

FREE
DVD

AD Protected Users

Web Servers

fedora 33
Server 64-bit

ISSUE 62/2021
ADMIN
Network & Security

ADMIN

Network & Security

ISSUE 62

LEAN WEB Servers

Go lighter and get faster

Hiawatha - A light-footed web server

Lighttpd - Lean, fast, and simple to configure

Kea - A modern DHCP server

4 Software-based load balancers

And more!

**Apache security, HTTP/2 and Nginx,
Apache and Nginx subdomains,
Apache Kafka**

AD Protected Users

Granular protection for highly privileged accounts

Microsoft 365 and Teams

Communication settings and security

Podman and systemd

Containers under systemd control

Grafana Dashboards

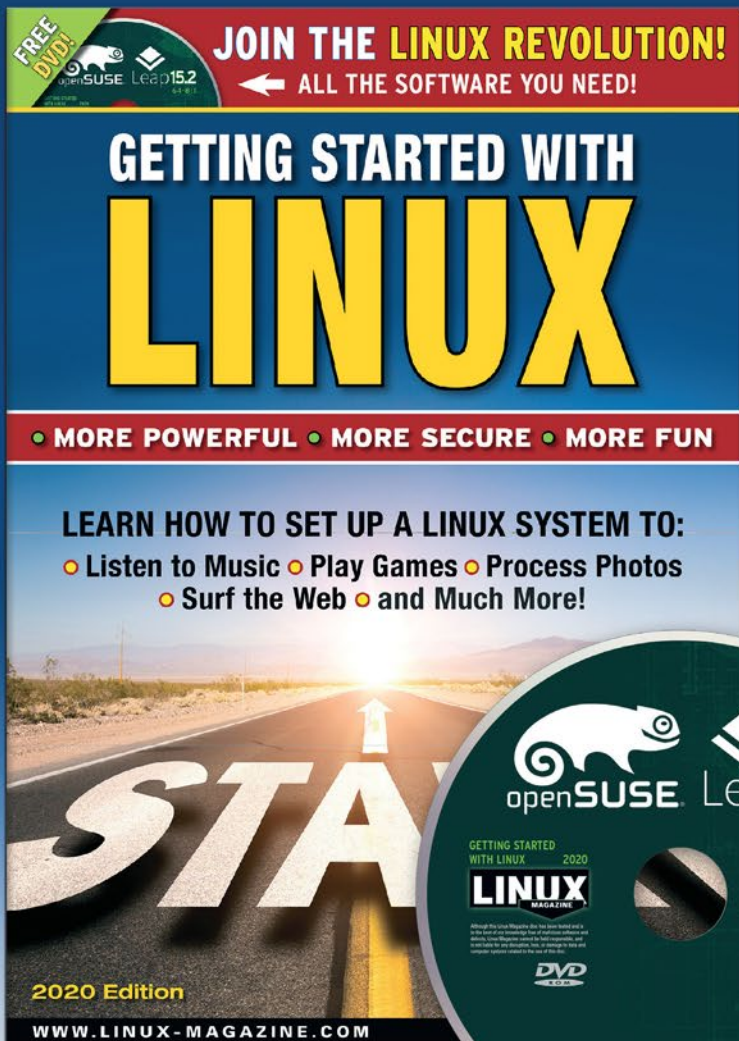
Grafana + Prometheus = custom visualization toolset

OpenEBS
Cloud-native storage

LINUX NEW MEDIA
The Pulse of Open Source

WWW.ADMIN-MAGAZINE.COM

Hit the ground running with Linux



Want your friends and colleagues to make the switch to Linux?

This single issue shows beginners how to:

- install Linux
- download and install free software for your Linux system
- play games
- create documents and spreadsheets
- process photos
- play music and videos
- and much more!

ORDER ONLINE: shop.linuxnewmedia.com/specials

Keep Positive

The road to success is paved with negative commentary.

I still remember telling one of my computer consulting clients that I was quitting my business and going back to a regular job, while also pursuing my dream of being a technology writer. He responded, "Oh, that's very competitive, are you sure you're doing the right thing?" Fast forward to now, almost 20 years later, and my answer is still, "Yes. Yes, I am." I heard pretty much the same thing from everyone I told. Everyone told me that writing, especially writing for a living, was too competitive and unrealistic. I'm glad I didn't listen to the naysayers.

You see, people will take every opportunity to discourage you from exploring your dreams. Sure, there's some rejection associated with writing, acting, filmmaking, or really doing anything you want to do, but it doesn't stop there. If you proclaim your innermost desires to do anything that's difficult, risky, or time-consuming, you don't have to look far to find someone who'll attempt to dissuade you from going further. My best advice is – spoiler alert – don't listen to them.

My son is a good example of someone who listened to the naysayers. He worked in a retail environment and wanted to become an Emergency Medical Technician (EMT), and he told a few people that he had enrolled in the EMT class. His manager discouraged him and disparaged the field as competitive, low paid, and not worth his time. His manager had attempted the class and exam a few years earlier himself and failed. My son quit the class a few days later. My response to him was, "Don't listen to someone who's unsuccessful at something. Of course, they're going to discourage you. They don't want you to be successful at something that they failed at." By then, of course, it was too late.

If you want to know something about a job or creative pursuit, don't listen to naysayers. Chances are very good that they aren't successful at anything, much less whatever it is that you want to do. The correct thing to do is ask someone who *is* successful in that role. If you want to be a system administrator, ask an experienced system administrator about the job and what it takes to get there. If you want to be a screenwriter, ask a successful screenwriter how you should proceed. Asking people who have failed at something will usually net you a disappointing response, whereas asking those who are successful will encourage you to go further.

Negative commentary is all too common, and it's very powerful. Regardless of how many times I told my son the opposite of what his former manager told him, it didn't help. We even had him talk to a local EMT, who gave him all kinds of encouragement, and that didn't help either. It's a psychological fact that it takes something on the order of 17 positives to offset one negative. So, in the minutes it took to destroy my son's aspirations of becoming an EMT, it would take hours of positive discussions to get him back to a psychological break-even point. He has since changed his goal to becoming a cybersecurity professional. I've preloaded his positive psychological scoreboard for that one. So far, there have been no failed security experts to kill his buzz.

All this boils down to some advice for you, regardless of your aspirations. First, listen only to those who are successful, as they will encourage you to stay the course. Second, keep your eye on the goal and don't stray from it. Rejection and disappointment are often indicators that you're doing something right. I know that sounds crazy, but people who do nothing never experience rejection. *Those who do nothing never experience failure.* Third, learn from your rejections and failures. Turn them into successes by asking for feedback and looking at the work of those who are successful. Finally, anything worth doing or having is worth a little time and a little pain to achieve it. Remember that, if it were easy, there would be no naysayers ready to discourage you. Success is often measured in aspiration, inspiration, and perspiration.

Ken Hess • ADMIN Senior Editor

ADMIN

Network & Security



Features

In this issue, we present a variety of solutions that resolve common web server needs.

10 Hiawatha Web Server

A lightweight web server with features that distinguish it from heavyweights such as Apache.

16 HTTP/2 for Nginx

The current representative of the HTTP family offers several advantages for website operators and their users if the protocol is correctly adapted to individual scenarios.

22 Lighttpd Setup

This long-established web server is lean and fast and can be set up quickly thanks to its simple configuration.

```
tim@ubuntu:~$ lighttpd -t -f /my/lighttpd.conf
2020-11-20 15:41:16: (conf:file.c:1316) source: ./my-lighttpd.conf
Time: 68 pos: 1 parser failed somehow near: http: ~
tim@ubuntu:~$
```

26 Subdomains

Set up virtual hosts on modern web servers for Apache and Nginx.

News

Find out about the latest plays and toys in the world of information technology.

8 News

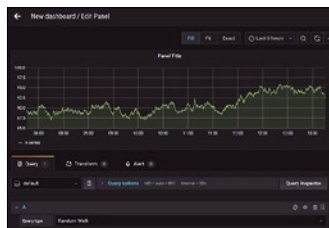
- Yet another botnet targeting Linux
- Linux kernel 5.11 released
- Ubuntu Core 20 officially released
- CloudLinux offers lifecycle support service for expired Linux distributions

Tools

Save time and simplify your workday with these useful tools for real-world systems administration.

30 Grafana Dashboards

Analytics and visualization dashboards coupled with Prometheus monitoring and alerting tools deliver custom reporting and alerting systems.

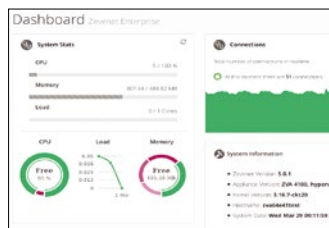


38 Kea DHCP Server

Modern underpinnings for dynamic IP address assignment by DHCP.

44 SD Load Balancing

We introduce the most important software load balancers, look at their strengths and weaknesses, and provide recommendations for use scenarios



50 Microsoft 365 and Teams Tips

Office 365 and Microsoft Teams come with useful settings for setting up communication channels and securing environments.

Containers and Virtualization

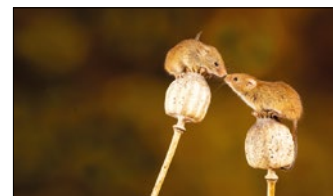
Virtual environments are becoming faster, more secure, and easier to set up and use. Check out these tools.

54 OpenEBS for Kubernetes

A cloud-native storage environment that makes block devices available to individual nodes in the Kubernetes cluster.

58 Podman and systemd

Put any software inside a container under the control of systemd.



Security

Use these powerful security tools to protect your network and keep intruders in the cold.

62 Hardening Apache

A smart configuration, timely updates, and careful security strategies can protect servers from attacks.



68 Attack Surface Reduction

Windows attack surface reduction policies help protect your entire IT infrastructure.



10 Hiawatha Web Server

A highly interesting project for applications that need a simple, well-functioning web server with basic security features.



38 Kea DHCP Server

This modular, highly available, and expandable modern DHCP server can connect to your management systems and does not require a service restart with configuration changes.



Management

Use these practical apps to extend, simplify, and automate routine admin tasks.

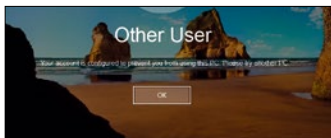
72 Store FIDO2 Info in LDAP

Secure passwordless login with LDAP and a schema to establish objects and attributes for FIDO2 authentication.



74 AD Privileged Accounts

Granular protection granted by the Protected Users group in Active Directory and Kerberos authentication policies.



80 Remora

Per-node and per-job resource utilization helps you understand how an application performs on the system through profiling and system monitoring.

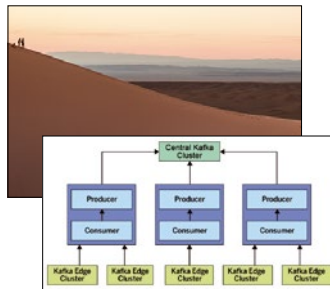


Nuts and Bolts

Timely tutorials on fundamental techniques for systems administrators.

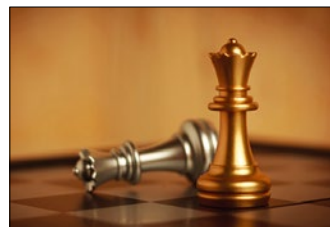
86 Apache Kafka Tuning

A guide to 10x scaling in Kafka with real-world metrics for high throughput, low latency, and cross-geographic data movement.



90 Rethinking RAID (on Linux)

Configure redundant storage arrays to boost overall data access throughput while maintaining fault tolerance.

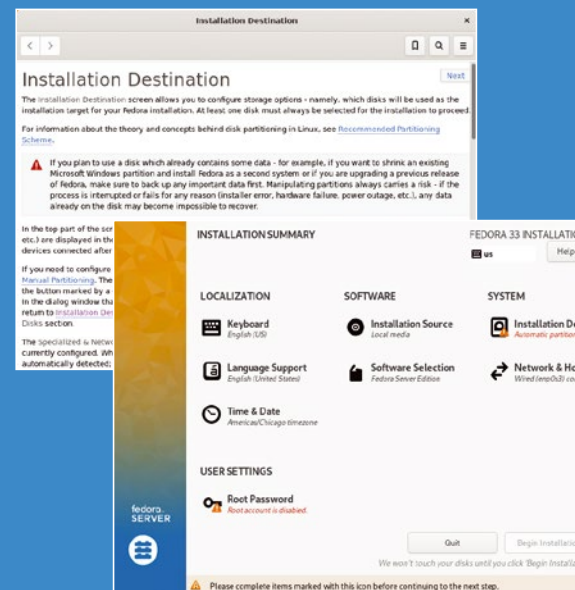


Service

- 3 Welcome
- 4 Table of Contents
- 6 On the DVD
- 97 Back Issues
- 98 Call for Papers

fedora 33 f
Server 64-bit

- Linux Kernel 5.8
- Btrfs default filesystem
- FreeIPA for advanced identity management
- Cloud versions



See p 6 for details



Fedora 33 Server (full install)

On the DVD

Fedora Server is an open source short-lifecycle server operating system sponsored by Red Hat and supported by a robust community. Its modular design [1] allows operating system upgrades that will not upset your established application or language stack, and Cockpit [2] presents system tasks in a web browser, allowing you to start and stop containers, administer storage, configure networks, and inspect logs.

Other features include:

- Linux kernel 5.8
- Btrfs default filesystem
- FreeIPA [3] for advanced identity management
- Cloud versions [4]

```

[ OK ] Listening on udev Kernel Socket.
[ OK ] Listening on User Database Manager Socket.
[ OK ] Mounting Huge Pages File System.
[ OK ] Mounting POSIX Message Queue File System.
[ OK ] Mounting Kernel Debug File System.
[ OK ] Mounting Kernel Trace File System.
[ OK ] Starting Create list of static device nodes for the current kernel...
[ OK ] Stopped Plymouth switch root service.
[ OK ] Stopped Journal Service.
[ OK ] Starting Journal Service.
[ OK ] Starting Remount Root and Kernel File Systems.
[ OK ] Starting Apply Kernel Variables.
[ OK ] Starting Coldplug All udev devices.
[ OK ] Mounted Huge Pages File System.
[ OK ] Mounted POSIX Message Queue File System.
[ OK ] Mounted Kernel Debug File System.
[ OK ] Mounted Kernel Trace File System.
[ OK ] Finished Create list of static device nodes for the current kernel.
[ OK ] Finished Remount Root and Kernel File Systems.
[ OK ] Finished Apply Kernel Variables.
[ OK ] Starting Reload Hardware Database.
[ OK ] Starting Load/Save Random Seed.
[ OK ] Starting Create System Users.
[ OK ] Finished Load/Save Random Seed.
[ OK ] Finished Create System Users.
[ OK ] Starting Create Static Device Nodes.
[ OK ] Started Journal Service.
  
```

INSTALLATION SUMMARY

LOCALIZATION

Keyboard
English (US)

Language Support
English (United States)

Time & Date
Americas/Chicago timezone

USER SETTINGS

Root Password
Root account is disabled

SOFTWARE

Installation Source
Local media

Software Selection
Fedora Server Edition

Installation Destination

The Installation Destination screen allows you to configure storage options - namely, which disks will be used as the installation target for your Fedora installation. At least one disk must always be selected for the installation to proceed.

For information about the theory and concepts behind disk partitioning in Linux, see [Recommended Partitioning Scheme](#).

If you plan to use a disk which already contains some data - for example, if you want to shrink an existing Microsoft Windows partition and install Fedora as a second system or if you are upgrading a previous release of Fedora, make sure to back up any important data first. Manipulating partitions always carries a risk - if the process is interrupted or fails for any reason (installer error, hardware failure, power outage, etc.), any data already on the disk may become impossible to recover.

In the top part of the screen, all locally available storage devices (SATA, IDE and SCSI hard drives, USB flash drives, etc.) are displayed in the Local Standard Disks section. Local disks are detected when the installer starts - any storage devices connected after the installation has started will not be shown.

If you need to configure additional local storage devices, select I will configure partitioning and press Done to move to Manual Partitioning. Then, connect any new hard drives you want to make available during the installation, and press the button marked by a circular arrow in the set of controls below the list of mount points on the left side of the screen. In the dialog window that opens, press Rescan Disks and wait until the scanning process completes. Then, press OK to return to Installation Destination; all detected disks including any new ones will be displayed in the Local Standard Disks section.

The Specialized & Network Disks section below shows advanced network storage (such as iSCSI and FCoE disks) currently configured. When you first open this screen, no such devices will be displayed because they can not be automatically detected; to search for network storage devices Add a disk button and proceed with Installation.

Next

DEFECTIVE DVD?

Defective discs will be replaced, email: cs@admin-magazine.com

While this *ADMIN* magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, *ADMIN* magazine cannot be held responsible and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.

Resources

- [1] Fedora Server modularity: [\[https://docs.fedoraproject.org/en-US/modularity/\]](https://docs.fedoraproject.org/en-US/modularity/)
- [2] Cockpit: [\[https://cockpit-project.org\]](https://cockpit-project.org)
- [3] FreeIPA: [\[https://www.freeipa.org/page/Main_Page\]](https://www.freeipa.org/page/Main_Page)
- [4] Fedora Cloud: [\[https://alt.fedoraproject.org/cloud/\]](https://alt.fedoraproject.org/cloud/)

CLEAR OFF YOUR BOOKSHELF WITH DIGITAL ARCHIVES

Complete your collection of *Linux Magazine* and *ADMIN Network & Security* with our Digital Archive Bundles.

You get a full year of issues in PDF format to access at any time from any device.



Lead Image © enki, 123RF.com

<https://bit.ly/archive-bundle>

2020
Archives
Available
Now!

News for Admins

Tech News

Yet Another Botnet Targeting Linux

Recently the drastic rise in cryptocurrency trading prices has led to numerous online systems falling prey to botnets, seeking to mine for profit. This botnet, dubbed WatchDog, was discovered by Unit 42 (<https://unit42.paloaltonetworks.com/>), who realized this particular threat has been active since January 2019.

WatchDog was written in Go and uses outdated enterprise applications as a point of entry.



© svedoliver, 123RF.com

So far, Unit 42 has found 33 exploits, targeting 32 vulnerabilities in open source software, such as Drupal, Elasticsearch, Apache Hadoop, Redis, and the ThinkPHP framework.

Unit 42 estimates around 500 to 1,000 infected systems are currently being used by WatchDog to mine for cryptocurrency and the total profit was estimated at 209 Monero coins (worth roughly \$32,000). However, the researchers have only been able to analyze a few binaries, so the figure will most likely be considerably higher.

The one silver lining is that Unit 42 has yet to discover that any credentials have been stolen. That, of course, could change at any moment. To that end, all admins are encouraged to keep all Linux systems (and the apps that run on said systems) updated.

For more information on WatchDog, read the full Unit 42 report (<https://unit42.paloaltonetworks.com/watchdog-cryptojacking/>).

Linux Kernel 5.11 Released

Linus Torvalds (the creator of Linux) has officially released the latest kernel for the open source operating system. Kernel 5.11 includes the usual dose of expanded hardware support, as well as a feature that should excite gamers, and an improvement for Wi-Fi.

As far as expanded hardware support, kernel 5.11 has brought improvements to RISC-V, as well as support for RISC-V CPU architecture such as OpenRISC support for the LiteX SoC controller driver.

On the Intel side of things, kernel 5.11 adds support for: Iris Xe GPU, Software Guard Extensions (SGX), and Intel Platform Monitoring Technology (PMT). Conversely, Intel Itanium support has finally been dropped.

With AMD, 5.11 enhances performance for Zen 2/Zen 3 CPUs and adds support for Van Gogh.

Finally, Nvidia RTX 30 GPU support has been added.

One of the more exciting additions to the kernel is the inclusion of the Syscall User Dispatch (SUD), which provides functionality for compatibility layers for quick capture of system calls issued by a non-native part of an application. This addition will greatly improve Windows games running via Proton or Wine.

Finally, the Linux kernel is adding support for the 6GHz band (Ultra High Band or UHB) support for WiFi 6E in the Intel WiFi "IWLWIFI" driver. This will make it possible to connect to wireless channels that use Ultra High Band.

Read more about kernel 5.11 in Linus' official release announcement (<http://lkml.iu.edu/hypermail/linux/kernel/2102.1/08310.html>).



**Get the latest
IT and HPC news
in your inbox**

**Subscribe free to
ADMIN Update
and HPC Update
bit.ly/HPC-ADMIN-Update**

Ubuntu Core 20 Officially Released

If you're an IoT or embedded device developer, the release of Ubuntu's latest "core" edition should have you excited. Why? Because this latest iteration of Ubuntu Core includes a number of features focused on security. In fact, as of this release, Mark Shuttleworth, founder and CEO of Canonical (the company behind Ubuntu), says, *"Every connected device needs guaranteed platform security and an app store."* He continues, *"Ubuntu Core 20 enables innovators to create highly secure things and focus entirely on their own unique features and apps, with confinement and security updates built into the operating system."*

Three specific security-minded features are:

- Cryptographically-authenticated boot.
- Full disk encryption.
- Manual and remote recovery modes.

Ubuntu Core 20 also addresses the cost of design by working with silicon providers and ODMs to streamline the process of bringing a new device to market. This is accomplished with a new service, dubbed "SMART START" (<https://ubuntu.com/smartstart>), which they call "smart things as a service." This new feature is targeted toward enterprise-class businesses seeking to become connected product manufacturers. SMART START combines hardware certification, software, and services to help accelerate the development process. SMART START offers a fixed-priced engagement for the launch of a device, which covers consulting, engineering, and updates for the first 1000 devices on certified hardware.

Read the full Ubuntu Core 20 release here: <https://ubuntu.com/blog/ubuntu-core-20-secures-linux-for-iot>.

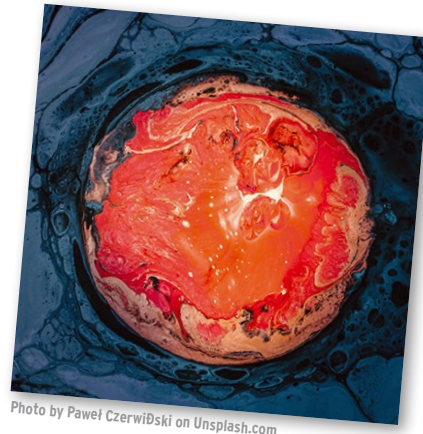


Photo by Paweł Czerwiński on Unsplash.com

CloudLinux Offering Lifecycle Support Service for Expired Linux Distributions

What happens when you have a server running a Linux distribution that's reached its end of life? Typically you migrate that server over to a newer release, and hope everything goes well. But what about those instances when you either don't have the time to make the switch or you know the software on the current platform isn't supported on the newer release? You desperately hold on to that deployment, trying to figure out a way to get updates, even with the EOL status.

That's where CloudLinux comes into play. The company responsible for AlmaLinux (a 1:1 binary compatible distribution with RHEL) has started offering Extended Lifecycle Support (ELS) support beyond EOL for some distributions. Here's what they currently offer:

- ELS for CentOS 6 to June 2024.
- ELS for Oracle Linux 6 starting February 2021 to February 2025

CloudLinux is also working on ELS support for Ubuntu 16 (set to expire April 2021) and Debian 9 (set to expire June 2022).

Igor Seletskiy, CEO and founder of CloudLinux, had this to say about why they're offering ELS support, "At CloudLinux we realized that a large volume of Linux users are at risk...relying on unsupported operating systems (OS). That is, unless these users rapidly move to an alternative OS which is not necessarily practical."

The ELS updates will include the kernel and other necessary packages like Apache, PHP, MySQL, GLibc, OpenSSL, and OpenSSH.

To find out more about CloudLinux ELS support (such as cost), check out their official Extended Lifecycle Support page: <https://www.cloudlinux.com/extended-lifecycle-support>.



© nyul, 123RF.com

The light-footed Hiawatha web server

Frugal Delivery

Hiawatha is a lightweight web server with a variety of features that distinguish it from heavyweights such as Apache.

By Martin Loschwitz

Most admins intuitively think of Apache first when they hear “web server.” The program is not only one of the longest serving of its kind, but it has built a reputation over the years of being a smart and stable product. Admittedly, there are other options: in the eyes of many admins, Nginx has long since become the better option. If you can’t handle the massive feature set of Nginx or Apache, you’ll regularly end up with Lighttpd, which is limited to the bare essentials in many respects. Only rarely do admins settle on Hiawatha (Figure 1).

The bottom line is that Hiawatha [1] very much offers a valid alternative to Apache, Nginx, Lighttpd, and other web servers if you don’t need exotic features. In this article, I introduce Hiawatha, detail its genesis, and describe how admins can get up and running quickly with the solution.

What Hiawatha Can Do

Anyone researching *Hiawatha* on the web will be confronted with a variety of possible explanations for the origin of the word. It is a social dance dating back to the 1920s, the mythical leader of the Iroquois Five Nations sometime between the 15th and 18th centuries, or the Iroquois euphemism for “he who rises early.” Although it seems likely

that the Indian chief was the inspiration, it remains unclear why the author of the software chose this name.

At least we know who programmed Hiawatha: Hugo Leisink was a computer science student from the Netherlands when he started working on the program in 2002. Unlike Linus Torvalds a good decade earlier, he did not do this on a mere whim. Instead, the web servers available at the time, especially Apache, got on his nerves for several reasons.

If you think back to that time, you will recall an era when powerful scripting languages like PHP existed,

but they didn’t have the quality or prevalence they have today. If you wanted to execute code in the context of a web server, you would tend to use Common Gateway Interface (CGI) scripts; however, web servers gave them practically free rein. Apache, for example, did not define a maximum length of time a CGI script could take to return either a positive or negative value. If you attacked CGI scripts with a certain amount of skill, you could take down an entire web server with a single script.

This made Leisink so mad that he devised a new web server. The idea



Figure 1: The W3Techs survey lists very few installations of Hiawatha, but that doesn’t begin to do justice to the server’s capabilities. © W3Techs



behind Hiawatha was born, and the program was designed to deliver the security features that Leisink very much missed in the competitors. Although the tool still claims to be a very lean, nimble server for HTTP(S), some features have been added over the years.

