them. You will be able to manage the virtual and physical resources just as we did with the CLI. END

By: Miren Karamta

The author is a project scientist and IT systems manager at the Bhaskaracharya Institute for Space Applications and Geo-informatics (BISAG), Gandhinagar, Gujarat. You can reach him via email at *mirenkaramta@yahoo.com*. LinkedIn: *https://in.linkedin.com/in/miren-karamta-2b929122*

rpcgen: The Simple Way to Develop Distributed Applications



In distributed computing, the data and the services processing the data are placed in different computers connected by a network. This makes developing a distributed application a daunting task. This article presents a simple method for developing a distributed application using *rpcgen*.

Many scientific problems are solved using distributed systems, as shown in Figure 1. Basically, a distributed system is a collection of autonomous computers connected by a network that appears to the users of the system as a single computer. The message passing mechanism plays a major role in the success of distributed computing. For a distributed application, developers have to worry about details such as sockets, network byte order, host byte order, etc. In this situation, Remote Procedure Call (RPC) presents a simple mechanism for distributed computing without the need to include socket code in the distributed application.

rpcgen

Open Network Computing (ONC) Remote Procedure Call (RPC) is a widely deployed remote procedure call system. ONC was originally developed by Sun Microsystems in the 1980s as part of the company's Network File System project, and is sometimes referred to as Sun RPC. *rpcgen* is an interface generator precompiler for Sun Microsystems ONC RPC. Now, *rpcgen* is supported by many systems. With the availability of the *rpcgen* application, developers can concentrate on developing the core features of their application, instead of spending most of their time on developing and debugging their network interface code.

Application development using RPC

Distributed computing uses the divide and conquer principle to solve computationally-intensive problems. The problem is divided into many tasks, each of which is computed by one or more computers in the distributed systems. For the sake of simplicity, let us start developing a distributed application with two computing services called *max* and *min* running on two different computers, as shown in Figure 2. The service *max* finds the maximum number in the given list of numbers and the service *min* finds the minimum number in the given list of numbers.

Distributed application development, using *rpcgen*, starts with designing the Interface Definition Language file. This file contains the details about the services that run on different autonomous computers in the distributed system. The file has to be saved with a *.x* extension like *dist_app.x*, as in this case.

```
struct ipair   {   // Data that will be processed by the
services
              int data[10];
       int count;
  };
program MINMAX{       // MINMAX is the name of interface
    version VER1    {       //VER1 is the version
          int max(ipair)=1;
          int min(ipair)=2;
    }=1;
}=0x3a3afeeb;       // 32 bit program number
```

The next step is compiling *dist_app.x* with *rpcgen*.

$*rpcgen –C dist_app.x* as shown in Figure 3 creates four new files—*dist_app_clnt.c*, *dist_app.h*, *dist_app_svc.c* and *dist_app_xdr.c*. The option *–C* is used to generate ANSI C code. The files *dist_app_clnt.c* and *dist_app_svc.c* are called the client stub and server stub, respectively. These are responsible for maintaining networking connections. The file *dist_app.h* is the header file which will be included in the distributed application. The file *dist_app_xdr.c* is called the *eXternal data representation* file. It will help in data conversions.

The next step is to create the distributed application's sample client and sample server programs using the options *–Sc* and *–Ss* with *rpcgen*, as shown in Figure 4.

We then slightly modify the sample files *dist_app_client.c*, *dist_app_server1.c* and *dist_app_server2.c* according to our

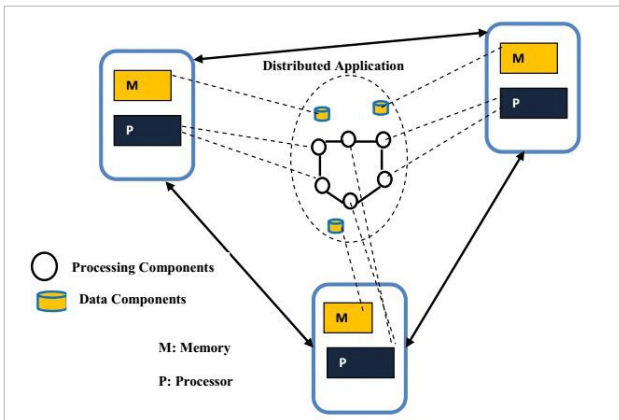


Figure 1: Distributed computing environment

needs, and then compile them using the gcc compiler as shown in Figure 5.

```
/*  This is  dist_app_client.c generated by rpcgen and
modified according to the need */

#include “dist_app.h”
void minmax_1(char *host1,char *host2,int *p,int length) //
Client Program has to communicate
{                  // with two Computers as per Figure 2.
     CLIENT *clnt1,*clnt2;
     int  *result_1,i;
     struct ipair op;
     int  *result_2;
     for(i=0;i<length;i++)           //making data ready
            op.data[i]=p[i];
            op.count=length;

#ifndef DEBUG
     clnt1 = clnt_create (host1, MINMAX, VER1, “udp”);  //
for making UDP connection
     if (clnt1 == NULL) {
       clnt_pcreateerror (host1);
       exit (1);
     }
     clnt2 = clnt_create (host2, MINMAX, VER1, “udp”); // for
making UDP Connection
     if (clnt2 == NULL) {
       clnt_pcreateerror (host2);
       exit (1);
     }
#endif  /* DEBUG */

     result_1 = max_1(&op, clnt1);        // sending data to
service Max as per Figure 2
     if (result_1 == (int *) NULL) {
       clnt_perror (clnt1, “call failed”);
     }
          printf(“\n The result from the Service Max is
```

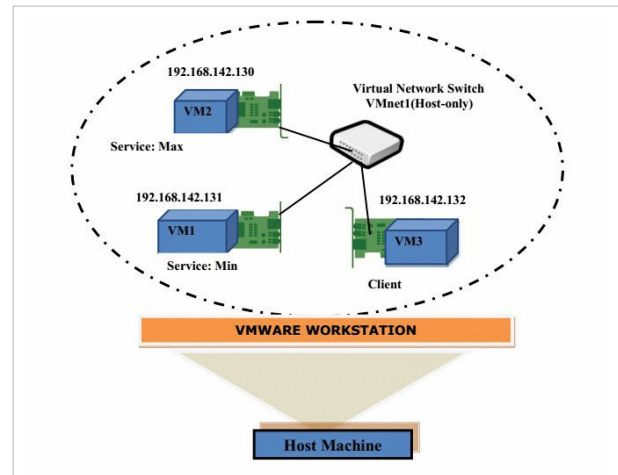


Figure 2: Test bed

```
root@client-virtual-machine: /home/client/RPC
root@client-virtual-machine:/home/client/RPC# rpcgen -C dist_app.x
root@client-virtual-machine:/home/client/RPC# ls
dist_app_clnt.c  dist_app.h  dist_app_svc.c  dist_app.x  dist_app_xdr.c
```

Figure 3: Compiling the IDL file with *rpcgen*

```
root@client-virtual-machine: /home/client/Desktop/RPC
root@client-virtual-machine:/home/client/Desktop/RPC# rpcgen -C -Sc dist_app.x >dist_app_client.c
root@client-virtual-machine:/home/client/Desktop/RPC# rpcgen -C -Ss dist_app.x >dist_app_server1.c
root@client-virtual-machine:/home/client/Desktop/RPC# rpcgen -C -Ss dist_app.x >dist_app_server2.c
root@client-virtual-machine:/home/client/Desktop/RPC# ls
dist_app_client.c  dist_app.h       dist_app_server2.c  dist_app.x
dist_app_clnt.c    dist_app_server1.c  dist_app_svc.c _    dist_app_xdr.c
```

Figure 4: Creating sample client and sample server programs with *rpcgen*

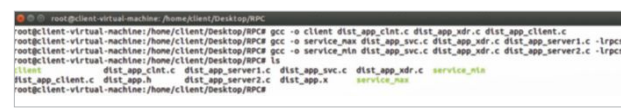


Figure 5: Generating a distributed application with gcc

```
:%d\n”, *result_1);
     result_2 = min_1(&op, clnt2);       // sending data to
service Min as per Figure 2
     if (result_2 == (int *) NULL) {
       clnt_perror (clnt2, “call failed”);
     }
        printf(“\n The result from the Service Min is
:%d\n”, *result_2);
#ifndef DEBUG
     clnt_destroy (clnt1);
          clnt_destroy (clnt2);
#endif   /* DEBUG */
}
int main (int argc, char *argv[])
{
     char *host1,*host2;

     int i,x[10];
          printf(“\n give any 10 elements”);  // For
collecting the list of elements from USER
                  for(i=0;i<10;i++)
               scanf(“%d”,&x[i]);
```

```
            host1="192.168.142.130"; //VM2 as per Figure 2
      host2 = "192.168.142.131";       //VM1 as per Figure 2
      minmax_1 (host1,host2,x,10);
            exit (0);
}

/*  This is dist_app_sever1.c generated by rpcgen and modified
according to the need */

#include "dist_app.h"
int * max_1_sec(ipair *argp, struct sec_rereq *rqstp) // The
service Max definition
{
      static int result;
      int i,big;
             printf("\n Got request for finding the max
element in the given array\n");
      big=argp->data[0];
       for(i=1;i<argp->count;i++)
             if(big<argp->data[i])
                  big=argp->data[i];
        result=big;
      return &result;
}
int * min_1_sec(ipair *argp, struct sec_req *rqstp)   // This
is not used here as it is not required.
{
      static int  result;
      /*
       * insert server code here
       */
      return &result;
}
/*  This is  dist_app_sever2.c generated by rpcgen and
modified according to the need */
#include "dist_app.h"

int * max_1_sec(ipair *argp, struct sec_req *rqstp) // This
is not used here as it is not required.
{
      static int  result;                          /*
       * insert server code here
       */
      return &result;
}
int *  min_1_sec(ipair *argp, struct sec_req *rqstp)  // The
service Min Definition
{
      static int  result;

      int i,small;
      printf("\n Got request for finding the min element in the
given array\n ");
      small=argp->data[0];
```

```
root@client-virtual-machine: /home/client/Desktop/RPC
root@client-virtual-machine:/home/client/Desktop/RPC# ./client

give any 10 elements 10 11 12 13 14 211 54 88 99 15

The result from the Service Max is :211

The result from the Service Min is :10
root@client-virtual-machine:/home/client/Desktop/RPC#
```

Figure 6: Executing the client program in VM3

```
root@kvsvn-virtual-machine: /home/kvsvn/RPC
root@kvsvn-virtual-machine:/home/kvsvn/RPC# ./service_max

Got request for finding the max element in the given array
```

Figure 7: Executing the service *max* in VM2

```
root@ratan-virtual-machine: /home/ratan
root@ratan-virtual-machine:/home/ratan# ./service_min

Got request for finding the min element in the given array
```

Figure 8: Executing the service *min* in VM1

```
      for(i=1;i<argp->count;i++)
            if(small>argp->data[i])
                 small=argp->data[i];

              result=small;
     return &result;
}
```

Executing the distributed application

We can execute the distributed application as shown in Figures 6, 7 and 8, respectively. The above modified code can also be downloaded from *http://opensourceforu.com/article_source_code/jan2016/rcpgen.zip*

Developing a distributed application is simple and easy with RPC, as compared to socket programming. Developers can use *rpcgen* to generate sample client and server programs, and then modify them easily, according to requirements. END

References

[1] Distributed Operating Systems (1st Edition) by Andrew S. Tanenbaum
[2] *https://en.wikipedia.org/wiki/Open_Network_Computing_Remote_Procedure_Call*
[3] *https://en.wikipedia.org/wiki/RPCGEN*
[4] *https://docs.freebsd.org/44doc/psd/22.rpcgen/paper.pdf*
[5] Power Programming with RPC by John Bloomer, Publisher: O'Reilly Media
[6] Implementing Remote Procedure Calls by Andrew D. Birrell and Bruce Jay Nelson, Xerox, Palo Alto Research Center

By: S. Ratan Kumar

The author currently works as the incubation centre coordinator at Anil Neerukonda Institute of Technology & Sciences (ANITS), Andhra Pradesh. He can be reached at *sratankumar.cse@anits.edu.in*. Web page: *https://sites.google.com/site/sajjaratankumar/*

"Cloud networking business is definitely growing in India"

The cloud has become an essential resource today, and has emerged as a major saviour for organisations at all levels. But the awareness of its vast potential is yet to be spread across the IT world. ***C. Ramanan, director*** of ***cloud networking – India subcontinent, Citrix,*** speaks about the importance of cloud technologies and open source in today's IT environment to ***Jagmeet Singh*** of ***OSFY***. Edited excerpts:

Q How is the recently launched NetScaler distinct from competing load balancing models?

Citrix understands Bimodal IT and has, thus, developed NetScaler on a Linux container that helps DevOps in application development and to stay agile. Our NetScaler is the first technology that offers a load balancer for Docker containers to enable the microservices load balancing functionality. This helps the new application development mode to do east-west load balancing between multiple microservices.

Q Why is there a need for a load balancer in the advanced world of cloud computing?

Traditionally, for any Web or mobile application that is built on the HTTP platform, customers generally prefer a load balancer to increase the application delivery performance and security. The advanced technology helps in reducing the back-end server workload. It also enhances the performance of application delivery and ensures secured delivery as well as protects the app from any web attacks.

But today, enterprise apps need NetScaler that is way beyond a simple load balancer. This solution helps to reduce the workload of back-end application and SSL servers and increases the application delivery performance by up to five times.

Q Why did you opt for the open source way of developing your load balancing technology?

In the places where microservices and Docker containers are required for developing applications, it is more efficient and highly effective to go for open source.

There is a vital need for every single microservice within that particular framework to communicate flawlessly. And to enable a flawless experience, you require a load balancer. It works as a reverse proxy and distributes network or application traffic on multiple servers to reduce workload.

We launched the open source NetScaler CPX around eight months back to solve the problem of application delivery through microservices and Docker containers. It gives developer teams the ability to customise and configure the tool to meet their requirements.

Q How do you view the parallel movements of open source and the cloud?

In the present open source world, there are two prime areas of focus—application development and orchestration.

Though the first area is a traditional one and developers are already aware of it, the second is critical for the current trend. If you look at the cloud model, it requires various components in your infrastructure such as the server,

storage, network and firewall, among others. A mediator is therefore required in this process to enable communication between different technologies and to simplify the whole process. This is the automation task of the orchestration layer.

Nowadays, there are various open source orchestration layers available in the market. CloudStack, OpenStack and OpenDaylight are few open source orchestration solution available today. NetScaler Nitro API seamlessly integrates with these Orchestration solutions and provides L4-L7 functionalities as a part of the complete automation, which is critical for enterprise app delivery.

Today, open source private cloud orchestration is attracting a large number of organisations. There is also quite a lot of adoption for open source in the private cloud market.

Q Is it profitable to choose open source for a new IT solution, instead of taking the proprietary path?

This would depend on customer requirements and the expectations in the market. The profitability is another factor that would be considered. Going the open source way obviously has both advantages and disadvantages. It is the same if you go the proprietary way.

> "Today, open source private cloud orchestration is attracting a large number of organisations."

Customers are not looking for any brand value. Instead, they prefer a solution that meets their needs and helps them to grow their business. The IT world is no longer looking for just some standardised solutions. The market has changed completely, and clients are now looking to fulfil their requirements from any efficient solution—whether it is open source or proprietary.

It is difficult to analyse the profitability matrix through open source or proprietary solution. However, if open source helps customers bridge the gap between what they plan for in the future and where they are today, it would be the best fit. Customers are also looking for flexibility, which more often comes from open source solutions. The flexibility can be with respect to enhancing the existing architecture, migration, seamless adoption to applications or the deployment of an existing application to a new area.

Q How is the Indian market different from developed markets like the US and UK when it comes to the open source space in the IT field?

The rate of adoption for new and emerging technologies is lower in India compared to the US and UK maybe in terms of percentage of adoption. But India has always been a volume player. The number of adoptions in the country will definitely supersede other parts of the world over server load balancer over supersedes. If we take the example of public cloud adoption, it took a little time in the Indian market to gain some success. In the same way, open source is slowly getting bigger in the IT industry in the country.

India is an IT and development hub, we see the growth of open source developer community in the country. Certainly, with more resources available in the market, open source adoption in India will certainly grow.

There is a mix of people who are migrating to the public cloud or building their own open source orchestration for their data centres. Also, some of the clients look for hybrid solutions. These dual-world technologies help them run their critical applications on their data centres and non-critical applications on the cloud. This trend is growing these days.

Compared to previous years, in the last year we have seen a lot of customers showing great interest in automation that uses private cloud deployments for data centres.

Q Do you think that it is hard to be a marketer of open source solutions in India?

New development areas as well as the latest software adoptions are inclined more towards open source. That is where we see the growth potential for the whole open source community in India.

So we can say that it is not difficult to market open source solutions in the Indian market. In fact, it is becoming easier even for new technologies like our NetScaler.

Q Is there any growth projection for cloud networking in India?

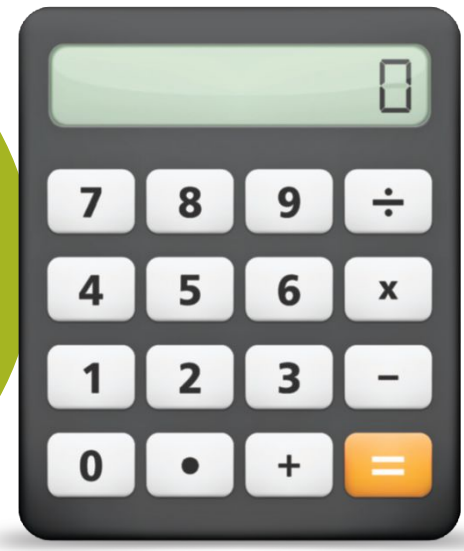
Cloud networking business is definitely growing in India. We see huge potential for cloud networking business in India.

Today, customers want the flexibility to host their apps and securely, seamlessly and optimally deliver these apps to the end users, who will be accessing them from any device on any network. Citrix NetScaler solution meets all these demands, and moves in parallel with all current and future technology trends through our 'software first' approach. Our NetScaler ADC, gateway, application firewall and SDWAN come as a part of our cloud networking offerings.

With all innovations, advanced cloud networking solution providers like NetScaler help customers keep their app anywhere (on-premise or on a public or hybrid cloud). We take care of optimally delivering these apps to the end users in a more secured manner, in a more controlled and managed the environment.

So the growth for cloud networking in India is indeed very high compared to what it was in the past. We see double-digit growth for NetScaler adoption in the Indian market, as the use-case for NetScaler today is way beyond just simple server load balancer. END

Develop a Tip Calculator Application in App Inventor 2



Moving on down the invention lane, let's create a simple Android application that calculates the tip we should give at a restaurant. Do have fun trying it out.

I hope that all of you are busy in developing creative Android apps, and doing the job well too. If you have uploaded any of your applications on to the Google Play Store, please share your links with me. On our journey so far, we have explored the various components and features of a smartphone. We can proudly say that we have mastered certain aspects like the components available for use in the palette, and found out that playing between the designer and block editor is so much fun.

In our Android app development journey, we have covered all those basic to complex applications that are essential for daily use. Now, let's develop a simple yet useful Android application. Leaving a tip is a tradition followed when we visit restaurants. With this app, we will calculate the amount of the tip.

Theme of the application

The theme is pretty simple—we will calculate the total bill at the restaurant by adding a percentage of the bill as the tip. We will have input boxes for the bill amount and tip percentage; then at the click of a button, we will get the total bill again, including the tip amount.

GUI requirements

For every application, we have a graphical user interface or GUI, which helps the user to interact with the on-screen components. How each component responds to user actions is defined in the block editor section.

GUI requirements for Screen1

1. *Label:* Labels are static text components that are used to display some headings or markings on the screen.
2. *Button:* A button will let you trigger the event and is a very essential component.
3. *Horizontal arrangement:* This is a special component, which keeps all child components horizontally aligned within it.
4. *Notifier:* A notifier is used to display some instructions or give users control over their existing components. You will see its functionality in more detail when we implement it in the app.
5. *Text box:* Text boxes are interactive components and are used to take user inputs. In our case, we need to input the bill amount and tip percentage to calculate the final sum.

The components that we require for this application are given in Table 1. We will drag them on to the designer from the left hand side palette.

1. Drag and drop the components mentioned in Table 1 to the viewer.
2. Visible components can be seen by you while the non-visible components will be located beneath the viewer under the tag 'Non-visible'.
3. Change the title of Screen1 to the name of the application.
4. Label and text boxes will be aligned using horizontal arrangements.
5. If you have dragged and placed everything, the layout will look something like what's shown in Figure 1.
6. Make the necessary property changes like we did when changing the text property for the label and button components (Figure 2).
7. Renaming the components helps to identify them in the block editor.