CGI and FastCGI

To date, Hiawatha provides comprehensive support for CGI and FastCGI. The constraints that Leisink created and built into the software back in 2002 are still available. If you as the admin want to restrict CGI scripts to a certain runtime, you can so configure it in Hiawatha. If a script goes over the top, Hiawatha takes care of it by firing off a hard-hitting `kill -9`, without any administrator intervention. However, this is not the only useful feature the solution offers. Hiawatha was also one of the first web servers to include large file support in its specifications. Admittedly, this doesn't sound particularly spectacular today, but there are still web servers from the small solutions segment that trip up when it comes to larger files.

Reverse Proxying Is Easy

An increasingly common use case for web servers, because of security concerns, is as a reverse proxy. A proxy server serves a client on a private network that uses it to connect to the outside world. Many corporate networks today are so strongly protected by firewalls that it is virtually impossible to access resources behind the various firewalls from the outside. This is where reverse proxies come in: They accept an incoming connection from the Internet on one side and have a route to the network where the destination service is located. The reverse proxy then uses this route as a transparent intermediary to handle the data traffic between the client and the service. For enterprises, this offers an attractive alternative to turning the firewall into Swiss cheese because it requires less

overhead and makes some use cases possible that could not be implemented otherwise because of excessively strict security measures. Apache and Nginx easily support use as a reverse proxy, but it seems slightly over the top to install the huge feature sets of the two servers to handle what is a relatively easy task. Leisink thought so, too, and implemented support as a reverse proxy in Hiawatha at an early stage. In a few thousand lines of code, you can implement what would otherwise mean considerable bloat in the system if you detoured via Apache or Nginx.

No Features? Think Again

If you hear that Hiawatha is a light-footed web server and immediately assume massively reduced functionality, you are thoroughly mistaken. The complicated rewriting of URLs that solutions like WordPress or Joomla often require is also supported by Hiawatha. The server uses the URL Toolkit, a custom Hiawatha component that bundles URL rewriting logic, for this purpose. Therefore, Hiawatha is perfectly suited to running Web 2.0 applications like WordPress, and because the web server has a complete transport layer security (TLS) implementation, it can be done on an encrypted basis. Hiawatha also supports basic HTTP login with digest access or basic authentication, which is essential, for example, for accessing the admin interfaces of many web projects.

No Data Flooding

Other features from the security corner include the ability to set up connection throttling at the web server level, which involves specifying the permitted upload speed for individual clients. This feature prevents users from wrecking individual web server instances by saturating their lines with targeted, parallel, multiple uploads. Of course, protection at the software level is not enough to protect servers against large-scale distributed denial of service (DDoS) attacks; instead,

you have to start on the network level, but at least small attacks are reliably stopped by Hiawatha in this way. Built-in caching also enables the server to process incoming and outgoing requests faster and with lower resource consumption. Support for HTTP compression by Gzip, which Hiawatha has also been offering for a few years, is in the same vein.

Preventing XSS and CSRF Attacks

Cross-site scripting (XSS) is a classic attack vector for targeting web servers. The danger potentially exists wherever web applications handle user input. Attacks are usually launched by bots running automatically.

In XSS attacks, attackers, in simple terms, access the infrastructure behind the web server. To do so, they exploit vulnerabilities in the respective web applications that do not consistently prevent unauthorized access. For example, if a web solution has a search form for usernames that triggers a MySQL call in the background, a typical XSS attack would be to enter a MySQL command in the search form. This particular form of XSS attack is quite common, and even has its own name: SQL injection. If the software is not hardened against XSS attacks, it forwards the illegal command directly to the database, where it ends up being executed.

Cross-site request forgery (CSRF) works the other way around, wherein attackers exploit the rights of legitimately logged-in users or their browser sessions to send commands to web servers that the legitimate users do not want to send.

The problem with these and other similar types of attacks is that, for a long time, web servers have almost completely ignored this attack vector and declared themselves not responsible. Hiawatha has been a clear exception to this rule for a long time: The server has code that tries to detect XSS and CSRF attacks on the basis of various parameters and prevents them if ongoing HTTP(S) sessions meet the attack criteria.

The framework that tries to detect attacks does even more in Hiawatha: It not only watches out for the signs of XSS and CSRF, it also detects clients that produce high volumes of traffic in a short time or send unusual HTTP requests. The admin defines the individual parameters for the engine that detects attacks in the server's configuration file.

Monitoring Included

Although virtually all modern software programs collect metrics data on their own usage and make the data available through a standardized interface, web servers have long held back on the subject of trending. Hiawatha also adopts a special approach: The tool writes copious amounts of metrics data as defined by its author. The web server includes a PHP-based mini-application named Hiawatha Monitor that retrieves the collected metrics data from the web server and outputs the data to the admin.

Because the Hiawatha Monitor is based on Banshee [2], the tool also visualizes received usage data. Although it cannot be compared in form and scope with the kind of data collection that a combination of Prometheus and Grafana archives, it is definitely better than nothing. Hiawatha cannot be connected to Prometheus in a meaningful way, because no suitable exporter can output the metrics data available in Hiawatha to Prometheus.

Tomahawk Command Shell

All relevant Hiawatha parameters can be set in a configuration file, but changes require restarting the tool. Depending on the situation, a restart is not always possible, which is why Hiawatha also comes with its own command line. The metrics data currently available to Hiawatha can be read out in the shell. More importantly, however, the command shell, dubbed Tomahawk, supports reconfiguration of the web server on the fly. IPs that have ended

up on Hiawatha's internal blacklist can be permanently removed here; otherwise, Hiawatha would simply no longer serve these clients. All current connections can be terminated by the kick command.

Getting Started

All in all, Hiawatha is clearly a powerful web server that can be implemented with very little in terms of resources, and it does not need to shy away from comparisons with competitors such as Apache or Nginx. What is less fortunate, however, is what admins have to do to set up a running instance of the service. Unlike Apache and Nginx, most distributions do not include packages for Hiawatha. You can't get by with just a little tinkering. That said, the reward is a clear-cut, easy-to-edit configuration.

The first step is to install Hiawatha, which will depend on the distribution you are using. Because Hiawatha is not very widespread, you will often search in vain for pre-built packages for the major distributions. Debian packages for "Buster" are available from a separate package repository [3]. After adding it to the package sources and enabling the GPG key for the repository, the command

```
apt install hiawatha
```

retrieves the package, including all its dependencies, and drops it on your system.

Life could also be great, theoretically, in the CentOS universe. Resourceful Hiawatha fans used to maintain packages for CentOS a few years ago. They were released in the Anku repository [4] with extensions for CentOS and RHEL. However, only one package for the ancient Hiawatha 10.8.4 is currently present, and that is only for CentOS 7. Packages for the current version 10.11 for CentOS 8 are missing.

Similar problems exist on Ubuntu: The only Hiawatha PPA delivers packages in a prehistoric state for Ubuntu 18.04 only. Users of macOS or the various BSD derivatives are

somewhat better off, because Hiawatha is part of the port system and can thus be compiled automatically from the sources.

Compiling

Hiawatha uses Cmake and doesn't have many parameters for the admin to specify when building. The recommendation is to build a Hiawatha package yourself instead of compiling the software on your production systems. To do so, first download the source code from the Hiawatha website. The compilation process then consists of the commands

```
mkdir build
cd build
cmake ..
sudo make install/strip
```

A quick look at the INSTALL file in the source code reveals which parameters Cmake basically supports for Hiawatha (Figure 2). Primarily, this means the paths on the filesystem you want Hiawatha to use, but you can also specify whether Tomahawk should be built, and support for TLS should be added, along with whether Hiawatha needs the extension to operate as a reverse proxy. Most parameters default to ON, which activates the respective function. However, you are free to change the values to your own liking – a feature you don't need in the first place doesn't have to bloat the Hiawatha binary unnecessarily.

After completing the build, it's time for the configuration. The following example assumes that /etc is set as the CONFIG_DIR – that is, that Hiawatha expects its configuration file to be in /etc/hiawatha/hiawatha.conf. It also requires that /var/www/ be set as WEBROOT_DIR, where Hiawatha looks for the web content at build time (although the parameter can be changed later by configuration).

The Simplest Web Server

If you only want Hiawatha to listen on port 443 with an SSL certificate,

your configuration file will be very simple.

```
Binding {
  Port = 443
  TLScertFile = 2
  /etc/ssl/www-certificate.pem
}
```

Unlike Apache, Hiawatha expects all components belonging to the SSL certificate to be in the same file. The referenced `ssl-certificate.pem` must therefore contain both the SSL certificate itself and the SSL key belonging to it, as well as any required secure sockets layer (SSL) certificate

authority (CA) and SSL intermediate CA certificates. The order is important: First the file must contain the private key, then the certificate, and then additional CA or intermediate certificates. Once all of this is in place, the configuration is complete, and whatever is in `WEBROOT_DIR` can be retrieved by Hiawatha.

Admittedly, this basic configuration might not make most admins very happy – a few parameters are probably still needed for regular operation. For example, you will not usually want to run the web server with the root rights of the system administrator, but as a less privileged user

account that you can specify with `ServerId` in the configuration file. `SystemLogFile` and `GarbageLogFile` let you define the logfiles that Hiawatha uses.

Virtual Hosts

One of the most used Apache features might be virtual hosts. The idea behind this is to have many web addresses point to one IP address. The web server then delivers the correct website according to the URL called. Hiawatha also offers this feature, and it is comparatively easy to set up.

In addition to the `Binding` statement used earlier, the section from [Listing 1](#) is all you need to define a `VirtualHost` that can execute PHP files in Hiawatha for `www.example.net`. Most important is that the `TLScertFile` statement is shifted from the `Binding` to the `VirtualHost` statement to allow Hiawatha to use different SSL certificates for different virtual hosts.

Security Settings

So far in the examples I have not covered the security features of and fraud detection in Hiawatha, which no admin would want to be without in everyday life. The following lines

```
1 ServerId = www-data
2 ConnectionsTotal = 150
3 ConnectionsPerIP = 10
4 SystemLogFile = /var/log/hiawatha/system.log
5 GarbageLogFile = /var/log/hiawatha/garbage.log
6
7 Binding {
8   Port = 443
9   MaxRequestSize = 128
10  TimeForRequest = 3,20
11  SSLcertFile = hiawatha.pem
12 }
13
14 BanOnGarbage = 300
15 BanOnMaxPerIP = 60
16 BanOnMaxReqSize = 300
17 KickOnBan = yes
18 RebanDuringBan = yes
19
20 CGIextension = cgi
21 CGIhandler = /usr/bin/perl:pl
22 CGIhandler = /usr/bin/php-cgi:php
23 CGIhandler = /usr/bin/python:py
24 CGIhandler = /usr/bin/ruby:rb
25 CGIhandler = /usr/bin/ssi-cgi:shtml
26
27 FastCGIServer {
28   FastCGId = PHP5
29   ConnectTo = 127.0.0.1:2005
30   Extension = php
31 }
32
33
34 # URL TOOLKIT
35 # These URL toolkit rules are made for the Banshee PHP framework,
36 # which can be downloaded from: http://banshee.leisink.org/
37 #
38 UrlToolkit {
39   ToolkitID = banshee
40   RequestURI isfile Return
41   Match ^/(favicon.ico|robots.txt|sitemap.xml)$ Return
42   Match .*?(.*) Rewrite /index.php?$1
43   Match .* Rewrite /index.php
44 }
45
46 Hostname = 208.77.188.166
47 WebsiteRoot = /var/www/hiawatha
48 StartFile = index.html
49 AccessLogFile = /var/log/hiawatha/access.log
50 ErrorLogFile = /var/log/hiawatha/error.log
```

Figure 2: Whereas other HTTP servers divide their configuration files into many individual parts, Hiawatha is content with a fairly small number of essential lines.

Listing 1: Virtual Hosts

```
CGIhandler = /usr/bin/perl:pl
CGIhandler = /usr/bin/php-cgi:php
CGIhandler = /usr/bin/python:py
CGIhandler = /usr/bin/ruby:rb
CGIhandler = /usr/bin/ssi-cgi:shtml
CGIextension = cgi

FastCGIServer {
  FastCGId = PHP7
  ConnectTo = /run/php/php7.0-fpm.sock
  Extension = php
}

VirtualHost {
  Hostname = www.example.net
  WebsiteRoot = /var/www/example.net/public
  StartFile = index.php
  AccessLogFile = /var/www/example.net/log/access.log
  ErrorLogFile = /var/www/example.net/log/error.log
  TimeForCGI = 5
  UseFastCGI = PHP7
  TLScertFile = /etc/ssl/example-net.pem
}
```

in the VirtualHost blob would enable attack detection:

```
PreventCSRF = yes
PreventXSS = yes
PreventSQLi = yes
```

To enable automatic client blacklisting, as well, you need to add the lines

```
BanOnGarbage = 300
BanOnMaxPerIP = 60
BanOnMaxReqSize = 300
```

Listing 2: Configuration Example

```
ServerId = www-data
ConnectionsTotal = 150
ConnectionsPerIP = 10
SystemLogfile = /var/log/hiawatha/system.log
GarbageLogfile = /var/log/hiawatha/garbage.log

Binding {
    Port = 443
    MaxRequestSize = 128
    TimeForRequest = 3.20
    SSLcertFile = hiawatha.pem
}

BanOnGarbage = 300
BanOnMaxPerIP = 60
BanOnMaxReqSize = 300
KickOnBan = yes
RebanDuringBan = yes

CGIextension = cgi
CGIhandler = /usr/bin/perl.pl
CGIhandler = /usr/bin/php-cgi.php
CGIhandler = /usr/bin/python.py
CGIhandler = /usr/bin/ruby.rb
CGIhandler = /usr/bin/ssi-cgi:shmtl

FastCGIServer {
    FastCGId = PHP5
    ConnectTo = 127.0.0.1:2005
    Extension = php
}

UrlToolkit {
    ToolkitID = banshee
    RequestURI isfile Return
    Match ^/(favicon.ico|robots.txt|sitemap.xml)$ Return
    Match .*?(.*) Rewrite /index.php?$1
    Match .* Rewrite /index.php
}

Hostname = 208.77.188.166
WebsiteRoot = /var/www/hiawatha
StartFile = index.html
AccessLogfile = /var/log/hiawatha/access.log
ErrorLogfile = /var/log/hiawatha/error.log
```

```
KickOnBan = yes
RebanDuringBan = yes
BanOnSQLi = 60
BanOnFlooding = 10/1:15
```

outside the VirtualHost block in hiawatha.conf. In fewer than 50 lines (**Listing 2**) you have a complete web server configuration, including various security features, with the lightweight web server application.

Ideal for Containers

The Hiawatha example can even be spun a little further. Because of its manageable size, Hiawatha is also a good candidate as a container web server that can display arbitrary websites. Once you have built a Hiawatha package as described, you can use it to conjure up a Hiawatha container image quickly on the basis of an existing distribution image for Docker or Podman. If you also use a bind mount to give it access to a configuration file in /etc on the host system, along with folders for storing logfiles, Hiawatha mutates into a generically usable web server container.

The benefits are that you can avoid Apache or Nginx, which tend to bloat container images. Hiawatha itself only consists of about 1.5MB of source code, plus about 5MB of code for the mbedTLS implementation that Leisink ships with Hiawatha. The Hiawatha binary can be practically ignored in terms of size, and because Hiawatha also has far fewer dependencies than Apache and others of that ilk, admins have to handle far fewer loose parts, all told, which also reduces the overall administrative overhead, and not just that for running the web server itself.

Interesting for Embedded

Hiawatha is also just as interesting as a web server in the embedded environment. Small computers like the Raspberry Pi and many devices from the embedded environment have sparse hardware reserves, which is where Hiawatha plays to its strengths. Although current Raspberry Pi

implementations no longer suffer so acutely from deficits in terms of CPU and RAM, every CPU cycle and megabyte of RAM you can avoid using still counts.

Even on embedded systems like the Turris Omnia open source router, Hiawatha offers an approach to running a web server in a resource-efficient way. The economy with which Hiawatha operates makes the service ideal in such environments.

Conclusions

Hiawatha is a great project and highly interesting for applications in which a simple, well-functioning web server with basic security features is needed. Many installations that use Apache, Nginx, or similar complex solutions could get by with just the features that Hiawatha offers: the ability to call CGI in the background (usually for PHP), SSL capability, and virtual hosts to run multiple websites on one server.

Hiawatha is far more frugal in its use of resources than its big-name competitors. Especially in environments where CPU cycles and RAM are not available in abundance (e.g., in the embedded environment), Hiawatha is very interesting. Anyone who claims that Hiawatha is more or less a do-it-yourself solution for limited applications is doing the tool an injustice: It is a full-blown web server with a feature set that is completely sufficient for most requirements. ■

Info

- [1] Hiawatha: [\[https://www.hiawatha-webserver.org\]](https://www.hiawatha-webserver.org)
- [2] Banshee framework: [\[https://www.banshee-php.org\]](https://www.banshee-php.org)
- [3] Hiawatha for Debian: [\[https://files.tuxhelp.org/hiawatha/\]](https://files.tuxhelp.org/hiawatha/)
- [4] Anku directory: [\[http://anku.ecualinux.com/7/x86_64/repoview/\]](http://anku.ecualinux.com/7/x86_64/repoview/)

The Author

Martin Gerhard Loschwitz is a Cloud Platform Architect at Drei Austria, where he works on topics such as OpenStack, Kubernetes, and Ceph.

Complete Your Open Source Library with Archive DVDs!

Each fully-searchable archive DVD includes past articles so you can find the content you need quickly.

Save hundreds off the single-copy rate with a convenient archive DVD!



ORDER YOUR DVD NOW!

<https://bit.ly/Archive-DVD>



Setting up HTTP/2 for Nginx

In Racing Trim

HTTP/2, the current representative of the HTTP family, offers several advantages for website operators and their users if the protocol is correctly adapted to the individual scenario. By Oliver Gutperl

Since its beginnings, the World Wide Web has primarily been based on two technologies: the Hypertext Markup Language (HTML) and the Hypertext Transfer Protocol (HTTP). HTML is by far the better known technology and the one with the more turbulent past. In addition to questionable technological overkill (e.g., the `<blink>` tag), conflicts between implementers have been a constant sideshow to the language's evolution. Some further developments such as XHTML have proved to be a technological dead end, causing considerable additional work in quite a few projects.

HTTP/2 at a Glance

Compared with HTML, however, the further development of HTTP has taken place rather covertly and has not been determined by political motives or marketing measures to any great extent. As it has become clear that many websites need to be available 24/7 and that failures could quickly become very expen-

sive – not just in the e-commerce sector – the focus has shifted to solid technology and extensive compatibility. Innovations are therefore slow to catch on and have undergone intensive testing. Today, only about half of all websites support HTTP/2 [1], even though it was standardized in RFC 7540 [2] back in May 2015. In the case of the Nginx web server, the `ngx_http_v2_module` module replaced its predecessor `ngx_http_spdy_module` in September 2015 (Nginx 1.9.5). April 2016 saw the module enter the stable branch (Nginx 1.10.0) [3]. By the way, the development of the Nginx module was financed by Dropbox and Automattic (WordPress). The big websites in particular were early adopters of HTTP/2, preferring it over HTTP/1.1. For them, features like superior compression or request multiplexing were attractive because they directly improved the user experience thanks to higher speed. Today, HTTP/2 for websites is no longer a unique selling point,

but rather a must-have feature that many users expect, if even unknowingly. HTTP/2 is supported by almost all state-of-the-art browsers, including Internet Explorer 11, at least as of Windows 10 [4].

On the major league web servers, however, HTTP/2 has not replaced HTTP/1.1, although it is available in parallel, ensuring downward compatibility with clients that cannot handle HTTP/2 without additional configuration overhead. Another advantage for the system administrator is that important features of HTTP/1.1 remain unchanged in HTTP/2. For example, most HTTP headers and the well-known status codes (1xx, 2xx, ...) are the same, as are the request methods (GET, PUT, etc.).

The bottom line is that it is high time for operators of websites that do not yet support HTTP/2 to make the jump. In this article, I show you how best to implement HTTP/2 with Nginx.

Nginx and HTTP/2

HTTP/2 is provided in Nginx by the `ngx_http_v2_module` module mentioned earlier. You enable it for a server context with the `http2` parameter in the `listen` directive:

Photo by Ludomil Sawicki on Unsplash

```
server {
    listen 443 ssl http2;
}
```

This example reveals one important prerequisite for the use of HTTP/2: The use of Transport Layer Security (TLS), formerly known as Secure Sockets Layer (SSL), is mandatory. HTTP/2 can therefore only be used for encrypted connections between the web browser and the web server because most web browsers request HTTP/2 by Application Layer Protocol Negotiation (ALPN), which in turn is part of TLS. However, this requisite should not be a major hurdle, because an encrypted connection is one of the basic requirements for the secure operation of a web server today. After the launch of Let's Encrypt [5], the creation of a widely accepted TLS certificate has become affordable, even for small projects. After successfully configuring TLS and adjusting the `listen` directive, HTTP/2 is enabled, which could mean the work is done. Important HTTP/2 features such as header compression, request multiplexing, and request pipelining are immediately available without further configuration. Whether further configuration options are necessary depends in practice on the web server's load. If a web server only needs to process a few access requests per unit of time, it can offer the benefits of HTTP/2 to users without any negative effect. As access increases, however, additional configuration steps may be required, and they differ from the optimizations known for HTTP/1.1.

Optimizing the Configuration

To find the optimal configuration, you must understand the differences in the way HTTP/2 and its predecessor protocols process requests. HTTP protocols before HTTP/2 (i.e., HTTP/1.0, HTTP/1.1) are basically based on the idea of the original HTTP protocol (HTTP/0.9), which assumes that a web browser establishes a dedicated TCP connection to the web server for

each request (i.e., for an HTML file, an image, etc.), sends the request, receives the response, and then closes the TCP connection again. With the further development of HTML, however, several requests quickly became necessary to display a complete page in the browser – not only many images, but also additional resources like stylesheets, JavaScript files, tracking pixels, and much more.

To process this flood of requests as quickly as possible with as little latency as possible, web browsers open as many as eight parallel TCP connections to the web server. To reduce the number of TCP connections, HTTP/1.1 introduced persistent connections. Additionally, the protocol specifies that web browsers should not establish more than two parallel connections to a web server (but browser vendors tend not to adhere to this). Nevertheless, it was primarily relevant for the administrator to optimize the configuration of a web server for the highest possible number of simultaneous connections. Moreover, special image domains were often introduced in collaboration with application development (*images.example.com*) to handle more connections, resulting in pressure on application development to reduce the number of resources required for a website. The result, among other things, was the use of bundling (e.g., with Webpack) and spriting (creating a collage of multiple images in a file that is cut up again in the web browser).

HTTP/2 introduces streams and multiplexing that significantly advance the concept of persistent connections. A single TCP connection between the web browser and the web server can now process any number of requests, even in parallel. The number of concurrent TCP connections is now more or less equal to the number of concurrent users and, therefore, significantly lower compared with previous protocol versions. At the same time, however, the average duration of a TCP connection increases and possibly its memory requirements, as well, because more data traffic is now handled over this one connection.

Important Configuration Options

When configuring Nginx, two HTTP/2-specific configuration directives come to the fore that let you customize the web server to suit your application.

1. The `http2_idle_timeout` directive specifies how long an HTTP/2 connection is kept open after the last data exchange. It is comparable to the older `keepalive_timeout` directive for HTTP/1.1. Even the default values provided by Nginx – 75 seconds for `keepalive_timeout`, 180 seconds for `http2_idle_timeout` – show that different use cases have been considered. Although 75 seconds might be considered an average value for potential navigation by the user to the next web page, 180 seconds is more of a maximum value. These times are intended to ensure that any navigation by the user will be handled by the existing connection to the extent possible.
2. The `http2_max_requests` directive specifies the maximum number of requests that can be processed over a given TCP connection. Again, it parallels HTTP/1.1 (`keepalive_requests`), and again the default values (100 vs. 1,000) show that the optimization goal has changed significantly.

Both directives are necessary to protect the web server resources. The `http2_idle_timeout` directive is intended to prevent concurrent connections to the web server from being blocked by clients that don't really need them. Because the amount of memory required per connection grows with the number of requests for that connection, `http2_max_requests` is intended to prevent a connection from taking up too much memory for itself without actually needing it yet.

Characteristics of a Web App

In common, the two HTTP/2 directives are much more sensitive to the anatomy of the website or application

compared with their older counterparts. To illustrate, I will draw on two (extreme) examples: the web version of RFC 7540 [2] and a web application such as Google Docs.

The RFC page is characterized by a minimal number of requests to display the page (the initial HTML document and a graphic as a favicon), a low probability of user navigation (because the page contains few links), and a long viewing time (because the document is very large and thus the reading time tends to be high).

Web applications, on the other hand, have completely different characteristics: Even the display of the first page requires dozens of requests. The multiple interaction options increase the likelihood that the user will navigate. Additionally, XHR requests can be triggered at any time by JavaScript. The user is likely to interact with the page for an extended period of time, which increases the probability that any of these interactions take place. If you look at these extremes, you can see that correctly chosen configuration values become enormously important in HTTP/2. A very high value for `http2_idle_timeout` is directly harmful in the case of RFC retrieval, because valuable connections remain unused. For the web application, on the other hand, a very high value directly improves the user experience, because the latency caused by the connection setup can be avoided in the best case.

A high value for `http2_max_requests` is probably not harmful in the RFC

example because the connection is very likely to be terminated before the maximum number of requests is reached. In the case of the web application, however, the advantages of HTTP/2 can only be exploited if the value is set high enough so that at least the initial page setup and the first user interactions can be handled completely after connection, so that no new connection setup is necessary.

These examples show two things: First, HTTP/2 can hardly realize its full potential if the administrator tends to be conservative in configuring it to average values. Second, the configuration is closely related to the application – not all websites are the same. Moreover, you can also deduce that bundling and spriting have significantly less effect on the speed of a website. Therefore, a cost-benefit analysis with HTTP/2 looks different from that for HTTP/1.1. New projects, especially, might not need a bundling pipeline set up, and the graphics department can dispense with spriting.

Measure, Don't Guess

With HTTP/2, determining the appropriate configuration through measurements becomes more important. Because all modern browsers offer a more or less sophisticated developer console that allows deep insights into the anatomy of a website, metrics are not difficult to come by. In the following example,

I use the Chrome developer tools (DevTools).

The *Network* tab (Figure 1) helps when trying to determine the number of requests for a common user interaction. Here, you select *Preserve Log* and *Disable Cache* and then use the search field to restrict the domain you want to examine (e.g., *domain:www.linux-magazin.de*). Now you load the website and make typical user actions. In the footer you can read the number of processed requests. The *Protocol* column, for which you might need to right-click on the column headers, shows that the website is using HTTP/2 (*h2*). In the *Connection ID* column, you can then see the ID of the TCP connection, which you can use to discover when the browser establishes a new TCP connection. The marketing department can give you an idea of how long users stay. The usual analysis tools (e.g., Google Analytics) provide good information in this regard. However, do not be deceived by low average values: You might need to look at the power user segment separately.

Freestyle: Server Push

HTTP/2 also offers another feature that significantly improves the user experience: server push, with which it is quite possible to predict which resources (images, JavaScript, etc.) the client will need, to display a web page fully.

To begin, it is enough to look into the header of an HTML file, which

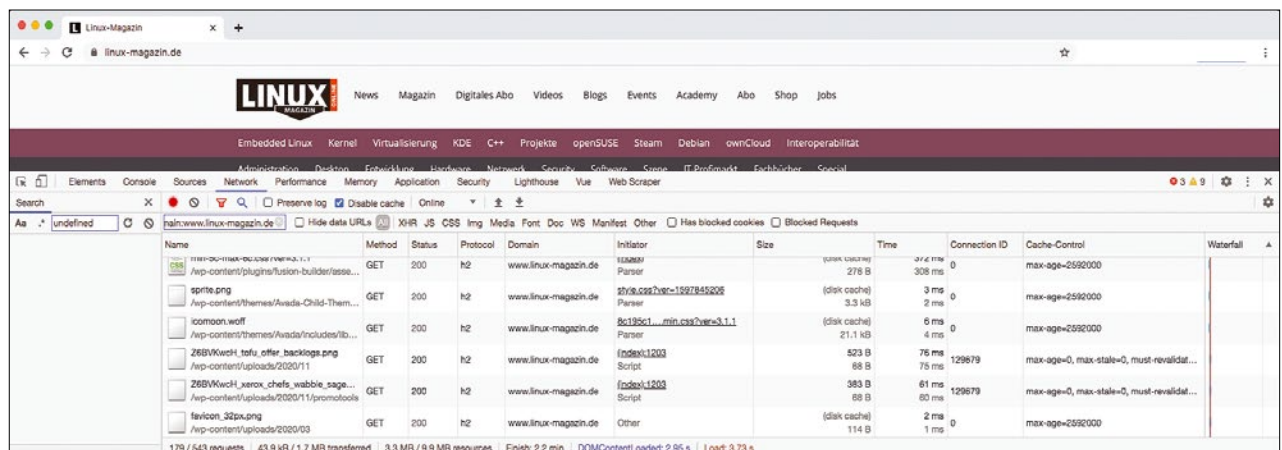


Figure 1: Analysis of web traffic with Chrome DevTools.

is usually a colorful mix of CSS and JavaScript resources. This information is also available to the web browser, but only after it has downloaded and parsed the initial HTML document. Only then does it send a request for the additional resources to the server and must wait for another network roundtrip before the responses arrive. With server push in HTTP/2, you can send these additional resources directly after the response for the initial HTML file – before the browser even knows it will need them. The browser accepts the additional responses and stores them in the browser cache. If a request for a specific resource is then made during parsing, the response comes immediately and at high speed from the local cache; a network roundtrip to the server is therefore unnecessary.

A requirement of the resources that are to be sent by server push is that they must be bufferable. The browser must be able to read from the response headers, which work the same way for server push as for a normal request, in that the response will remain valid for a while. Nginx handles this requirement with the usual `expires` configuration directive. Once set, you can use the `http2_push` directive to push individual files to the client. In the example

```
location /index.html {
    http2_push /css/styles.css;
    http2_push /js/scripts.js;
}
```

the `styles.css` and `scripts.js` files are sent directly with the response to a request for `index.html`. These actions can be tracked with Chrome DevTools by looking at the *Initiator* column. If *Disable cache* is enabled, the *Push / ...* entry shows that the file arrived at the browser by HTTP/2 push. Be careful, though: Not all files the server has sent by push will appear here. If the current HTML page (so far) did not contain any request for a file, then it will not appear in the list. However, as soon as such a request is made, it appears there. With *Disable cache* checked, however, the browser does not behave

like a normal user. Alternatively, you can check the Nginx access log to see whether the server push is working correctly. Files sent by server push initially appear in the access log as normal requests, but if you add the keyword `$msec` to the Nginx log output,

```
log_format http2 '$time_iso8601 ⚡
    $msec$status $connection ⚡
    $http2 "$request";'
```

you can see that all requests were processed at exactly the same time (**Listing 1**), which indicates a push. If you know the behavior of users on your own site well, you can consider pushing the most frequently visited follow-up page directly:

```
location /index.html {
    http2_push /css/styles.css;
    http2_push /js/scripts.js;
    http2_push /<next-page>.html;
}
```

If the user follows this navigation path, the browser serves the corresponding request for `/<next-page>.html` from the local cache with virtually no time delay.

Note, though, that Nginx does not remember which resources have been sent to a client and are most likely cached there. Therefore, configuration errors can cause Nginx to send resources that are already available to the client, wasting network bandwidth. Especially for resources with very different cache durations, you should pay very close attention to this possibility.

Nginx as Proxy

When Nginx works as a proxy in front of an application server, customizing all details of the web application in the HTTP/2 push configuration is

Listing 1: Server Push in Log

```
2020-11-22T12:01:10+01:00 1606042870.567 200 605 h2 "GET /index.html HTTP/2.0"
2020-11-22T12:01:10+01:00 1606042870.567 200 605 h2 "GET /css/styles.css HTTP/2.0"
2020-11-22T12:01:10+01:00 1606042870.567 200 605 h2 "GET /js/scripts.js HTTP/2.0"
```

not practical. At this point, the `http2_push_preload` directive helps:

```
location /proxy/ {
    proxy_pass http://applicationserver/;
    http2_push_preload
    on;
}
```

This directive processes link headers of upstream resources and sends resources marked with `rel=preload` automatically (i.e., whenever a response from the application server sends this header along). For example, with the following statement, Nginx automatically sends the `/css/styles.css` resource as a push to the client in addition to the response itself:

```
link: </css/styles.css>; rel=preload; ⚡
    as=style
```

This mechanism allows control of the push behavior directly from the web application. Because the web application probably has state management, it is also much easier to avoid unnecessary pushes at this point. For example, resources can only be sent by push at the start of a session.

Listing 2 shows an example Nginx configuration for HTTP/2 that can serve as a starting point for your own experiments.

Conclusion

Even without special configuration, enabling HTTP/2 brings a speed advantage to your website. With careful tuning of the maximum requests and idle times per connection, you can reduce the load on your server. Push mechanisms can also significantly improve the speed perceived by the user. In practice, implementing these optimizations will require the cooperation of the software development department, which in return, will

Listing 2: Example Configuration

```

01 http {
02
03     log_format http2 '$time_iso8601 $msec $status $connection $http2
    "$request"';
04
05     server {
06
07         listen 443 ssl http2; # TLS and HTTP/2
08
09         http2_idle_timeout 3m;
10         http2_max_requests 1000;
11
12         access_log /usr/local/var/log/nginx/http2.log http2;
13
14         ssl_certificate /test.crt;
15         ssl_certificate_key /test.key;
16
17         root /var/www/html;
18
19         # Required for server push:
20         location /css/ {
21             expires 3h;
22         }
23
24         location /js/ {
25             expires 3h;
26         }
27
28         location /index-2.html {
29             expires 3h;
30         }
31
32         # Example for server push
33         # Call: GET /
34         location /index.html {
35             # These files are sent by push,
36             # when the client calls /index.html:
37             http2_push /css/styles.css;
38             http2_push /js/scripts.js;
39             # expected next navigation destination:
40             http2_push /index-2.html;
41         }
42
43         # Example of Nginx as proxy with http2_push_preload.
44         # The (simulated) application server is below.
45         # Call: GET /applicationserver/
46         location /applicationserver/ {
47             proxy_pass http://localhost:1025/;
48             # Process link header with rel=preload:
49             http2_push_preload on;
50         }
51     }
52
53     # Simulation application server:
54     server {
55         list 1025;
56         root /var/www/html;
57
58         location ~ .html$ {
59             # Hints for http2_push_preload:
60             add_header link "</css/styles.css>; rel=preload; as=style";
61             add_header link "</js/scripts.js>; rel=preload; as=script";
62         }
63     }
64 }

```

reduce their efforts required for bundling and spriting.

Overall, the development of HTTP/2 clearly shows a considerable potential for optimization in closer coordination between software development and the web server infrastructure. With the release of the first draft of HTTP/3 in November 2020, it also became clear that such investments are worthwhile. Most likely, HTTP/3 will mainly optimize the multiplexing mechanisms introduced with HTTP/2 with respect to the underlying network layer. It

is therefore safe to assume that, if HTTP/2 mechanisms are used sensibly, you will soon be able to benefit directly from the optimizations to be expected from HTTP/3. Therefore, the use of HTTP/2 already appears to be a worthwhile optimization that can be strongly recommended to every web server operator.

Info

- [1] Usage statistics for HTTP/2:
[\[https://w3techs.com/technologies/details/ce-http2\]](https://w3techs.com/technologies/details/ce-http2)

- [2] RFC 7540:

[\[https://tools.ietf.org/html/rfc7540\]](https://tools.ietf.org/html/rfc7540)

- [3] Changes in Nginx 1.10:

[\[http://nginx.org/en/CHANGES-1.10\]](http://nginx.org/en/CHANGES-1.10)

- [4] HTTP/2 support in browsers: [\[https://caniuse.com/?search=HTTP%2F2\]](https://caniuse.com/?search=HTTP%2F2)

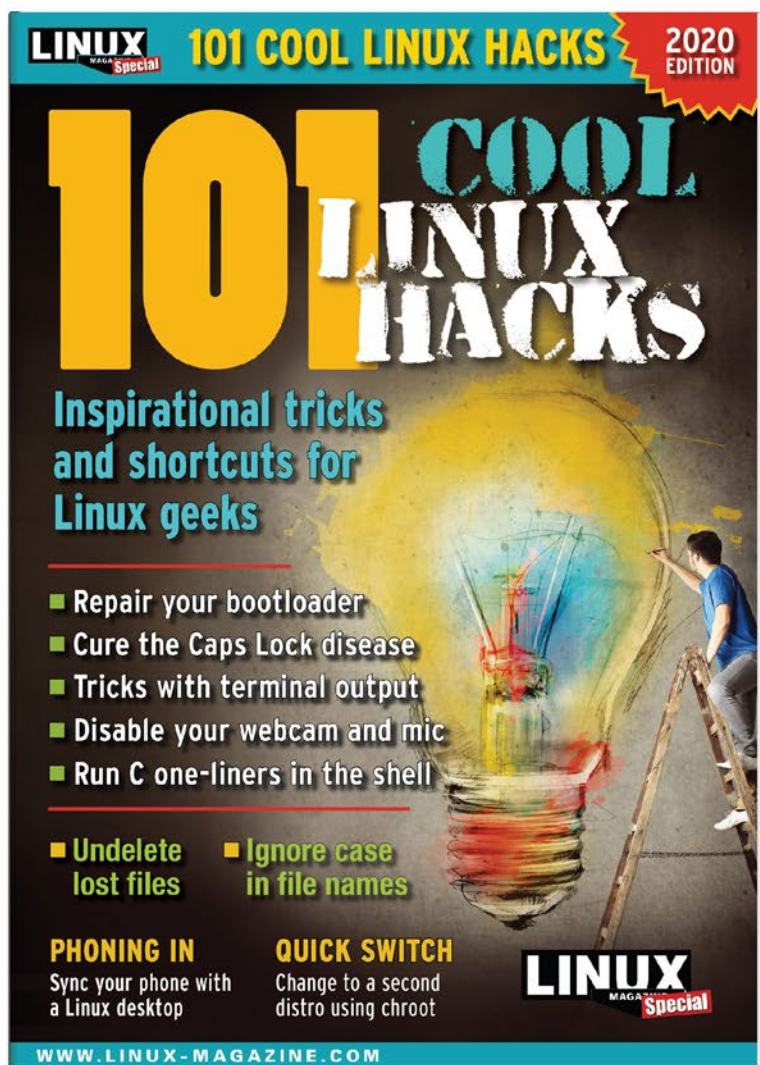
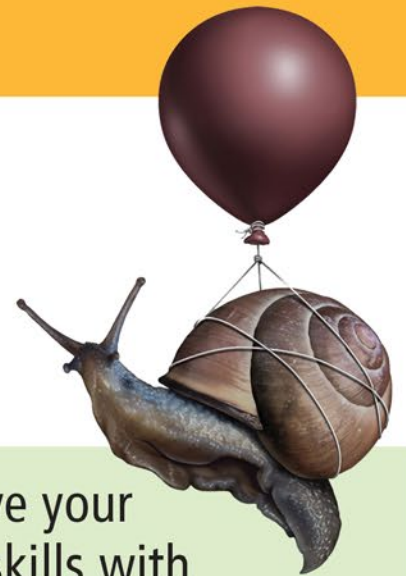
- [5] Let's Encrypt: [\[https://letsencrypt.org\]](https://letsencrypt.org)

The Author

Oliver Gutperl has been involved in the optimization of server applications and the interaction between infrastructure and software development for more than 20 years. He is the author of the book *Nginx richtig konfigurieren* (*Configuring Nginx Properly*, in German).

SHOP THE SHOP
shop.linuxnewmedia.com

GET PRODUCTIVE WITH 101 LINUX HACKS



Improve your Linux skills with this cool collection of inspirational tricks and shortcuts for Linux geeks.

- Undelete lost files
- Cure the caps lock disease
- Run C one-liners in the shell
- Disable your webcam and mic
- And more!

ORDER ONLINE:
shop.linuxnewmedia.com/specials



Setting up the lightweight Lighttpd web server

Fast Delivery

The long-established Lighttpd web server is lean and fast and can be set up quickly thanks to its simple configuration. By Tim Schürmann

In terms of functionality, the small Lighttpd [1] web server need not shy away from comparison with more prominent competitors. Among other features, it supports virtual host strategies and can change its configuration to reflect incoming requests. If necessary, further functions can be added with the use of modules. For example, access to individual websites can be restricted, connections can be secured by TLS, and scripting languages can be integrated over the FastCGI interface, which also has a built-in load balancer that distributes incoming requests to multiple PHP servers.

By the way, the web server's name is pronounced "Lighty"; some websites even use this phonetic spelling as a synonym of the official name. Lighttpd supports HTTP 1.0/1.1 and encrypted connections over HTTPS. Versions 1.4.56 or later support the newer HTTP/2. Because of its small footprint, Lighttpd is often used on smaller or less powerful systems. For example, it is the engine for the popular Pi-hole [2] network filter and ad blocker. The easy-to-understand configuration file allows lightning-fast setup.

Quick Installation

Most distributions have Lighttpd in their repositories. On Ubuntu, you can install it with the command:

```
$ sudo apt install lighttpd lighttpd-doc
```

Some distributions keep only rarely needed modules in separate packages with names that often start with *lighttpd-*. In Ubuntu's case, *lighttpd-webdav* retrofits a WebDAV interface, for example. For your first steps with the web server, though, you will not normally need these modules.

To build Lighttpd from the source code, you need at least a C compiler, Make, the *pcrc-config* tool, and the developer packages for Zlib and Bzip2. On Ubuntu, the command

```
$ sudo apt install build-essential \
libpcre3-dev zlibg-dev libbz2-dev
```

retrieves everything you need. The numerous dependencies for all optional Lighttpd modules are listed in the Lighttpd wiki [3]. You can compile the Lighttpd source code from the project website with the familiar three-step process:

```
$ configure
$ make
$ sudo make install
```

In the *doc/systemd/* source code directory, you will find matching configuration files for *systemd*.

Quick Setup

All of the settings for the web server are grouped in the *lighttpd.conf* file, which is located by default in the */etc/lighttpd/* directory (see also the "File Split" box). As a starting point for your own *lighttpd.conf*, I would recommend the simple configuration shown in Listing 1. On each line, you have the name of a setting on the left, followed by the associated value on the right, separated by an equals sign.

File Split

To avoid getting lost in complex configurations, Lighttpd settings can be split up among several files. The files in turn usually end up in the */etc/lighttpd/conf.d/* and */etc/lighttpd/vhosts.d/* subfolders. In the *lighttpd.conf* file, the lines

```
include "/etc/lighttpd/conf.d/*"
include "/etc/lighttpd/vhosts.d/*"
```

add all the configuration files from the two directories. Which subdirectories Lighttpd uses depends on the distribution. Ubuntu uses the *conf-enabled/* and *conf-available/* subfolders. In *conf-available/*, each file groups settings that belong together. For example, *15-fastcgi-php.conf* stores the configuration for the FastCGI module. However, Lighttpd only evaluates the files in the *conf-enabled/* directory in the configuration provided on Ubuntu. Like Apache, a symbolic link to the *15-fastcgi-php.conf* file in the *conf-enabled/* directory is all you need to enable the FastCGI settings.

Photo by Brett Jordan on Unsplash

Lighttpd ignores lines that start with the hash symbol (#).

In more complex configurations, some domain names, directories, or URL components appear in several places. To save typing, you should define variables for such information to use as placeholders later.

Listing 1 makes use of this possibility right at the beginning. The `/var/www` directory is assigned to the `var.dir` variable.

Another advantage of variables becomes apparent if the directory changes at some point: All you need to do is adjust the `var.dir` variable. The variable name must begin with `var.` but can be freely chosen otherwise. The only exceptions are `var.PID` with the process ID of the running Lighttpd and `var.CWD` with the current working directory.

The `server.document-root` setting is followed by the directory where all the web pages are stored (the document root). Note that the `var.dir` variable is used here with `/html` appended to its value. In general, the `+` operator allows any two strings to be concatenated. In the example, this specifies a folder named `/var/www/html`.

The web server later listens on the port that follows `server.port`. To access port 80, Lighttpd must be launched with root privileges. To avoid vulnerabilities, the server in **Listing 1** automatically switches to the `www-data` account in a group of the same name (i.e., to the user ac-

count provided by Debian and Ubuntu for this purpose).

Food for Talk

The `server.errorlog` parameter specifies the text file in which Lighttpd logs its messages and will also contain all the warnings and notices. Adding `server.errorlog-use-syslog = "enable"` would tell the web server to write these messages to the syslog. Finally, the additional line `server.use-ipv6 = "enable"` would enable support for IPv6.

In its response, the web server sends along the MIME type of the transmitted data, such as `text/html` for an HTML document. To assign the appropriate MIME type to individual file extensions, you use `mimetype.assign`. The setting demonstrates the use of a list: The assignments are each placed between parentheses as a comma-separated list. In **Listing 1**, the individual values of the list are on a separate line, just for the sake of clarity, but the formatting is not significant.

PHP scripts should be executed by the web server and not delivered to the browser. The extensions of files to be ignored are listed in `static-file.exclude-extensions`. When a browser

accesses a directory, Lighttpd looks for one of the files listed after `index-file.names` and returns it. If none of these index files exists, Lighttpd returns a 403 error message. Unlike many competitors, Lighttpd prohibits its clients from accessing directories directly – this kind of access

Listing 1: Simple Lighttpd Config

```
var.dir = "/var/www"

# Define basic settings:
server.document-root = var.dir + "/html"
server.port = 80
server.username = "www-data"
server.groupname = "www-data"
server.errorlog = "/var/log/lighttpd/error.log"

mimetype.assign = (
    ".html" => "text/html",
    ".txt" => "text/plain",
    ".jpg" => "image/jpeg",
    ".png" => "image/png"
)

static-file.exclude-extensions = ( ".fcgi", ".php", ".~", ".inc" )
index-file.names = ( "index.html" )
```

has to be permitted explicitly with the `dirlisting` module.

Supplying the Configuration

As an experiment, save your `lighttpd.conf` file if it already exists, and then create a new configuration file with the contents from **Listing 1**. Type

```
lighttpd -t -f /etc/lighttpd/lighttpd.conf
```

to first check for syntax errors (**Figure 1**). A typo like `server.port=80` would not be detected. You should therefore still check the error log after reloading the configuration. **Figure 2** shows a configuration test with the parameter `-D`, which outputs messages to the terminal.

The new settings are applied by the web server as soon as it receives the `SIGUSR1` signal. If you have a distribution that uses `systemd`, use:

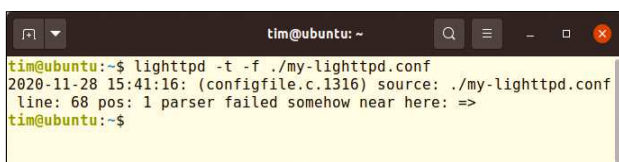
```
sudo systemctl reload lighttpd.service
```

Otherwise the manual approach is to run

```
sudo kill -n SIGUSR1 <PID>
```

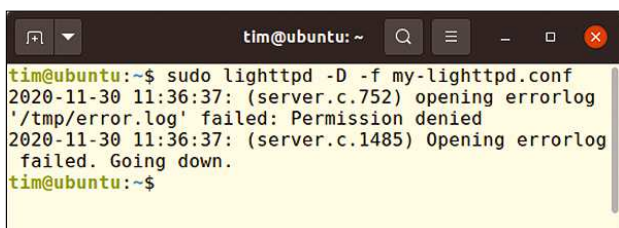
To discover the `<PID>`, run:

```
ps -A | grep lighttpd
```



```
tim@ubuntu: ~
tim@ubuntu:~$ lighttpd -t -f ./my-lighttpd.conf
2020-11-28 15:41:16: (configfile.c.1316) source: ./my-lighttpd.conf
line: 68 pos: 1 parser failed somehow near here: =>
tim@ubuntu:~$
```

Figure 1: The `-t` parameter tells Lighttpd to check syntax.



```
tim@ubuntu: ~
tim@ubuntu:~$ sudo lighttpd -D -f my-lighttpd.conf
2020-11-30 11:36:37: (server.c.752) opening errorlog
'/tmp/error.log' failed: Permission denied
2020-11-30 11:36:37: (server.c.1485) Opening errorlog
failed. Going down.
tim@ubuntu:~$
```

Figure 2: The `-D` switch keeps Lighttpd in the foreground and outputs messages to the terminal, which would be useful for a quick test of a new configuration.

Table 1 lists the signals that Lighttpd understands. If the web server is not yet running, typing either of the lines,

```
sudo systemctl start lighttpd.service
sudo lighttpd -f /etc/lighttpd/lighttpd.conf
```

will enable it at once.

Flexible Condition

Imagine you want Lighttpd always to return pages from `/var/www/html/blog` when a browser requests the blog from `blog.example.com`. You can manage this in the configuration file with conditionals. The lines

```
$HTTP["host"] == "blog.example.org" {
    server.document-root =
        var.dir + "/html/blog"
}
```

test whether the requested hostname is identical to `blog.example.com`. The `$HTTP["host"]` element stands for the requested host name; all data available as an alternative is summarized in **Table 2**. If the condition is true, Lighttpd evaluates all the settings in the brackets. The example defines `/var/www/html/blog/` as the document

root folder. All settings between the curly braces thus only apply to `blog.example.org`. In this way, completely different configurations can be stored for different domains and, in turn, virtual hosts can be implemented. Instead of `==`, a `!=` tests for inequality. Additionally, more complex rules can be formulated with regular expressions. Lighttpd evaluates settings in the curly brackets if the regular expression to the right of `=~` is true or the expression to the right of `!~` is false. For example,

```
$HTTP["url"] =~ "^/blog/"
```

would check to see whether the URL requested by the browser starts with `/blog/`. Lighttpd also supports conditionals. Consequently, a conditional can contain other conditions between the curly braces.

Communication Secret

Most of Lighttpd's functions are provided by modules that can be activated as needed. Encrypted connections over TLS are provided by the `mod_openssl` module,

```
server.modules = ("mod_openssl")
$SERVER["socket"] == ":443" {
```

```
    ssl.engine = "enable"
    ssl.pemfile = "/etc/lighttpd/server.pem"
}
```

which in turn relies on OpenSSL. In Lighttpd 1.4.56, four more modules joined the mix, handling encryption with mbed TLS (`mod_mbedtls`), wolfSSL (`mod_wolfssl`), GnuTLS (`mod_gnutls`), and NSS (`mod_nss`). However, they are still considered experimental.

To use the SSL module, you must first load it in `lighttpd.conf`:

```
server.modules = ( "mod_openssl" )
```

The settings for the module end up back in the central configuration file. In the case of `mod_openssl`, the `mod_openssl` module lines shown previously are all you need: The conditional checks whether the request has come in through port 443. In this case, `ssl.engine` enables encryption with the certificate specified by `ssl.pemfile`. You can generate a suitable self-signed certificate with OpenSSL:

```
$ openssl req -new -x509 -keyout /etc/lighttpd/server.pem -out /etc/lighttpd/server.pem -days 365 -nodes
```

Lighttpd also supports the Let's Encrypt project. The somewhat more complex configuration required for this is explained in detail on a wiki page [\[4\]](#).

Account Control

The `mod_auth` module takes care of authentication. To activate it, either add it to the list after `server.modules` or add it later to `lighttpd.conf` with the `+=` operator:

```
server.modules += (
    "mod_auth", "mod_authn_file")
```

Two modules are needed for access control: `mod_auth` fields a username and password and then asks a backend whether the user has sufficient access rights. The backends in turn provide other modules, one of which

Table 1: Signals Supported by Lighttpd

Signal	Response
SIGTERM	Terminates Lighttpd immediately; existing connections are interrupted
SIGINT	Lighttpd responds to all current requests and then terminates
SIGUSR1	Lighttpd responds to all current requests and then reloads its configuration
SIGHUP	Lighttpd reopens its logfiles but does not reload the configuration

Table 2: Data in Conditionals

Field	Meaning
<code>\$REQUEST_HEADER["..."]</code>	The information specified in the quotes from the request header (e.g., <code>\$REQUEST_HEADER["User-Agent"]</code> refers to the user agent)
<code>\$HTTP["request-method"]</code>	Request method
<code>\$HTTP["scheme"]</code>	Schema of incoming connection, either http or https
<code>\$HTTP["host"]</code>	Hostname
<code>\$HTTP["url"]</code>	Complete URL path without the domain name and query strings
<code>\$HTTP["querystring"]</code>	Query string (everything after the <code>?</code> in the URL)
<code>\$HTTP["remoteip"]</code>	Remote IP or remote network; only works with IPv4
<code>\$SERVER["socket"]</code>	Socket; only works with the <code>==</code> operator; additionally, the value must have the format <code><IP>:<Port></code> .

is `mod_authn_file`. This module provides several backends, all of which read user data from text files. Other available backends use PAM or an LDAP server instead.