Table 1

Component's name	Purpose	Location
Label	To display a label	Palette-->User Interface-->Label
Button	To trigger events	Palette-->User Interface-->Button
Horizontal arrangement	To arrange the child components	Palette-->Layout-->Horizontal Arrangement
Notifier	To display on-screen information	Palette-->User Interface-->Notifier
Text box	To get user input	Palette-->User Interface-->Text Box

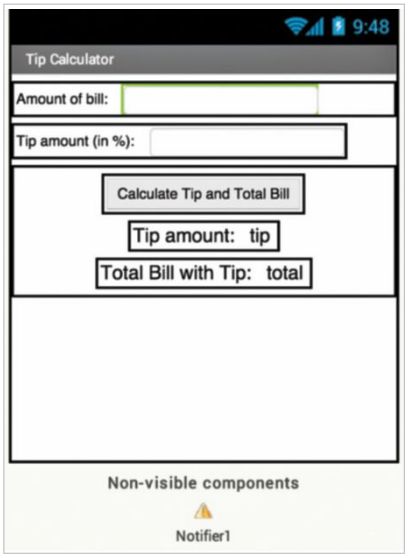


Figure 1: Designer screen

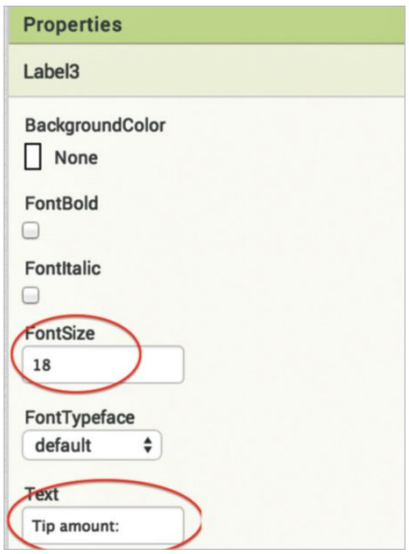


Figure 2: Label3 property changes

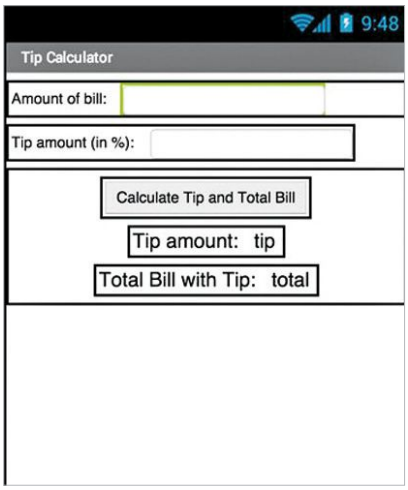


Figure 3: An overview of the application

8. Your graphical user interface is ready. Figure 3 shows exactly what the application will look like after the installation.
9. The hierarchy of the components that we have dragged to the designer is shown in Figure 4.

If you are confused by the designer and the components viewer, let me explain things a bit more. Here is the hierarchy that we have placed for our application.

- At the top, we have the title for the application. It can be set by changing the text property of the screen.
- Next, we have the horizontal arrangement, which holds a label and text box as child elements. This text box will be used to get the bill amount from the user.
- In the next series, we have another label and text box components — this second text box will be used to get the percentage (%) value of the tip.
- Below these we have a button which, upon clicking, will calculate the tip value and add it to the bill, to give the final bill amount.
- The labels placed next will display the tip and total bill amount separately.

Now, let's head towards the block

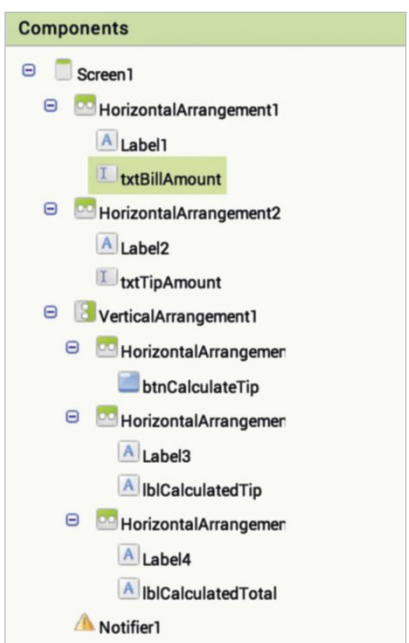


Figure 4: Components view

editor to define the behaviour. Let's discuss the actual functionality that we are expecting from the application.

- First, the user will enter the bill amount.
- Using the second text box, the user will input the percentage (%) of the tip applicable for the service.
- Upon clicking the button, the user should get the tip and total bill amount.
- We will check for any invalid inputs in the text fields.

So let's move on and add these features using the block editor. I hope you remember how to switch from the designer to the block editor. There is a button available right above the *Properties* pane to do so.

Block editor blocks: I have already prepared the blocks for you. All you need to do is drag the relevant ones from the left side palette and drop them on the viewer. Arrange the blocks in the same way as shown in Figure 5. I will explain what each one does and how it is called.

- We have initialised two variables, namely, *IclBillAmount* and

IclTipPercent; when the button is clicked, the click block of the button will be called.

- First, we will check whether the entries made are only numerical as we can't make calculations on text values.
- If the inputs entered are invalid, we will display a suitable error message using the notifier so that users can correct it accordingly.
- Next, we will calculate the tip amount mathematically.
- Adding the tip amount to the initial bill will give you the total amount payable.

Now, you are done with the block editor too. Next, we will move to download and install the app on your phone to check how it is working.

Packaging and testing

To test the app, you need to get it on your phone. First, you have to download the application to your computer and then move it to your phone via Bluetooth or USB cable. I'll tell you how to download it.

1. On the top row, click on the 'Build' button. It will show you the option to download the apk to your computer.
2. Downloading will show the progress and, after a successful download, the application will be placed to the download folder of your directory or the location you selected for it.
3. Now you need to get this apk file to your mobile phone either via Bluetooth or USB cable. Once you have placed the apk file to your SD card, you need to install it. Follow the on-screen instructions to install it. You might get some notification or warning saying *Install from un-trusted source*. Allow this from the settings and after successful installation, you will see the icon of your application in the menu of your mobile. Here, you will see the default icon, which can be changed and we will tell you how to do this as we move ahead.

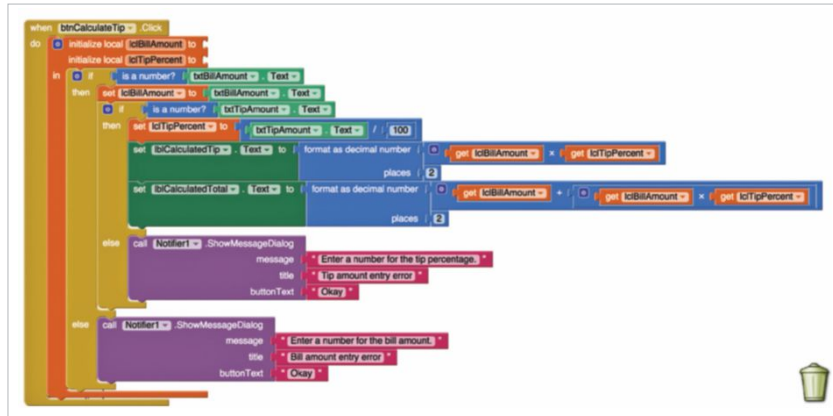


Figure 5: Block Editor Image 1

I hope your application is working exactly according to the requirements you have given. Now, depending upon your usability and customisation, you can change various things like the image, sound and behaviour.

Debugging the application

We have just created the prototype of the application with very basic functionality, but what else would the user expect of it? Let's look at various use cases that your app should be able to operate under—this will require serious thought so as not to annoy the user. Consider the following cases:

- Can we make a record of the calculations using the database component?
- Can we share the calculations we have made using the sharing component?

These are some of the scenarios that might occur, and users of your app will be pretty happy to see them implemented.

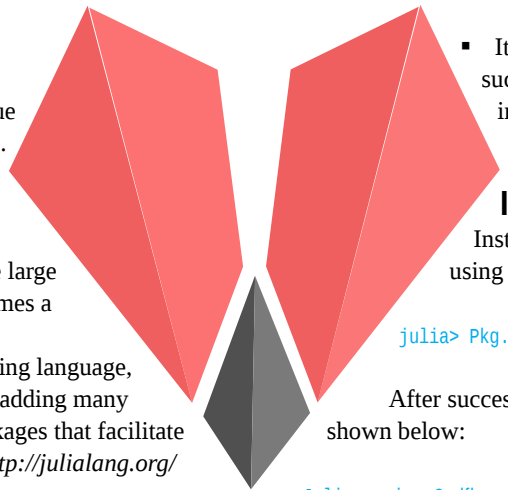
Think through all these scenarios and explore ways in which you can integrate them into the application. Do ask me if your app fails to operate well in any of the above cases. END

By: Meghraj Singh Beniwal

The author has a B. Tech in electronics and communication, is a freelance writer and an Android app developer. He is currently working as an automation engineer at Infosys, Pune. He can be contacted at *meghrajsingh01@rediffmail.com* or *meghrajwithandroid@gmail.com.*

Gadfly: Enabling Publication-Quality Plotting with Julia

Visualisation of data in the form of plots is an important requirement in today's Big Data scenario. This article introduces Gadfly – a visualisation system for Julia. Gadfly enables developers to build publication-quality graphics in various forms such as SVG, PNG, PDF, etc. With its wide variety of plot types, Gadfly provides a rich canvas for aesthetic statistical graphics.



Julia is becoming increasingly popular with developers from various domains, primarily due to the simplicity and power it offers. One important feature provided by most modern day programming languages is the ability to visualise data. As today's applications handle large volumes of data, visualisation becomes a mandatory component.

Julia is an extensible programming language, i.e., its features can be extended by adding many custom packages. The external packages that facilitate plotting in Julia are listed below *(http://julialang.org/downloads/plotting.html):*

- PyPlot
- Gadfly

PyPlot uses the Python calling feature (PyCall) to directly call MatPlotLib of Python. To use PyPlot in Julia programs, Python and MatPlotLib need to be available in the system.

Gadfly

The focus of this article is to illustrate the features of Gadfly, which is based on the Wickham-Wilkinson style 'Grammar of Graphics' in Julia. Detailed information on the Grammar of Graphics is available at *https://www.cs.uic.edu/~wilkinson/TheGrammarOfGraphics/GOG.html*. The ggplot2 for R forms the basis of Gadfly. Daniel C. Jones is the pioneer in developing Gadfly. An active community of Julia developers now maintains this package (*https://github.com/GiovineItalia/Gadfly.jl*).

Features of Gadfly

The major features of the Gadfly Julia package are listed below:

- Gadfly has the important feature of rendering publication quality graphics. It enables rendering in various formats such as SVG, PostScript, PD, etc.
- It offers support for 20+ plot types.
- Gadfly is available in Julia, out-of-the-box.
- The facility to integrate with *DataFrames.jl* is another key feature of GadFly. *DataFrames.jl* enables working with tabular data in Julia.
- It provides interactive features in plots such as panning, zooming, etc. These interactivity features are enabled through snap.svg *(http://snapsvg.io/)*.

Installing Gadfly

Installation of Gadfly can be easily done using the following command (at Julia REPL):

```
julia> Pkg.add("Gadfly")
```

After successful installation, it can be loaded as shown below:

```
Julia> using Gadfly
```

Gadfly: A simple plot

Julia programs can be executed in two different ways—one is by installing Julia locally in the system and the other is to execute it directly in JuliaBox *(https://juliabox.com/)*.

The steps involved in plotting a simple graph are illustrated in this section.

- *Step 1:* Load the Gadfly package, as follows:

```
using Gadfly
```

- *Step 2:* Load the values in X and Y. In this example, we are loading 1000 random values in two variables—*xvalues* and *yvalues*.

```
xvalues = rand(1000)
yvalues = rand(1000)
```

- *Step 3:* Plot the graph with the *Gadfly.plot* function.

```
Gadfly.plot(x=xvalues, y=yvalues, Geom.point)
```

The output of Step 3 is shown in Figure 2.

The following code sequence generates a line graph as shown in Figure 3:

```
using Gadfly
```

Figure 1: Features of Gadfly

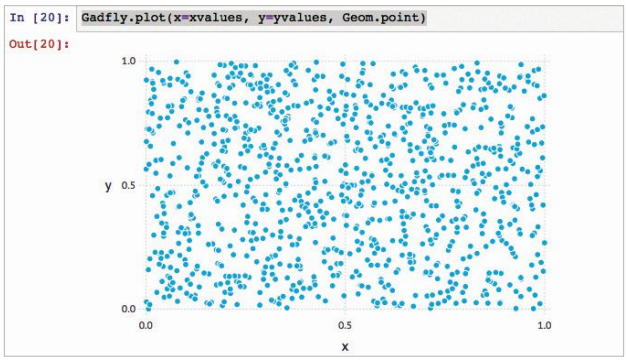


Figure 2: Gadfly - a simple plot

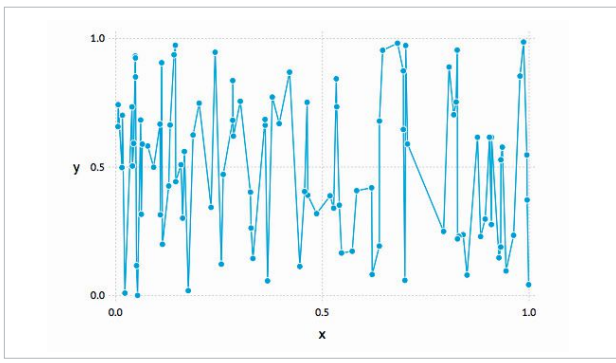


Figure 3: Gadfly - generating a line graph

```
xvalues = rand(100)
yvalues = rand(100)
Gadfly.plot(x=xvalues, y=yvalues, Geom.point, Geom.line)
```

The smoothened version is generated with *Geom.smooth*, as shown in Figure 4.

The plot can be customised with details such as *axis title* with the following code:

```
Gadfly.plot(x=1:10, y=2.^rand(10),
Scale.y_sqrt, Geom.point, Geom.smooth,
Guide.xlabel("Stimulus"), Guide.ylabel("Response"), Guide.
title("Dog Training"))
```

The output of the code is shown in Figure 5.

Gadfly: Histogram with RDataSets

Gadfly enables the creation of histograms effortlessly. A sample histogram with RDataSets is shown in Figure 6.

```
using RDatasets
Gadfly.plot(dataset("car", "SLID"), x="Wages",
color="Language", Geom.histogram)
```

Multilayer plots

Gadfly provides options to layer and stack plots. A sample code to layer a point plot and line plot is shown below:

```
Gadfly.plot(layer(x=rand(10), y=rand(10), Geom.point,
order=1),
     layer(x=rand(10), y=rand(10), Geom.line, order=2))
```

Here, the keywords *layer* and *order* are used to provide multilayer plots in a specified order.

Gadfly backends

Gadfly provides support to write to various backgrounds such as SVG and SVGJS. The support for SVG and SVGJS is provided by default. However, the other backends such

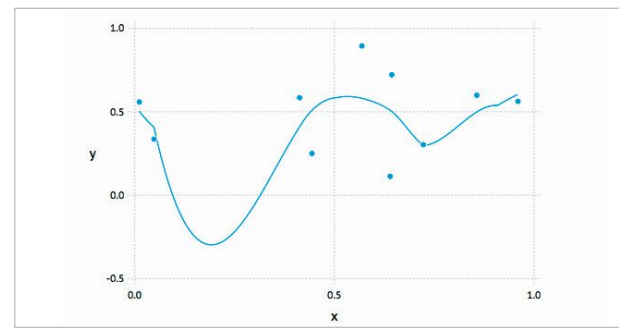


Figure 4: Gadfly – Geom.smooth

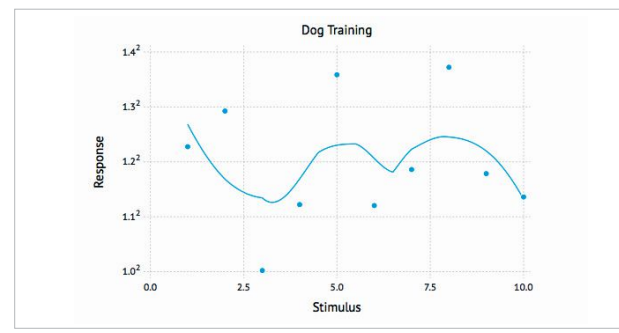


Figure 5: Gadfly plot with axis title information

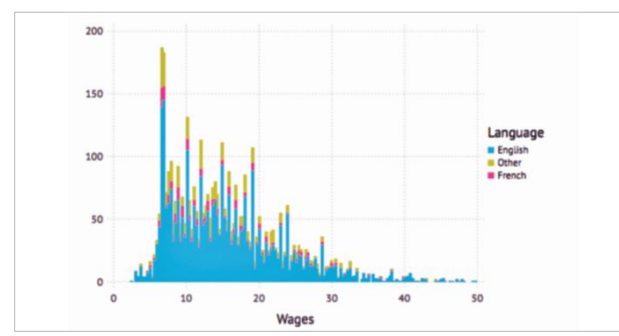


Figure 6: A sample histogram

as PNG, PDF, etc, require *Cairo* Julia bindings, which is a 2D graphics library *(https://github.com/JuliaGraphics/Cairo.jl):*

```
samplePlot = Gadfly.plot(layer(x=rand(10), y=rand(10), Geom.
point, order=1),
     layer(x=rand(10), y=rand(10), Geom.line, order=2))
draw(SVG("myplot.svg", 4inch, 3inch), samplePlot)
draw(PDF("myplot.pdf", 4inch, 3inch), samplePlot)
```

Themes

Gadfly has various options to customise the plot with themes. A sample code is shown below and its output is given in Figure 8. In this plot, the panel background colour is *gray* and the *default_color* is set as *orange*.

```
using Gadfly
dark_panel = Theme(
panel_fill=colorant"gray",
    default_color=colorant"orange"
)
Gadfly.plot(x=rand(10), y=rand(10), dark_panel)
```

The plots can be customised with various parameters. Some of them are listed below:

- *line_width*
- *panel_fill*
- *panel_opacity*
- *grid_color*
- *discrete_color_scheme*
- *continuous_color_scheme*

Geometries

The core drawing of the plot is made by geometries, and those which are available are represented using 'Geom'. Gadfly provides more than twenty styles. Some of the Geom types are listed below:

- *Geom.point*
- *Geom.line*
- *Geom.polygon*
- *Geom.boxplot*
- *Geom.ribbon*
- *Geom.violin*

The complete list of all Geoms is available at *http://gadflyjl.org/stable/man/geometries.html.*

For example, a Boxplot with RDataSets can be easily built as shown below:

```
using Gadfly
using RDatasets
Gadfly.plot(dataset("lattice", "singer"), x="VoicePart",
y="Height", Geom.boxplot)
```

To summarise, the features provided by Gadfly are really useful in making publication-quality plots. At the same time, making these plots is fairly simple. This excellent combination of features makes Gadfly a great choice with respect to visualisation of data. END