Which backend `mod_auth` needs to use is defined by a setting in `lighttpd.conf`. **Listing 2** uses the plain backend, which expects the passwords in plain text in a simple text file. `auth.backend.plain.userfile` tells the appropriate module to fetch the user data from the `/etc/lighttpd/user.txt` file. It contains the user names and passwords on each line in a `<User>:<Password>` format. Finally, `auth.require` specifies that access to the `/blog` URL is restricted, the password is in the clear (`"method" => "basic"`), and any authorized user is granted access (`"require" => "valid-user"`).

The example shows an easily set up but relatively insecure form of authentication. Lighttpd and its modules do not support the `.htaccess` files known from Apache. As a consequence, a web application that builds on that basis could be vulnerable on Lighttpd.

Distribution Box

Lighttpd prefers to integrate scripting languages over the FastCGI interface, which the `mod_fastcgi` module installs retroactively. At the same time, it includes a load balancer that distributes the load across multiple FastCGI servers. Suitable settings for `lighttpd.conf` are shown in the example in **Listing 3**; the module distributes the requests evenly among the servers by a round-robin method (line 2). The settings in the `fastcgi.server` module reveal to which FastCGI servers Lighttpd sends a script for execution. In the example in **Listing 3**, the web server passes all files with a `.php` extension to one of two FastCGI servers. The first of the two has IP address `192.168.0.1` and port `1026`, and the second sits at the same port on IP address `192.168.0.2`.

On Ubuntu, you just need to install the `php7.4-fpm` package so PHP is

```
auth.backend = "plain"
auth.backend.plain.userfile = "/etc/lighttpd/user.txt"
auth.require = ( "/blog" => ("method" => "basic", "realm" => "application", "require" => "valid-user") )
```

waiting on a socket for incoming PHP code. In this case, you only need to point the web server to the appropriate socket:

```
fastcgi.server = ( ".php" =>
    ( ( "socket" => "/run/php/php-fpm7.4" ) )
)
```

Lighttpd always executes the modules in the order in which they appear after `server.modules` or in `lighttpd.conf`. Therefore, you should always load the modules that control access first (e.g., `mod_auth`) and only then load modules that generate and return content (e.g., `mod_fastcgi`). Otherwise, Lighttpd might skip authentication.

Detailed documentation (**Figure 3**) of all official modules and the web server can be found in the Lighttpd wiki [\[5\]](https://redmine.lighttpd.net/projects/lighttpd/wiki).

Conclusions

The work on Lighttpd is currently limited to maintenance and careful ongoing development of the 1.4 branch. The web server is lagging behind its competitors slightly when it comes to new technologies. Nevertheless, deployment of Lighttpd would make sense if a workgroup needed a web server at short notice, because it can be set up quickly, thanks to its compact and intelligible configuration, and it can be adapted to a team's needs just as quickly through the use of modules.

Info

- [1]** Lighttpd: [\[http://www.lighttpd.net/\]](http://www.lighttpd.net/)
- [2]** Pi-hole: [\[https://pi-hole.net/\]](https://pi-hole.net/)
- [3]** Lighttpd module dependencies: [\[https://redmine.lighttpd.net/projects/lighttpd/wiki/OptionalLibraries\]](https://redmine.lighttpd.net/projects/lighttpd/wiki/OptionalLibraries)
- [4]** Let's Encrypt in Lighttpd: [\[https://redmine.lighttpd.net/projects/lighttpd/wiki/HowToSimpleSSL\]](https://redmine.lighttpd.net/projects/lighttpd/wiki/HowToSimpleSSL)

Listing 2: Determining Backend

Listing 3: FastCGI Config for Lighttpd

```
server.modules += ( "mod_fastcgi" )
fastcgi.balance = "round-robin"
fastcgi.server = (
    ".php" => (
        ( "host" => "192.168.0.1",
          "port" => 1026,
        )
        ( "host" => "192.168.0.2",
          "port" => 1026,
        )
    )
)
```

[5] Lighttpd wiki: [\[https://redmine.lighttpd.net/projects/lighttpd/wiki\]](https://redmine.lighttpd.net/projects/lighttpd/wiki)

The Author

Tim Schürmann is a freelance computer scientist and author. Besides books, Tim has published various articles in magazines and on websites.



Figure 3: The Lighttpd wiki provides an overview of existing modules, which cover all the features that could be important in daily work, such as URL rewriting (`mod_rewrite`).

Set up subdomains with Apache and Nginx

Sublet

Set up virtual hosts on modern web servers for Apache and Nginx. By Stefan Wintermeyer

The shortage of addresses and the quest for efficient resource usage established web server virtualization well before the topic of virtual machines and containers gained its current prominence. In this article, I demonstrate how to use different fully qualified domain names

(FQDNs) on the same physical system with virtual hosts.

Virtual Hosts

In the mid-1990s, a single web server (usually the Apache server, which quickly became the de facto standard on Linux) typically ran with a single IP address. Once installed and configured, it reliably delivered a web page – always the same page – after requests to that IP address. Not only was it impractical to have a separate computer for each web server, no matter how small, but IPv4 addresses were becoming scarce. Virtualized hosts didn't exist yet, so in version 1.1 the Apache developers conjured up the strategy of virtual hosts as a solution. Suddenly it didn't matter how many FQDNs pointed to the same IP address: The web server took care of the correct distribution of requests. That's why the web browser doesn't just contact the plain vanilla IP address that it gets from the domain name server (DNS) after resolving the FQDN, it additionally specifies the desired host name directly after the GET request with a `HOST` specification (e.g., over Telnet on port 80, as in [Listing 1](#)).

In this example, the web server responded with a *301 Moved Permanently* HTTP response, which means that the resource no longer exists and the new URL should be used because the server no longer delivers the web page over HTTP, but only has an SSL-encrypted version over HTTPS. To make life as easy as possible for end users and search engines, the server operator then defines a 301 redirect. In this case, the browser tries a second request to the new URL. For temporary rather than permanent redirection, you need to use HTTP status code 302. With SSL encryption, you have no way around this functionality, even if you have the luxury of a dedicated IP address for your web server.

Apache vs. Nginx

Even though Apache was one of the pioneers, if not the decisive pioneer for stable and fast web servers on Linux, Nginx has a slightly higher market share today. As of January 2021, it accounts for 33.3 percent of all web servers worldwide, compared with 26.4 percent for Apache [1]. As you can see, both web servers are very popular in the community, so in

Listing 1: Virtual Host Output

```
$ telnet www.wintermeyer.de 80
```

```
Trying 89.221.14.204...
Connected to www.wintermeyer.de.
Escape character is '^['.
```

```
GET / HTTP/1.1
HOST: www.wintermeyer.de
```

```
HTTP/1.1 301 Moved Permanently
Server: nginx/1.10.3
Date: Fri, 27 Nov 2020 14:00:25 GMT
Content-Type: text/html
Content-Length: 185
Connection: keep-alive
Location: https://www.wintermeyer.de/
```

```
<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.10.3</center>
</body>
</html>
```

the following examples, I look at the configuration for both Nginx (with Server blocks) and Apache (with VirtualHost blocks).

The starting point will be a freshly installed Debian 10.6 “Stable,” on which I configure virtual hosts for the imaginary domains *www.example1.de* and *www.example2.de*. The packages from the distribution’s repository will come into play. The IP address for the server is 192.168.2.120.

Apache

The first task is to install the web server. In Apache’s case, you can use:

```
# apt-get update
# apt-get -y install apache2
```

To check that the web server is installed and running, enter:

```
systemctl status apache2
```

The default website can be retrieved by pointing a web browser to *http://192.168.2.120*. The website returned is located in the */var/www/html/index.html* file.

For each of the two new web pages, you create a separate directory:

```
# mkdir /var/www/www.example1.de
# mkdir /var/www/www.example2.de
```

In each of the two directories, store an *index.html* with the content from [Listing 2](#), each with an appropriately customized domain name. Then, set the permissions to *rwX-rX-rX*

```
# chmod -R 755 /var/www/2
www.example{1,2}.de
```

for these directories.

Configuring Apache

The configurations for the web servers available and enabled on the system are located in the directories shown in [Listing 3](#). Now you need to create a minimal basic configuration for *www.example1.de* in */etc/apache2/sites-available/www_exam-*

ple1_de.conf ([Listing 4](#)), and do the same for *www.example2.de*.

The two configurations are not yet live, however, so you either need to link the configuration files manually with */etc/apache2/sites-enabled/* or use the *a2ensite* tool:

```
# a2ensite www_example1_de.conf
```

The next step is to tell Apache to parse the new configuration and make the virtual hosts available:

```
# systemctl reload apache2
```

Again, you can use Telnet to test this, as in [Listing 1](#).

To disable a virtual host, you could delete the corresponding link with *rm*; however, it is better to use the appropriate tool and parse the changed configuration:

```
# a2dissite www_example1_de.conf
# systemctl reload apache2
```

To make your work as easy as possible as an admin, you will want to use only one FQDN per configuration file, but feel free to use it for both HTTPS and HTTP. Technically you could put 100 different FQDNs in one file, but that would be far more confusing.

Redirects with Apache

One simple example of the use of virtual hosts for the same FQDN is redirecting all *http://example1.de* to *http://www.example1.de*, which might also work the other way around, depending on what the marketing department or in-house search engine optimization guru thinks is more fashionable at the moment. The file */etc/apache2/sites-available/www_example1_en.conf* has to contain the code from [Listing 5](#). To test, run:

```
curl -I http://example1.de
```

This command outputs the headers and initially returns a *301 Moved Permanently* status message stating the new location, *http://www.example1.de*. The command

```
curl -I http://www.example1.de
```

should then generate the output from [Listing 6](#).

Nginx

Nginx also needs to be set up according to the previous example,

```
apt-get -y install nginx
```

Listing 2: index.html

```
<html>
<head>
  <title>Test for www.example1.de</title>
</head>
<body>
  <h1>Test for www.example1.de</h1>
</body>
</html>
```

Listing 3: Configuration Directories

```
# tree /etc/apache2/sites-*
/etc/apache2/sites-available
--- 000-default.conf
default-ssl.conf
/etc/apache2/sites-enabled
--- 000-default.conf -> ../sites-available/000-default.conf
```

Listing 4: Minimal Configuration

```
<VirtualHost *:80>
  ServerName www.example1.de
  DocumentRoot /var/www/www.example1.de/
</VirtualHost>
```

Listing 5: Redirects

```
<VirtualHost *:80>
  ServerName example1.de
  Redirect permanent / http://www.example1.de/
</VirtualHost>

<VirtualHost *:80>
  ServerName www.example1.de
  DocumentRoot /var/www/www.example1.de/
</VirtualHost>
```

Listing 6: Accessing a Redirected Website

```
HTTP/1.1 200 OK
Date: Sat, 28 Nov 2020 08:25:05 GMT
Server: Apache/2.4.38 (Debian)
Last-Modified: Fri, 27 Nov 2020 17:20:15 GMT
ETag: "85-5b519dfafde34"
Accept-Ranges: bytes
Content-Length: 133
Vary: Accept-Encoding
Content-Type: text/html
```

which you can check with `systemctl` as demonstrated earlier to make sure it is up and running. The default page is at `http://192.168.2.120`, which resides in the `/var/www/html/index.nginx-debian.html` file.

For the two new Nginx web pages, you will again be creating separate directories, storing suitable `index.html` files, and setting appropriate permissions in each case, as demonstrated in the Apache example.

Configuring Nginx

The configuration for the web servers available and enabled on the system is located in the directories shown

Listing 7: Configuration Directories

```
# tree /etc/nginx/sites-*
/etc/nginx/sites-available
--- default
/etc/nginx/sites-enabled
--- default -> /etc/nginx/sites-available/default
```

Listing 8: Minimal Configuration

```
server
{
    listen 80;
    server_name www.example1.de;
    root /var/www/www.example1.de;
    index index.html;
}
```

Listing 9: sites-available/www.example1.de

```
server {
    listen 80;
    server_name example1.de;
    return 301 http://www.example1.de$request_uri;
}

server {
    listen 80;
    server_name www.example1.de;
    root /var/www/www.example1.de;
    index index.html;
}
```

in **Listing 7**. Now you can create a minimal basic configuration for `www.example1.de` in the `/etc/nginx/sites-available/www.example1.de` file (**Listing 8**), and do the same for `www.example2.de`.

Again, you need to fire up both configurations by first manually linking the configuration files:

```
# ln -s /etc/nginx/sites-available/
www.example1.de
/etc/nginx/sites-enabled/
```

Then, tell Nginx to parse the new configuration, making the virtual hosts available:

```
# systemctl reload nginx
```

Again, test the new configuration manually over Telnet as before. To disable a virtual host, just use `rm` to delete the matching link and reparse the configuration. In terms of the number of FQDNs per configuration file, the same applies as for Apache.

Redirects in Nginx

As a simple example of the use of virtual hosts for the same FQDN, you will again be redirecting all `http://example1.de` to `http://www.example1.de` (**Listing 9**). After reloading the configuration, it's time for another test with `curl -I`.

SSL with Let's Encrypt

Let's Encrypt [2] is the easiest way to get SSL working on your web server. Fortunately, the approach is identical for Apache and Nginx, except for one small parameter in the script call.

The installation of Let's Encrypt relies on the Snap package manager, which can be installed with:

```
# apt-get -y install snapd
[... New registration ...]
```

At this point you have to log off and log back on again to make sure that all paths are set correctly; then, set up Let's Encrypt with the commands:

```
# snap install core
# snap install --classic certbot
# ln -s /snap/bin/certbot
/usr/bin/certbot
```

The next command is a call to Certbot with either `--apache` or `--nginx` as parameters. For Apache, that would be:

```
# certbot --apache
```

At this point, Certbot asks for a valid email address and then for the FQDN to be activated. The Certbot script automatically configures the FQDN selected in this process and adjusts the Apache or Nginx configuration accordingly. The charming thing about this process is that the system now takes care of updating the certificates without any further intervention. The new web server can now only be reached at `https://www.example1.de`, and it automatically redirects requests to `http://www.example1.de` to HTTPS with a 301.

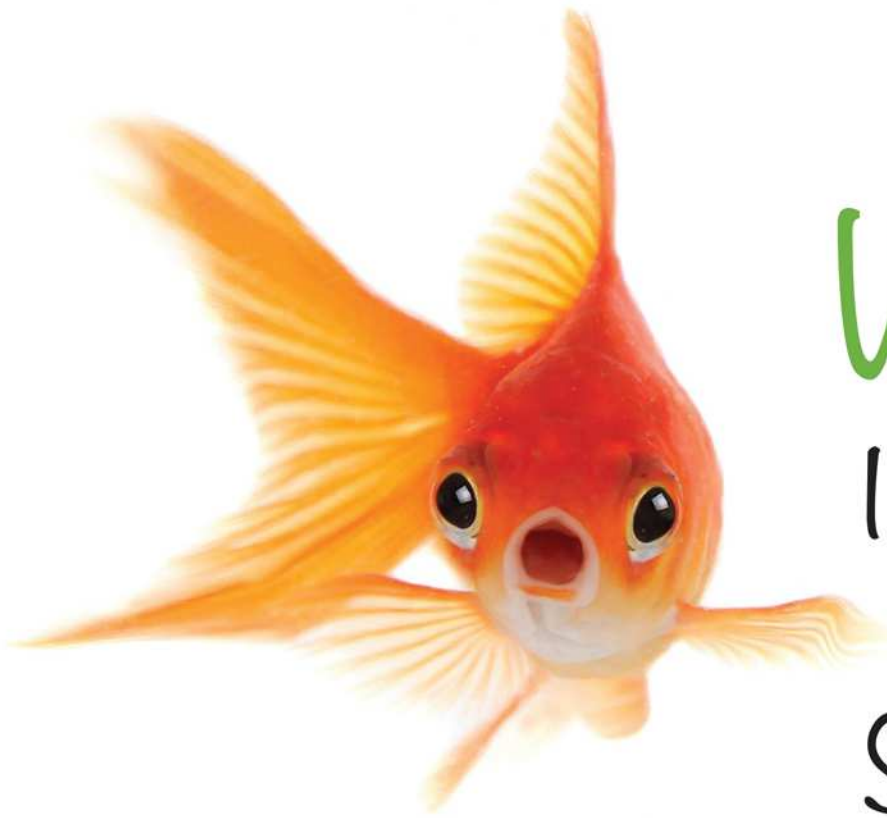
Info

- [1] Web server market shares: [\[https://news.netcraft.com/archives/2021/01/28/january-2021-web-server-survey.html\]](https://news.netcraft.com/archives/2021/01/28/january-2021-web-server-survey.html)
- [2] Let's Encrypt: [\[https://letsencrypt.org\]](https://letsencrypt.org)

The Author

Stefan Wintermeyer ([\[https://www.wintermeyer-consulting.de\]](https://www.wintermeyer-consulting.de)) is a consultant, coach, and book author for Ruby on Rails, Phoenix, and web performance.





What?!

I can get my
issues
SOONER?

ADMIN
Network & Security

Secure Containers
with a hypervisor DMZ

CYBERSECURITY
Machine learning defends the IT infrastructure

4 Password Managers

Safe Containers
Automate updates of container components

MinIO
Local object store with REST interface

Lock Down Microsoft IIS

Secure Containers

NEW features in PHP 8

Apache Kafka
Better stream processing

Teler
Analyze logs and identify suspicious activity in real time

Keycloak
In-house single sign-on server

How to slow down attackers

NEW features in PHP 8

LINUX NEW MEDIA
The Power of Open Source

WWW.ADMIN-MAGAZINE.COM

Available anywhere, anytime!

Sign up for a digital subscription to improve our admin skills with practical articles on network security, cloud computing, DevOps, HPC, storage and more!

Subscribe to the PDF edition: <https://bit.ly/digital-ADMIN>

Now available on ZINIO: bit.ly/ADMIN-ZINIO

Grafana and Prometheus customized dashboards

Stand and Deliver

Grafana analytics and visualization dashboards plus the Prometheus monitoring and alerting tool make possible extensive custom reporting and alerting systems. By Chris Binnie

Visualizing data trends and issues when they arise is clearly of significant benefit, whether for applications or the systems on which they run. Having access to sophisticated tools that can provide highly customizable options

to create bespoke visual dashboards is key and an important facet in being able to maintain high levels of uptime. A welcome bonus is if such tools are compatible with multiple types of data sources, so you can

centralize your single-pane-of-glass viewing platform. Step forward, Grafana [1]. It has the capability to gather metrics from all types of applications and treat each as an individual data source. From there, it is possible to query and filter results precisely to your needs. Coupled with another exceptional bedmate, Prometheus [2], it is possible to incorporate monitoring and alerting into a uber-capable, bespoke visualization toolset. Whereas Grafana is open source visualization software, Prometheus is an open source event monitoring and alerting tool. The two applications do have some cross-over [3], though, and the dashboards from Grafana sit well with Prometheus' sophisticated time series handling [4]. In this article, I look at Grafana and Prometheus and how to get started with dashboarding.

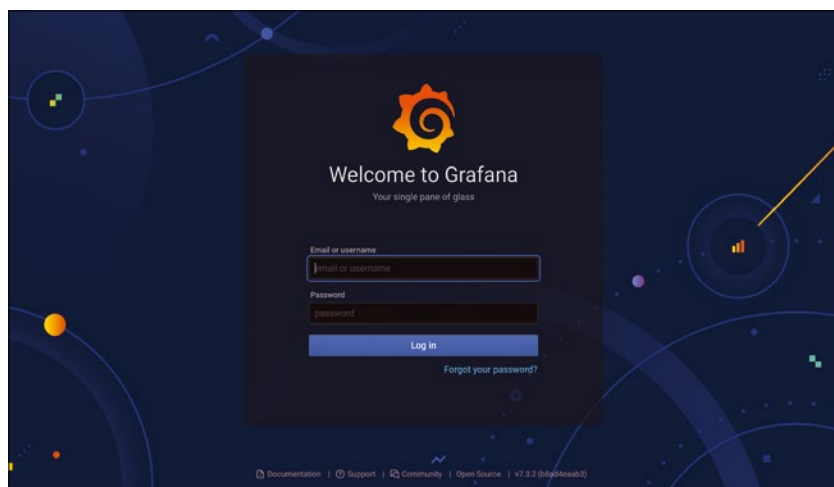


Figure 1: The login page of the open source Grafana user interface.

Listing 1: Check for Running Containers

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
3fa2f72de6bc	grafana/grafana	"/run.sh"	25 seconds ago	Up 24 seconds

Spirographs

For the example in this article, I'm using Linux Mint 20 (Ulyana), which has the long-term support (LTS) version of Ubuntu 20.04 running under

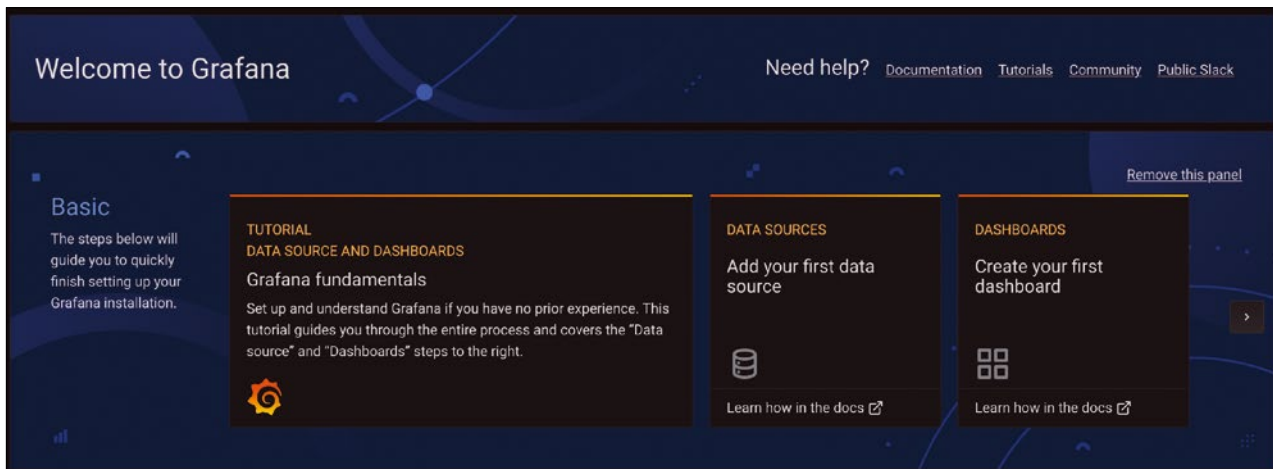


Figure 2: Creating your first dashboard in Grafana.

the bonnet. In true DevOps style, I will not install Grafana locally, because in the past, I have seen some compatibility issues when running with Prometheus if it was also installed locally. Instead, I will use Docker containers for both Grafana and Prometheus. For Grafana, I am using the clear instructions on the Grafana website [5]; remember to get the latest versions and decide whether to use the Alpine or Ubuntu version, depending on your needs. The simple command to fire up the latest version is:

```
$ docker run -d -p 3000:3000 grafana/grafana
3fa2f72de6bc4fd61a0cecd1d18a8bf1aaf7b7
71b49f731fe4ae6eab841d3fd
```

If you want to run a specific version, which some applications might prefer for compatibility, you can use a command that explicitly states a version as the image tag:

```
$ docker run -d -p 3000:3000 --name grafana grafana/grafana:6.5.0
```

After running the non-specific version of that command, you should see the output in Listing 1 (abbreviated here) to see which containers are running.

Goey

If you now check TCP port 3000 in a browser by visiting `http://localhost:3000/login`, you will see

the aesthetically pleasing Grafana interface (Figure 1). The user and password are `admin`, which you are immediately prompted to change at login.

On the right side of the Welcome page, click *Create your first dashboard* (Figure 2). Next, choose the *Add new panel* button to continue (Figure 3). You are then offered the Random Walk dashboard (Figure 4). If you select the small blue question mark icon next to *default* and choose the – *Grafana* – option in the pull-down menu under the *Query* box on the left side, you can read the description for that data source.

Here, you can flick through the different data source types and read the descriptions, which I'll look at in more detail in a moment. The default data source (Grafana Fake Data Datasource

– Native Plugin) demonstrates nicely what you can expect from a simple panel; you should try zooming into it by selecting specific time periods to become familiar with this functionality. Options allow you to adjust your query alongside transformations and alert settings. Under *Transform*, the following narrative is provided: “Transformations allow you to join, calculate, re-order, hide and rename your query results before being visualized.” You can find more information on the Grafana website [6], which explains that “Transformations process the result set of a query before it's passed on for visualization.” Apparently, when you have multiple dashboards with a high volume of queries, “the ability to reuse the query result from one panel in another

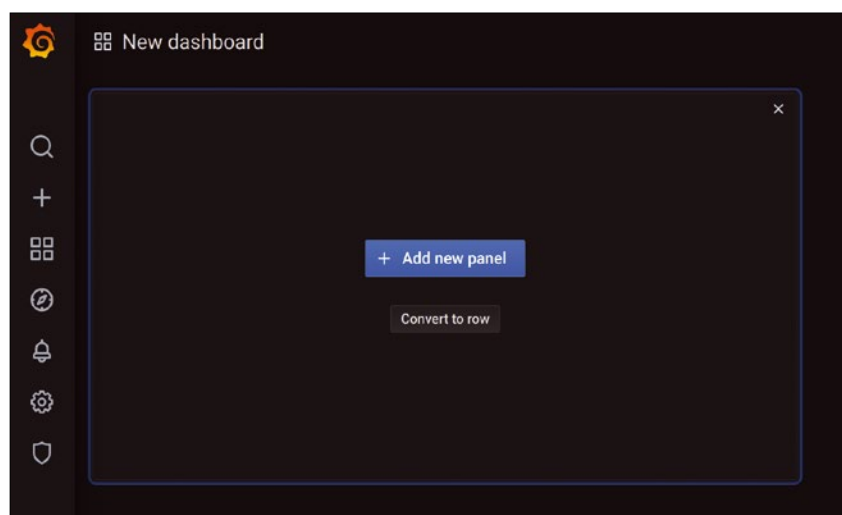


Figure 3: Add a new panel to get started.



Figure 4: Random walk example dashboard.

panel can be a huge performance gain.” As a result, it’s well worth reading up more about transforming your data if you use dashboarding extensively.

If you click the *Save* button at top right and name your dashboard something like *Random Walk*, you can come back to it later. Next, click the Grafana logo at top left and go back to the Welcome screen.

Golden Source

As mentioned, Grafana supports a number of data sources that you might want to explore (Table 1). The

documentation notes that, as well as setting up custom data sources, three other sources are treated a little differently, as you saw when you clicked the small blue question mark icon a moment ago to read their descriptions. First, the – *Grafana* – test source helps you test graphing and dashboards’ visual aspects. The – *Mixed* – data source allows you to query several sources in the same panel, so that every time you create a new query, you can choose a different data source. Finally, the – *Dashboard* – source lets you “use a result set from another panel” within the same dashboard. All in all, it’s possible to use these special data sources

to add extra flexibility to your graphing.

Many applications can push data to Grafana in a sane and compatible format. If you look at the documentation for Grafana plugins [7], you might be pleasantly surprised at the multitudinous array of community and officially provided

plugins that help you get data into Grafana. Of the tens of plugins available, a number of well-known names include the monitoring solution Zabbix, the container orchestrator Kubernetes, the analytics engine Elasticsearch, the code repository GitHub, the databases MySQL and PostgreSQL, and the popular alerting tool Datadog. Incidentally, Grafana helpfully provides a GitHub repository [8] to help you get started creating your own plugins.

God of Fire

The next step is to connect Prometheus to Grafana. Again, I’ll go down the Docker route with the official documentation [9]. Before beginning, create a configuration file and save it locally as `/root/prometheus.yml` (Listing 2). Now you can start up the Prometheus container with:

```
$ docker run \
  -p 9090:9090 \
  -v /root/prometheus.yml:/etc/prometheus/prometheus.yml \
  prom/prometheus
```

Table 1: Supported Data Sources

AWS CloudWatch	Microsoft SQL Server
Azure Monitor	MySQL
Elasticsearch	OpenTSDB
Google Cloud Monitoring	PostgreSQL
Graphite	Prometheus
InfluxDB	Tempo
Jaeger	Testdata
Loki	Zipkin



Figure 5: Happiness is a working Prometheus instance.

The command shows a volume is mounted for a specific file at the local `/root` file path. From that command, you are left with a running Docker container, but because you didn't detach the container (with the `-d` switch), if you close that terminal, it will stop; anything else you do will have to be in a second terminal. The STDOUT information is useful (Listing 3, abridged).

The logging output looks relatively sane, so you should try to connect the two pieces of software. For proof that Prometheus is indeed working, however, point your browser to its web interface at `http://localhost:9090/graph` (Figure 5). If you want, you can run queries in the field above the *Execute* button, called the expression browser. You can find instructions on how this works on the main website [10].

Jigsaw Puzzle

By now, you should have a recognizable data source that Grafana can ingest. Figure 6 shows where to navigate to choose plugins: Click *Plugins* and the cog icon in the left sidebar. However, after typing *Prometheus* into the search box at the top of the page, the entry simply states “Grafana ships with built in

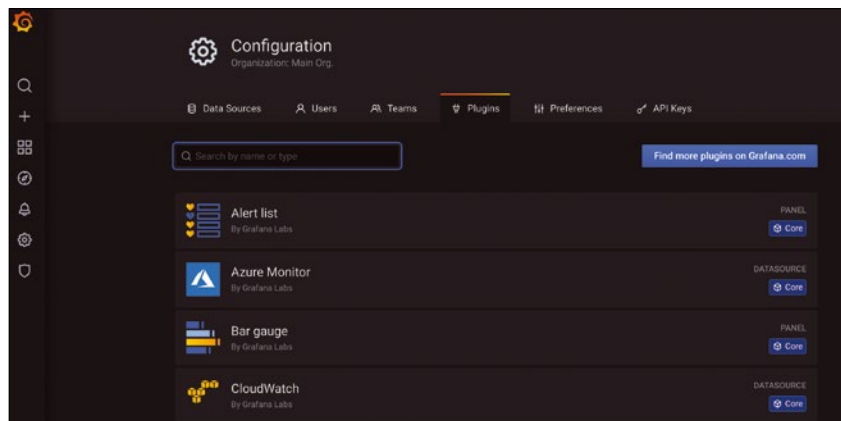


Figure 6: Plugins for data sources.

support for Prometheus, the open-source service monitoring system and time series database” and encourages you to look at this in more detail on the Data Sources page [11]. In other words you are being politely redirected; therefore, you will instead try and use the Prometheus data source directly.

If you go back to the settings cog icon and choose *Data Sources* this time, click *Add Data Source* (Figure 7). After clicking *Prometheus*, assuming the Name box is already populated with the name of

your data source, you can add the URL of your Prometheus server and continue.

To find the correct URL, you should open a new terminal (or the Prometheus container will stop) and run the command:

```
$ docker ps
```

Listing 2: Prometheus Config File

```
global:
  scrape_interval: 15s # Scrape frequency
  evaluation_interval: 15s # Check rules

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['127.0.0.1:9090'] # Open up Prometheus on TCP port 9090
```

Listing 3: Prometheus Container STDOUT

```
level=info ts=2020-11-15T19:11:23.675Z caller=web.go:516 component=web msg="Start listening for connections" address=0.0.0.0:9090
level=info ts=2020-11-15T19:11:23.697Z caller=head.go:642 component=tsdb msg="Replaying on-disk memory mappable chunks if any"
level=info ts=2020-11-15T19:11:23.697Z caller=head.go:656 component=tsdb msg="On-disk memory mappable chunks replay completed" duration=3.766µs
```

Once you have found the hash ID for the Prometheus container on the left side (mine started with *da33*), you can find its IP address with:

```
$ docker inspect da33 | grep IPAddress
"SecondaryIPAddresses": null,
  "IPAddress": "172.17.0.3",
  "IPAddress": "172.17.0.3",
```

Now you can add that IP address to the *URL* field in the Grafana GUI as *http://172.17.0.3:9090*. Click *Save & Test*, and if all is going well, you should see a big green banner that says *Data source is working*. Now you can set up some dashboarding to show off Grafana.

Showing Off

To begin, navigate to the *Dashboards* tab at the top of the screen and click *import* on the right for the three example dashboards provided. If you want to delve deeper, you can find the Prometheus Stats JSON file online [12].

Once the import has completed, the name of the dashboard becomes a link that you can mouse over and select with a click. As if by magic, you have multiple metrics on display (Figure 8).

To speed up the data refresh frequency, mouse over the top right pull-down menu and select *5s* (as opposed

to *1m*). If your graphs are lightly populated, fret not; waiting for a few minutes might resolve the issue. If using a tablet, try the “pinch” trick to focus on a specific time period and dig deeper into a specific display and practice with some of the settings to get used to how Grafana interacts. If you click up near the top left on the name of the dashboard, you will see a list of dashboards you have created. The others might not be set up for you, so don’t despair if they offer empty displays.

The End Is Nigh

Having the ability to customize each and every query along with the way that those queries are formatted and output visually is a powerful addition to any modern server estate. I would recommend having a look at hooking up the Kubernetes plugin and the MySQL plugin and configuring them to suit your needs. In this article, I have barely scratched the surface of either Prometheus or Grafana.

Although I have purposely tried to extol the virtues of Grafana’s visualizations, I have left out the exceptionally sophisticated Prometheus to some extent. I would be remiss not to include at least one query, so

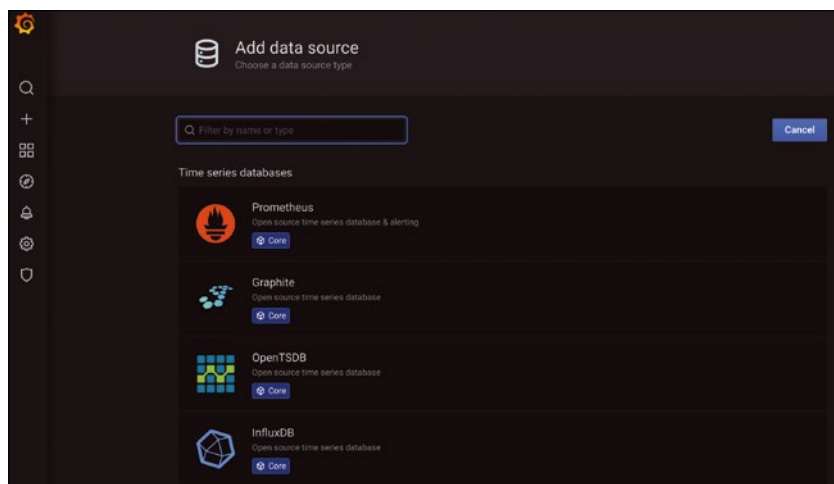


Figure 7: The Prometheus data source.



Figure 8: The Prometheus Stats 2.0 dashboard.

Figure 9 shows, from `http://localhost:9090`, the output from the CPU query:

```
process_cpu_seconds_total
```

The built-in queries available in the pull-down menu next to the blue *Execute* button are numerous and should offer enough interest to whet

your appetite for more. With some effort, you can create a fully bespoke, comprehensive reporting and alerting system with these exceptional tools. I'll leave the rest to you. ■

Info

[1] Grafana: [\[https://grafana.com\]](https://grafana.com)

[2] Prometheus: [\[https://prometheus.io\]](https://prometheus.io)

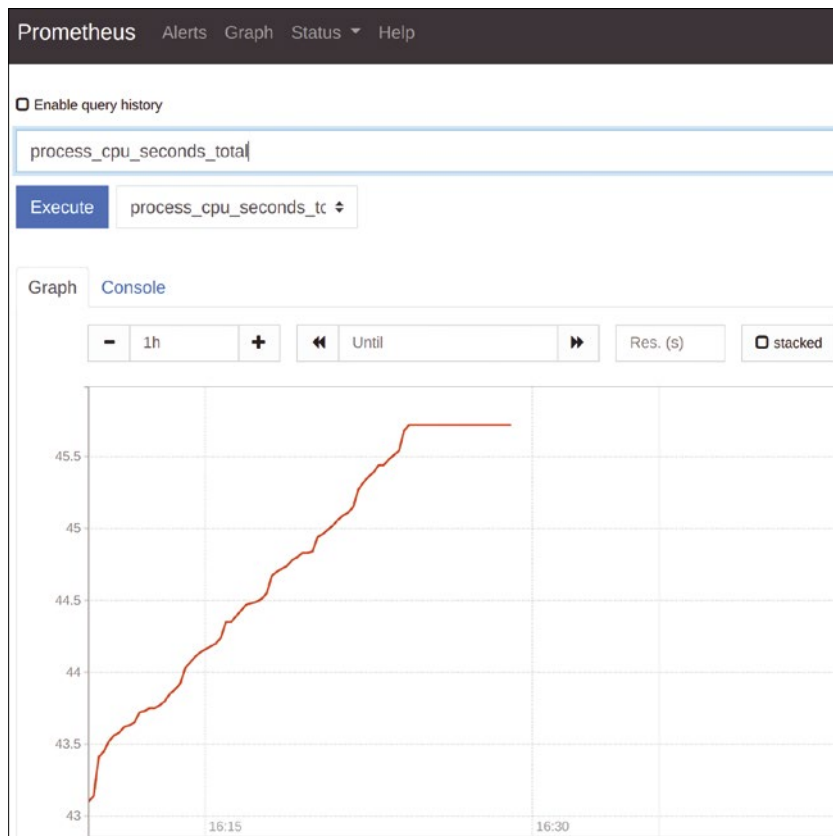


Figure 9: Checking CPU performance.

- [3] Grafana vs. Prometheus: [\[https://wisdomplexus.com/blogs/grafana-vs-prometheus/amp/\]](https://wisdomplexus.com/blogs/grafana-vs-prometheus/amp/)
- [4] Time series: [\[https://en.wikipedia.org/wiki/Time_series\]](https://en.wikipedia.org/wiki/Time_series)
- [5] Grafana Docker image: [\[https://grafana.com/docs/grafana/latest/installation/docker/\]](https://grafana.com/docs/grafana/latest/installation/docker/)
- [6] Transformations: [\[https://grafana.com/docs/grafana/latest/panels/transformations/\]](https://grafana.com/docs/grafana/latest/panels/transformations/)
- [7] Grafana plugins: [\[https://grafana.com/grafana/plugins\]](https://grafana.com/grafana/plugins)
- [8] GitHub repository: [\[https://github.com/grafana/grafana-starter-datasource\]](https://github.com/grafana/grafana-starter-datasource)
- [9] Prometheus on Docker: [\[https://prometheus.io/docs/prometheus/latest/installation/\]](https://prometheus.io/docs/prometheus/latest/installation/)
- [10] Expression browser: [\[https://prometheus.io/docs/prometheus/latest/querying/basics/\]](https://prometheus.io/docs/prometheus/latest/querying/basics/)
- [11] Prometheus data sources: [\[http://docs.grafana.org/datasources/prometheus\]](http://docs.grafana.org/datasources/prometheus)
- [12] Prometheus Stats JSON: [\[https://grafana.com/grafana/dashboards/2\]](https://grafana.com/grafana/dashboards/2)

Author

Chris Binnie's new book, *Cloud Native Security*, teaches you how to minimize attack surfaces across all of the key components used in modern cloud native infrastructure. Learn with hands-on examples about container security, DevSecOps tooling, advanced Kubernetes security, and Cloud Security Posture Management: [\[https://www.cloudnativesecurity.cc\]](https://www.cloudnativesecurity.cc).

REAL SOLUTIONS *for* REAL NETWORKS

ADMIN is your source for technical solutions to real-world problems.

It isn't all Windows anymore - and it isn't all Linux.

A router is more than a router. A storage device is more than a disk. And the potential intruder who is looking for a way around your security system might have some tricks that even you don't know.

Keep your network tuned and ready for the challenges with the one magazine that is all for admins.

Improve your admin skills with practical articles on:

- Security
- Cloud computing
- DevOps
- HPC
- Storage and more!

SUBSCRIBE NOW
SAVE 30%



shop.linuxnewmedia.com



Modern implementation of DHCP with Kea

Address Jockey

The Kea project offers modern underpinnings for Dynamic IP address assignment by DHCP. By Benjamin Pfister

Dynamic IP address assignment

by the Dynamic Host Configuration Protocol (DHCP) usually performs its service reliably. However, some implementations lack modern interfaces to connect them to existing management systems and databases. Some outdated solutions also lack functions for IPv6. The Kea project, launched in 2014, modernizes the DHCP server. Kea was designed as a successor to the Internet Systems Consortium's open source DHCP server (ISC DHCP), which has been widely used on enterprise networks since 1999. Version 1.8.1 [1] of the DHCP server was recently released and is available for the Linux, Unix, and macOS operating systems. The service has a modular structure with, for example, separate daemons for DHCPv4 and DHCPv6, as well as for dynamic DNS registration.

The Kea server configuration relies on a JSON-based data structure. Configuration changes do not require a service restart, with the exception of an interface change, and was one of the key criteria for one of the early adopters of the project: Facebook. According to its own statements, the social media

company used the predecessor, ISC DHCP, for provisioning the operating systems of bare metal servers and out-of-band management systems. However, automated mechanisms were complex to implement with static configuration files and required frequent restarts of services to apply the changes. Additionally, extended availability requirements, as well as the dynamic connection to the internal inventory system as a "single point of truth," were additional arguments for establishing Kea as the replacement.

Highly Available and Expandable

Some optional functions can be reloaded with dynamically loaded "hook modules." Hooks written in C++ extend the range of functions, which makes it possible to keep the core of the service small, yet not limit the possible range of functions. Moreover, the ISC provides a web-based monitoring GUI named Stork [2] that requires agents on the monitored Kea servers. Connecting Kea to SQL databases for DHCP lease data, DHCP reservations, and some

configuration data also appears interesting and supports, for example, unified access from clustered DHCP servers to a lease dataset. A RESTful API offers the ability to connect to your own management systems. High availability (HA) is also feasible. In contrast to the ancient ISC DHCP, Kea also offers HA for IPv4 and IPv6. Redundancy for IPv6 is implemented according to RFC8156. In addition to two active servers, you can set up classic 1:1 redundancy and run multiple backup servers.

In the classic ISC DHCP, dividing address pools was a very flexible process. Kea can only do this in a 50/50 distribution.

Added to all these functions, Kea offers integration with RADIUS servers. For example, if a MAC address is known, the RADIUS server can suggest a specific IPv4 or IPv6 address and thus offer a quasi-RADIUS-based DHCP reservation. Alternatively, the RADIUS server can also name a specific address pool, which can be useful if dedicated sets of rules depend on certain IP addresses at other points, such as firewalls.

Lead Image © Cheryl Ann Quigley, Fotolia.com

Installation with Dependencies

Before you install, you need to answer two crucial questions that need to be considered later in the build process: (1) Is redundancy intended? (2) Do you want to use a comma-separated values (CSV) file for the lease data or a database?

After downloading Kea, you need to compile it yourself. However, pre-built packages can be downloaded from the Cloudsmith website [3]. Some Linux distributions offer such packages through their package managers, but the versions usually are not up to date. Starting in Kea 1.6.0, you can generate a build with a Python 3 script called `Hammer.py`, which is included in the Git repository.

In the example here, you will first install some dependencies, including a MySQL server that uses `memfile` (the simplest and fastest database back end, which uses an in-memory database and stores DHCP lease data on disk in a CSV file). MySQL, PostgreSQL, and Cassandra are supported as alternative back ends. Afterward, you will be cloning the current GitLab repository and executing the build process (Listing 1). The binaries are then stored under `/usr/local/sbin`.

Preparing Logging and the Database

To define globally which modules need to be loaded by Kea and which paths will be used to access the configuration files, edit the `keactrl.conf` configuration file, where you can disable the dynamic DNS service (e.g., if you do not need dynamic DNS updating on the network). You can start and stop the service and query the current status with:

```
keactrl start
keactrl stop
keactrl status
```

In the respective module configuration, you can define a logging target. For DHCPv4 this would be `kea-dhcp4.conf` if you do not use a different configuration file as an argument.

Now you need to initialize the MySQL database before using it the first time. The `kea-admin` binary provides the `db-init` argument for this purpose. I will not go into detail on the steps for granting the MySQL user permissions here. Listing 2 initializes the MySQL database and shows the created table structure.

Creating the Initial Configuration

Among other things, the `/usr/local/etc/kea` directory contains the configuration files of the modules for DHCPv4 and DHCPv6, the dynamic DNS registry, and the parameterization of the `keactrl` script.

To customize the configuration, use either the configuration files or the RESTful API. The associated daemon is named Kea Control Agent, which also lets you connect to your own management systems. However, it does not provide encrypted access natively. You would have to connect a reverse proxy such as Apache or Nginx to enable encrypted management access – which is of course recommended.

After the initial configuration, start the DHCPv4 service with

```
keactrl start -s dhcp4
```

and verify the status by typing `keactrl status`.

Static Config vs. Database

To begin, start with the static DHCPv4 configuration in a configuration file (Listing 3). Note that these sample configurations are not intended for production and partly include IP addresses from reserved address ranges specifically for documenting the process.

A DHCPv4 configuration file initially starts and ends with the outer element `"Dhcp4": { }`. The individual parameters must be comma-separated cleanly to create a valid JSON file. This example first defines some timers, including timers for the maximum validity of a DHCP lease and for the extension of a

Listing 1: Build Process

```
sudo apt-get -y install liblog4cplus-dev libboost-all-dev
make build-essential autoconf libtool libssl-dev
mysql-server libmysqlclient-dev
cd /opt/
sudo git clone https://gitlab.isc.org/isc-projects/kea.git
cd kea
sudo autoreconf -install
sudo ./configure --with-mysql
sudo make
sudo make install
```

Listing 2: MySQL Table Structure

```
pfister@dhcp:~$ kea-admin db-init mysql -u dhcpsvc -p
t0ps3cr3t -n dhcp
mysql> show tables;
+-----+
| Tables_in_dhcp |
+-----+
| dhcp4_audit |
| dhcp4_audit_revision |
| dhcp4_global_parameter |
| dhcp4_global_parameter_server |
| dhcp4_option_def |
| dhcp4_option_def_server |
| dhcp4_options |
| dhcp4_options_server |
| dhcp4_pool |
| dhcp4_server |
| dhcp4_shared_network |
| dhcp4_shared_network_server |
| dhcp4_subnet |
| dhcp4_subnet_server |
| dhcp6_audit |
| dhcp6_audit_revision |
| dhcp6_global_parameter |
| dhcp6_global_parameter_server |
| dhcp6_option_def |
| dhcp6_option_def_server |
| dhcp6_options |
| dhcp6_options_server |
| dhcp6_pd_pool |
| dhcp6_pool |
| dhcp6_server |
| dhcp6_shared_network |
| dhcp6_shared_network_server |
| dhcp6_subnet |
| dhcp6_subnet_server |
| dhcp_option_scope |
| host_identifier_type |
| hosts |
| ipv6_reservations |
| lease4 |
| lease4_stat |
| lease6 |
| lease6_stat |
| lease6_types |
| lease_hwaddr_source |
| lease_state |
| logs |
| modification |
| parameter_data_type |
| schema_version |
+-----+
```

Listing 3: DHCPv4 Config File for Memfile

```
{
  "Dhcp4": {
    "renew-timer": 900,
    "rebind-timer": 1800,
    "valid-lifetime": 3600,
    "interfaces-config": {
      "interfaces": ["ens33"],
      "dhcp-socket-type": "raw"
    },
    "lease-database": {
      "type": "memfile",
      "persist": true,
      "lfc-interval": 3600
    },
    "option-data": [
      {
        "name": "domain-name-servers",
        "data": "192.0.2.1"
      },
      {
        "code": 15,
        "data": "lab.pfisterit.de"
      },
      {
        "name": "domain-search",
        "data": "lab.pfisterit.de"
      }
    ],
    "subnet4": [
      {
        "subnet": "192.0.2.0/24",
        "pools": [
          {
            "pool": "192.0.2.20 -- 192.0.2.199"
          }
        ],
        "option-data": [
          {
            "name": "routers",
            "data": "192.0.2.1"
          }
        ],
        "reservations": [
          {
            "hw-address":
              "1a:1b:1c:1d:1e:1f",
            "ip-address": "192.0.2.201"
          }
        ],
        "loggers": [ {
          "name": "kea-dhcp4",
          "output_options": [ {
            "output": "/tmp/kea-dhcp4.log"
          } ],
          "severity": "WARNINGS",
          "debuglevel": 0
        } ]
      }
    ]
  }
}
```

lease with the previous DHCP server or with other DHCP servers. After that, it defines the interface on which a service binding will be made and determines that it accepts raw requests with `dhcp-socket-type`.

Alternatively, to accept DHCP requests by a DHCP relay on the DHCP server, you could parameterize the socket type as `udp`. Next, the file configures the leases database to be a `memfile` type and stores it persistently in the `/var/lib/kea/dhcp4.leases` directory. Next are the global DHCP options. The example defines the DNS server of the organization, the domain name, and the search domain.

Additionally, it creates a DHCP scope with the option to assign the default gateway; that is, it has set up a DHCP reservation for the MAC address `1a:1b:1c:1d:1e:1f` and the IP address

`192.0.2.201`. The path `/usr/local/var/log/kea-dhcp4.log` is used as the logging destination. The corresponding log level is set to `WARNINGS`.

Figure 1 shows a DHCPv4 process in Wireshark as an example. The global DHCP option for the *lab.pfisterit.de* DNS domain was passed to the client in the offer from the DHCP server residing on the IPv4 address `192.168.178.194`. In contrast to the example in **Listing 3**, with a `memfile` lease database, the next example connects to a MySQL database by storing the corresponding connection parameters such as the TCP port, database name, and corresponding credentials in the configuration snippet (**Listing 4**).