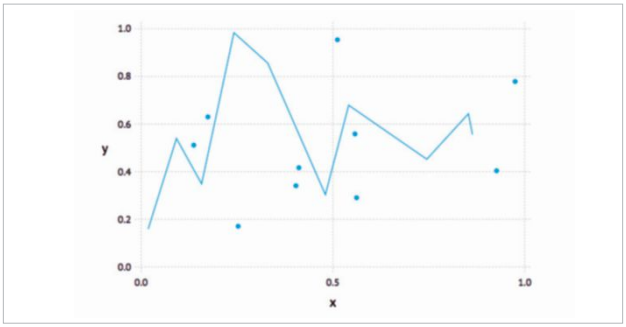


Figure 7: Gadfly – stacking and layering

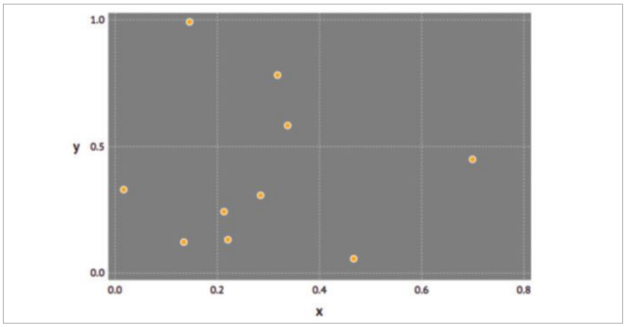


Figure 8: Gadfly-customising plots with themes

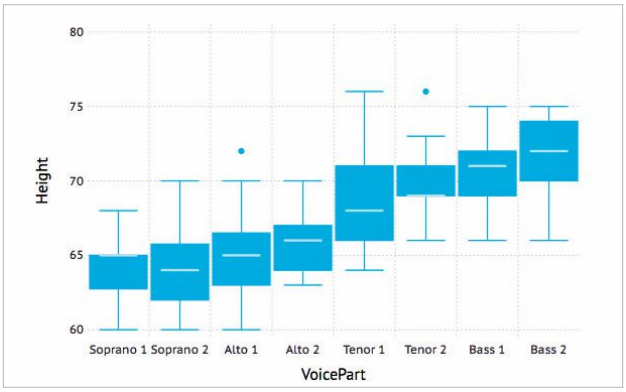


Figure 9: Boxplot

References

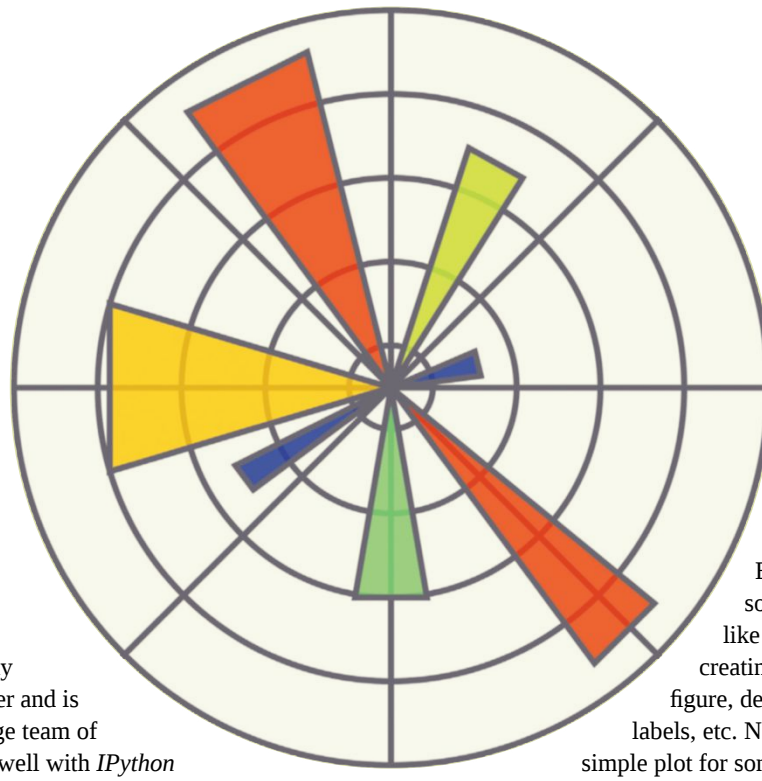
[1] *http://gadflyjl.org/stable/index.html*
[2] *https://juliabox.com/*
[3] *https://github.com/GiovineItalia/Gadfly.jl*

By: Dr K.S. Kuppusamy

The author is assistant professor of computer science, School of Engineering and Technology, Pondicherry, and has more than 11 years of teaching and research experience in academia and industry. He can be reached via mail at *kskuppu@gmail.com.*

2D Plotting Using the *matplotlib* Library

Publication quality 2D plots can be produced by *matplotlib*, which is an open source object-oriented Python library. With this article, we begin a series that will take the reader through the nuances of 2D plotting with *matplotlib*.



matplotlib is a Python library for creating 2D plots. It was originally created by John D. Hunter and is now maintained by a large team of developers. It integrates well with *IPython* – the component in the standard scientific Python toolset. *matplotlib* uses NumPy – a fundamental package for scientific computing with Python. The *matplotlib* code is conceptually divided into three parts -- the *pylab interface*, the front-end and the back-end. The pylab interface is a set of functions which allow the user to create plots. The front-end (matplotlib API) is a set of classes for creating and managing figures, text, lines, plots, etc. Basically, this is an abstract interface. The back-end is a drawing device (renderer) that transforms the front-end representation to a hard copy or a display device.

Installation

Use the following commands to install *matplotlib*:

```
python –m pip install  -U pip setuptools
python –m pip install matplotlib
```

Pyplot

matplotlib.pylot is a collection of command style functions. Each function makes some change to a figure, like creating a figure, creating a plotting area in a figure, decorating the plot with labels, etc. Now, let us create a very simple plot for some given data, as shown below:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('Y values')
plt.show()
```

If you provide a single list array to the *plot()* command, *matplotlib* assumes it is a sequence of Y values and internally generates the X value for you. The output is shown in Figure 1.

We do not need to worry about window creation and other basic event handling. *matplotlib* provides some basic operations of the plot. The basic operations are: *Reset original view, Back to previous view, Forward to next view, Pan axes with left mouse, Zoom with right, Zoom to rectangle, Configure subplots* and *Save the figure*. If you click on the 'Configure subplots' button, another window with the following parameters will open: *left, bottom, right, top, wspace*, and *hspace* to adjust the plot (see Figure 2).

Table 1

Property	Value type
alpha	float
animated	[True | False]
antialiased or aa	[True | False]
clip_box	a matplotlib.transform.Bbox instance
clip_on	[True | False]
clip_path	a Path instance and a Transform instance, a Patch
color or c	any matplotlib colour
contains	the hit testing function
dash_capstyle	['butt' | 'round' | 'projecting']
dash_joinstyle	['miter' | 'round' | 'bevel']
dashes	sequence of on/off ink in points
data	(np.array xdata, np.array ydata)
figure	a matplotlib.figure.Figure instance
label	any string
linestyle or ls	['-' | '--' | '-.' | ':' | 'steps' | ...]
linewidth or lw	float value in points
lod	[True | False]
marker	['+' | ',' | '.' | '1' | '2' | '3' | '4']
markeredgecolor or mec	any matplotlib color
markeredgewidth or mew	float value in points
markerfacecolor or mfc	any matplotlib color
markersize or ms	float
markevery	[None | integer | (startind, stride)]
picker	used in interactive line selection
pickradius	the line pick selection radius
solid_capstyle	['butt' | 'round' | 'projecting']
solid_joinstyle	['miter' | 'round' | 'bevel']
transform	a matplotlib.transforms.Transform instance
visible	[True | False]
xdata	np.array
ydata	np.array
zorder	any number

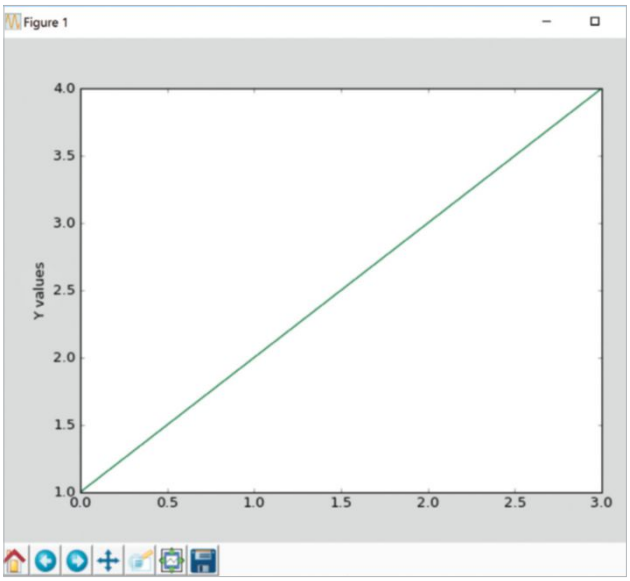


Figure 1: Plot for the values [1, 2, 3, 4]

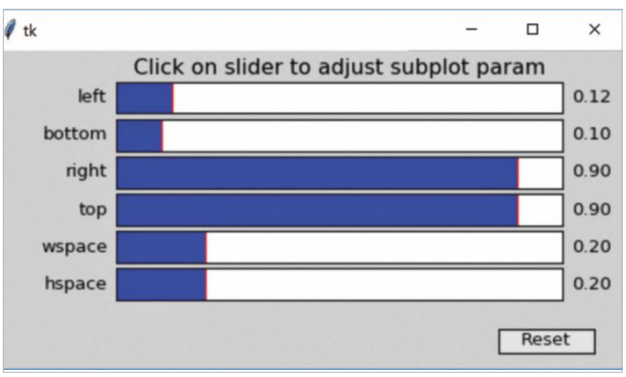


Figure 2: Parameters to adjust the plot

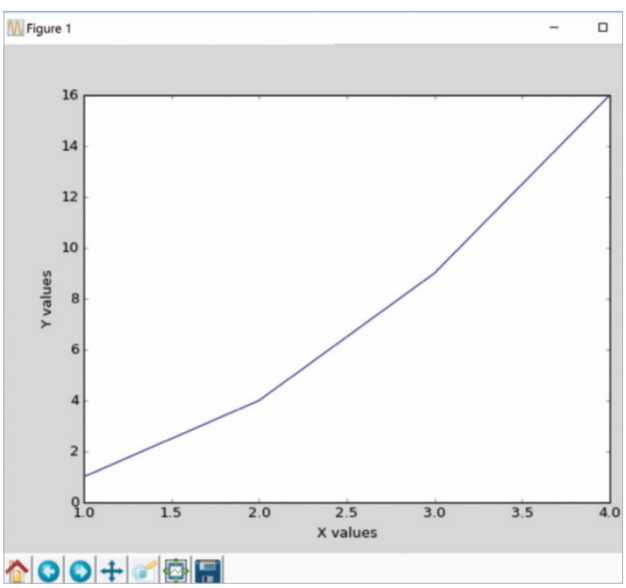


Figure 3: Specifying X and Y values in *plot()* function

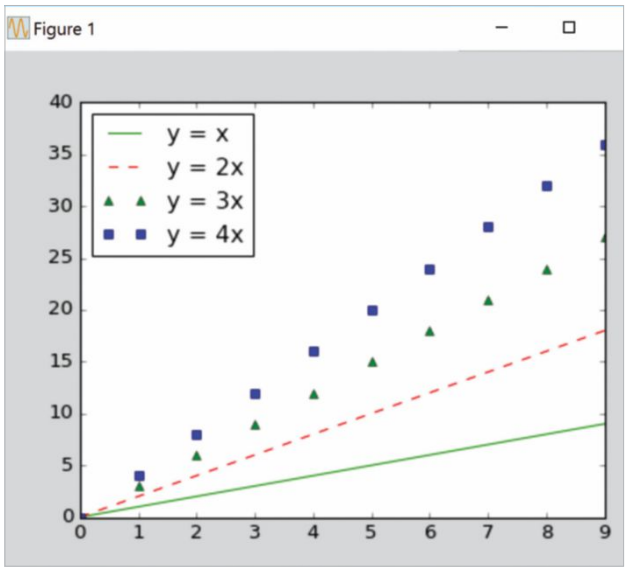


Figure 4: Plotting multiple lines with different styles

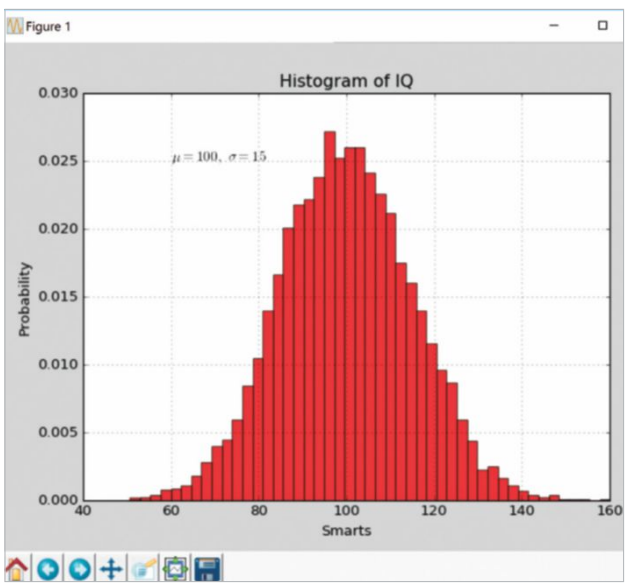


Figure 5: Histogram with text and grid

To plot X versus Y, we can use the commands shown below, as *plot()* takes an arbitrary number of arguments.

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.xlabel('X values')
plt.ylabel('Y values')
plt.show()
```

The output of the code above is shown in Figure 3.

To control the axis, *matplotlib* has the *axis()* command; it takes a list of values (Xmin, Xmax, Ymin, Ymax) and also specifies the viewport of the axes.

To plot several lines with different styles, use the following commands:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(10)
plt.plot(x, x, 'g')
plt.plot(x, 2 * x, 'r--')
plt.plot(x, 3 * x, 'g^')
plt.plot(x, 4 * x, 'bs')
plt.legend(['y = x', 'y = 2x', 'y = 3x', 'y = 4x'],
loc='upper left')
plt.show()
```

Controlling line properties

We can set many attributes to a line. The Table 1 shows a list of available *Line2D* properties.

Adding text into the plot

The *text()* command can be used to add text in an arbitrary location. The *xlabel()*, *ylabel()* and *title()* are used to add text in the indicated locations. The code below creates a histogram and some text. The output is shown in Figure 5.

```
import numpy as np
import matplotlib.pyplot as plt
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)
# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g',
alpha=0.75)
plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

This is only an introductory article on *matplotlib*. Next month, we will explore the topic in greater depth along with examples. The code is taken from *http://matplotlib.org* and readers are advised to look at the examples provided at *http://matplotlib.org/examples/index.html*. END

References

[1] *http://matplotlib.org/examples/index.html*

By: Nelson Joseph G.

The author works at EinNel Technologies, Chennai, and can be contacted at *nelson@einnel.com*.

Bower: The Package Manager for Web Applications

Websites comprise frameworks, libraries, assets and utilities. Keeping track of all these packages and making sure they are up to date (or set to the specific versions you need) is tricky. Bower manages all this for you, seamlessly.

Bower is an open source package manager for Web applications. We need to install a lot of packages while building a website. Bower helps in automatically fetching and installing all these packages. The main objective of Bower is not to minimise code but to install the right version of packages and their dependencies that we require for a project.

Installing Bower

To install Bower, you have to have the Node package manager, i.e., *npm* installed on your system.

To install *npm*, go to the *Node.js* website *(www.nodejs.org)* and download the current version of *Node.js* for your system. It is a command line utility.

Install Bower as a global node module. To do so, use the following command:

```
C:\>npm install -g bower
```

Note: Bower is a user command; there is no need to execute it with super user permissions.

Once the installation is complete, Bower is ready for use.

Creating the *bower.json* file

Bower uses a file called *bower.json* to keep track of all the dependencies that are used in your project. To build a *bower.json* file, use the following command:

```
C:\>bower init
```

When you type this command, you will be asked a few questions and then you can initialise the *bower.json* file.

The description of the file properties of *bower.json* is as follows:

1. name – The name of your application
2. version – A version number for your application

By using Bower, we can add a new package to our project, automatically. Here, we will add Bootstrap to our project by using Bower. To install packages with Bower, use the following commands:

```
# install dependencies listed in bower.json
C:\> bower install
# install a package and add it to bower.json
C:\> bower install <package> -S
```

While installing packages using Bower, there may be a chance that we get this specific error:

```
ENOGIT git is not installed or not in the PATH
```

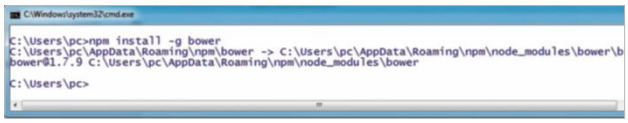


Figure 1: Installing Bower

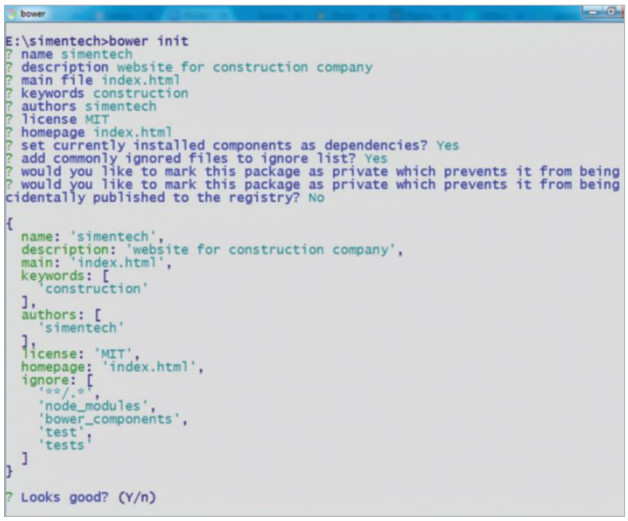


Figure 2: Creating *bower.json*

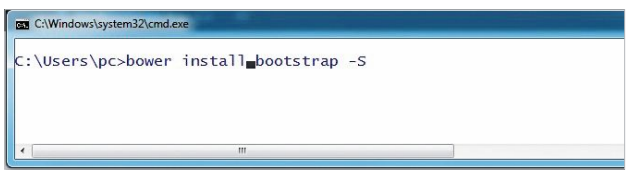


Figure 3: Installing the Bootstrap package using Bower

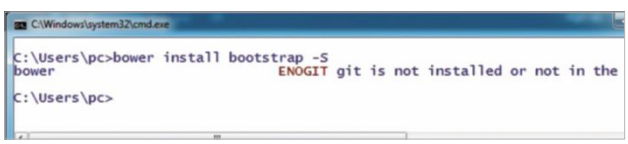


Figure 4: Error while installing the package using Bower

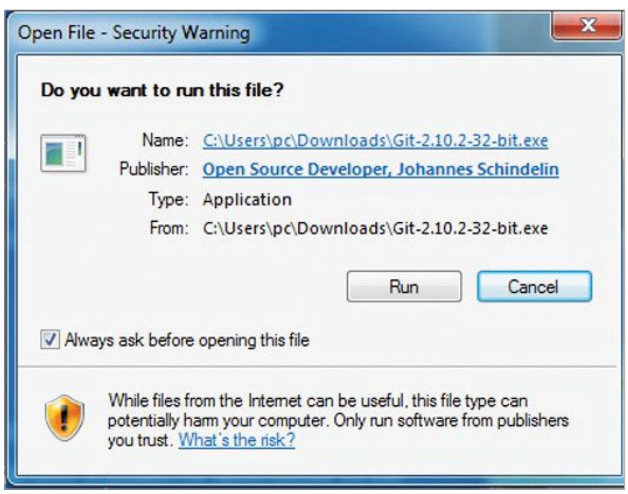


Figure 5: Installing Git

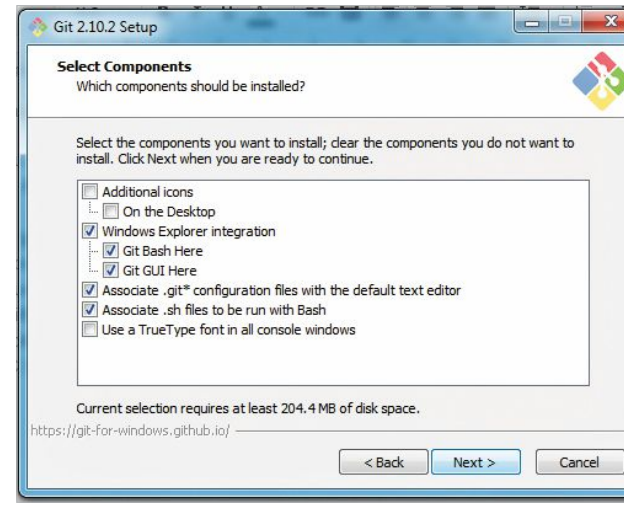


Figure 6: Selecting components

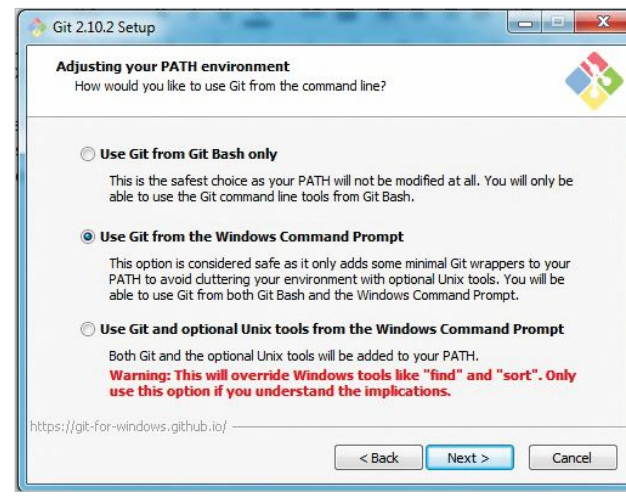


Figure 7: Setting the path

Note: To use Bower on Windows, you must have Git installed on your machine as some packages require it to be installed. Also, add Git to your path variable.

To install Git on Windows, follow the instructions given below:

- Download Git from *https://git-for-windows.github.io/*.
- Run the *Git2.110.2.exe* file.
- Follow the installation instructions. The path to my Git installation is *C:\Program Files*. Choose a different one if you want.
- Select the Git component that you want.
- Select the second or third option from the list. It will automatically add Git to your PATH variable.
- Configure the line ending conversations.
- Click *Next* and then click *install* to install Git on your machine.

After successful installation of Git, open Git Bash or the Git Shell *(Program files -> Git->Git Bash\Git Shell\Git*

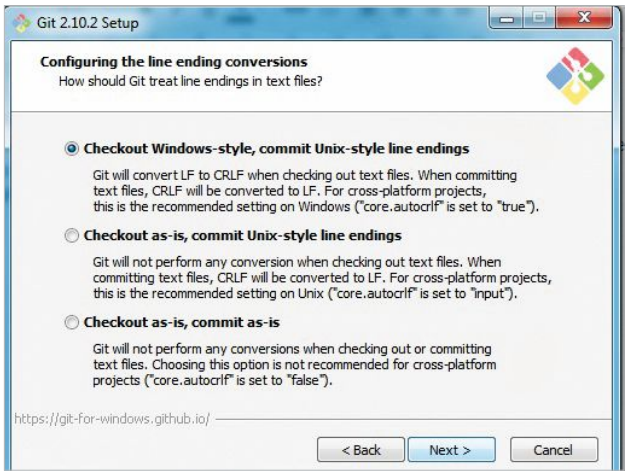


Figure 8: Selecting line ending conversations

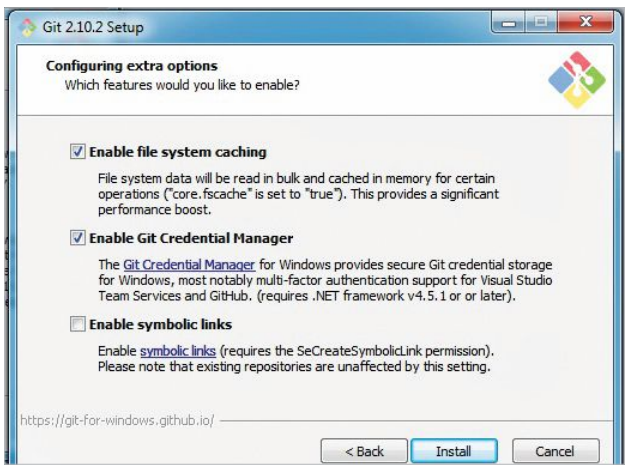


Figure 9: Extra options

cmd) and go to your project (using *cd*). Git Shell is installed by default with the GitHub Windows installation. To install Bootstrap, enter the following command:

```
C:\>bower install bootstrap -S
```

** In the above code snippet, '-S'means it will save Bootstrap as a dependency for your project in the *bower.json* file.

Now, Bower will automatically install Bootstrap to your project. It recognises that Bootstrap depends on JQuery. So, Bower will automatically install JQuery for us.

In your project folder, Bower has created a *bower_components* folder. The components that Bower fetched and installed are saved in this folder. Now, open the *bower.json* file to see that Bower has added new information about the dependencies.

Updating Bower components

To update Bower components, use the following command:

```
C:\>bower update
```

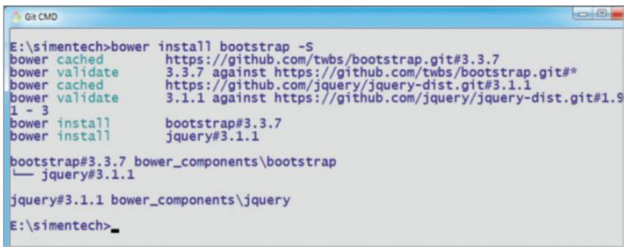


Figure 10: Successful installation of the package using Bower

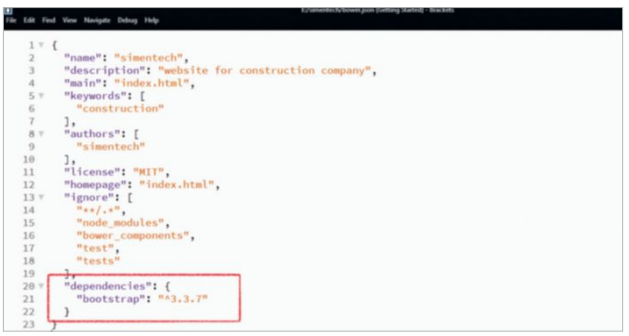


Figure 11: *bower.json* file

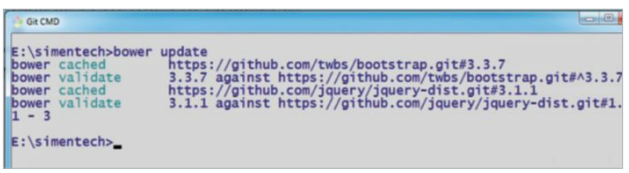


Figure 12: Update packages using Bower

Using Bower components

To make use of the Bower components, include CSS and JS scripts into the Web pages. Since all JS and CSS files reside in the *bower_component* folder, we will have to write the following code.

For CSS, the code is:

```
<link href="bower_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">
```

For JS, the code is:

```
<script src="bower_components/jquery/dist/jquery.min.js"></script>
<script src="bower_components/bootstrap/dist/js/bootstrap.min.js"></script>
```

END

References

[1] *https://bower.io/*
[2] *https://git-for-windows.github.io*

By: Shweta Tyagi

The author currently works as a Web developer and tester. Her areas of interest are website development and automation testing.

Waf: An Excellent Build Automation Tool

Build automation automates the process of a software build and removes the drudgery of having to do it all manually. Waf is a relatively new build automation tool that is open source and platform-independent. Written in Python, it is maintained by Thomas Nagy.

Build automation tools are used for automatic compilation and installation of computer software. Let us assume you are working on a large software project which is divided into a 100 program files. It is possible to compile all these files manually to generate an executable file. But what if you are frequently making changes to a particular file, and the functions used in it are also used by all the other program files? Then you'd have to recompile all the 100 files again and again. You'd be forced to execute hundreds of commands manually. Such a tedious process to compile the project might make you overlook some minor errors. In such a situation, a build automation tool is quite handy. It helps to build the executable file from hundreds of program files by automating the build process.

An executable file or a binary file contains instructions that cause a computer to perform certain tasks. We are familiar with the *.exe* executable files in Windows. The *a.out* file associated with C programs is actually an old executable file format in Linux. And nowadays, the most popular executable file format in Linux is the Executable and Linkable Format (ELF).

Apache Ant, Bazel, BitBake, CMake, Make, Waf, etc, are some popular build automation tools. The most popular build automation tool is Make, which dates back to the 1970s. I used Make for a long time till I decided to explore Waf, a relatively new build automation tool, which I found to be excellent.

Installing Waf

Waf is developed using Python and maintained by Thomas Nagy. The latest version is *waf-1.9.6*. It is open source software and its source code is released under the new BSD licence. The associated documentation of Waf is under the Creative Commons Licence, and prohibits commercial reuse and modification. Hence, it is overlooked by many of the Linux distributions. For example, Debian Linux does not have Waf included in it by default. But having Waf in your

system is very simple because you only need to download the Waf executable file. You can use the following Linux command *wget* to download the Waf executable file if you have Internet connectivity:

```
wget https://waf.io/waf-1.9.6
```

The command *wget* will download the Waf executable file to your current working directory. You can run this executable file with the command *'./waf-1.9.6'*. But remember to rename the file as *waf* and copy this file to any one of the directories pointed by the PATH environment variable of Linux to make matters easy. This step is important because the Linux environment will search for executable files only in the directories pointed by the environment variable PATH and, in most cases, the current working directory will not be included. Execute the command *'echo $PATH'* to see the various directories searched by the Linux Shell for executables. Also remember to change the access permissions by using the command *chmod*. In this case, the command *'chmod 755 waf'* or *'chmod 755 waf-1.9.6'* (depending on whether you have renamed the Waf executable or not) will be sufficient. This command gives read, write and execute permissions to the owner, and gives read and execute permissions for the group and others.

If you do not have Internet connectivity, just download the Waf executable file from the URL *https://waf.io* and copy it to one of the directories pointed by the environment variable PATH, as mentioned earlier. But do remember that this is not the way we are supposed to use Waf. The developers of Waf want us to copy its executable into the main directory of each and every project we are developing along with a Waf script to build the project. This is the single greatest advantage of Waf over most other build automation tools. If you are using the tool Make for building your project, then the target system should have Make installed in it. But with Waf, you do not have this constraint because all you have to do is to bundle an executable of Waf along with the source files of your project. But while writing this article, I primarily had students in mind—those who are trying to customise their systems with a build automation tool and hence chose the method we are discussing.

Waf can be built from the source files but I prefer the easier method, whereby the executable of Waf is downloaded. As an aside, Waf can also be used in MS Windows and the working of the executable is similar in Windows and Linux.

A sample program to test Waf

In order to demonstrate the working of Waf, a simple C program is used in this article. But do remember that Waf can be used to build projects developed in languages like C++, C#, Java, Fortran, Python, D, etc. The simple C program is divided into three files — *main.c*, *data.c* and *show.c*. The file *main.c*, shown below, calls two functions *data( )* and *show( )*. But the function definitions are given in two separate files.

```
void data( );
void show( );
int main( )
{
            data( );
            show( );
            return 0;
}
```

The file *data.c* given below contains the definition of the function *data( )*.

```
#include <stdio.h>
void data( )
{
            printf("\nLet us learn about Waf\n");
}
```

The file *show.c* given below contains the definition of the function *show( )*.

```
#include <stdio.h>
void show( )
{
            printf("\nWaf is a build automation tool\n");
}
```

These three files can be compiled using the C compiler *gcc* from the GNU Compiler Collection (GCC) to generate an executable called *hello* by executing the following single command:

```
gcc main.c data.c show.c -o hello
```

But the problem with this sort of compilation is that all the three files are recompiled irrespective of whether they are modified or not. So we need the following sequence of commands, which will compile each file individually rather than compiling all the files at a single go, in order to avoid unnecessary recompilation of unmodified files.

```
gcc -c main.c
gcc -c data.c
gcc -c show.c
gcc main.o data.o fun.o -o hello
```

The option *–c* of *gcc* tells the compiler to produce object code rather than executable code. But the option *–c*

```
[root@deepu waf]# ls
data.c  main.c  show.c
[root@deepu waf]# gcc -c main.c
[root@deepu waf]# gcc -c data.c
[root@deepu waf]# gcc -c show.c
[root@deepu waf]# gcc main.o data.o show.o -o hello
[root@deepu waf]# ls
data.c  data.o  hello  main.c  main.o  show.c  show.o
[root@deepu waf]# ./hello

Let us learn about Waf

Waf is a build automation tool
```

Figure 1: Manual compilation output

```
[root@deepu waf]# waf configure
Setting top to                           : /root/Desktop/waf
Setting out to                           : /root/Desktop/waf
Setting top == out (remember to use "update_outputs")
Checking for 'gcc' (c compiler)          : /usr/lib/ccache/gcc
'configure' finished successfully (0.135s)
[root@deepu waf]# waf
Waf: Entering directory `/root/Desktop/waf'
Building from the build directory, forcing --targets=*
[1/4] c: main.c -> main.c.1.o
[2/4] c: data.c -> data.c.2.o
[3/4] c: show.c -> show.c.3.o
[4/4] cprogram: main.c.1.o data.c.2.o show.c.3.o -> hello
Waf: Leaving directory `/root/Desktop/waf'
'build' finished successfully (0.062s)
[root@deepu waf]# ls
c4che  config.log  data.c  data.c.2.o  hello  main.c  main.c.1.o  show.c  show.c.3.o  wscript
[root@deepu waf]# ./hello

Let us learn about Waf

Waf is a build automation tool
```

Figure 2: Output of *wscript* execution

also makes sure that the linker is not called into action. The first three commands create three object files called *main.o*, *data.o* and *show.o*. The fourth command uses the object files *main.o*, *data.o* and *show.o* to produce an executable file called *hello* with the option *-o*. This executable *hello* is executed with the command *./hello* and the output is shown in Figure 2.

A simple Waf script

What if the program contains 10, 100 or 1000 files? Well, we need 11, 101 or 1001 commands, respectively, to generate an executable file. This is alarming and almost as difficult as writing the program itself. But here comes Waf to our rescue. The same effect as one thousand and one command line instructions can be achieved with the help of a single Waf script. The Waf executable will search for a script called *wscript* containing all the rules to build a particular project in the current working directory of the project from which the Waf executable is called. The code below shows the *wscript* to compile and build the sample project.

```
top = '.'
out = '.'
def options(opt):
    opt.load('compiler_c')
def configure(conf):
    conf.load('compiler_c')
def build(bld):
            bld.program(source='main.c', target='hello',
use='data show')
            bld.objects(source='data.c', target='data')
            bld.objects(source='show.c', target='show')
def test(tst):
            print("\nThis is a waf test function\n\n")
```

Always remember to save the Waf script files with the name *wscript*. In order to build our sample project, open a terminal and set the directory containing the sample C programs and the Waf script *wscript* as the current working directory. If you have manually created the executable file *hello*, make sure that you have deleted the executable as well as the object files. The sample project can be built by executing the following commands.

```
waf configure
waf
```

If you haven't renamed the file *waf-1.9.6* as *waf*, then you need to execute the command *waf-1.9.6* on the terminal instead of *waf*. Figure 2 shows the output of these command executions.

Figure 2 also shows the associated files generated by Waf other than the executable *hello*. So it is better to store the output data into a separate directory other than the current working directory. For example, the line of code *out = './dest'* instead of the line of code *out* = '.' will store the executable *hello* and other associated directories into a directory called *dest* inside the current working directory. And now, if you run the executable *hello* in the current working directory or the directory *dest* with the command *./hello* you will get the same output shown in Figure 2.

The Waf script demystified

Now, it is time for us to understand the working of the Waf script file *wscript*. The line '*top* = '.' defines the top directory of the project. This is almost always the directory containing the top level *wscript* of your project. Remember that '.' denotes the current working directory in Linux and this is where Waf initially searches for source files, unless you are recursively calling other Waf scripts from one of the parent directories or sub-directories (a topic that we are going to discuss later). The line *out* = '.' defines the name of the output directory to which the executable file of the project and other dependent files are to be saved. As mentioned earlier, in our case, the executable file will be saved to the current working directory.

If you are familiar with the programming language Python, you will see that the remaining lines of code in the file *wscript* are nothing but the definition of four Python functions—*options*, *configure*, *build* and *test*. Even if you don't know Python, there's no need to worry because you don't need Python to use Waf. The most

important fact to remember while learning Waf is '*Waf commands map to Python functions*'. In simple terms, for each Waf command there should be a Python function defined in the Waf script *wscript*. So now we can use the four Waf commands— *options*, *configure*, *build* and *test*, which are available with the Waf script, and all of them need a Python function to define their actions. Each of these four functions has a parameter defined for them, which can be further used by other functions. In our case, the variables used to denote these parameters are *opt*, *conf*, *bld* and *tst*, respectively.

The first Python function from our Waf script *wscript* invokes a command called *configure*. The *configure* command is used to check if the requirements for working on a project are met and if it is possible to store the executable file and other associated files in your destination directory. The parameter of the *configure* command is then stored for use by other commands, such as the *build* command. The function *options* defines a command named *options* which is called once, before any other command is executed in *wscript*. In our case, the function *options* with the parameter *compiler_c* is used to load the C compiler in a platform-independent manner. The term 'platform-independent' means that when using Waf, it doesn't matter whether you have gcc, LLVM or some other compiler to compile the given C program. Instead, Waf will simply identify a compiler capable of compiling C programs on your system or show an error message if it fails to find a suitable C compiler. Other extensions like *compiler_cxx*, *compiler_fc* and *compiler_d* are provided to support the compilation of projects developed using C++, FORTRAN and D, respectively.

The most important function in *wscript* is called *build*, which can be executed with the command *waf* or *waf build*. In our *wscript*, the second and third lines of code in the *build* function create object files for the source files *data.c* and *show.c*. The first line of code in the *build* function generates the final executable file *hello* by using the file *main.c* and the object files generated from the files *data.c* and *show.c*. So, in order to build our sample project, we only need the three functions— *options*, *configure*, and *build*. You might then wonder what the purpose of the function *test* is, in the given *wscript*.

User defined Waf commands

Earlier, we learned the rule '*Waf commands map to Python functions*'. So if you want a user defined Waf command, all you need to do is define a Python function in *wscript*. Execute the command *waf test* on the terminal and see the output. I am sure you will be able to guess the output.

Multiple Waf scripts for a project

For the simple project we are discussing, we only need a single *wscript*. It is possible to have more than one *wscript* for even a single project. Consider the *wscript* shown below in the current working directory of our project:

```
def configure(conf):
        print("Waf File  Configured!!!")
def test(tst):
        print("This is the main wscript")
        tst.recurse('src')
```

The *wscript* shown below is in the directory *src* inside the current working directory.

```
def test(tst):
        print("This is the wscript from the directory
src")
```

The commands *waf configure* and *waf* executed from the current working directory will give you the output shown in Figure 3. The line of code *tst.recursive('src')* calls the *wscript* inside the directory *src* along with the *wscript* from the main directory of the project. Now execute the command *cd src* and *waf*. The output of these commands (shown in Figure 3) might seem a bit surprising because the execution of the *wscript* inside the current working directory and the *wscript* inside the directory *src* are both showing the same output.

Even if you are calling Waf from a directory containing a *wscript*, the *wscript* executed is the one that is configured last in one of the parent directories. So if you want only the *wscript* from the directory *src* to take charge of affairs, you need to add a *configure* function in it. This example also tells you something about the default destination of Waf. If you haven't provided the command *out = '.'* or say *out = './dest'*, a default directory called *build* is generated by Waf to store the executable and other associated files.

A few final words about Waf

So we have learned about the installation and use of Waf with a simple C program. But this example and article just cover the bare minimum essential for a novice to

```
[root@deepu waf]# waf configure
Setting top to                           : /root/Desktop/waf1/waf
Setting out to                           : /root/Desktop/waf1/waf/build

Main Waf script  Configured!!!
'configure' finished successfully (0.197s)
[root@deepu waf]# waf test
This is the main wscript
This is the wscript from the directory src
'test' finished successfully (0.001s)
[root@deepu waf]# cd src
[root@deepu src]# waf test
invalid lock file in /root/Desktop/waf1/waf/src
This is the main wscript
This is the wscript from the directory src
'test' finished successfully (0.001s)
```

Figure 3: Execution of multiple Waf scripts

```
[root@deepu waf]# waf
Waf: Entering directory `/root/Desktop/waf/de'
[1/4] c: main.c -> de/main.c.1.o
[2/4] c: data.c -> de/data.c.2.o
[3/4] c: show.c -> de/show.c.3.o
[4/4] cprogram: de/main.c.1.o de/data.c.2.o de/show.c.3.o -> de/hello
Waf: Leaving directory `/root/Desktop/waf/de'
'build' finished successfully (0.794s)
[root@deepu waf]# waf
Waf: Entering directory `/root/Desktop/waf/de'
Waf: Leaving directory `/root/Desktop/waf/de'
'build' finished successfully (0.011s)
```

Figure 4: Back-to-back execution of *wscript*

start working with Waf. The features of Waf that I have left out are innumerable. For example, Waf can build executables in parallel which, all together, reduce the code generation time. Another stunning (and disturbing) fact regarding Waf is that it offers a Turing complete language as part of its repertoire. In simple terms, a tool can be called 'Turing complete' if and only if it is as powerful as the C programming language—at least theoretically. The command *waf distclean* will delete all the files generated by the command *waf* or *waf build*. This is quite an improvement from the Make utility, with which you need to define the function *clean* independently, to remove the files automatically built by Make.

Finally, to understand the single most important benefit of Waf or any other build automation tool in comparison to manual compilation, execute the command *waf* twice on the terminal. I am sure that the second time, Waf will not do anything at all. All you are going to see is the message shown in Figure 4.

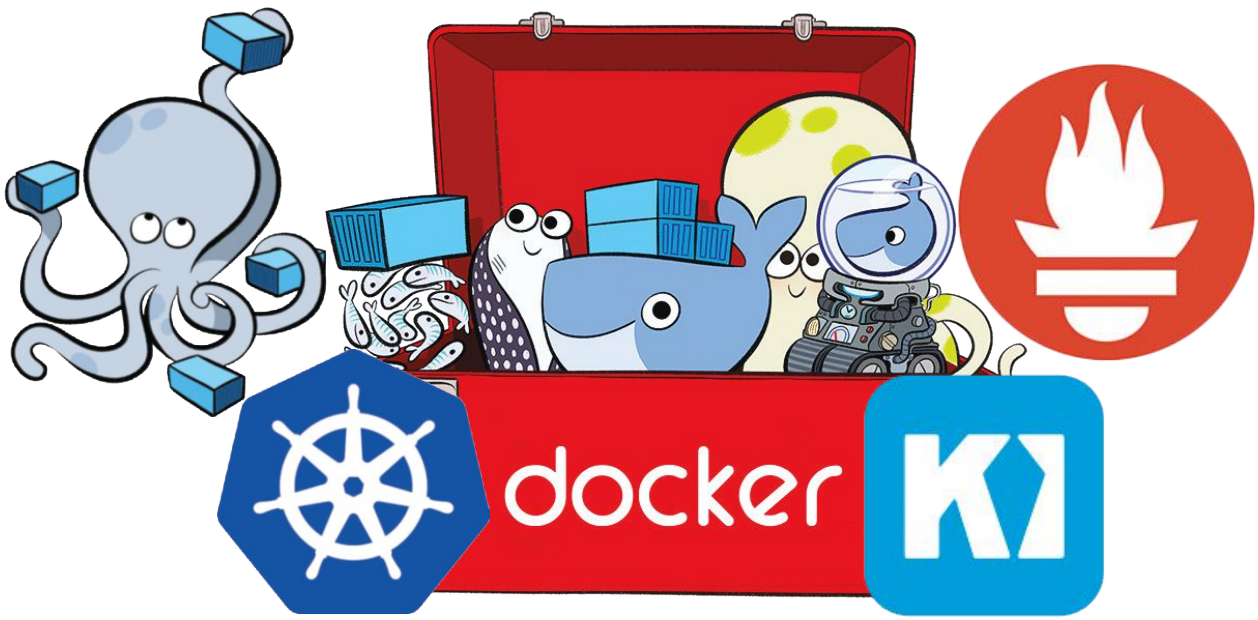
We can conclude that for Waf, '*if there's no modification, then there's no compilation*'. In simple terms, Waf will recompile a source file if and only if it is modified. But it is also possible to forcefully recompile all the source files even when no changes are made. This can be achieved by executing the command *waf clean* on the terminal. END

By: Deepu Benson

The author has nearly 16 years of programming experience, and is currently working as an assistant professor in Amal Jyothi College of Engineering, Kerala. He maintains a technical blog *www.computingforbeginners.blogspot.in* and can be reached at *deepumb@hotmail.com*.