DHCPv6 and Data Models

The example in **Listing 5** uses a dedicated configuration file for IPv6. The

Listing 4: IPv4 Config File for MySQL

```
{
  "Dhcp4": {
    "renew-timer": 900,
    "rebind-timer": 1800,
    "valid-lifetime": 3600,
    "interfaces-config": {
      "interfaces": ["ens33"],
      "dhcp-socket-type": "raw"
    },
    "lease-database": {
      "type": "mysql",
      "name": "dhcp",
      "host": "",
      "connect-timeout": 5,
      "max-reconnect-tries": 5,
      "reconnect-wait-time": 3000,
      "user": "dhcpsvc",
      "password": "t0ps3cr3t",
      "port": 3306,
      "persist": true,
      "lfc-interval": 3600
    },
    "option-data": [
      {
        "name": "domain-name-servers",
        "data": "192.0.2.1"
      },
      {
        "code": 15,
        "data": "lab.pfisterit.de"
      },
      {
        "name": "domain-search",
        "data": "lab.pfisterit.de"
      }
    ],
    "subnet4": [
      {
        "subnet": "192.168.178.0/24",
        "pools": [
          {
            "pool": "192.168.178.20 -- 192.168.178.199"
          }
        ],
        "option-data": [
          {
            "name": "routers",
            "data": "192.168.178.1"
          }
        ],
        "reservations": [
          {
            "hw-address":
              "1a:1b:1c:1d:1e:1f",
            "ip-address": "192.168.178.201"
          }
        ],
        "loggers": [ {
          "name": "kea-dhcp4",
          "output_options": [ {
            "output": "/tmp/kea-dhcp4.log"
          } ],
          "severity": "WARNINGS",
          "debuglevel": 0
        } ]
      }
    ]
  }
}
```

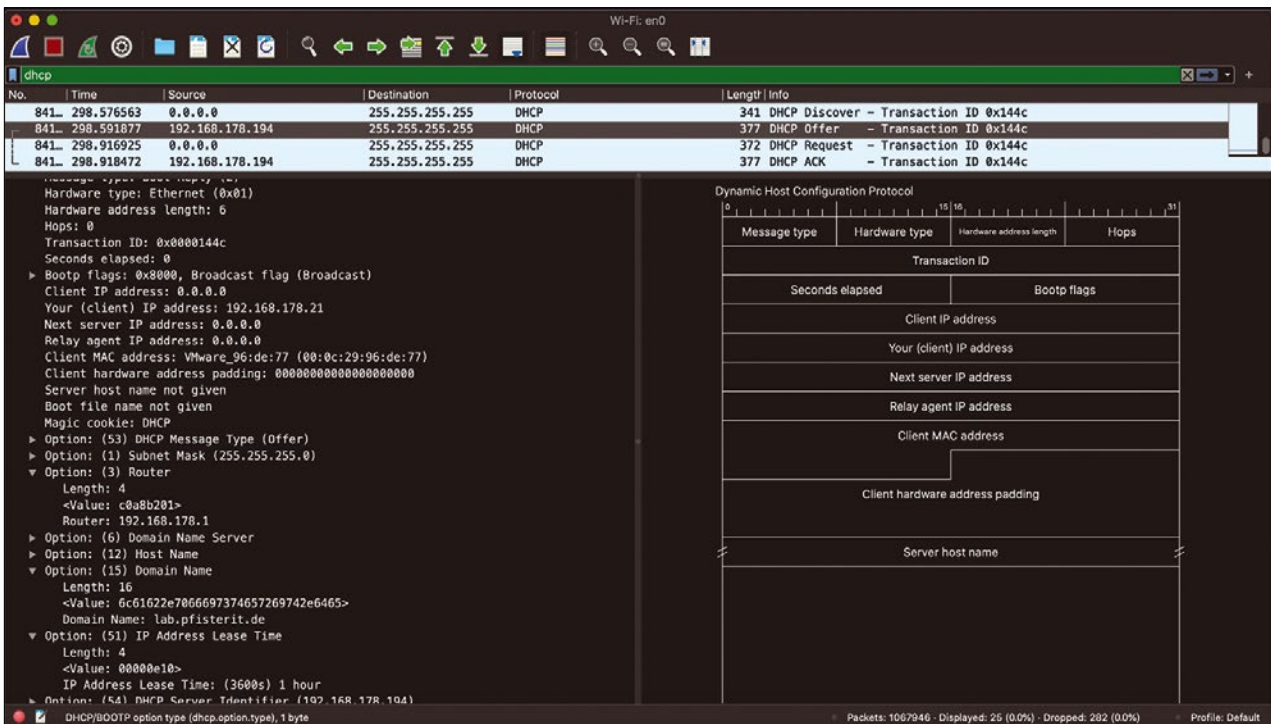


Figure 1: Example of a packet capture of a DHCP offer. The global option of the domain name was adopted.

matching timers and interface bindings are already known from the IPv4 configuration. The central differences are recognizable in this very simple JSON example in the form of specifications for Dhcp6 instead of Dhcp4, as well as subnet6 instead of subnet4 for the corresponding scopes.

As the level of automation in the network environment increases, the YANG (Yet Another Next Generation) data model has become increasingly popular. A unified data model is particularly recommended on heterogeneous networks. Kea also offers YANG support based on the open source Sysrepo engine. To work with this data model, the NETOPEER2 NETCONF server, which is under BSD license, offers an interface for configuration adjustments.

Migration Path from ISC DHCP

Anyone who uses an existing ISC DHCP server and finds the new features interesting will now be asking themselves how they can migrate their existing configuration to Kea. A migration wizard known as KeaMA [4] is available for this purpose.

The wizard uses an existing ISC configuration file and translates it into a Kea-compatible configuration file. You need to run the process twice: once for IPv4 and once for IPv6 if you use both protocols. If you encounter any errors during the conversion, the migration wizard will offer appropriate tips. However, the tool is still considered experimental according to the GitLab page.

Conclusions

Kea offers interesting approaches and functions. However, switching from an existing ISC DHCP server to Kea needs to be well thought out and planned. If functions such as high availability for IPv6 are not used on your network and you do not need APIs for management systems, it is questionable whether a switch to Kea will offer any advantages. One argument, of course, is that switching removes the need to restart for scope changes. For data centers with a large number of configuration changes within short time intervals, Kea could be interesting. The documentation is also very useful.

Info

- [1] Kea download: <https://www.isc.org/download/>
- [2] Kea dashboard Stork: <https://gitlab.isc.org/isc-projects/stork>
- [3] Precompiled Kea packages: <https://cloudsmith.io/~isc/repos/>
- [4] KeaMA: <https://gitlab.isc.org/isc-projects/dhcp/tree/master/keama>

Listing 5: IPv6 Sample Config

```
{
  "Dhcp6": {
    "renew-timer": 900,
    "rebind-timer": 1800,
    "valid-lifetime": 3600,
    "interfaces-config": {
      "interfaces": ["ens33"],
    },
    "lease-database": {
      "type": "memfile",
      "persist": true,
      "name": "/var/lib/kea/dhcp6.leases"
    },
    "subnet6": [
      {
        "subnet": "2001:db8:1::/64",
        "pools": [
          {
            "pool": "2001:db8:1::5 - 2001:db8:1::ffff"
          }
        ]
      }
    ]
  }
}
```

The background of the entire page is a photograph of a modern, multi-level library. The library features white bookshelves filled with books, a wide staircase with a white railing, and a person sitting on a bench. The image is overlaid with a blue gradient that curves across the top and bottom, framing the text.

Hone your skills with special editions!

Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

Check out the full library!

shop.linuxnewmedia.com

FREE DVD!
Ubuntu 15.2

JOIN THE LINUX REVOLUTION!
← ALL THE SOFTWARE YOU NEED!

GETTING STARTED WITH LINUX

• MORE POWERFUL • MORE SECURE • MORE FUN

LEARN HOW TO SET UP A LINUX SYSTEM TO:

- Listen to Music • Play Games • Process Photos
- Surf the Web • and Much More!

2020 Edition

WWW.LINUX-MAGAZINE.COM

300+ BEST BASH COMMANDS **SAVE 15%**
on Linux Certification
See details inside

LINUX SHELL

HANDBOOK 2019 Edition

SUPERCARGE YOUR LINUX SKILLS

Travel Light
with fast and graceful keyboard commands

Power at Your Fingertips

- Manipulate text strings
- Pipe and redirect output
- Monitor processes
- Manage users and groups
- Create easy automation scripts

Keep this comprehensive guide as a permanent command reference

WWW.LINUX-MAGAZINE.COM

FREE DVD!
LibreOffice Full Version

Become a LibreOffice Expert!
2020 Edition

LibreOffice

Dive deep into the world's greatest free office suite

Write Your Own LO Macros
Save time and automate common tasks

Digital Signatures
Lock down your private documents

Edit and Save MS Office Files

Create Professional:

- Text Documents
- Spreadsheets
- Presentations
- Databases

Replace MS Office and Google Docs!

LibreOffice*
Includes full versions for Windows, macOS, and Linux

WWW.LINUX-MAGAZINE.COM

LINUX SPECIAL **101 COOL LINUX HACKS** **2020 EDITION**

101 COOL LINUX HACKS

Inspirational tricks and shortcuts for Linux geeks

- Repair your bootloader
- Cure the Caps Lock disease
- Tricks with terminal output
- Disable your webcam and mic
- Run C one-liners in the shell
- Undelete lost files
- Ignore case in file names

PHONING IN
Sync your phone with a Linux desktop

QUICK SWITCH
Change to a second distro using chroot

LINUX SPECIAL

WWW.LINUX-MAGAZINE.COM

Software instead of appliances: load balancers compared

Balancing Act



We introduce the most important software load balancers, look at their strengths and weaknesses, and provide recommendations for use scenarios. By Martin Loschwitz

For a long time now, load balancers have supported any number of back-end servers. If the load in the setup becomes too high, you just add more servers. In this way, almost any resource bottleneck can be prevented. Understandably, the load balancer is a very important part of almost every web setup – if it doesn't work, the website is down. Appliances from F5, Citrix, Radware, and others were considered the quasi-standard in the load balancer environment for a long time, although powerful software solutions that ran as services on normal Linux servers (e.g., HAProxy or Nginx) were available.

The cloud-native trend is a major factor accelerating the shift toward software-defined load balancing these days, not least because typical appliances are difficult to integrate into cloud-native applications – reason enough for a brief market overview. What tools for software-based load balancing are available to Linux admins? What are their specific strengths and weaknesses? Which software is best suited for which scenario?

Before I get started, however, note that not all of the load balancers presented here offer the same features.

OSI Rears Its Head

As with hardware load balancers, different balancing techniques exist for their software counterparts. In a nutshell, many load balancers primarily differ in terms of the Open Systems Interconnection (OSI) layer on which they operate, and some load balancers offer support for multiple OSI layers. What sounds dull in theory has significant implications in practice. Most software load balancers for Linux operate on OSI Layer 4, which is the transport layer. They field incoming connections by certain protocols – usually TCP/IP – on one side and route them to their back ends on the other. Load balancing at this layer is agnostic with respect to the protocol being used. Therefore, a Layer 4 load balancer can distribute HTTP connections just as easily as MySQL connections, or indeed any other protocol. Conversely, Layer 4 also means that protocol-specific options are not

available for balancing. For classical load balancing on web servers, for example, many applications expect their session to be sticky, which means that the same clients always end up on the same web server for multiple successive requests according to various parameters (e.g., session IDs). However, the web server can only guarantee this stickiness if it not only forwards packets at the protocol level, but also analyzes and interprets the data flow. Many load balancers support this behavior, but it is then referred to as Layer 7 (i.e., the application layer) load balancing. For the choice of load balancer to be used, the question of support for Layer 7 load balancing is of great relevance. Some balancers in this comparison explicitly support Layer 4 only; others also offer protocol support. In this article, I discuss the OSI layers supported by the solutions presented individually for every solution tested.

High Availability

When admins are considering software load balancers, they need to

Photo by Patrick Fore on Unsplash

keep the load balancer itself in mind as a potential source of problems, rather than just considering the website's functionality. Appliance vendors draw on this as a welcome opportunity to cash in once again. Load balancer appliances from all manufacturers can be operated in high-availability (HA) setups, but you need at least two of the devices. That the firmware has this functionality doesn't make any difference. In addition to the second device, a license extension is usually required to install the two balancers in an HA cluster. With self-built load balancers, admins resolve the HA issue themselves by designing for hardware redundancy. However, now that the infamous half-height pizza boxes come with 256GB of RAM and multicore CPUs, the financial outlay is manageable. Moreover, the software load balancers presented here are not particularly greedy when it comes to resources. The software side is a bit more uncomfortable: IP addresses that switch in combination with certain services can be implemented in Linux as a typical HA cluster with Pacemaker.

Classic Choice: HAProxy

The first test candidate in the comparison is a true veteran that most admins will have encountered. HAProxy [1] is one of the oldest solutions in the test field; the first version saw the light of day at the end of 2001. With regard to the OSI model, HAProxy is always based on Layer 4; it implements balancing for any TCP/IP connection. HAProxy is therefore always suitable as a load balancer for MySQL or other databases. Complementary Layer 7 functionality can be used, but only for HTTP – other protocols are excluded.

Because HAProxy has been on the scene for such a long time, it offers several features that often are not needed in many use cases. However, HAProxy has no weaknesses in terms of basic functions. It supports HTTP/2 as well as compressed connections with Gzip, HTTP URL rewrites, and rate limiting to counter distributed denial of service

(DDoS) attacks. Because the program has been capable of multithreading for several years, it benefits from modern multicore CPUs, especially those that also support hyperthreading. In any case, a reasonably up-to-date multicore processor is not likely to faze HAProxy. Secure sockets layer (SSL) and load balancing is a complex issue, because a middleman like HAProxy and SSL encryption are mutually exclusive. Therefore, an established practice is to let the load balancer handle SSL termination instead of the HTTP server. In such cases, of course, the admin can additionally set up SSL encryption between the individual web servers and the load balancer. HAProxy can handle both: The service terminates SSL connections on demand and also talks to the back ends in encrypted form (e.g., when checking their states). HAProxy has extensive options for checking the functionality of its own back ends. If a back end goes offline or the web server running there does not deliver the desired results, HAProxy automatically removes it from the rotation setup.

All told, HAProxy is a reliable load balancer with a huge range of functions, which, however, makes the program's configuration file difficult to understand. That said, any admin in a state-of-the-art data center will probably generate the configuration from the automation setup anyway, so complexity is not a difficulty. HAProxy connections are available for all common automation systems,

either from the vendor or from third-party providers. I also need to mention the commercial version of HAProxy: In addition to support, it comes with a few plugins for additional functions, such as a single sign-on (SSO) module and a real-time dashboard in which various HAProxy metrics can be displayed.

Speaking of metrics data, integrating HAProxy into conventional and state-of-the-art monitoring systems is very simple thanks to previous work by many users and developers. The tool comes with its own status page, which provides a basic overview (Figure 1). All told, HAProxy can be considered a safe choice when it comes to load balancers.

Unexpected: Nginx

Most admins will not have the second candidate in this comparison on their score cards as a potential load balancer, although they know the program well: Nginx [2] is used in many corporate environments, usually as a high-performance web server. However, Nginx comes with a module named Upstream that turns the web server into a load balancer. Keeping to what it does best, Nginx restricts itself to the HTTP protocol on Layer 7 of the OSI model.

Although Nginx is primarily a web server and not a load balancer, it can compete with HAProxy in terms of features. The program's modular structure is an advantage: Nginx can

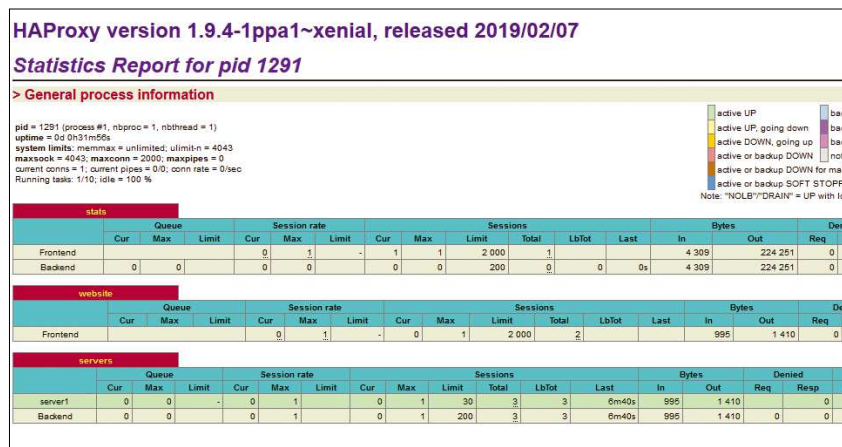


Figure 1: HAProxy offers a status page with basic details on the current usage of the service. © HAProxy

handle SSL and the more modern HTTP/2 protocol. Compressed connections and metrics data in real time are also part of the feature set. Thanks to the Upstream module ([Figure 2](#)), admins can then simply define another hop in the route taken by incoming packets. If SSL is used, Nginx handles SSL termination without a problem. Like HAProxy, Nginx supports several operating modes when it comes to forwarding clients to the back-end servers. In addition to the typical *Round Robin* mode, *Least Connections* is available, which selects the back end with the lowest load. The *IP Hash* algorithm calculates the target back end according to a hash value that it obtains from the client's IP address.

A commercial version of Nginx, Nginx Plus, offers a *Least Time* mode, as well, wherein the back end that currently has the lowest latency from Nginx's viewpoint is always given the connection. However, session persistence is also supported by the normal Nginx (minus Plus). Whether the IP hashing algorithm or a random mode with persistent sessions is the better option depends in part on the setup and can ultimately only be found out through trial and error.

Nginx admins also can solve the HA problem described at the beginning in a very elegant way by buying the product's commercial Nginx

Plus distribution, because it has a built-in HA mode and can be run in active-passive mode or even active-active mode. Potentially unwelcome tools like Pacemaker can be left out in this scenario.

Robust Solution: Seesaw

Not surprisingly, Google, one of the most active companies in the IT environment, has its own opinion on the subject of load balancing. The company is heavily involved in the development of a load balancer, even if it is not officially a Google product. The name of this program is Seesaw [\[3\]](#), which describes the core aspect of a load balancer quite well.

Seesaw is based on the Linux virtual server (LVS) functions and therefore tends to follow Keepalived in terms of functionality. It is aimed exclusively at users of OSI Layer 4, which makes it explicitly the first load balancer in the test that does not offer special functions for HTTP. However, the World Wide Web protocol was probably not the focus of the Google developers when they started working on Seesaw a few years ago, which is clear from the basic operating concept of the solution.

Hacks that use Pacemaker and the like to achieve high availability do not exist in Seesaw. Instead, it always needs to be operated as a cluster of at least

two instances that communicate with each other. In this way, Seesaw practically avoids the HA problem. On the Seesaw level, the admin configures the virtual IPs (VIPs) to be published to the outside world, as well as the IPs of the internal target systems.

Because Seesaw supports Anycast and load balancing for Anycast, the setup can become a little more complex. If you bind the Quagga Border Gateway Protocol (BGP) daemon to Seesaw, it will announce Anycast VIPs as soon as you enable them on one of the Seesaw systems. Anycast load balancing is something that Nginx and HAProxy only support to a limited extent, so this is where Seesaw provides innovative features.

In return, however, Seesaw has strict requirements: at least two network ports per machine on the same Layer 2 network. One is used for the system's own IP, and the others announce the VIPs for running services. However, the developers note that Seesaw can easily be run as a group of multiple virtual machines, which takes some of the stress out of the setup.

Written in Go, Seesaw ideally should not cause too much grief after the initial setup. The clearly laid out configuration file takes the INI format and usually comprises no more than 20 lines, although it only plays a minor role in the Seesaw context anyway.

It was important to Google that the service could be managed easily from a central point. Seesaw therefore also includes a config server service, where the admin dynamically configures the back-end servers and then passes the information on to Seesaw ([Figure 3](#)). This setup might sound a bit like overkill at first, and it only pays off to a limited extent if you are running a single load balancer pair. However, if you think in Google terms and are confronted with quite a few Seesaw instances, you will quickly understand the elegance behind this approach, which is all the more true because Seesaw tries to keep its configuration simple and does not try to replicate one of the common automators. If you are looking for a versatile load balancer for OSI Layer 4, you



```

1 listen 443 ssl;
2 ssl on;
3 ssl_certificate /etc/nginx/ssl/example.com/server.crt;
4 ssl_certificate_key /etc/nginx/ssl/example.com/server.key;
5 ssl_trusted_certificate /etc/nginx/ssl/example.com/ca-certs.pem;
6
7 upstream mywebapp1 {
8     server 10.42.0.2;
9     server 10.42.0.3;
10 }
11
12 server {
13     listen 80;
14     listen 443 ssl;
15     server_name example.com www.example.com;
16
17     ssl on;
18     ssl_certificate /etc/nginx/ssl/example.com/server.crt;
19     ssl_certificate_key /etc/nginx/ssl/example.com/server.key;
20     ssl_trusted_certificate /etc/nginx/ssl/example.com/ca-certs.pem;
21
22     location / {
23         proxy_pass http://mywebapp1;
24         proxy_set_header Host $host;
25         proxy_set_header X-Real-IP $remote_addr;
26         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
27         proxy_set_header X-Forwarded-Proto $scheme;
28     }
29 }

```

Figure 2: Even though most admins will tend to think of Nginx as a web server rather than a load balancer, the program can do both; the configuration is straightforward.

```

1 [cluster]
2 anycast_enabled = false
3 name = au-syd
4 node_ipv4 = 192.168.
5 node_ipv6 = 2015:cafe::2
6 peer_ipv4 = 192.168.10.3
7 peer_ipv6 = 2015:cafe::3
8 vip_ipv4 = 192.168.10.1
9 vip_ipv6 = 2015:cafe::1
10
11 [config_server]
12 primary = seesaw-config1.example.com
13 secondary = seesaw-config2.example.com
14 tertiary = seesaw-config3.example.com
15
16 [interface]
17 node = eth0
18 lb = eth1

```

Figure 3: Seesaw is very easy to configure because it takes its main configuration from its own service instead of from a file.

might want to take a closer look at Seesaw – it's worth it.

Commercial: Zevenet

The last candidate in the comparison is Zevenet [4], which proves that load balancers can be of commercial origin without being appliances. If you want, you can also get the service with hardware from Zevenet, but it is not mandatory. Zevenet also describes itself as cloud ready, and the manufacturer even delivers its product as a ready-to-use appliance for operation in various clouds. In terms of content, Zevenet is not that spectacular. It is a standard load balancer, with community and professional versions available. The community version has basic balancing support for OSI Layers 4 and 7; the focus in Layer 7 is on HTTP. However, the strategy behind this is clearly do-it-yourself; support from the manufacturer is only available in the form of best-effort requests in their forum. The Enterprise Edition from Zevenet is far more fun. Like Nginx and Seesaw, it offers built-in high availability for cluster operation, delivers its own monitoring messages over SNMP, and offers its own API for access. Unlike the tools presented so far, Zevenet Enterprise also has an extended GUI that admins can use to configure Zevenet.

A basic version of the GUI is also available for the Community Edition (Figure 4), but the Enterprise variant offers significantly more functions (Figure 5). What may be a thorn in the side of experienced admins can save the day in meetings with management. Zevenet can be connected to an external user administration tool in the GUI or, alternatively, provide its own with an audit trail. The ability to change the load balancer configuration can therefore be defined in Zevenet at the service level. The Layer 4 and 7 capabilities of the Community version are significantly expanded in the Enterprise environment. For example, different log types can be monitored and logged sepa-

ately, with the ability to specify the destination for the log messages. On OSI Layer 7 for HTTP, Zevenet offers features such as support for SSL wildcard certificates, cookie injection, and support for OpenSSL 1.1.

Not all of these features are equally relevant for all use cases, but all told, Zevenet's range of functions is impressive. Attention to detail can be seen in many places. For example, Zevenet transfers connections from one node of a cluster to another without losing state. Zevenet takes security seriously by integrating various external domain name system blacklist list (DNSBL) or real-time blackhole list (RBL) services, as well as services for DDoS prevention and the ability to deny access to arbitrary clients when needed. Zevenet already leaves the spheres of load balancing and mutates into something more like a small firewall.

Load Balancing for Containers

In common among all the load balancers presented so far is that admins often run them on separate hardware and integrate them into an HA setup, which is of great relevance for classic applications, conventional web server environments, and other standard software. However, it does not apply to cloud-native computing, because in this case, everything is software-defined to the extent possible and is managed by the users themselves. The centralized load balancer that handles forwarding for a

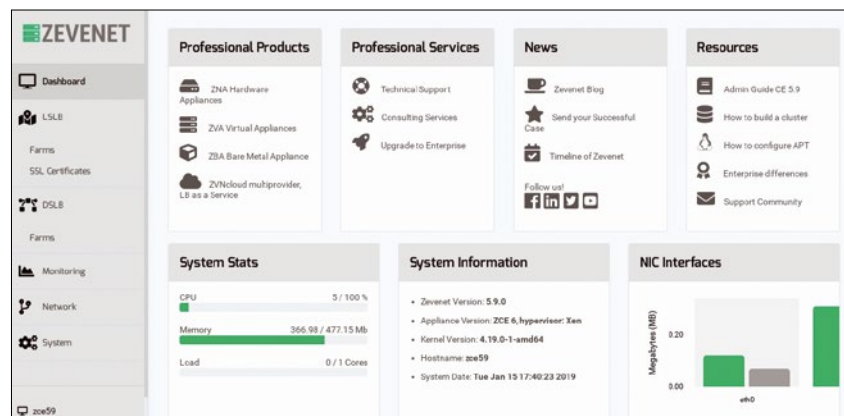


Figure 4: Zevenet not only offers API-based access to the configuration, but also comes with a GUI. © Zevenet

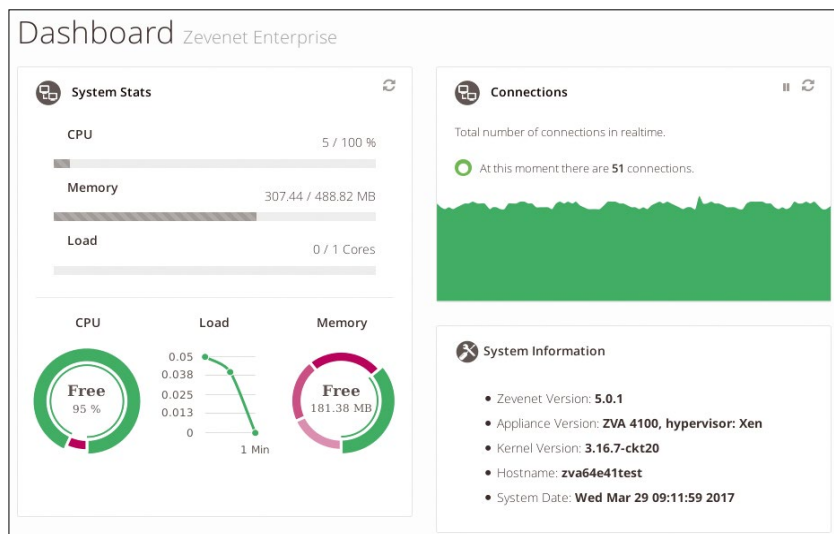


Figure 5: The Enterprise GUI in Zevenet offers various features missing in the Community variant. © Zevenet

website or several installations does not exist in this scenario.

Instead, the admin has Kubernetes (or one of the distributions based on it), which rolls out the app in pods that the respective container orchestrator distributes arbitrarily across physical systems. The network is purely virtual and cannot be tied to a specific host, so the typical load balancer architecture comes to nothing. If a setup requires load balancing (e.g., to allow various microcomponents of an application to communicate with each other in a distributed manner), the legacy approach fails.

Don't Do It Yourself

As a developer or administrator, you could create a container with a load balancer yourself, which you then roll out as part of the pod design. Technically, there is nothing to prevent an appropriately preconfigured Nginx from acting reliably as a balancer. However, setups that run in container orchestrations are usually very dynamic. The number of poten-

tial source and target servers on both sides changes regularly, for which classic software load balancers do not cope well.