Docker Tools that Developers will Find Useful

Docker allows IT departments to focus on applications, not virtual machines, as a standard unit of production. Looking at recent trends, there have been many projects created by the Docker community that have advanced the developer experience. This article takes a look at a few Docker tools that may be useful for developers.



Docker was introduced to the world with no prior announcement, by Solomon Hykes, founder and CEO of dotCloud at the Python Developers Conference in Santa Clara, California on March 15, 2013. Hykes started Docker in France as an internal project within dotCloud, a PaaS company, with contributions from other dotCloud engineers including Andrea Luzzardi and Francois-Xavier Bourlet. The Docker project was quickly made open source and became available to the public on GitHub in March 2014, with the release of version 0.9.

Docker is an open source project that automates the deployment of Linux applications inside software containers.

As per the official website, *docker.com*, "Docker containers wrap up a piece of software in a complete file system that contains everything it needs to run: code, runtime, system tools, system libraries -- anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in."

Docker can be integrated into various infrastructure tools, including Amazon Web Services, Ansible, CFEngine, Chef, Google Cloud Platform, IBM Bluemix, Jelastic, Jenkins, Microsoft Azure, OpenStack Nova, OpenSVC, HPE Helion Stackato, Puppet, Salt, Vagrant and VMware vSphere integrated containers.

Key benefits of using Docker

Currently, Docker is regarded as the hottest open source project that allows users to deploy applications inside containers, adding a layer of abstraction. It is gaining a lot of traction in the development and DevOps worlds for its consistency across environments. Docker technology has seen consistent and rapid adoption in key enterprises like Microsoft, Google, etc, because of a number of advantages, which are listed below.

- **Simple configuration:** Docker provides simple configuration. VMs, too, can run on any platform with their own *config* on top of the infrastructure. Docker, however, offers the same capability without the overhead of a virtual machine. It lets the user put environment and configuration into code for deployment to be done. Docker gives the user freedom to run applications across multiple IaaS/PaaS solutions without any tweaks.
- **Developer friendly:** Docker provides a developer-friendly environment via a low memory capacity and allows easy running of dozens of services.

- **Rapid application deployment:** Containers include the minimal runtime requirements of the application, which reduces their size and allows them to be deployed quickly.
- **Portability across machines:** An application and all its dependencies can be bundled into a single container that is independent from the host version of the Linux kernel, platform distribution or deployment model. This container can be transferred to another machine that runs Docker, and executed there without compatibility issues.
- **Version control and component reuse:** Users can track successive versions of a container, inspect differences, or roll back to previous versions. Containers reuse components from the preceding layers, which makes them noticeably lightweight.
- **Debugging capabilities:** Docker provides many tools that are not necessarily specific to containers, but work well with the concept of containers. They also provide extremely useful functionality. This includes the ability to checkpoint containers and container versions, as well as to differentiate between two containers. This can be immensely useful in fixing an application.
- **Multi-tenancy:** Using Docker, it is easy and inexpensive to create isolated environments for running multiple instances of app tiers for each tenant. This is possible given the spin up speed of Docker environments and its easy-to-use API, which we can use to spin containers programmatically.

Open source Docker tools for developers

The success of any software project is often measured by the ecosystem it spawns. Projects built around, next to, and on the top of the core technology enhance its power and usability, and they often move the needle forward. Docker allows IT departments to focus on applications, not virtual machines, as a standard unit of production. Looking at recent trends, there have been many projects created by the Docker community that have advanced the developer experience. Although choosing among the great contributions is really difficult, here is a roundup of the most useful open source Docker tools for developers.

Kubernetes

Kubernetes is an open source container cluster manager designed by Google for managing containerised applications in a clustered environment. The objective behind the development of Kubernetes is to provide developers a strong platform for automatic deployment, scaling and operation of application containers across clusters of hosts, providing container-centric infrastructure.

Kubernetes is regarded as portable (public, private, hybrid and multi-cloud), extensible (modular, pluggable, hookable and composable) and self-healing (with auto-replacement, auto-restart, auto-replication and auto-scaling). It can schedule and run application containers on clusters of physical or virtual machines.

Kubernetes satisfies a number of common needs of applications for developers, like co-locating helper processes, mounting storage systems, distributing secrets, checking the application's health, replicating application instances, horizontal auto-scaling, rolling updates, resource monitoring, log access and ingestion, and support for introspection and debugging.

Dockersh

Dockersh has been developed by Yelp as part of its testing and service management infrastructure suite. Dockersh is designed to be used as a login shell on machines with multiple interactive users. When a user invokes Dockersh, it brings up a Docker container, and then spawns a new interactive shell in the container's namespace.

With Dockersh, users enter their own individual Docker containers (acting like a lightweight virtual machine), with the user's home directory mounted from the host system (so that user data is persistent between container restarts), but with its own kernel namespaces for processes and networking. This means that the users are isolated from the rest of the system, and can only see their own processes respectively, and have their own network stack which, in turn, provides better privacy between users. Dockersh eliminates the need for any of these techniques by acting like a regular shell, which can be used in */etc/passwd* or as an SSH ForceCommand.

Dockersh tries hard to drop all privileges as soon as possible, including disabling the *suid*, *sgid*, *raw sockets* and *mknod* capabilities of the target process.

Prometheus

Prometheus is an open source system-monitoring and alerting toolkit developed by SoundCloud. It addresses many aspects of monitoring such as generation and collection of metrics, graphing the results on dashboards, and raising alerts when anomalies occur. It works well for recording any purely numeric time series, and fits both machine-centric monitoring as well as highly dynamic service-oriented architectures.

Its components are:

- *Prometheus server:* Scraps and stores time series data
- *Client libraries:* Used for instrumenting application code
- *Push gateway:* Supports short-lived jobs
- GUI-based dashboard builder based on Rails/SQL
- Special-purpose exporters
- Alert manager, command querying tool and other tools

Prometheus is designed for reliability to allow users to quickly diagnose problems. The Prometheus server is standalone, and does not depend on network storage or other remote services.

The features of Prometheus are:

- A multi-dimensional data model (time series identified by the metric name and key/value pairs)

- A flexible query language to leverage this dimensionality
- Non-reliance on distributed storage; single server nodes are autonomous
- Time series collection happens via a pull model over HTTP
- Pushing time series is supported via an intermediary gateway
- Targets are discovered via service discovery or static configuration
- Multiple modes of graphing and dashboarding support

Docker Compose

Another useful tool for developers is Docker Compose. The prime importance of this tool is that it provides a base for running multi-container Docker applications, along with its dependencies, in single file. Also, applications can run with a single command. All the applications are defined in a YAML file, which comprises all the options used in 'Docker run'.

Docker Compose is a three-step process:

- Defines your app's environment with a Docker file so it can be reproduced anywhere.
- Defines the services that make up your app in *docker-compose.yml*, so that they can be run together in an isolated environment.
- Lastly, runs *docker-compose up* and Compose will start and run your entire app.

Kitematic

Kitematic is an open source project designed to simplify and streamline using Docker on a Mac or Windows system. Kitematic aims to make Docker useful as a desktop-environment developer's tool. It makes the process of downloading Docker images, spinning them up, and managing them into a task no more difficult than, say, using VMs in an application like VMware Workstation.

The Kitematic GUI launches from the home screen and presents curated images that can run instantly. The user can search for any public image on Docker Hub from Kitematic just by typing in the search bar. The GUI is used to create, run and manage containers just by clicking on buttons. Kitematic allows you to switch back and forth between the Docker CLI and the GUI. Kitematic also automates advanced features such as managing ports and configuring volumes. Users can use Kitematic to change environment variables, stream logs, and single-click terminals into your Docker container -- all from the GUI.

The features of Kitematic include:

- *Fast and easy set-up:* One-click install gets Docker running on your Mac, and lets users control app containers from the GUI.
- *Docker Hub integration:* Allows users to easily search favourite images on Docker Hub from Kitematic GUI to create and run your app containers.
- *Seamless experience between the CLI and GUI:* Enables users to seamlessly switch between Kitematic GUI or Docker CLI to run and manage your application containers.
- *Advanced features:* Automatically maps ports and visually changes environment variables, configuring volumes, streamline logs and CLI access to containers.

Logspout

Logspout is a great tool for managing logs generated by programs running inside Docker containers. It manages the log router for the Docker container by attaching all containers to a host, and consists of an extensible module system.

The size of Logspout is just 14MB and it has Busybox at its core, which enables developers to route all the logs of Docker containers to a central location like a single JSON object.

Flocker

Flocker is an open source container data volume orchestrator for Dockerised applications. It migrates data along with containers as they change hosts. Flocker gives users the tools they need to run containerised services like databases in production. Flocker manages Docker containers and data volumes together.

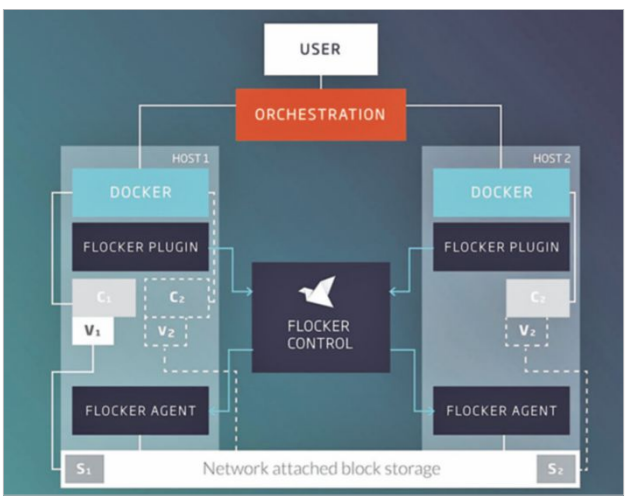


Figure 1: Flocker

Flocker is designed to work with the other tools to build and run distributed applications. It can be used with popular container managers or orchestration tools like Docker, Kubernetes, Mesos, etc. Flocker supports block-based shared storage such as Amazon EBS or OpenStack Cinder to choose the storage back-end that is best for an application.

Its features are:

- Seamless database migrations for micro services
- Integrates into existing Docker workflow

- Runs multiple containers on multiple machines
- Easily moves between development, staging and production

The official website of Flocker is *https://clusterhq.com/flocker/introduction/*.

Powerstrip

Powerstrip, a new open source project from ClusterHQ, provides a useful platform for developers to build powerful Docker prototypes known as Powerstrip adapters. Multiple Powerstrip adapters can be integrated behind the familiar Docker CLI and API.

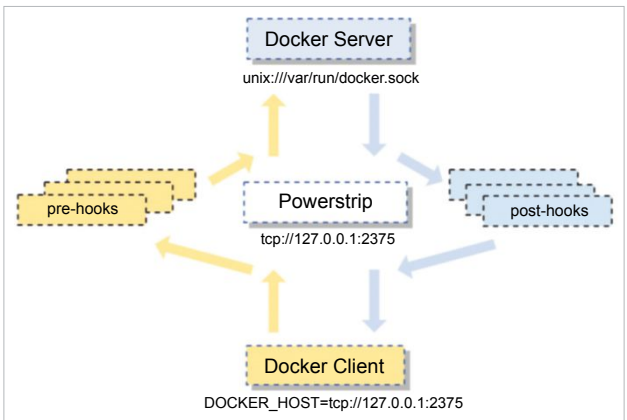


Figure 2: Powerstrip

For example, you can have a storage adapter like Flocker running alongside a networking adapter such as Weave, all playing well with the orchestration framework you choose.

Powerstrip is implemented as a configurable, pluggable, HTTP proxy for the Docker API, which lets users plug multiple Docker extension prototypes into the same Docker daemon.

Powerstrip allows you to build Docker extensions by implementing chained blocking webhooks triggered by arbitrary Docker API calls. The objective behind Powerstrip development is to make it possible for a rich ecosystem of tool builders to offer specialised services, without forcing end users to choose between different Docker experiences.

Helios

Helios, an open source Docker orchestration platform, helps developers in the deployment and management of containers

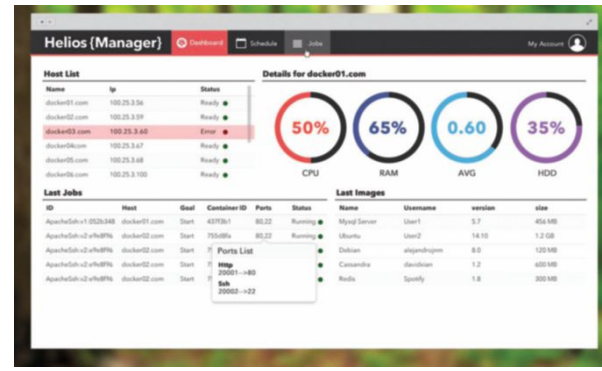


Figure 3: Helios

across a cluster of servers. It is equipped with an API based on HTTP as well as a command based client to connect to servers running Docker containers. Its key element is keeping track of event history, which includes varied information like Docker deployment, Docker restart and even updates or changes of any sort in the Docker version. END

References

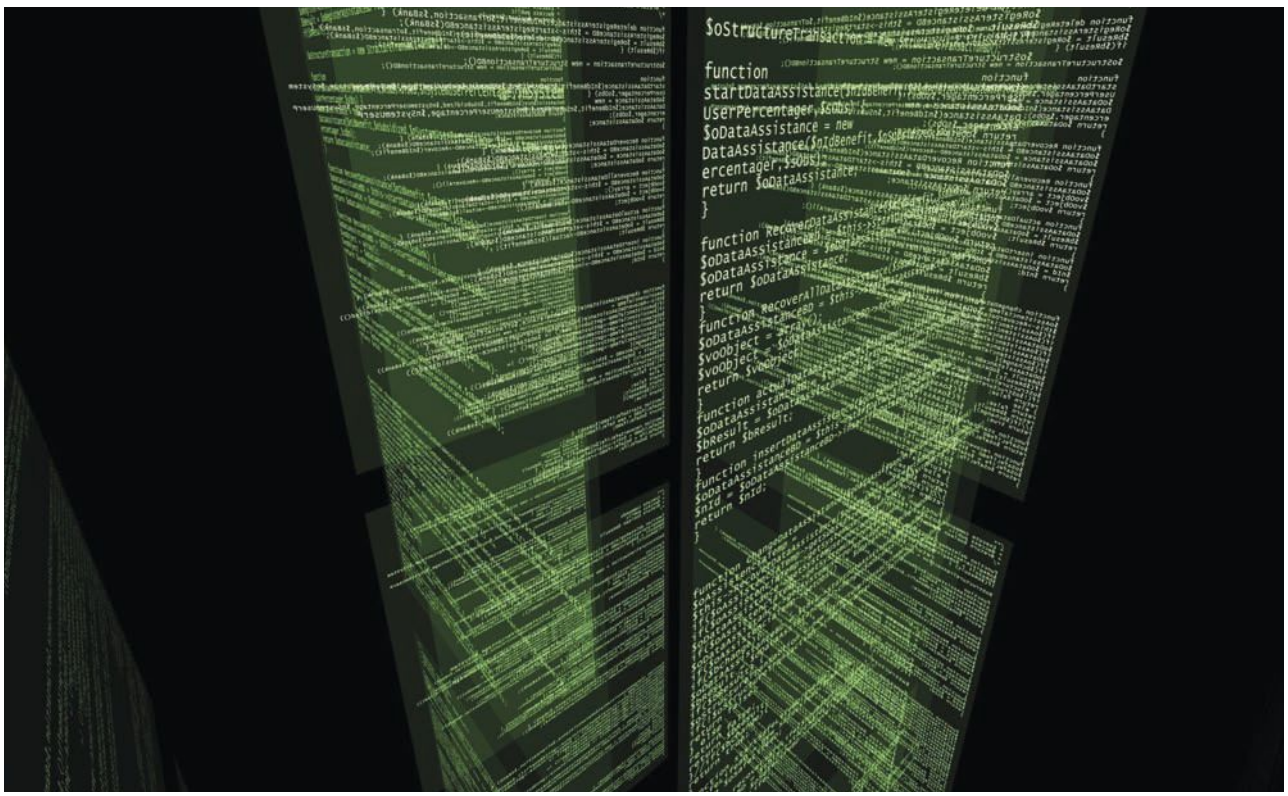
[1] *http://kubernetes.io/*
[2] *https://github.com/Yelp/dockersh*
[3] *https://prometheus.io/*
[4] *https://www.docker.com/products/docker-compose*
[5] *https://kitematic.com/*
[6] *https://github.com/progrium/logspout*
[7] *https://clusterhq.com/flocker/introduction/*
[8] *https://github.com/ClusterHQ/powerstrip*
[9] *https://github.com/spotify/helios*

By: Prof. Anand Nayyar

The author is an assistant professor in the department of computer applications and IT at KCL Institute of Management and Technology, Jalandhar, Punjab. He loves to work on and research open source technologies, cloud computing, sensor networks, hacking and network security. He can be reached at *anand_nayyar@yahoo.co.in*. You can watch his YouTube videos at *youtube.com/anandnayyar*.

Scilab: The Open Source Equivalent of MATLAB

Scilab is free and open source software for numerical computation. It provides engineers and scientists with a powerful environment to do their computation.



Engineers and scientists all over the world use computers for their calculations, yet many of us remain unaware of the software or tools used by a majority of them. Since the invention of transistor in 1947, computing power has increased tremendously, more or less according to the predictions made in Moore's Law. The law states that the number of transistors in a given area will be doubled approximately every two years. In 2015, the transistor count was 5.5 billion in an Intel processor. That is around 5.5 per cent of the 100 billion neurons in the human brain. So, artificial intelligence (AI) is getting closer to human intelligence.

MATLAB is a high level numerical computational platform with IEEE floating point precision of approximately 16 decimal digits. However, space exploration demands a greater level of decimal precision—approximately 25 digits is the count for ISRO. So do NASA and ISRO use MATLAB? Yes, they do, for most of their computation. But they also use Scilab, an open source, high level numerical computational platform.

Scilab can be downloaded from *www.scilab.org*. Taking up about 150MB, it can be installed on your Linux, Windows or Mac OS. Learning Scilab is easy with *http://www.scilab.in* and with the initiatives taken by Prof. Kannan M. Moudgalya at the Indian Institute of Technology, Bombay. The professor's textbook companion project at *www.scilab.in/Completed_Books* contains almost all the engineering numerical problems solved with Scilab. These can be accessed freely, but sadly, most engineering students are unaware of this.

MATLAB

Cleve Moler, the chairman of the computer science department of the University of New Mexico, started developing MATLAB in the late 1970s. The latest release of this proprietary programming language has a memory size of more than 7GB. It has many inbuilt functions and simulation tool bars, which make it easy to compute on various platforms according to a user's needs. It has well written help documentation and wide online community support. For

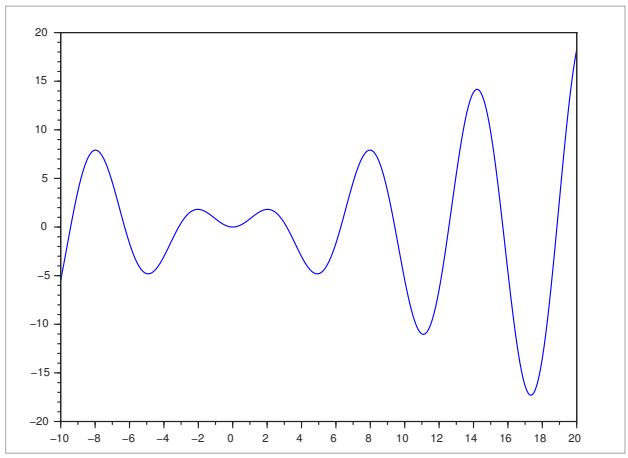


Figure 1: Function

example, if you want help on *function random*, type 'help rand' without quotes in the command window.

Why Scilab then?

Scilab was created in 1990 by researchers from Inria and ENPC in France. It is freely available for all computation platforms. Scilab stores numbers in IEEE 754 double-precision binary floating-point format using 64 bits or 8 bytes of computer memory, which is also 16 decimal digit accuracy.

It has computation for elementary mathematics, linear algebra, numerical integration, differentiation, optimisation, etc. Its interface is similar to MATLAB's and includes a workspace, editor, command window, working directory, etc. It has *xcos*, which is equivalent to *simulink* in MATLAB. Since it doesn't have as many inbuilt functions and tool bars as in MATLAB, many find it less interesting to work with Scilab. However, real researchers, who build from scratch, find Scilab a useful platform. But engineers whose problems are too complex to solve in an exact manner, prefer MATLAB. Engineering research employs many semi-empirical methods that are foreign to pure scientific research–one example being the method of parameter variation. Such methods can be easily implemented in Scilab.

Fire-fly algorithm in Scilab

To feel the power of Scilab, let's implement a swarm intelligence based algorithm in it. Note that we use a single variable for ease of visualisation. The function $f(x) = x \sin(x)$ has a single variable x, and we want to find the value for x such that the function is minimum. Essentially, the above example is a minimisation problem, and these kinds of problems come under optimisation. If we plot the function for variable x, by looking into the function plot, we can infer that the minimum value lies somewhere between 17 and 18. To find it with the fire-fly algorithm, we have the help of 25 fire-flies. We scatter the flies along the function, so that we have flies in different locations along the function plot. The brightness of the flashes given out by each fly increases

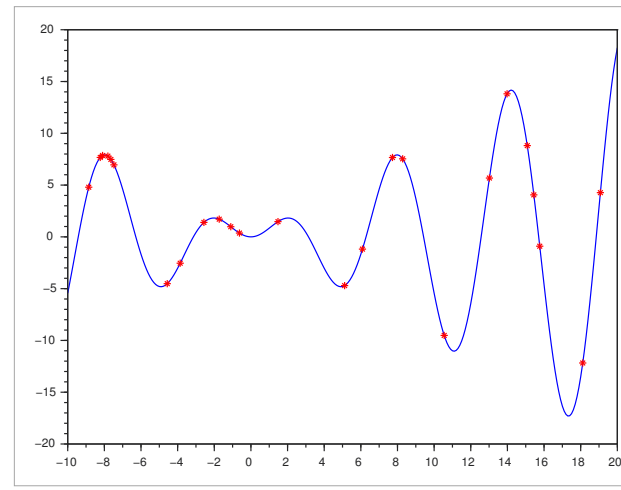


Figure 2: Function with flies

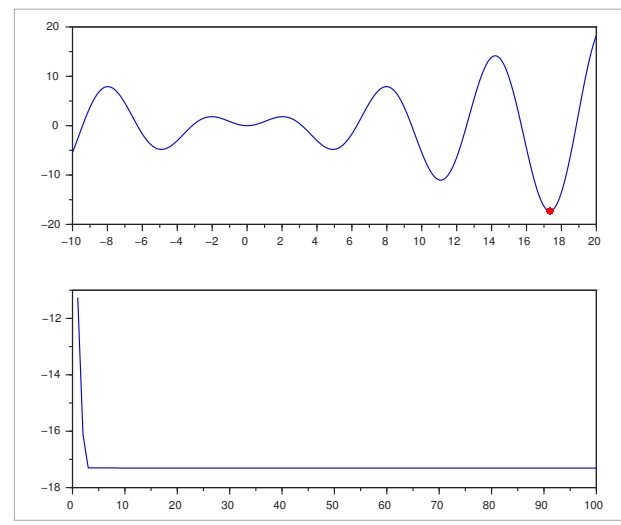


Figure 3: Function convergence

with its proximity to the exact location of the solution. The equation governing this process is:

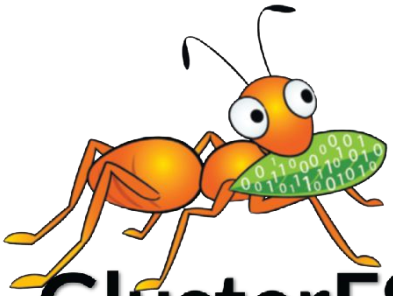
$$x_i = x_i + \beta_j \times e^{rx - r^2_{ij}} \times (x_j - x_i) + \alpha\epsilon_i$$

After 100 iterations, the result obtained is 17.307. This is shown with the convergence, as seen in Figures 1, 2 and 3. Hence, we have seen the action of the fire-fly algorithm in finding the minimum, using Scilab.

Engineering problems, which are more like real world problems, tend to be highly non-linear in nature. They can be easily tackled with evolutionary methods like this. Unfortunately, Scilab is mostly ignored by academia, and its power is often overlooked. END

By: Hithu Anand

The author has a teaching background and is currently researching on smart grids. He can be reached at *hithuanand@gmail.com*.



GlusterFS: A Dependable Distributed File System

The Gluster file system is an open source distributed file, which can scale out in a 'building block' manner to store several petabytes of data. This article will be of interest to newbies and those who wish to know more about file systems.

A distributed file system gives us a way of storing and accessing files in a client-server architecture. We can use more than one server to store data and use multiple clients (local or remote) that can access data from these servers. A distributed file system organises and displays files and directories from multiple servers as if they were stored in your local system, thereby projecting a simple interface to the user or application. The main advantage is that it is easier to distribute documents to multiple clients and provide a centralised storage system so that client machines do not use their resources to store the data.

GlusterFS concepts

Some of the basic terminology that we need to know to understand GlusterFS is given in Table 1.

Introducing GlusterFS

GlusterFS is a distributed file system that can scale up to several petabytes and can handle thousands of clients. It clusters together storage building blocks over RDMA or TCP/IP, and aggregates disk and memory resources in order to manage data in a single global namespace. These blocks do not need any special hardware to store data and are based on a stackable user space design. GlusterFS is a user space file system and hence uses FUSE (file system in user space) to hook itself with the virtual file system (VFS) layer. FUSE is a loadable kernel module that lets non-privileged users create their own file systems without editing kernel code. Gluster uses already tried and tested disk file systems like ext3, ext4, xfs, etc, as the underlying file system to store the data.

Daemons

Before we can create a volume, we must have a trusted storage pool consisting of storage servers that provide bricks to create the volume. Every node has a daemon - *glusterd* - running (this is a management daemon). Each *glusterd* can interact with the *glusterd* on other nodes to add them to its trusted storage pool. This is called peer probing. A node can have multiple

Table 1

Brick	This is the basic unit of storage, represented by an export directory on a machine
Server	This is the machine that hosts the actual file system in which the data will be stored. A node is a server capable of hosting bricks.
Client	This is the machine that mounts the volume (this may or may not be the server).
Trusted storage pool	A storage pool is a trusted network of storage servers. You can dynamically add or remove nodes from the pool. Only nodes that are in a trusted storage pool can participate in volume creation.
Volume	A volume is a logical collection of bricks. Most of the Gluster management operations happen on the volume. Multiple volumes can be hosted on the same node.
Translator	These are the building blocks of GlusterFS. All functions are implemented as translators. They are stacked together to achieve the desired functionality.
Volfile	*.vol* files are configuration files used by the *glusterfs* process. A set of translators stacked to achieve a particular functionality is stored in a volfile, which is usually located at */var/lib/glusterd/vols/volume-name/*

peers. For example, if nodes A and B are peers, and if node A adds node C as a peer, then nodes A, B and C all become peers and can participate in the creation of volumes. We can also remove nodes from the trusted storage pool. To create a volume we can use bricks from any number of nodes, provided they are in the same trusted storage pool. On one node, we can create multiple bricks and hence can host multiple volumes.

As a part of volume creation *glusterd* will generate client and server *volfiles*. A *volfile* is a configuration file that has a collection of translators, where each translator has a specific function to perform and together implement the desired functionality. These files are stacked in a hierarchical structure. A translator receives data from its parent translator, performs the necessary operations and then passes the data down to its child translator in the hierarchy.

Winding fop via the client-server graph

On mounting a volume on a client machine, a *glusterfs* client process is spawned, which interacts with the *glusterd* process and gets a configuration file or the client *volfile*. *Glusterd* also sends a list of ports on which the client can talk to each brick.

When we issue a file operation or *fop*, VFS hands the request to the FUSE module which, in turn, hands the request to the GlusterFS client process. The GlusterFS client processes the request by winding the call to the first translator in the client *volfile*, which performs a specific operation and winds the call to the next translator. The last

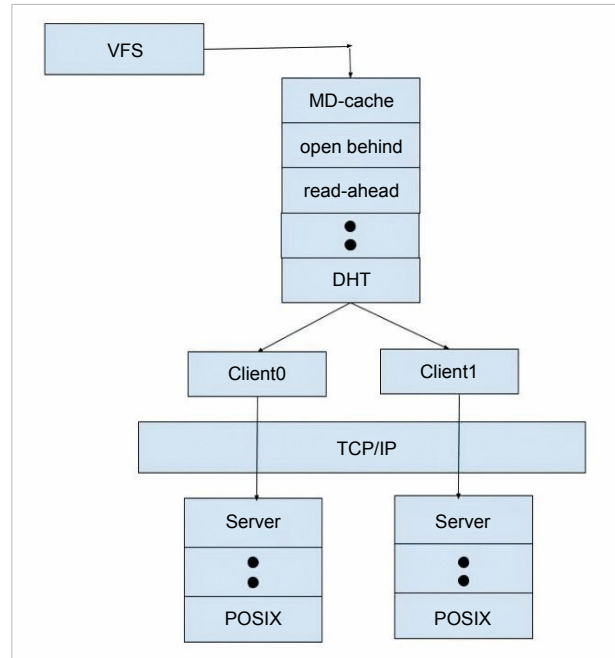


Figure 1: The client-server graph

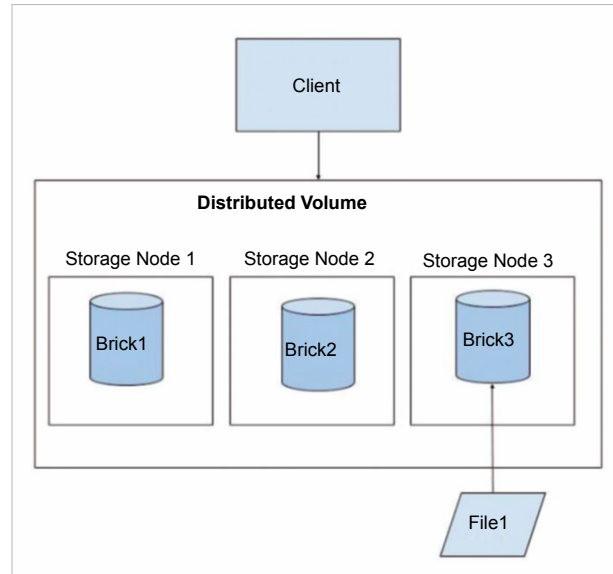


Figure 2: Distributed GlusterFS volume

translator in the client *volfile* is the protocol client, which has the information regarding the server side bricks and is responsible for sending the request to the server (each brick) over the network. On the server stack, the first translator loaded in the protocol server is responsible for getting the package. Depending on the options set, there might be a few more translators below it. Finally, the request reaches the POSIX file system.

Types of volumes

The Gluster file system supports different types of volumes,

depending on the requirements. Some volumes are good for scaling storage volumes, some for improving performance and some for both.

Distributed GlusterFS volume: This is the default GlusterFS volume, i.e., while creating a volume, if you do not specify the type of the volume, the default option is to create a distributed type of volume. In this volume, files are randomly stored across the bricks in the volume. There is no data redundancy. The purpose for such a storage volume is to scale easily. However, this also means that a brick failure will lead to complete loss of data and one must rely on the underlying hardware for protection from data loss.

Replicated GlusterFS volume: In this volume, we overcome the data loss problem faced in the distributed volume. Here, an exact copy of the data is maintained on all the bricks. The number of replicas in the volume can be decided by the client while creating the volume. One major advantage of such a volume is that even if one brick fails, the data can still be accessed from its replica brick. Such a volume is used for better reliability and data redundancy.

Distributed replicated GlusterFS volume: This volume combines the advantages of both a distributed and a replicated volume. In this volume, files are distributed across replicated sets of bricks. The number of bricks must be a multiple of the replica count. Also, the order in which we specify the bricks matters since adjacent bricks become replicas of each other. This type of volume is used when high availability of data due to redundancy and scaling storage is required. So, if there are eight bricks and the replica count is 2, then the first two bricks become replicas of each other; then the next two and so on. This volume is denoted as 4x2. Similarly, if there are eight bricks and the replica count is 4, then four bricks become a replica of each other and we denote this volume as 2x4.

Striped GlusterFS volume: Consider a large file being stored in a brick that is frequently accessed by many clients at the same time. This will cause too much load on a single brick and might reduce its performance. In a striped volume, the data is stored in the bricks after dividing it into different stripes. So, the large file will be divided into smaller chunks (equal to the number of bricks in the volume) and each chunk is stored in a brick. Now, the load is distributed and the file can be fetched faster but no data redundancy is provided.

Reliability

To maintain availability and ensure data safety, it is recommended that you create a distributed replicated volume. We must have the replica pairs on different nodes, so that even if a node comes down, the other replica pair can be used to access data, which is immediately replicated on all pairs. Another data safety feature in GlusterFS is geo-rep. Geo-replication provides asynchronous replication

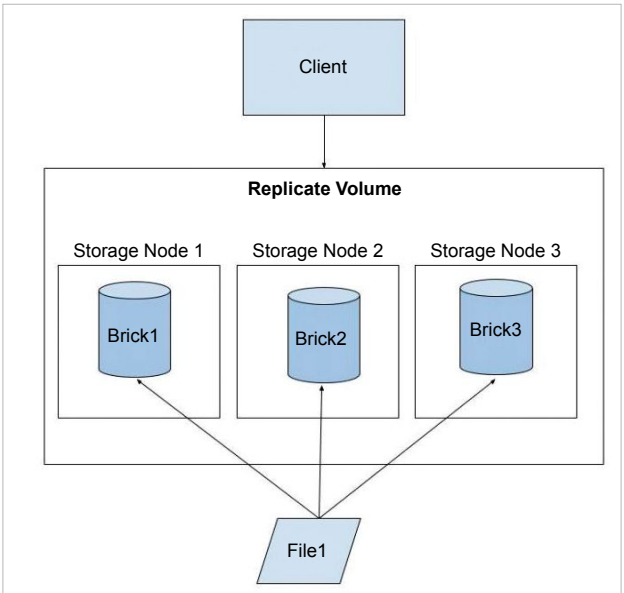


Figure 3: Replicated GlusterFS volume

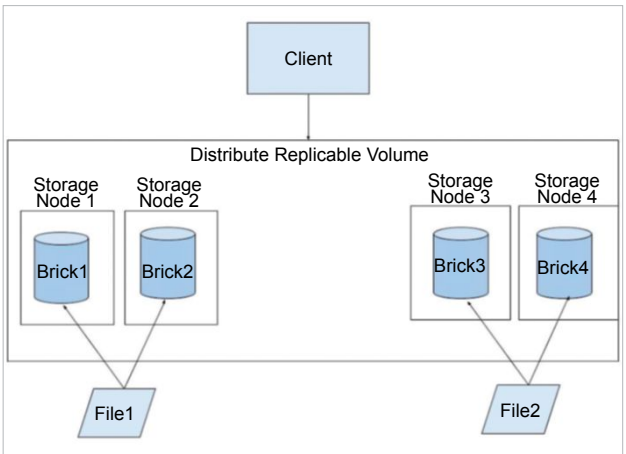


Figure 4: Distributed replicated GlusterFS volume

of data across geographically distinct locations. It mainly works across a WAN and is used to replicate the entire volume, unlike Automatic File Replication (AFR) which is intra-cluster replication (used for replicating data within the volume). This is mainly useful for backup of the entire data for disaster recovery.

GlusterFS performance

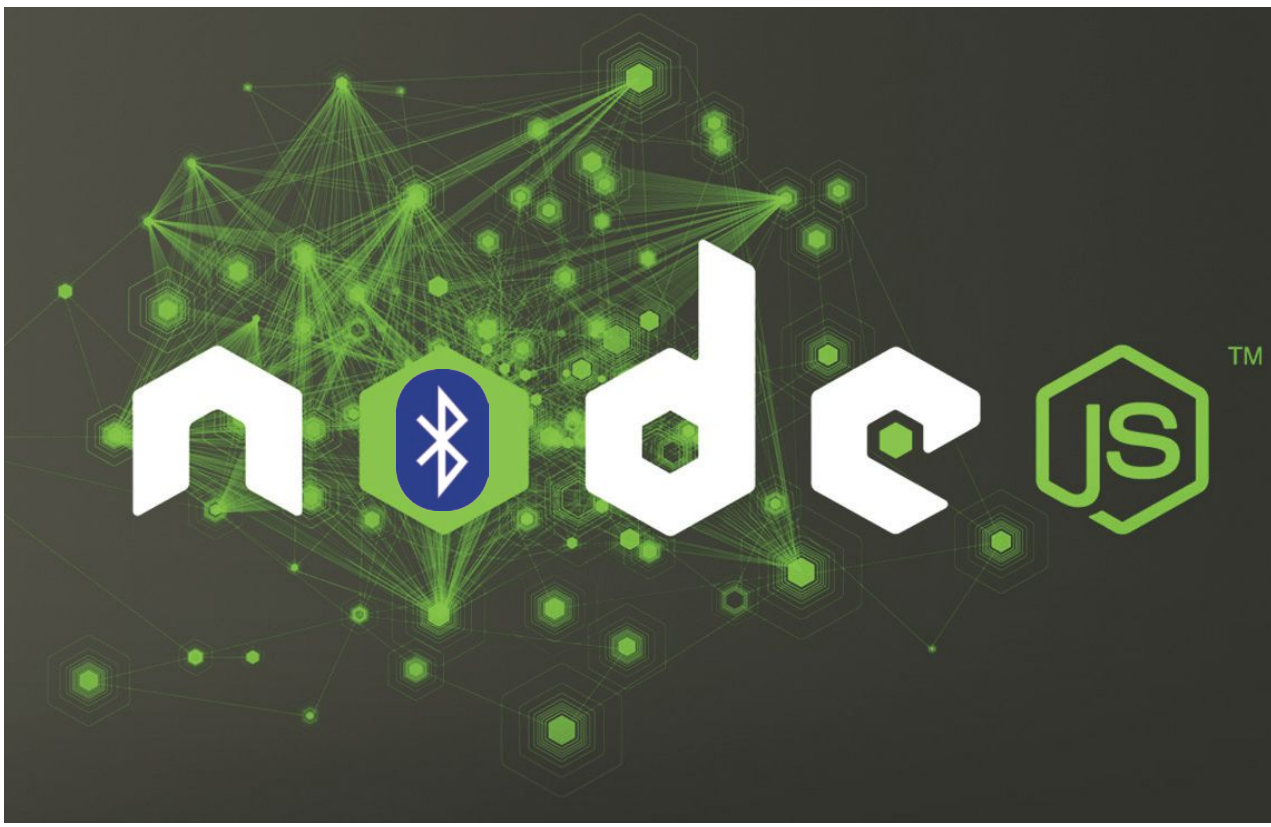
There are some translators that help in increasing the performance in GlusterFS. These, too, are present in the client *volfile*.

Readdir-ahead: Once a read request is completed, GlusterFS pre-emptively issues subsequent read requests to the server in anticipation of those requests from the user. If a sequential read request is received, the content is ready to be immediately served. If such a request is not received, the data is simply dropped.

Continued on Page 86...

Talk to Your Bluetooth Device Using Node.js

Node.js is an event-driven I/O server-side JavaScript environment based on Google's Chrome V8. It is a powerful runtime environment for developing varied tools and applications. This article talks about interfacing Node.js with Bluetooth devices.



Change in the world of technology is constant. This requires every one of us to upgrade ourselves with new technologies and adapt to them. With the Internet of Things (IoT) emerging as a new technology, companies are building new ecosystems for 'things' so that they too can have a piece of cake. Open source frameworks facilitated by the Linux Foundation are adding a dimension to the open things ecosystem.

Embedded computing, which once used to be closer to hardware languages like C, is slowly moving away to high level languages like Java. The reasons for such movement could be multi-fold, ranging from high power hardware that can still provide a better user experience, to the scalability of new generation applications. Such movement necessitates newer frameworks for the high level languages to be able to adapt to the new ecosystem. Node.js is one such framework that is getting adopted to develop a wide range of solutions.

An overview of Node.js

Node.js is an open source JavaScript runtime framework built on Chrome's V8 JavaScript engine. The community activity is governed by the Node.js Foundation and the framework is available from *https://nodejs.org*. The Node.js framework has the largest ecosystem of open source libraries, which extend its functionality considerably. These packages are available at *https://www.npmjs.com/*. The implementation of the framework can be explored at *https://github.com/nodejs*. Having understood in brief how the framework works, let's explore it in detail and understand the internal architecture.