Sidecar Solution

Fortunately, developers have already come up with a solution to this problem: software-based load balancers designed specifically for such container architectures. Istio [5] is something of a shooting star in this software segment. The software initially implements plain vanilla load balancing with integrated termination for SSL connections. If desired, Istio can also create Let's Encrypt SSL certificates with an ACME script. For this purpose, Istio provides the containers in pods with sidecars, which act as a kind of proxy for the individual containers. In addition to simple balancing, Istio implements security rules and can make load balancing dependent on various dynamic parameters (e.g., on the load to which the individual services on the back end are currently exposed). Moreover, Istio comes preconfigured as a con-

tainer and can be integrated with Kubernetes without too much trouble. For applications that follow microarchitecture principles, approaches like Istio that are specifically designed for this purpose offer the better approach. They should be given preference to homegrown, statically rolled out load balancer containers in any case, because they offer a significantly larger feature set and are usually backed by a large community, which takes a lot of work off the admin's shoulders.

Conclusions

Load balancers come as ready-to-use appliances or software for Linux servers, and a number of capable and feature-rich, software-based load balancers can be had for practically any environment. HAProxy, Nginx, Seesaw, and Zevenet all have very different backgrounds and therefore focus on slightly different scenarios. However, they do their job so well that you can deploy any of the programs discussed in this article in standard use cases without hesitation. At the end of the day, your personal preferences are also a factor that needs to be taken into consideration. ■

Info

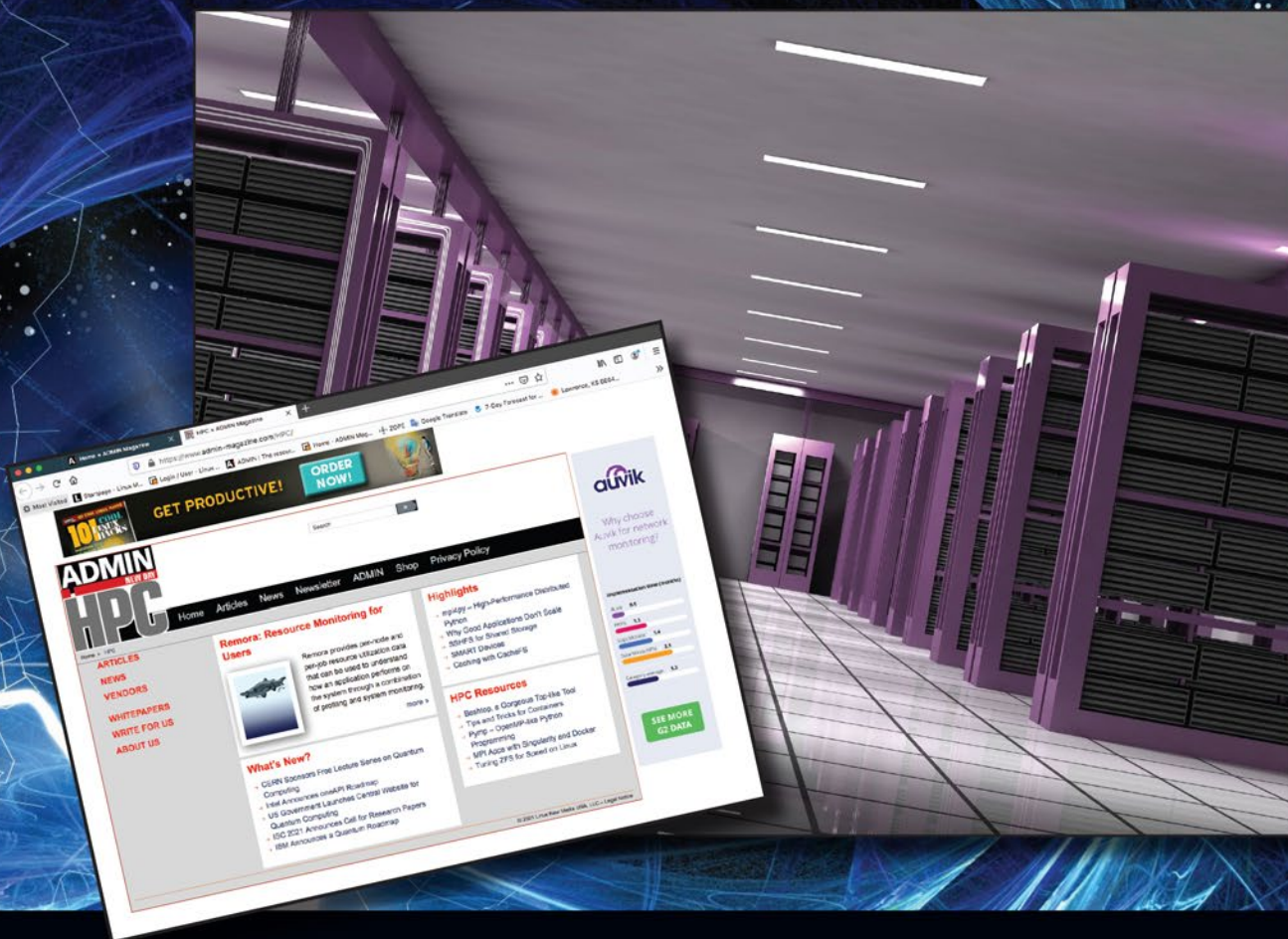
- [1] HAProxy: [<https://www.haproxy.org>]
- [2] Nginx: [<http://www.nginx.com>]
- [3] Seesaw: [<https://github.com/google/seesaw>]
- [4] Zevenet: [<https://www.zevenet.com>]
- [5] Istio: [<https://istio.io>]

The Author

Martin Gerhard Loschwitz is Cloud Platform Architect at Drei Austria and works on topics such as OpenStack, Kubernetes, and Ceph.



A Webzine for High-Performance Computing Specialists



If you work with high-performance clusters, or if you're ready to expand your skill set with how-to articles, news, and technical reports on HPC technology.

ADMIN
Network & Security

admin-magazine.com/HPC

Microsoft 365 and Teams settings and security

Little Tricks

Office 365 and Microsoft Teams come with useful settings for setting up communication channels and securing environments. *By Thomas Joos*

Corporations are increasingly turning to Microsoft's cloud services in the form of Exchange Online, SharePoint Online, or OneDrive for Business. The environment previously known as Office 365, and now renamed Microsoft 365, is enjoying increasing popularity. The same is true for Microsoft Teams – and not just since the pandemic. In this article, I offer simple but effective tips for the administration of both worlds. Although Microsoft recently changed the name of Office 365 to Microsoft 365, in many cases it's still referred to as Office 365, and the names are interchangeable. Nothing has changed in terms of administration. I start with user management, which does not always have to take place at a central IT location, because in Office 365 you can delegate administrative authorizations to users or lower level administrators.

Services are managed in the Microsoft 365 Admin Center, which you can

reach quickest by going to <https://admin.microsoft.com>. You can see the user roles there in the *Manage roles* menu item in user management by clicking on the user to access the menu item. The most important roles appear first in the window. You can view all the roles that are available by clicking on the item *Show all by category*.

Now you will see the roles and a description. If you click on a role, you will see its members in *Assigned admins*. Use the *Export admin list* link to create a CSV file listing all roles and their members. You will only see roles with members; the report will not show empty roles. If you click on several roles, you can compare their rights in context with the menu item *Compare roles*.

PowerShell Management

PowerShell also lets you manage authorizations and other settings for

Office 365; you need the Azure AD module:

```
Install-Module AzureAD
```

To log in, you need the `Connect-MsolService` cmdlet. For example, if you want to add user *Thomas Joos* to the *Teams Service Administrator* group, use the command:

```
Add-MsolRoleMember -RoleMemberEmailAddress "thomas@joos-test.de" -RoleName "Teams Service Administrator"
```

You can display all roles with `Get-MsolRole`.

Spam and Antivirus Protection

In the Exchange Admin Center for Office 365 (<https://outlook.office365.com/ecp>) you can configure the spam settings and the virus protection from the *Protection* menu item. The settings correspond to the options available to you in local installations of Exchange Online. However, quaran-

tine is not managed in the Exchange Admin Center in Exchange Online, but with the *Office 365 Security & Compliance* (<https://protection.office.com/quarantine>) item.

Connecting Smartphones and Tablets

To manage the security of mobile devices connected to Office 365 you need *Office 365 Security & Compliance* (<https://protection.office.com/>). In the Data Loss Prevention section, you will find the *Manage devices* link, which you can use to create policies. Setting up Mobile Device Management (MDM) in Office 365 is a wizard-based process in the Microsoft 365 Admin Center [1]. In several steps, you first configure the cloud environment and create the policies. If users connect their endpoints to Microsoft 365, the policies are transferred automatically, but users first need to register their devices and agree to the policies for private devices. If users refuse to implement the policy on individual devices, they will not be given a connection to Office 365. You first need to enable MDM in the corresponding subscription, because the functions are not enabled by default.

PowerShell Connection

To connect by PowerShell to Exchange Online, first store the credentials in a variable:

```
$user = Get-Credential
```

Now open a session in PowerShell by saving the session data in a variable and then starting the session with that variable:

```
$session = New-PSSession `
-ConfigurationName Microsoft.Exchange `
-ConnectionUri `
https://outlook.office365.com/ `
powershell-liveid/ `
-Credential $user `
-Authentication Basic -AllowRedirection
```

If no error message appears, you can import the session with the

saved data into PowerShell or PowerShell Core,

```
Import-PSSession $session
```

which imports the Exchange Online management cmdlets into the current session and lets you manage Exchange Online in the session. Microsoft customizes the login for Exchange Online with the new Exchange Online PowerShell V2 module. Further information can be found online [2].

Managing with OneDrive Admin Center

The OneDrive Admin Center lets you manage various settings centrally for all users of a subscription for OneDrive for Business. You can reach the OneDrive Admin Center from <https://admin.onedrive.com>. The *Sharing* and *Sync* options are important. You decide whether users are allowed to synchronize and share data from OneDrive, and you can also configure sharing. Microsoft also provides group policy extensions [3], which can be used to control OneDrive for Business by group policies. The download includes ADMX and ADML files.

Once you have copied the group policy files for OneDrive, they will be available in *User Configuration | Administrative Templates | OneDrive* and *User Configuration | Administrative Templates | OneDrive*. In the Group Policy Management Console, settings can also be found in *Computer Configuration | Administrative Templates | Windows Components | OneDrive*. The setting *Prevent the usage of OneDrive for data storage* is interesting. The option *Prevent OneDrive files from syncing over metered connections* lets you stipulate that locally stored data are not synchronized with OneDrive if the computer is on an external network or a slow WLAN.

Message Tracing in Office 365

In the Office 365 Security & Compliance Center at <https://protection.office.com>

under *Mail flow*, you will find the dashboard and message expiration tracking sections. Under *Dashboard* you can view general information about the messages of the users in the subscription. From the *Message trace* item you can create your own queries. If the default reports are not enough, you can also create custom reports. The reports can be exported by *Export results* and saved as a CSV file. If you create your own report, you can run it in real time or save it for later use.

Creating Connectors in Exchange Online

Connectors are an important basis for sending and receiving email in on-premises deployments of Exchange. In Exchange Online, you can also configure in the Admin Center from the *connectors* menu item. For example, you can define settings to stipulate where email is to be sent from the Internet, from partner organizations, or from other sources. The email that local Exchange servers send to Exchange Online can also be configured by connectors.

From the *recipients | groups* menu you can create your own distribution lists and groups in the Exchange Online Admin Center. Distribution lists receive their own email address. If an email goes to this address, all members of the distribution list will receive the message. This process also works for receiving email from outside the Office 365 subscription. In this case, you need to enable the *Senders inside and outside of my organization* option in *delivery management* in the distribution list properties.

Encrypting Email

Microsoft has integrated functions for encrypting email in Office 365, which means that users no longer have to take action themselves or make a special request to clients. The corresponding options for configuring message encryption can be found in the Microsoft 365 Admin Center.

In the web interface, the options are available under *Admin Center* and the selection of *Exchange*. You can access the page directly from <https://outlook.office365.com/ecp>.

Under *mail flow* | *rules* you can access the email rules for the organization. At this point you can create various transport rules. To create a new rule for message encryption, select the *Apply Office 365 Message Encryption and rights protection to messages* option when creating a new rule (Figure 1). The *Apply this rule if* dropdown lets you specify which email messages should be encrypted. The functions for encryption are available in the *Follow these rules* item, which is where you determine how Office 365 should handle the messages. Under *Do the following*, activate the option *Change message security* | *Apply Office 365 message encryption and rights protection* to encrypt email.

Connecting Exchange with Office 365

With the Microsoft Hybrid Agent, local Exchange servers can be operated together with Office 365. Microsoft thus offers a tool that lets you connect local Exchange installations to Exchange Online. A script is also

available online [4] that you can use to test the connection. After downloading, copy the file into the PowerShell Modules directory. The quickest way to open this directory is to use:

```
Cd $PSHome\Modules
```

For the script to work, you temporarily need to set the PowerShell script execution policy to *Unrestricted*, import the PowerShell script into PowerShell as a module, and test the connection:

```
Set-ExecutionPolicy Unrestricted
Import modules .\HybridManagement.psml
Test-HybridConnectivity -testO365Endpoints
```

By the way, Microsoft offers the free Office 365 IdFix tool that you can use to fix problems and incorrect data during Office 365 synchronization with local directories (e.g., Active Directory). The tool detects duplicates and incorrect formatting. You can download the IdFix DirSync Error Remediation Tool on GitHub [5].

Managing Rights in Teams

Microsoft Teams offers a number of roles:

- Skype for Business administrator: Full user access to all Skype for

Business features in Office 365 and admin features in Microsoft Teams.

- Teams service administrator: Admin rights for Microsoft Teams.
- Teams communications administrator: Management of voice and telephony features in Teams and Skype.
- Teams communications support engineer: Access to tools that address communication issues.
- Teams communications support specialist: Management of call records of all participants.

Microsoft Teams management takes place in the Microsoft Teams Admin Center (<https://admin.teams.microsoft.com>). Users access Teams from the <https://teams.microsoft.com> address.

Security in Microsoft Teams

In *Messaging policies*, you can manage many important settings, including security settings, that apply to the channels and chats in each team. *Global* settings include allowing giphy's, memes, and stickers, as well as activating message priority control.

External applications can also be integrated into Teams to improve productivity. In *Teams apps*, you can see the applications that users and ad-

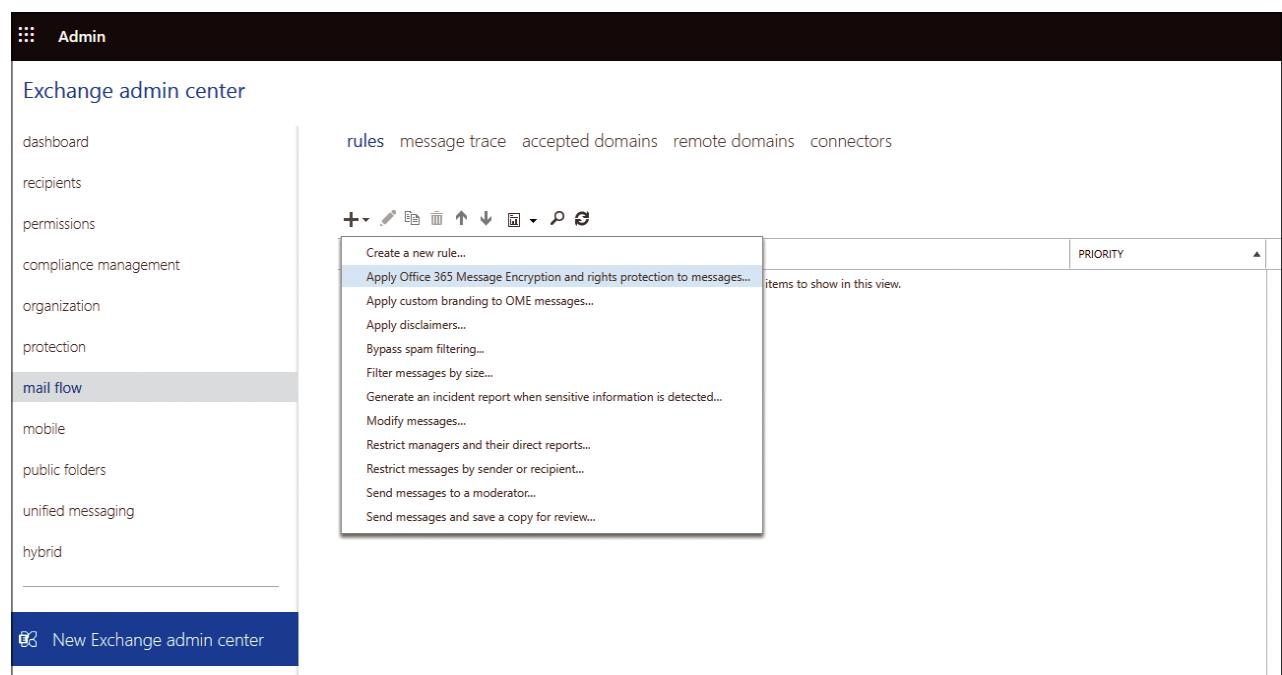


Figure 1: Message encryption rules enable secure communication without user intervention.

ministrators can integrate into Teams through the *Manage apps* item. Under *Org-wide settings*, you can define whether third-party apps should be allowed or whether you are only allowing access to Microsoft apps for Teams in your organization. In Teams Admin Center, you again configure settings that apply to all teams in the Office 365 subscription. Important parameters can be found primarily under *Org-wide settings* | *Teams settings*. The defaults then apply to all teams in the Office 365 subscription.

Regulated Communications

In terms of communication, you determine in *Teams settings* under the Email Integration section whether the channels in teams should also receive messages by email. Each team usually uses the *General* channel, which is also available for all groups across teams. The channels are displayed below the team in the client. Various tasks are available there via the context menu.

The channel can be managed and notifications controlled there, including, among other things, email addresses that can be assigned to a channel. When an email is sent to the channel's address, the message appears in

the channel and is available to users who have access to that channel. Under the Files section, you can also set up whether certain cloud storage – with the exception of OneDrive for Business – should be available in teams (e.g., under the *Files* tab). As far as collaboration with external parties is concerned, you can allow users to invite guests to meetings from *Org-wide settings* | *Guest access*.

In most cases, people from outside the organization are not allowed to access meetings in Teams, even if they have been invited. By enabling the option *Allow guest access in Teams*, planners of a meeting or users in the company are allowed to invite guests. Microsoft also provides a checklist in this section [6] on whether and when it makes sense to allow guest access.

Conclusions

Office 365 is becoming increasingly popular in the corporate environment. With a few simple steps, useful settings can be made in Office 365 and Microsoft Teams, whether you want to secure the environments or set up the appropriate communication channels for users.

In this article, I revealed, without claiming to be exhaustive, a number of such contact points. ■

Info

- [1] Set up MDM:
[\[https://portal.office.com/adminportal/home#/MifoDevices\]](https://portal.office.com/adminportal/home#/MifoDevices)
- [2] Using Exchange Online PowerShell V2 module:
[\[https://docs.microsoft.com/en-us/powershell/exchange/exchange-online-powershell-v2?view=exchange-ps\]](https://docs.microsoft.com/en-us/powershell/exchange/exchange-online-powershell-v2?view=exchange-ps)
- [3] Group policy extensions for Office 365:
[\[https://docs.microsoft.com/en-us/onedrive/use-group-policy\]](https://docs.microsoft.com/en-us/onedrive/use-group-policy)
- [4] Script for testing the Exchange connection:
[\[http://aka.ms/hybridconnectivity\]](http://aka.ms/hybridconnectivity)
- [5] IdFix directory synchronization error remediation tool:
[\[https://github.com/Microsoft/ifix\]](https://github.com/Microsoft/ifix)
- [6] Checklist for inviting team guests:
[\[https://docs.microsoft.com/en-us/microsoft-365/solutions/collaborate-as-team?view=o365-worldwide\]](https://docs.microsoft.com/en-us/microsoft-365/solutions/collaborate-as-team?view=o365-worldwide)

The Author

Thomas Joos is a freelance IT consultant and has been working in IT for more than 20 years. In addition, he writes hands-on books and papers on Windows and other Microsoft topics. Online you can meet him on [\[http://thomasjoos.spaces.live.com\]](http://thomasjoos.spaces.live.com). ■

Cloud-native storage with OpenEBS

Home in the Clouds

Software from the open source OpenEBS project provides a cloud-native storage environment that makes block devices available to individual nodes in the Kubernetes cluster. By Oliver Frommel

In the beginning, running containers in a stateless mode seemed to be the best and simplest approach because it was possible to scale container-based applications by starting additional containers on additional hosts. If you had problems with a container, it could simply be terminated and restarted on another node – or at least that was the theory.

In practice, users unfortunately achieve less with stateless containers than they would like, prompting the Kubernetes container platform in early days to add new features in each release, such as the stateful sets or the storage interface for persistent volumes. Kubernetes also implemented the Container Storage Interface (CSI), which the Cloud Native Computing Foundation (CNCF) made standard.

In addition to the persistent volume capabilities, numerous storage vendors in the Kubernetes world cater to container clusters. If Kubernetes is running on one of the major cloud platforms, you can typically find an interface to the corresponding storage service, such as GCE Persistent

Disk (Google) or AWS Elastic Block Store (EBS; Amazon). Moreover, you have the option to use classic storage protocols, such as NFS, Internet SCSI (iSCSI), and FibreChannel, or modern distributed filesystems such as Ceph or GlusterFS.

Cloud-Native Storage

Another group of storage solutions tries to get to the root of the problem with technologies developed from scratch that understand how to deal with the specifics of a container landscape. Examples of these cloud-native storage approaches include Rook [1], Portworx, StorageOS, and OpenEBS [2], which I take a closer look at in this article.

OpenEBS is an open source product from MayaData, who also provides corresponding offerings with support for enterprise customers. Company founder Evan Powell coined the term “container-attached storage” for the technology, with the essential difference from so-called legacy systems being that all management components of the storage layer run as pods in the

Kubernetes cluster itself. Of course, this is not a unique selling point because other cloud-native offerings (e.g., Rook) also work in this way; however, in the case of classic storage providers or cloud providers, the storage management runs outside the cluster.

One key difference between OpenEBS and Rook, for example, is how storage is handled behind the scenes.

Whereas Rook essentially takes over the task of automating a Ceph cluster running in Kubernetes, OpenEBS has developed its own approach that implements various components for integration into Kubernetes and relies on iSCSI for storage provisioning. The provisioning is completely transparent to the cluster admin, so there is no need to run an iSCSI server, as would be the case with classic Kubernetes-iSCSI integration. You will see exactly how this works in a moment when I take you through installing OpenEBS on a Kubernetes cluster.

In principle, individual worker nodes in the cluster act as storage nodes by making their local storage space available to the cluster. They can be appropriately optimized, dedicated

nodes or even normal worker nodes, provided they have the necessary storage. According to the OpenEBS developers, this flexibility even improves the reliability of the storage, because as you add more nodes for more workloads, storage availability also increases. At the same time, according to the manufacturer, the effects of the failure of individual nodes can be minimized, because OpenEBS internal replication means that the metadata is automatically available on every node.

Installing OpenEBS

To use OpenEBS, you need a Kubernetes cluster (Figure 1) of at least version 1.13 – preferably 1.14 if you want to use some new features like snapshots and volume cloning. Nodes running OpenEBS services must have an iSCSI client (“initiator” in iSCSI-speak) installed, which assumes that you have enough control over the nodes to allow the installation of packages.

To install OpenEBS, you must be in the cluster admin context; in other words, you need maximum permissions for the Kubernetes cluster, in part because various elements of OpenEBS are implemented as custom

resource definitions. To install the OpenEBS components in the cluster, use either the Kubernetes package manager Helm version 2 or 3 or the YAML file of the OpenEBS operator. Instead of a default installation, you can also perform a customized install by downloading and editing the YAML file, which means you can specify, for example, node selectors that specify which Kubernetes worker nodes run the OpenEBS control plane, the Admission controller, and the Node Disk Manager. You can apply the default YAML file directly from the repository:

```
kubectl apply -f https://openebs.github.io/charts/openebs-operator.yaml
```

After that, a quick look at the list of running pods reveals whether everything is working. Here, you have to specify the *openebs* namespace, which is where the OpenEBS components are located (Listing 1). A call to

```
kubectl get storageclass
```

lists the available storage classes, including four new ones: *openebs-device*, *openebs-local*, *openebs-jiva-default*, and *openebs-snapshot-promoter*.