The Node.js framework is part of the server side JavaScript, with the goal of offering a scalable framework to develop high performance network applications. This goal is realised by providing a single-threaded, event driven, non-blocking I/O framework. The framework is implemented in C/C++ with interface APIs in JavaScript. The block diagram

in Figure 1 illustrates the overall structure of the Node.js framework.

The major blocks that the Node.js framework consists of are the interface APIs, a thin core and low lying libraries for event management, the HTTP server, security and the V8 engine. The framework also provides for scaling up the functionality as add-ons, as detailed in the next section.

Event loop

In an embedded firmware solution that has no scheduler to manage its different activities, a design involving interrupts and an infinite loop is used. The system's activities, triggered by external interrupts or internal timer interrupts, are created as events and managed in the infinite loop outside the interrupt context. The way the loop manages the events is based on the application and the needs of the embedded solution.

On similar principles, the Node.js framework uses the event loop to manage different activities of the framework. This implementation is done in the lower library *libuv* available in *http://libuv.org/*. *Libuv* is a cross-platform library aimed at providing a framework for event-driven asynchronous I/O. Figure 2 illustrates the event loop of the Node.js framework.

A detailed explanation of how the event loop is implemented within the Node.js framework is available at *https://github.com/nodejs/node/blob/master/doc/topics/event-loop-timers-and-nexttick.md*.

Non-blocking I/O

Asynchronous I/O operations using call back form the core part of the non-blocking I/O operation of the Node.js framework, as illustrated in Figure 3.

To provide a non-blocking design, the framework offloads I/O operations to the underlying operating system and is managed within the event loop. Once the operation is completed, the operating system will get back to the Node.js framework. This control back to the framework is via the call back that is passed while calling the I/O operation.

Talking to your Bluetooth device over Node.js

To scale up, Node.js extends the functionality by installing add-ons as packages. In this following example, let us explore how to interface Node.js with the Linux Bluetooth framework. The set-up is illustrated in Figure 4.

This section assumes that the reader has prior understanding of the BlueZ Linux driver and the DBus framework.

For example, the application that runs on the Node.js framework is a simple Web server that discovers Bluetooth devices in the vicinity and lists them on the browser. Let's assume that a Linux system with a BlueZ stack is running a Node.js Web server with the package *dbus-native*. Given below is the code snippet of the application that creates a

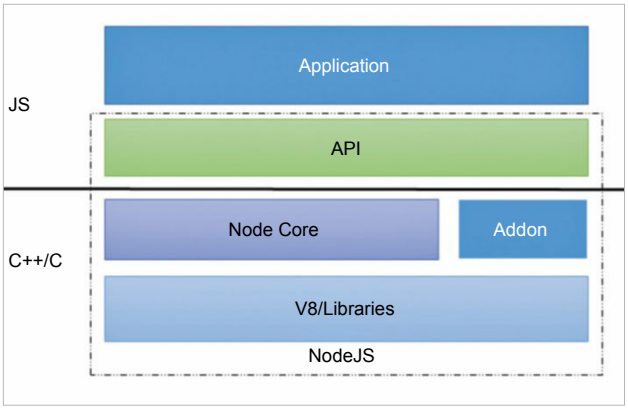


Figure 1: Blocks of the Node.js framework

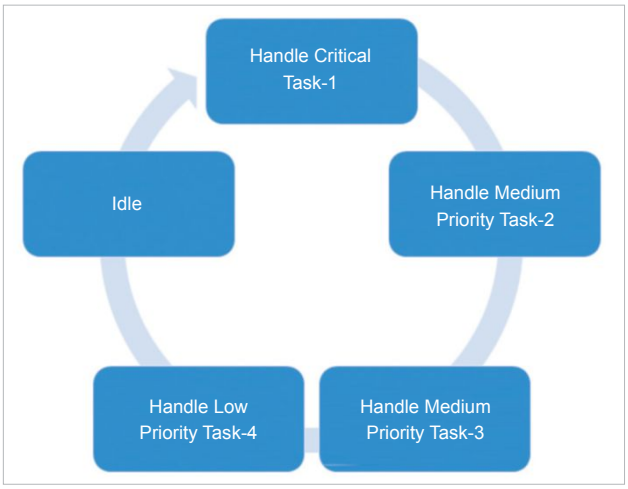


Figure 2: Illustrative representation of an event loop

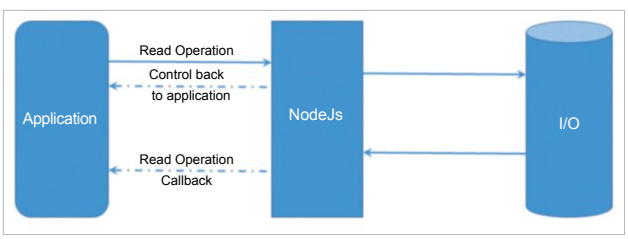


Figure 3: A representation of non-blocking I/O

Web server on port '8080' and starts discovering Bluetooth devices.

```
var dbus = require('dbus-native');
var sBus = dbus.systemBus();

var systemBus = dbus.systemBus();
var btService = systemBus.getService('org.bluez');

bt.getInterface('/org/bluez/hci0','org.bluez.
Adapter1',function(err,Intf){
   Intf.StartDiscovery();

});
```

```
var http = require("http");

const server = http.createServer((request,response)=>{

btService.getInterface('/','org.freedesktop.DBus.ObjectManage
r',function(err,device_intf){
     device_intf.on('InterfacesAdded',function(devname){
       console.log(devname);
       response.statusCode = 200;
       response.setHeader('Content-Type', 'text/plain');
       response.write('New Device Path : '+devname);
       response.end();
     });
});
}).listen(8080);

console.log('Server started');
```

After running the application on the Node.js framework, open a browser and listen to port 8080 of the Web server. The browser will start listing the Bluetooth devices in the vicinity, as illustrated in Figure 5. END

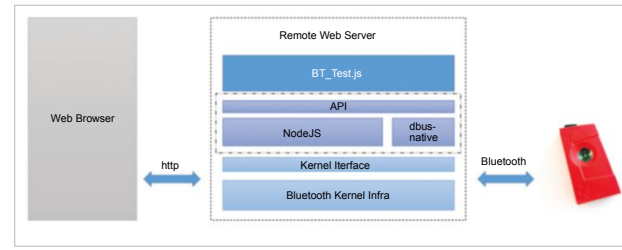


Figure 4: A representation of the Node.js HTTP server in the Linux Bluetooth ecosystem

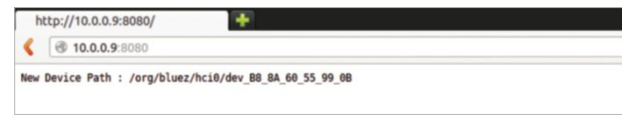


Figure 5: Node.js HTTP server response on a browser

By: Rajaram Regupathy

The author is an embedded software professional with hands-on experience in successfully delivering software/hardware products/solutions. He has published books and articles on various topics ranging from open source to FPGA which has got multiple citations.

Continued from Page 83...

IO-cache: This translator is used to cache the data that is read. This is useful if many applications read data multiple times and reads are more frequent than writes. The cache is maintained as an LRU list. When *io-cache* receives a write request, it will flush the file from the cache. It also periodically verifies the consistency of the cached files by checking the modification time on the file. The verification timeout is configurable.

Write-behind: This translator improves the latency of a write operation by relegating it to the background and returning to the application even as the write is in progress. Using the write-behind translator, successive write requests can be pipelined, which helps to reduce the network calls. Without the write-behind translator, once an application has issued a write call, it goes to the VFS and then to FUSE, which initiates a write call that is wound till it reaches the protocol client translator, which sends it to the server. Once the client receives a reply from the server, it returns it to the FUSE. With a write-behind translator, once the application issues a write call, it reaches VFS, then FUSE and then initiates the write call which is wound only till the write-behind translator. Write-behind adds the write buffer to its internal queue and returns with success. Write-behind initiates a fresh *writev()* call to its child translator, whose replies will be consumed by write-behind itself. Write-behind doesn't cache the write buffer.

Stat-prefetch: The purpose of this translator is to store the stat information. It helps to optimise operations like *ls -l*, where a readdir is followed by a stat on the entry. This translator needs to maintain the following *fops*.

1. *Lookup:* Checks if the entry is present, and if so, then the stat information is returned from this translator rather than winding the call till the server. If the entry is not present, then we wind the lookup call and cache the stat information.
2. *Chmod/chown:* If a *chmod* or *chown* is performed, then the entry must be removed, since these calls change the *ctime* of directories.
3. *truncate/ftruncate/setxattr:* The entry must be removed, since these calls change the *mtime* of the directories.
4. *Rmdir:* Deletes the entry from the cache and then proceeds to remove the entry from the back-end. END

References

[1] *https://gluster.readthedocs.io/en/latest/*
[2] *https://www.gluster.org/wp-content/uploads/2012/05/Gluster_File_System-3.3.0-Administration_Guide-en-US.pdf*

By: Sakshi Bansal

The author has completed her bachelor's in computer science engineering from Amrita Vishwa Vidyapeetham. She has contributed to various open source projects such as Mozilla Thunderbird and Mediawiki. She blogs at *http://sakshiii.wordpress.com/*.

Be Up to Date with Technology
Strengthen Reading Habits

And, enjoy these FREE Gifts too...

You & Me

YOGG Wrist Band

Your Personal Fitness and Activity Tracker

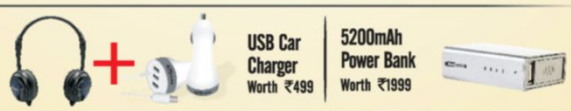
Worth ₹2999

Worth ₹1599

Sound Pot

Portable, Aesthetically Designed and Rubberised Premium Finish Bluetooth Speaker

Wireless Headphone Worth ₹1999 + USB Car Charger Worth ₹499 | 5200mAh Power Bank Worth ₹1999



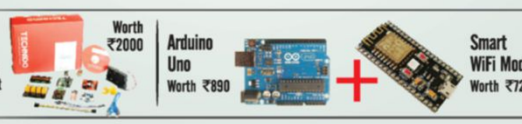
Techies

Raspberry Pi 3

Latest version of Raspberry Pi is a low cost, credit-card sized computer

Worth ₹3300

IoT Starter Kit – Self Learning Kit for Hobbyist, Enthusiast and Students – Worth ₹2000 | Arduino Uno Worth ₹890 + Smart WiFi Module Worth ₹725



ORDER FORM

Magazine	Duration	Cover Price (₹)	You Pay (₹)	Gifts For You & Me (Please Tick Any One)	Gift for Techies
Open Source For You with DVD	1 Year	1440	1440	☐ 5200mAh Power Bank - worth Rs 1999	☐ Arduino Uno worth Rs 890 + Smart WiFi Module –worth Rs 725
	2 Years	2880	2880	☐ Wireless Headphone - worth Rs 1999 + USB Car Charger – worth Rs 499	☐ IoT Starter Kit – worth Rs 2000
	5 Years	7200	7200	☐ Yogg WristBand - worth Rs 2999 + Sound Pot – worth Rs 1599	☐ Raspberry Pi 3 – worth Rs 3300

Free e-zine Access With Every Subscription | To Subscribe Online, visit: http://subscribe.efyindia.com

Name_____ Designation_____ Organisation_____

Mailing Address_____

City_____ Pin Code_____ State_____ Phone/Mobile_____ Email_____

Subscription No. (for existing subscribers only_____. **I would like to subscribe Open Source For You starting with the next issue.** Please find enclosed a sum of Rs_____ by DD/MO/crossed cheque*bearing the No._____ dt._____ in favour of **EFY Enterprises Pvt Ltd,** payable at Delhi. *(*Please add Rs 50 on non-metro cheque)*

Send this filled-in form or its photocopy to: **EFY Enterprises Pvt Ltd** D-87/1, Okhla Industrial Area, Phase 1, New Delhi 110 020 | Ph: **011-26810601-03** | Fax: **011-26817563** | e-mail: **info@efy.in** | **www.efy.in**

EFYGROUP Technology Drives Us

Terms & conditions: # This offer is applicable for new subscriptions and renewal by existing subscribers. # The rates are valid for subscribers within India only. # Please allow 4-6 weeks for processing your subscription and 6-8 weeks to send your gifts # Replacement copies will be sent only if intimation of damaged / non-receipt of copies is received within 15 days of publication # If you are not satisfied with the magazine or our services, we shall refund the balance amount after three months. # Disputes, if any, are subject to exclusive jurisdiction of competent courts and forums in Delhi/New Delhi only.

Open source offers more flexibility, agility and security to Bharti Airtel

The New Delhi-headquartered Bharti Airtel has over 257 million subscribers across the country, as per the recent data released by the Telecom Authority of India (TRAI). It is also one of the youngest firms to use open source software on a large scale.

Apart from being a market leader in the telecom world, Airtel is widely known for its broadband and DTH (direct-to-home) services. Though the group, founded by veteran entrepreneur Sunil Bharti Mittal, is not directly related to the IT industry, it has recently started contributing back to the community.

"Open source has helped us build elastic, scalable systems at low cost and hence increase our productivity," says Harmeen Mehta, global chief information officer, Bharti Airtel.

Airtel has deployed a large number of open source technologies to improve efficiency. These include solutions such as Redis, Nginx, Apache Tomcat, Drools, JBoss Wildfly and Spring. Besides this, the company's vast user data is stored primarily on community-based offerings like AeroSpike DB, MongoDB, OrientDB and PostgresDB.

"The telecom industry is all about technology and innovation. And today, more innovation is happening using open source technology than with proprietary software. This is a testament to all the committers and contributors around the world who have helped completely change this landscape," Mehta told *Open Source For You* in an exclusive interaction.

Making the switch from proprietary to open source software

Earlier, however, Airtel had opted for proprietary software over open source solutions. Anant Kumar, general manager and head of product engineering, has helped the company make a swift shift from using proprietary solutions to open source technologies. Kumar heads a team of more than 80 engineers at Airtel's Gurgaon office.

"Open source lets us take control in order to make our own decisions smoothly, along with contributing back to the

community. It enables us to be more lean, agile and secure and, hence, move faster in the competitive world," says Kumar.

'More exciting than challenging'

Although shifting to open source turned out to be a fruitful decision for Airtel, the transition was not a breeze. However, Kumar terms this transition "more exciting than challenging."

The main challenge that Kumar and his team faced while switching to open source was to bridge the knowledge gap. Airtel preferred reskilling the existing team and hired fresh talent only to make the transition smoother.

"Having the right team is very important. The engineering team does look for new co-workers who enjoy doing their work and are curious to discover the given solutions. At times, we ask them to code during interviews just to see how they think," Kumar says.

Kumar adds that his engineering crew picks solutions that are tried and tested by various large- and small-scale organisations, have a large community base, elaborate documentation and are available without any restrictive or expensive licensing terms and conditions. Moreover, the team follows the agile development methodology and believes in prototyping technologies before their final deployment.

"Following agile development methodology helped us," says Kumar, adding, "With rapid prototyping, we can judge the fitness of a technology for an application."

Anant Kumar, head of engineering, Bharti Airtel, exploring new open source developments with him team

Six major projects that use open source technologies

Airtel has as many as six major projects based on open source solutions. There is the Mitra app for retailers to acquire new customers and reverse wrongly made recharges; the Fibre Force app to help NMT engineers measure a patroller's performance; the Decision Tree app for call centre employees; the Hive social network app and the Project Leap open network initiative. Also, the telecom giant has recently developed its eCAF (electronic customer acquisition form) to digitise customer acquisition using Aadhaar biometric data.

The development of eCAF was aimed at offering instant activation of mobile connections and to enable paperless processes for account activation. "We were getting almost

Anant Kumar, general manager and head of product engineering, Airtel

Airtel's open source based projects

EeCAF: An app to completely digitise customer acquisition using the Aadhaar number and biometric fingerprint, hence reducing activation TAT (turnaround time) from a few days to a few minutes.

Mitra app: An app empowering retailers to acquire new customers, reverse wrong recharges and keep them up-to-date with their account and services -- ensuring complete synergy between the company's and retailer's achievements.

Decision Tree: A one-stop-shop for call centre employees, which reduces the average call handling and training time considerably.

Fibre Force app: This app enables NMT engineers to measure a patroller's performance for inter-city and intra-city fibre maintenance. It helps to improve operational efficiency and also increase GIS accuracy in the long run. This will also ensure that risk-prone areas are proactively allocated additional resources.

Hive: A social networking app for Airtel employees, which enriches internal communication.

Project Leap: An app that provides an aggregate experience metric, thus enabling the planning for future networks as well as resolving existing network issues.

300,000 acquisitions a day through the traditional paper-based process pan-India. But with our innovative eCAF, we are now getting almost 200,000 acquisitions in a day with just 10 per cent of our total retailers on-board," Kumar reveals.

Likewise, the building of the Decision Tree app for optimising its call centre agents helped Airtel in reducing average call handling time. "The algorithm of the Decision Tree engine diagnoses the customer's problem in real-time and provides insight to the agent on how to resolve it. The agent does not have to toggle between screens and multiple back-end applications, or check customer history and usage records to solve the query," explains Kumar.

Decision Tree increased agent efficiency and reduced average handling time for calls by almost 50 per cent. These benefits also directly resulted in some unexpected cost savings for the telco.

Airtel has deployed MongoDB, Redis and Drools to enable Decision Tree for its agents.

Going forward, Airtel is also in the process of integrating its open source-backed eCAF solution within its payment bank. Kumar says that this move will help the masses open a bank account with Airtel instantly and support the ongoing digital payment movement in the country.

Community support to enhance developments

Airtel is using open source communities to improve its existing models and find ways to develop some innovations. "A great community can help to solve issues and provide a resolution to a problem," Kumar states.

The engineering team members at Airtel's premises mainly rely on Stack Overflow and GitHub. Kumar feels case studies are a vital source for gaining knowledge. "Case studies of companies using open source products provide us inputs on architecture and technology fitments," he says.

Contributing back through strategic partnerships

In addition to leveraging community support, Airtel is contributing back to the open source world through some strategic partnerships with startups and enterprises that are building new open source based solutions. "We do not want to limit ourselves to the proprietary world at all, and would like to fully embrace open source; we are now even starting to contribute back," Mehta proclaims.

California-based Aerospike is among the enterprises that has so far received support from Airtel. The startup offers a Flash-optimised in-memory open source NoSQL database that operates in three distinct layers -- a Flash-optimised data layer, a self-managed distribution layer and a cluster-aware client layer.

Harmeen Mehta, global chief information officer, Bharti Airtel

Kumar told *Open Source For You* that his team used skills from Aerospike to address some issues in the telecom space. "Aerospike has open sourced its product for the developer community. We have involved the firm to help us crack complex problems in some of the core telco-domain applications," he says.

The right mix of open source and proprietary software

Choosing an appropriate combination between open source and proprietary technologies is not child's play. According to Kumar, the engineers at Airtel are using the following thumb rule. "Wherever we find that our application requirements are not being met by a licensed product, we go for open source," states Kumar.

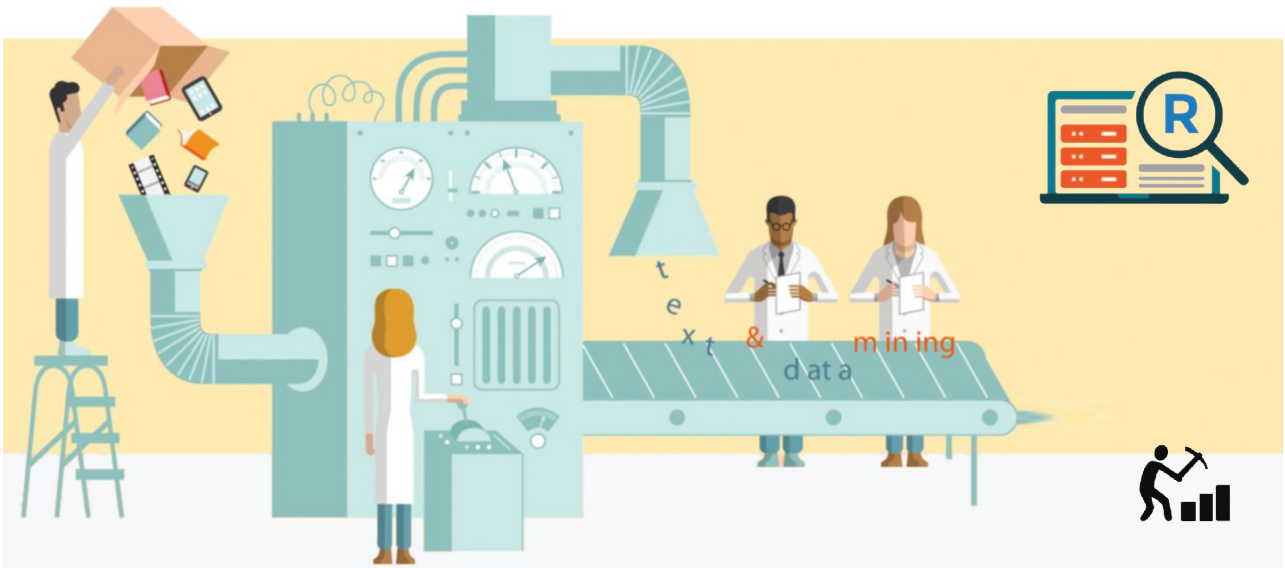
The team mainly contemplates open source solutions rather than proprietary technologies because of the former's stability, available documentation and ease of implementation.

"We are really happy with the choices that we have made," Mehta affirms. END

By: Jagmeet Singh
The author is an assistant editor at EFY.

An Introduction to Text Mining with R

The vast quantity of data, textual or otherwise, that is generated every day has no value unless processed. Text mining, which involves algorithms of data mining, machine learning, statistics and natural language processing, attempts to extract some high quality, useful information from the text.



Text mining, in general, means finding some useful, high quality information from reams of text. More specifically, text mining is machine-supported analysis of text, which uses the algorithms of data mining, machine learning and statistics, along with natural language processing, to extract useful information. It covers a wide range of applications in areas such as social media monitoring, recommender systems, sentiment analysis, spam email classification, opinion mining, etc.

Whatever be the application, there are a few basic steps that are to be carried out in any text mining task. These steps include preprocessing of text, calculating the frequency of words appearing in the documents to discover the correlation between these words, and so on. R is an open source language and environment for statistical computing and graphics. It includes packages like tm, SnowballC, ggplot2 and wordcloud, which are used to carry out the earlier-mentioned steps in text processing.

Getting started

The first prerequisite is that R and R Studio need to be installed on your machine. R Studio is an integrated development environment (IDE) for R. The free open source versions of R Studio and R can be downloaded from their respective websites.

Once you have both R and R Studio on your machine, start R Studio and install the packages tm, SnowballC, ggplot2 and wordcloud, which are usually not installed by default. These packages can be installed by going to *Tools -> install packages* in R Studio.

Loading data

First, let us start by loading the data, which can be any text (or collection of texts, commonly referred to as a corpus) from which we want to extract useful information. To illustrate the process in this tutorial, we have used five text documents related to our favourite superhero, Batman!

For this step, first load the library tm and then use the function *Corpus()* to create a collection of documents. This loads our text documents residing in a particular folder into a corpus object.

```
library(tm)
docs <- Corpus(DirSource("path to your folder"))
docs
<<VCorpus>>
```

Gift Yourself Knowledge

And get more gifts from us

OFFERS FOR YOU & ME

Portronics

YOGG Wrist Band

Your personal fitness and activity tracker

10W LED Bulb
Branded 10W LED bulb with a two-year warranty

USB Car Charger
Three USB ports with micro USB cable

10W LED Bulb + Three Night Lamps
Set of one 10W LED bulb with three 0.5W night lamps

ORDER FORM

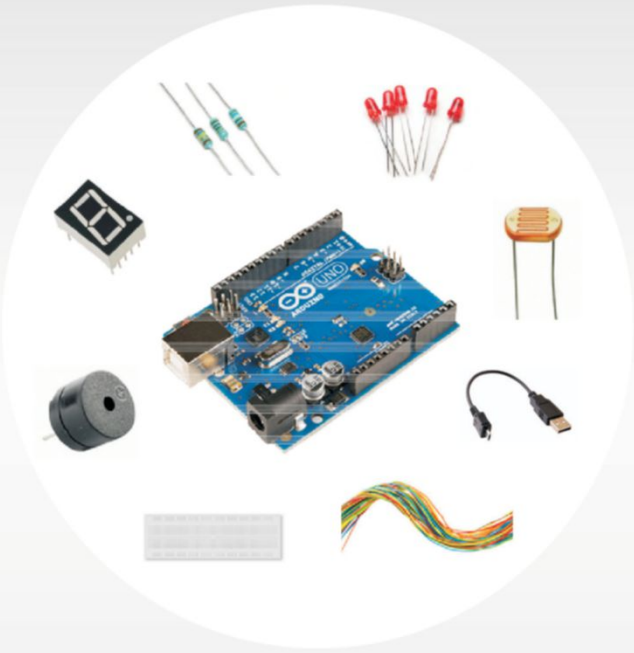
Magazine	Duration	Cover Price (₹)	You Pay (₹)	You Save (₹)	Gifts For You & Me
Electronics For You	1 Year	720	625	95	☐ 10W LED Bulb
	2 Years	1440	1150	290	☐ USB Car Charger
	3 Years	2160	1725	435	☐ 10W LED Bulb+3 Night Lamps
	5 Years	3600	2880	720	☐ Yogg Smart Wristband

To Subscribe Online, visit:
http://subscribe.efyindia.com

Technology Advisers for Businesses and Techies

SUBSCRIBE TO ELECTRONICS FOR YOU, SAVE MONEY & GET GIFTS

OFFERS FOR TECHIES



Arduino Starter Kit

A perfect kit to help you get started with Arduino projects.

Digital Multimeter

Use it to measure voltage, current, resistance and continuity of the circuit.

Basics Of Electronics Kit

A simple electronic kit which is packed with all the important components to help you learn by doing it yourself

IoT Development

A compact IoT embedded module for development of Internet based projects

ORDER FORM

Magazine	Duration	Cover Price (₹)	You Pay (₹)	You Save (₹)	Gifts For Techies
Electronics For You	1 Year	720	575	145	☐ Digital Multimeter
	2 Years	1440	1150	290	☐ Basic Of Electronics Kit/Components Kit
	3 Years	2160	1725	435	☐ IoT Development Board
	5 Years	3600	2800	720	☐ Arduino Starter Kit

Free ezine
Access With Every Subscription

Name_____ Designation_____ Organisation_____

Mailing Address_____

City_____ Pin Code_____ State_____ Phone/Mobile_____ Email_____

Subscription No. (for existing subscribers only)_____. **I would like to subscribe to Electronics For You starting with the next issue.** Please find enclosed a sum of Rs_____ by DD/MO/crossed cheque*bearing the No._____ dt._____ in favour of **EFY Enterprises Pvt Ltd,** payable at Delhi. *(*Please add Rs 50 on non-metro cheque)*

Send this filled-in form or its photocopy to: **EFY Enterprises Pvt Ltd** D-87/1, Okhla Industrial Area, Phase 1, New Delhi 110 020 | Ph: **011-26810601-03** | Fax: **011-26817563** | e-mail: **info@efy.in** | **www.efy.in**

EFYGROUP *Technology Drives Us*

Terms & conditions: # This offer is applicable for new subscriptions and renewal by existing subscribers. # The rates are valid for subscribers within India only. # Please allow 4-6 weeks for processing your subscription and 2 weeks to send your gifts # Replacement copies will be sent only if intimation of damaged / non-receipt of copies is received within 15 days of publication # If you are not satisfied with the magazine or our services, we shall refund the balance amount after three months. # Disputes, if any, are subject to exclusive jurisdiction of competent courts and forums in Delhi/New Delhi only.

```
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 5
```

The above method is to be used when you want to perform text processing on data present on your local machine in a folder, but if you want to load a dataset from an online repository, R allows you to directly load the dataset into your project as mentioned below.

To understand how to load data in R directly from the Internet, we use the famous Iris dataset (stored in CSV format) available from the UCI Machine Learning Repository (*http://archive.ics.uci.edu/ml/*), which provides free access to several datasets that can be used to test many machine learning algorithms.

Download and load the library RCurl which allows fetching of URIs, HTTP requests, *get* and *post* forms, and then process the results returned from the Web server, as follows:

```
library(RCurl)
url <-'https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data'
 docs <- getURL(url, ssl.verifypeer=FALSE)
connection <- textConnection(docs)
dataset <- read.csv(connection, header=FALSE)
head(dataset)
  V1  V2  V3  V4          V5
1 5.1 3.5 1.4 0.2 Iris-setosa
2 4.9 3.0 1.4 0.2 Iris-setosa
3 4.7 3.2 1.3 0.2 Iris-setosa
4 4.6 3.1 1.5 0.2 Iris-setosa
5 5.0 3.6 1.4 0.2 Iris-setosa
6 5.4 3.9 1.7 0.4 Iris-setosa
```

Preprocessing

Preprocessing the text means removing punctuations, numbers, common words, etc. This step is performed to clean the data to increase the efficiency and robustness of our results by removing words that don't necessarily contribute to our analysis.

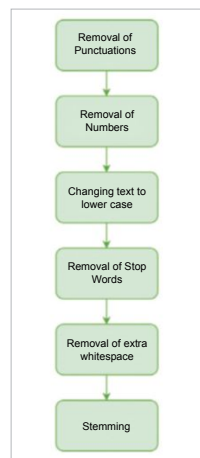


Figure 1: Text preprocessing

Removal of punctuation: To remove all punctuation, use the following command:

```
docs <- tm_map(docs, removePunctuation)
```

Removal of numbers: The following command will remove the numbers in the text:

```
docs <- tm_map(docs, removeNumbers)
```

Changing the text to lower case: The command shown below will change the text being mined into lower case:

```
docs <- tm_map(docs, tolower)
```

Removal of stop words: Common words like 'but', 'and', etc, are called stop words. The following command will remove them:

```
docs <- tm_map(docs, removeWords, stopwords("english"))
```

Removal of extra whitespaces: The following command will remove the extra whitespaces:

```
docs <- tm_map(docs, stripWhitespace)
```

Stemming: Stemming is the procedure of converting words to their base or root form. For example, *playing*, *played*, *plays*, etc, will be converted to *play*.

To perform stemming, use the command shown below:

```
library(SnowballC)
docs <- tm_map(docs, stemDocument)
```

Document term matrix

A document term matrix (DTM) depicts the frequency of a word in its respective document. In this matrix, the rows are represented by documents and the columns are represented by words. How often a word occurs in a document, is what determines the value ascribed to it—if it appears only once, its value is 1; if it appears twice, its value is 2; and if it does not appear at all, then the value is 0.

To create a DTM for our corpus, type:

```
dtm<- DocumentTermMatrix(docs)
dtm
<<DocumentTermMatrix (documents: 5, terms: 2566)>>
Non-/sparse entries: 3449/9381
Sparsity           : 73%
Maximal term length: 23
Weighting          : term frequency (tf)
```

The summary of our DTM states that there are five documents and 2566 unique words. Hence, let's give a matrix with the dimensions of 5x2566, in which 73 per cent of the rows have value 0.

To get the total frequency of words in the whole corpus, we can sum the values in a row, as follows:

```
freq <- colSums(as.matrix(dtm))
```

For that, we have to first transform the DTM into a matrix, and then sum up the rows to give a single value for each column.

Next, we sort this in descending order to get to know the terms with the highest frequency, as follows:

```
ord <- order(freq,decreasing=TRUE)
freqn[head(ord)]
```

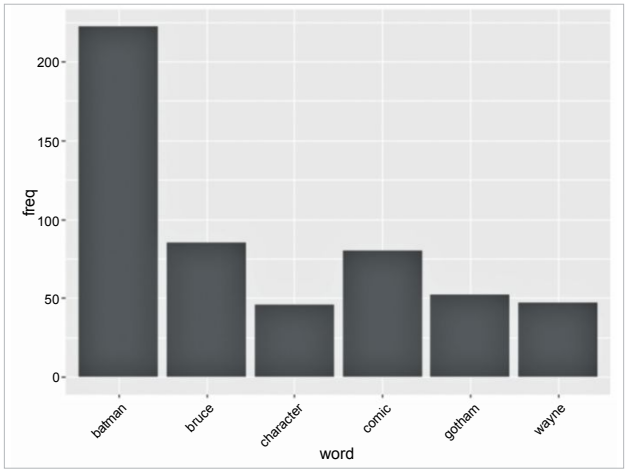


Figure 2: Depicts the plot of words with a frequency greater than 30

```
   batman    bruce     comic    gotham     wayne character
      222       85        80        52        47        46
```

Finding associations

After the previous step, we get a rough idea that the terms 'batman', 'bruce', 'comic', 'wayne' and 'character' are central to our corpus. Thus, if we want to find terms that highly correlate with them we can use the *findAssocs()* function, which takes as its parameter the DTM, the word and the correlation limit (which ranges from 0 to 1). A correlation of 1 means 'always together', while a correlation of 0.5 means 'together for 50 per cent of the time'.

```
findAssocs(dtm, "bruce", corlimit=0.90)
$bruce
   able      pain     times     order      will      time
   0.99      0.98      0.98      0.97      0.96      0.95
   access    day       defeat    earth     eyes      idea
   0.94      0.94      0.94      0.94      0.94      0.94
 individual  minds  nightwing  sometimes  unlike  character
  0.94       0.94      0.94   0.94     0.94      0.93
 associated      life      part personality    alfred      man
 0.92       0.92      0.92      0.92      0.91      0.91
 parents
    0.91
```

Plotting the term frequencies

To graphically represent our term frequencies as a plot, the ggplot2 library is used as follows:

```
library(ggplot2)
wf <- data.frame(word=names(freq), freq=freq)
p <- ggplot(subset(wf, freq>30), aes(word, freq))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
p
```

Figure 3: Word Cloud of words with a frequency greater than 30

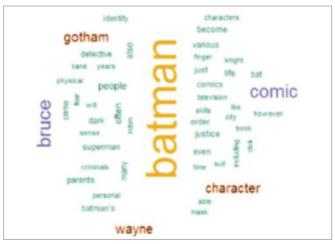


Figure 4: Word Cloud of the 50 words that occur most often in the document

Word Cloud

Word Cloud is another way of representing the frequency of terms in a document. Here, the size of a word indicates its frequency in the document corpus.

Load the *wordcloud* package, as follows:

```
library(wordcloud)
```

For Word Cloud comprising terms with a frequency greater than 30, use the following command:

```
wordcloud(names(freq), freq, min.freq=30,colors=brewer.
pal(3,"Dark2"))
```

For a Word Cloud for the 50 words that occur most often, use the command given below:

```
wordcloud(names(freq), freq, max.words=50,colors=brewer.
pal(6,"Dark2"))
```

Text processing is about extracting useful information from text, which includes basic steps of pre-processing data, stemming the data, representing the corpus using the document term matrix and obtaining the associations between terms. R provides several libraries and functions to efficiently carry out these tasks. END

References

[1] *https://cran.r-project.org/web/packages/tm/tm.pdf*
[2] *https://cran.r-project.org/web/packages/wordcloud/wordcloud.pdf*
[3] *https://cran.r-project.org/web/packages/SnowballC/SnowballC.pdf*

By: Barkha Patel and Sapan H. Mankad

Barkha Patel is a Text Mining enthusiast. Her agenda is to solve real life problems through technology products that are user friendly and user-centric. She can be contacted at *barkha95@yahoo.com*.

Sapan H. Mankad is assistant professor, department of IT, Institute of Technology, Nirma University. He can be contacted at *sapan.mankad@gmail.com* and he blogs at *htttp://www.shmlive.wordpress.com*. Web page: *http://www.nirmauni.ac.in/ITNU/Faculty/Prof-Sapan-H-Mankad*.

How to check the type of file system on an unmounted device

Here are a few simple steps for checking the file system type of an unmounted device.

Run the following command in the terminal:

```
# fdisk /dev/sdX -l
```

This will display details that give you a basic idea of the file system's structure. The output is something like what's shown below:

```
Device Boot Start End Blocks  Id  System
/dev/sdb1  1     986 7912515+ b W95FAT32
```

Next, you can try the following command:

```
# blkid
```

The output is something like what follows:

```
/dev/sdb1: LABEL="HP V220W" UUID="74C0-177C" TYPE="vfat"
```

You can also use *parted* as shown below:

```
# parted /dev/sdX -l
```

This is similar to *fdisk* but the output is more formatted as shown below:

```
Model: hp v220w (scsi)
Disk /dev.sdb :8103MB

Number Start End Size Type FileSystem
1    979kB 8103MB 8102MB primary fat32
```

—Sharmin Shaikh, imsharmeen@gmail.com

Find out your hard disk's serial number without physical access to it

Here is a trick to find out your hard disk's serial number in Linux when physical access to the disk is not possible. You need to run the following command in the terminal to get these details:

```
#hdparm -I /dev/sda
```

To get only the serial number, you can run the following command:

```
#hdparm -I /dev/sda | grep Serial
```

—Suresh Jagtap, smjagtap@gmail.com

Access control lists to secure the file/ directory

The access control list is used to define the permissions for the file and directory on an individual level. It allows you to give permissions for any user or group to any disk resource.

Thus, it provides more security. Rules must be specified in the following formats.

1 . To add the permissions to the user, use the following code:

```
setfacl -m "u:username:permissions"
  or
setfacl -m "u:username:permissions"
```

2. To add permissions to the group, use:

```
setfacl -m "g:groupname:permissions"
  or
setfacl -m "g:gid:permissions"
```

3. To remove all permissions, use:

```
setfacl -b
```

4. To remove each entry, use:

```
setfacl -x "entry"
```

5. To display permissions, use:

```
getfacl filename
```

—Sandhya Bankar, bankarsandhya512@gmail.com

Get your IP address in the command line

If you ever need the public IP address of your computer, you can get it from the command line by adding one simple alias statement to the *.bashrc* file located in the home directory.

Add the alias as follows:

```
myip='echo $(curl --silent ipecho.net/plain)'
```

… at the end of *.bashrc* and reopen the terminal; or you can run sources, as follows:

```
~/.bashrc
```

The next time you need a public IP address, there's no need to visit the browser—just type the following command:

```
#myip
```

Your computer's public IP address will be displayed. An example is:

```
$user@linux]$ myip
```

The output will be your public IP, i.e.,

```
1xx.2xx.0.58
```

—Ankith Herle, ankitherle@gmail.com

Start/stop services automatically on booting up

To start/stop a service when booting up is easy using the startup manager tool, but this doesn't show the status of all the services on bootup—whether they start automatically or not. To change this or view the status of all services on booting up, you can use the *systemctl* utility on Linux systems that have *systemd* as the *init* manager. Here are a few generic commands and what they do.

To see all services available in your Linux system, use the following command:

```
$systemctl  | grep service
```

To see all active services, use:

```
$systemctl list-units
```

To start or stop a service on booting up, use:

```
$sudo systemctl enable application.service
$sudo systemctl disable application.service
```

To see the status of the service, use:

```
$systemctl status application.service
```

For this, you can also try the following command:

```
$systemctl is-enabled application.service
```

—Shubham Dubey, sdubey504@gmail.com

Base conversion using shell

We often need to convert a number from one base to another—for instance, convert from decimal to hexadecimal or from octal to hexadecimal, and so on. There are many GUI applications available for this; however, we can do this very quickly and efficiently from the CLI as well. It simply uses shell's built-in 'printf' command. Let us understand it with simple examples.

The prototype of the 'printf' function is:

```
printf FORMAT [ARGUMENT] ...
```

It uses the '%o' format specifier for octal (base 8) numbers, '%f' for hexadecimal (base 16) numbers and '%d' or '%i' for decimal (base 10) numbers. Shown below are simple examples that illustrate conversions:

```
[bash]$ printf "DEC(%i) = HEX(%X)\n" 255 255   # decimal to
hexadecimal conversion
DEC(255) = HEX(FF)

[bash]$ printf "HEX(%X) = OCT(%o)\n" 0xFF 0xFF # hexadecimal
to octal conversion
HEX(FF) = OCT(377)

[bash]$ printf "OCT(%o) = DEC(%i)\n" 255 255   # octal to
decimal conversion
OCT(377) = DEC(255)
```

—Narendra Kangralkar, narendrakangralkar@gmail.com

END

Share Your Linux Recipes!

The joy of using Linux is in finding ways to get around problems—take them head on, defeat them! We invite you to share your tips and tricks with us for publication in *OSFY* so that they can reach a wider audience. Your tips could be related to administration, programming, troubleshooting or general tweaking. Submit them at *www.opensourceforu.com*. The sender of each published tip will get a T-shirt.

DVD OF THE MONTH

Try out this distro for its stability.

openSUSE Leap 42.2

This is a community distribution that shares code and infrastructure with SUSE Linux Enterprise. This latest release includes a long-term support kernel (Linux 4.4) and KDE's Plasma 5.8 desktop environment. openSUSE is the makers' choice for sysadmins, developers and desktop users. With openSUSE, you can discover the best open source tools developed by the community. The complete documentation on how to install and use openSUSE 42.2 can be found at *https://doc.opensuse.org/*.

This distro provides new as well as experienced Linux users with the most usable Linux distribution and stable operating system.

The bundled DVD is a bootable install disk. Before installing the OS on your system, you are advised to do a media check, as shown in the screenshots below.

R N I No. DELENG/2012/49440, Mailed on 27/28th of Advance month Delhi Postal Regd. No. DL(S)-01/3443/2016-18
Published on 27th of Advance month Licenced to Post without Pre-Payment Licence No. U(S)-56/2016-17