Listing 1: openebs Namespace

```
kubectl get pods -n openebs

NAME
maya-apiserver-7b4988fcf6-f4q9q
openebs-admission-server-54dd65b4c9-hkpdh
openebs-localpv-provisioner-68bb775959-g7z8q
openebs-ndm-ggndz
openebs-ndm-hdp5s
openebs-ndm-operator-6b678c6f7f-tp9t6
openebs-ndm-qh4zs
openebs-provisioner-67bfd5bff-p45mz
openebs-snapshot-operator-85d8d495c-g7b77
```

Listing 2: PVC for Jiva Storage

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: demo-vol1-claim
spec:
  storageClassName: openebs-jiva-default
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 4G
```

Now, in principle, you can start using storage from OpenEBS for pods, even if the storage classes set up so far are not yet complete or optimal.

A simple example of a persistent volume claim (PVC) is shown in Listing 2; it uses

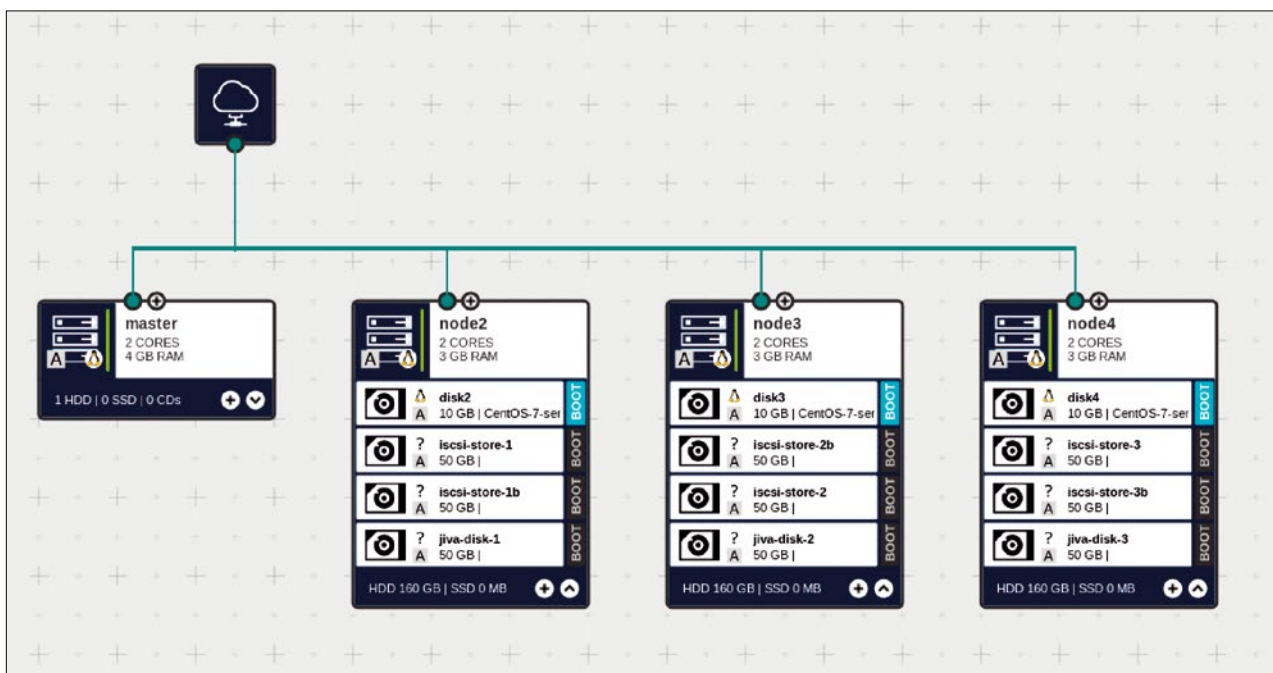


Figure 1: A Kubernetes cluster with storage in the worker nodes.

```
kubectl apply -f pvc.yaml
```

to create a Jiva volume that can then be used in a database's YAML deployment file.

Jiva is one of four available storage engines and, in addition to the two special cases Local Hostpath and Local Device, the more sophisticated storage type implemented by OpenEBS. Each participating node contributes a directory to the Jiva storage pool. A volume created with it is automatically replicated between the nodes. Without further intervention, the default storage pool for Jiva is already configured, creating three replicas of the data, which require three nodes in the cluster.

Managing Storage

For a quick and more detailed look at the configuration of the storage class, you can enter:

```
kubectl describe sc openebs-jiva-default
kubectl describe sp default
```

The first command points you to the associated "default" storage pool, and the second reveals more about it, such as the location (`/var/openebs`) on each storage node. If you want to have more control over your storage, you can add another disk to each node, install a filesystem, and mount it as a new Jiva storage pool. The associated storage class can then be used to define

various parameters, such as the number of replicas.

Basically, data replication happens at the volume level and not the storage pool level. You can see this when you request a persistent volume with the PVC in [Listing 2](#). You will then see four pods in the OpenEBS namespace that contain the *repl number* name component in addition to the PVC's ID PVC. Each pod runs on a different node and takes care of its part of the replication. Two other storage options offered by OpenEBS are Local PV Hostpath and Local PV Device. Whereas the Hostpath engine can make a directory of a node available to the pods running on the same machine, Local PV Device accesses a dedicated block device that is available on a node. One advantage of both approaches over the local volume provisioner, which Kubernetes comes with out of the box, is the ability to provision storage dynamically (i.e., to serve an application's request instead of provisioning a volume in advance in the admin context). Additionally, both storage options, like all other OpenEBS storage engines, support backup and restore with the Kubernetes application Velero.

cStor Storage Engine

The fourth storage engine in OpenEBS is cStor, which is getting ready to replace the good old Jiva. Although Jiva is easy to use, it does not offer as many options as cStor. To use cStor, you first need to create a cStor storage pool – which OpenEBS does not do by default – by making available block devices on each storage node not used elsewhere. The Node Disk Manager scans the available devices and uses a (configurable) blacklist to filter out those that are not suitable for storage, such as

already mounted disks. Potentially, several disks per node then form a storage pool that, together with the storage pools of the other nodes, forms the cStor storage pool. A list of the block devices available in the cluster is output by the call:

```
kubectl get blockdevice -n openebs
```

These devices can then be added to a YAML file and used to compile a storage pool. You can specify, for example, how a node writes the data to its local disks: striped, mirrored, raidz, or raidz2. The fact that these terms are reminiscent of RAID is not a coincidence; they are underpinned by the ZFS filesystem, which OpenEBS uses behind the scenes for cStor for local data storage. However, it should be noted that this only applies to local data storage: As mentioned, replication of the data otherwise takes place on the volume level. A maximum of five replicas are possible, of which at least three must be available for the volume to be usable.

As an alternative to the manual process described previously for creating the cStor storage pool, OpenEBS has developed new Kubernetes operators in version 2.0 under the `cstorPoolCluster` (CSPC) umbrella, which are intended to simplify the provisioning and management of cStor pools, including such things as adding new disks to pools, replacing broken disks, resizing volumes, backing up and restoring, and automatically upgrading the software. More information can be found on the GitHub page [\[3\]](#). If OpenEBS is already installed as described above, you just need to deploy the operator on top:

```
kubectl apply -f https://openebs.github.io/charts/
cstor-operator.yaml
```

Now you can create a YAML file as shown in [Listing 3](#) and enter the block devices you discovered in the process described above. This creates a storage pool that mirrors the data between the two disks (`dataRaidGroupType: "mirror"`). Calls to

Listing 3: cStor Pool

```
apiVersion: cstor.openebs.io/v1
kind: CstorPoolCluster
metadata:
  name: simplepool
  namespace: openebs
spec:
  pools:
  - nodeSelector:
    kubernetes.io/hostname: "node2"
    dataRaidGroups:
    - blockDevices:
      - blockDeviceName: "blockdevice-3f4e3fea1ee6b86ca85d2cde0f132007"
      - blockDeviceName: "blockdevice-db84a74a39c0a1902fced6663652118e"
    poolConfig:
      dataRaidGroupType: "mirror"
```

```
kubectl get cspc -n open-ebs
kubectl get cspi -n openebs
```

show the current status of the pool. To now retrieve volumes from the pool, all you need is a storage class as described in [Listing 4](#). The replica count of this storage class is 1 because more than one pool is not available. If replication is desired, you would have to extend the `cstor-PoolCluster` (CSCP) API accordingly below the `pools` key. The `cstor-simple` storage class can now be used in PVCs to request persistent volumes from the storage.

Conclusions

Storage is a massive challenge for container clusters. OpenEBS provides container-attached storage that

makes block devices available to individual nodes in the Kubernetes cluster. Operation is supported by a Kubernetes operator, which also supports features such as snapshots, backup, and restore. One prerequisite for using OpenEBS is control over the Kubernetes worker nodes, which you need to extend with physical or virtual disks. However, Kubernetes users should not expect performance miracles, because the storage is distributed over the cluster by iSCSI behind the scenes. If you want to run a database with high-performance requirements, you can use the local provisioners from OpenEBS, which offer more convenience than the local provisioners from Kubernetes but are, of course, ultimately tied to the respective Kubernetes node.

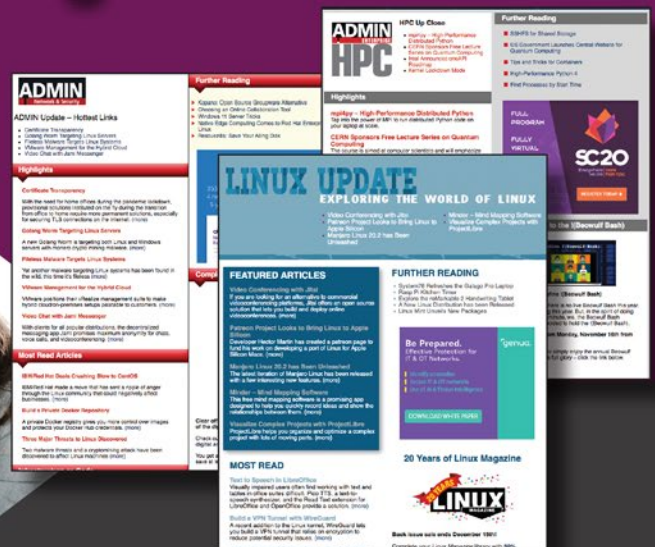
Info

- [1] "Cloud-native storage for Kubernetes with Rook" by Martin Loschwitz, *ADMIN*, issue 49, 2019, pg. 47, [<https://www.admin-magazine.com/Archive/2019/49/Cloud-native-storage-for-Kubernetes-with-Rook/>]
- [2] OpenEBS: [<https://openebs.io>]
- [3] CSCP operator: [<https://github.com/openebs/cstor-operators>]

Listing 4: cStor Storage Class

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: cstor-simpl
provisioner: cstor.csi.openebs.io
allowVolumeExpansion: true
parameters:
  cas-type: cstor
  cstorPoolCluster: simplepool
  replicaCount: "1"
```

IT Highlights at a Glance




Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox.

Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

ADMIN and HPC: bit.ly/HPC-ADMIN-Update

Linux Update: bit.ly/Linux-Update



Integrating Podman and systemd

Perfect Harmony

With the integration of Podman and systemd, you can put any software inside a container under the control of systemd and see almost no difference between running the service directly on the host or inside a container. By Thorsten Scherf

Podman is considered the standard when it comes to managing containers and pods. Seamless integration into the systemd world is a big help, especially in environments where Kubernetes is not used and users want to deploy arbitrary software within containers. After the release of Podman version 2.0, the two components now work in even closer collaboration.

Some readers might frown at the thought of linking Podman [1] and systemd [2] more closely, and with good reason. After all, microservices should run primarily within containers, without the need for a service

manager like systemd. This idea is correct as far as it goes, and in an ideal world, Kubernetes exclusively manages orchestration for containers with microservices. However, experience shows that this is not always possible and that other architectures must also be taken into account. One reason is certainly that software often still exists in classic form and does not necessarily follow a microservice-based architecture.

Administrators don't want to miss out on the benefits that running software inside a container bring, and there are use cases for running software

in containers like regular services on a host. Because Podman is known to be based on a fork/exec model, unlike Docker, it is easy to put containers under the control of systemd

to leverage all of the benefits that the system and service manager offers. In this article, I show examples of both use cases.

Inside and Outside Containers

To begin, you will discover how easy it is to put any software inside a container under the control of systemd.

Listing 1 shows a container file that you can use to create a new container image. The description file for the new image installs the desired software (just the Apache web server in this case), creates a simple configuration for the service, and creates a systemd unit file. Finally, instructions follow for starting the systemd-init service inside containers based on this image. With

```
podman build -t myhttpd ?  
-f <containerfile>
```

you create a new image based on the container file, which is then used directly afterward to start a new container with the command:

```
podman run -d --name=myhttpd ?  
-p 8080:80 myhttpd
```

Listing 1: Container File

```
FROM fedora:latest  
RUN dnf -y install httpd psmisc ; yum clean all; systemctl enable httpd;  
RUN echo "Hello administrator" > /var/www/html/index.html  
RUN mkdir /etc/systemd/system/httpd.service.d; echo -e '[Service]\nRestart=always' > /etc/systemd/system/httpd.service.d/httpd.conf  
EXPOSE 80  
CMD [ "/sbin/init" ]
```

If you call the Podman command as a non-root user, this method basically works because Podman supports rootless containers, but you cannot bind the container service to a privileged host port (≤ 1024) in this case. Instead, container port 80 is mapped to host port 8080 with the `-p 8080:80` option. Finally, the following test confirms that the web server started correctly inside the container and is now accessible from the host on port 8080:

```
curl --silent http://localhost:8080
Hello administrator
```

If you call the `ps tree` command inside the container, you will see that the `httpd` process (Listing 2) is now under the control of `systemd`. Of course, you can also launch other software in the same way using `systemd` inside a container. Depending on the software used, the unit file may be somewhat

larger, which is why it is a good idea in such a case to copy the file from the host system to the container,

```
COPY httpd.conf /etc/systemd/
system/httpd.service.d/
```

as shown here with the `COPY` command within the container file.

Create systemd Configuration with Podman

Conversely, you can also tell Podman to create a `systemd` unit file for a service and then use it to manage containers like regular services on the host system. For example, you can start and stop containers and the applications running in them with `systemctl`. To prevent SELinux from blocking access by a container process to the host's cgroup system, you first need to allow it with the SELinux boolean:

Listing 2: Systemd Control of Httpd Process

```
podman exec myhttpd /usr/bin/ps tree
systemd--dbus-broker-lau---dbus-broker
|-httpd--httpd
|   |-2*[httpd---64*[{httpd}]]
|   `--httpd---80*[{httpd}]
|-systemd-homed
|-systemd-journal
`--systemd-logind
```

```
setsebool -P container_manage_cgroup on
```

If you already have a running container named *apache* and now want to create a `systemd` configuration for it, use the commands:

```
podman generate systemd --files --name apache
sudo cp ./container-apache.service /etc/systemd/system/
```

The resulting unit file (Listing 3) is first placed in the current directory

GOT CLUSTER?

Tune in to the HPC Update newsletter for news, views, and real-world technical articles on high-performance computing.

<https://bit.ly/HPC-ADMIN-Update>

HPC Up Close

- OpenACC - Parallelizing Loops
- DES Completes Dark Matter Survey
- Open Compute Project Call for Posters
- Dolphin Announces New Switch for Composable Architectures
- Usenet

Highlights

OpenACC - Parallelizing Loops
OpenACC is a great tool for parallelizing applications for a variety of processors. In this article, I look at one of the most powerful...

DES Completes Dark Matter Survey
Scientists will continue to analyze the results until 2021

Open Compute Project Call for Posters
Submissions for the OCP Future Technologies Symposium are due on January 31

Dolphin Announces New Switch for Composable Architectures
The XOS624 will support software-defined configuration and device sharing at low latency.

Usenet
Usenet, is a gigantic Internet forum with thousands of subforums. The Usenet system is designed as a federated network, which means you just need...

Most Read

GUI or Text-Based Interface?
Sysadmins are like smokejumpers who parachute into fires, fighting them until they are out, or at least under control. When you jump into the...

Exploring the /proc Filesystem with Python
The `proc` filesystem (stands for `process`), is a Linux pseudo-filesystem that provides an interface to the operating system's kernel data structures.

Quantum Computing Milestone Achieved
Researchers at OQNL, perform independent operations on two qubits encoded on photons of different frequencies.

ISC 2019 Call for Submissions

The ISC High Performance 2019 conference (June 18-20) is currently open to various opportunities for engineers, researchers, and scientists working in high performance computing.

[Learn more here.](#)

Further Reading

- Shared Storage with NFS and SSHFS
- Human Brain Supercomputer Wakes Up
- Resource Management with Slurm
- Intel Unveils New Processor Line
- Discovering ROCm

7 YEARS OF ADMIN On One DVD

Order Now!
Shop.linuxnewmedia.com

2019 Archives Available Now!

Clear off your bookshelf and get every issue from 2018 at 50% off the digital price!

The 2018 ADMIN Network & Security magazine digital archive bundle is available now (Issue #43 through #48). Order now, and get all 2018 issues added to your digital account for access at any time from any device.

[Order the archives today!](#)

Listing 3: etc/systemd/system/container-apache.service

```

container-apache.service
autogenerated by Podman 2.0.4

[Unit]
Description=Podman container-apache.service
Documentation=man:podman-generate-systemd(1)
Wants=network.target
After=network-online.target

[Service]
Environment=PODMAN_SYSTEMD_UNIT=%n
Restart=on-failure
ExecStart=/usr/bin/podman start apache
ExecStop=/usr/bin/podman stop -t 10 apache
ExecStopPost=/usr/bin/podman stop -t 10 apache
PIDFile=/var/run/containers/storage/overlay-containers/
0b43a756b2eb9947239827e571675cef10095515be02a05d7318ba8701e819ff/userdata/common.pid
KillMode=none
Type=forking

[Install]
WantedBy=multi-user.target default.target

```

and then copied to the systemd directory so that the init system has direct access to the file.

Figure 1 shows how you now use `systemctl enable` to activate the new container service. The call to `curl` finally confirms that the container has been started and the web server is running in the container. If you restart the computer, systemd will also take care of activating the new container service from now on.

Another useful feature is that you can now create portable systemd unit files that can be used on other machines to start containers with systemd. To do this, simply add the `--new` option when calling `podman`. The unit file created in

this way can then be used on the local computer or on other computers to place the desired container under systemd control with the command:

```
podman generate systemd --files \
--new --name apache
```

If the image file for the container does not yet exist locally, the registry downloads it as soon as systemd starts the container.

Update Container Images with Podman

Podman now has an auto-update feature for images. If you use the option

`--label io.containers.autoupdate=image` when starting a new container, you can use the `podman auto-update` command to make Podman first check whether a new image exists for this container in the registry before starting a container. If so, it will download it, tell systemd to stop the running container, and then start a new container based on the new image. Note that this feature is only available for containers that are under systemd control.

Conclusions

Podman offers a number of interesting features in the current version. Once you have created a systemd unit file with Podman, there is almost no difference, at least from an administrative point of view, between running a systemd service directly on the host or inside a container. The auto-update feature is also very useful for keeping container images up to date and avoiding the risk of using container instances with old CVEs. ■

Info

[1] Podman project page: <https://podman.io>

[2] systemd project page: <https://systemd.io>

The Author

Thorsten Scherf is a Senior Principal Product Experience Engineer who works on the global Red Hat Identity Management team. You can see him as a speaker at various conferences.



```

>
> sudo systemctl enable --now host-myshttpd
> sudo systemctl status host-myshttpd
● host-myshttpd.service - Podman container-apache.service
   Loaded: loaded (/etc/systemd/system/host-myshttpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2020-09-03 10:16:20 CEST; 2min 52s ago
     Docs: man:podman-generate-systemd(1)
  Main PID: 48227 (common)
    Tasks: 2 (limit: 13946)
   Memory: 3.2M
      CPU: 232ms
   CGroup: /system.slice/host-myshttpd.service
           └─48227 /usr/bin/common --api-version 1 -c 0b43a756b2eb9947239827e571675cef10095515be02a05d7318ba8701e819ff -u 0b43a756b2eb9947239827e571675cef10095515be02a05d7318ba8701e819ff -r /usr/bin/crun

Sep 03 10:16:20 kermit.tuxgeek.de systemd[1]: Starting Podman container-apache.service...
Sep 03 10:16:20 kermit.tuxgeek.de podman[48094]: apache
Sep 03 10:16:20 kermit.tuxgeek.de systemd[1]: Started Podman container-apache.service.
Sep 03 10:16:24 kermit.tuxgeek.de systemd[1]: /etc/systemd/system/host-myshttpd.service:14: PIDFile= references a path below legacy directory /var/run/, updating /var/run/containers/storage/overlay-contain
Sep 03 10:16:02 kermit.tuxgeek.de systemd[1]: /etc/systemd/system/host-myshttpd.service:14: PIDFile= references a path below legacy directory /var/run/, updating /var/run/containers/storage/overlay-contain
Sep 03 10:19:09 kermit.tuxgeek.de systemd[1]: /etc/systemd/system/host-myshttpd.service:14: PIDFile= references a path below legacy directory /var/run/, updating /var/run/containers/storage/overlay-contain
>
> curl --silent http://localhost
<html><body><h1>it works!</h1></body></html>

```

Figure 1: The `systemctl enable` command enables the container service.

Too Swamped to Surf?



ADMIN
Network & Security

ADMIN offers additional news and technical articles you won't see in our print magazine. Subscribe today to our free ADMIN Update newsletter and receive:

- Helpful updates on our best online features
- Timely discounts and special bonuses available only to newsletter readers
- Deep knowledge of the new IT



bit.ly/HPC-ADMIN-Update



Harden your Apache web server

Batten the Hatches

Cyberattacks don't stop at the time-honored Apache HTTP server, but a smart configuration, timely updates, and carefully considered security strategies can keep it from going under. By Christoph Mitasch

Whether you compile Apache yourself or use a package from a repository, you need to keep the software up to date and shut down vulnerabilities as soon as possible. One way to keep on top of important information is to subscribe to Apache's Announce mailing list [1].

In addition to the web server software itself, you also need to ensure that interpreters such as PHP, Python, and Perl and the web applications you use are secured. Last but not least, every security-conscious admin patches the underlying operating system on an ongoing basis.

Installation, Modules, and Updates

As a first step, you should disable unnecessary modules. On Debian and Ubuntu this task is quite easy thanks to the `a2dismod` command. Otherwise, you will have to search for the `LoadModule` directive.

The primary candidates for disabling include `autoindex`, `CGI/CGId`, `Include`, `UserDir`, and `suEXEC`. To determine which modules are already

built-in, type `apache2 -l`, and after disabling modules, call the `apache2ctl -t` command before restarting Apache; this action triggers a syntax check of the configuration. The results will show, among other things, whether the module you wanted to disable is still referenced.

The use of a firewall – either centrally or directly in the operating system – is also useful. In this way you can limit, for example, the number of incoming connections per IP (`connlimit` with `iptables`; meters or dynamic sets with `nftables`). Also, it is often not necessary to allow all outgoing connections on a web server. For example, the `iptables` rule

```
# iptables -A OUTPUT -m owner --uid-owner www-data -m state --state new -j DROP
```

disallows outgoing traffic from the `www-data` system user that does not belong to any existing connection. You can achieve additional security at the operating system level with SELinux (Red Hat, SUSE) or AppArmor (Debian, Ubuntu). Both extensions

implement mandatory access control (MAC) and are extremely powerful, but they also require some training. To be in a position to respond promptly to failures, you will want to monitor the web server with a monitoring solution like Icinga or Zabbix. Other useful extensions include intrusion prevention and detection systems (IPS/IDS) from the open source sector such as Suricata or OSSEC. Comprehensive security management also includes checking logfiles for potential attacks and, ideally, integrating them into a security information and event management (SIEM) system.

Configuration

Currently Apache 2.4 Core has 90 configuration directives [2]. In any case, you should be familiar with the `Directory`, `Files`, `Limit`, `Location`, and `VirtualHost` configuration groups. For those who have been using Apache for many years, it is best to take a look at the changes in version 2.4 [3], which include the new `mod_authz_host` access module with the `Require` direc-

Photo by Jean-Pierre Brungs on Unsplash

tive, which replaces the previous `Order/Allow/Deny` syntax.

Without explicit configuration, the `Directory` directive allows access to the entire root filesystem, which can be prevented by the configuration shown in [Listing 1](#). Depending on requirements, you can then proceed to assign more privileges to the subdirectories you actually use.

Enabling `FollowSymLinks` allows symbolic links to be followed on the filesystem. Remember that a symbolic link could be used to access the `/etc/passwd` file. A more secure variant here is `SymLinksIfOwnerMatch`, which only follows the link if the symlink and target owners are identical.

`Options` controls the functions available in a directory. `Indexes` affects the directory listing; `Include` and `ExecCGI` allow or disallow server-side includes with `mod_include` and CGI scripts with `mod_cgi`, respectively. A very restrictive configuration of the `Options` directive might be:

```
<Directory "/var/www/html/">
    AllowOverride None
    Options -Indexes -Includes -ExecCGI
    -FollowSymLinks
</Directory>
```

`Options` also support inheritance through the use of `+` and `-`. In the following configuration, `FollowSymLinks` and `Includes` are in effect for the `/var/www/html/help` directory:

```
<Directory "/var/www/html/">
    Options Indexes FollowSymLinks
</Directory>
<Directory "/var/www/html/help">
    Options +Includes -Indexes
</Directory>
```

The `AllowOverride` directive deserves special attention because it determines the handling of the very powerful `.htaccess` files, which `None` disables. You can use `AuthConfig`, `FileInfo`, `Indexes`, and `Limit` to allow certain settings. If direct access to the Apache configuration is possible, you generally want to avoid using `.htaccess`. It makes sense to separate the configuration from the content.

Like `Directory`, the `Files` directive can be used to impose restrictions for specific file names, as in the example in [Listing 2](#), which prohibits downloading `.htaccess` and `.htpasswd` files. `Limit` lets you restrict access to certain HTTP methods. In this case, only a successfully authenticated user has access to the specified methods. The `Location` directive refers to the URL and should only be used when dealing with content outside the filesystem.

`Directory` and `Files` are always processed first. `DirectoryMatch`, `FilesMatch`, and `LocationMatch` support the use of regular expressions, as in:

```
<FilesMatch "\.(gif|jpe?g|png)$">
```

Always pay attention to the context in which individual directives apply, as illustrated in the official documentation ([Figure 1](#)).

Security Issues

The `ServerTokens` and `ServerSignature` directives are often referred to in security discussions. In general, obfuscating the web server software or version number does not genuinely improve security, but you will definitely want to update. After all, you don't want to paint a target on your forehead because the web server reports an outdated version. `ServerSignature` is set to `Off` by default, which means that Apache normally does not add a footer to the pages with *Apache Server at www.example.com Port 443*. In contrast, `ServerTokens` defaults to `Full`, and the resulting output is a header. You can query this header with `wget -s (--server-response)`.

The usual recommendation is to use a different setting to avoid revealing too much information immediately about the web server. The minimal variant `Prod` means that the server only outputs the *Apache* name:

Listing 1: Directory Configuration

```
<Directory "/">
    Options None
    AllowOverride None
    Require all denied
</Directory>
<Directory "/var/www/html/">
    Require all granted
</Directory>
```

Listing 2: Restrictions

```
<Files ".ht*">
    Require all denied
</Files>
<Limit POST PUT DELETE>
    Require valid-user
</Limit>
<Location /status>
    SetHandler server-status
    Require ip 192.0.2.0/24
</Location>
```

`ServerSignature Off`

`ServerTokens Prod`

The `FileETag` and `TraceEnable` directives also need to be disabled by setting them to `None` and `off`, respectively, for security reasons.

Fending Off DoS Attacks

The following directives are useful for hardening Apache against denial-of-service (DoS) attacks:

- `RequestReadTimeout`
- `Timeout`
- `KeepAliveTimeout`
- `LimitRequestBody`
- `MaxRequestWorkers` (was `MaxClients`)
- `MaxConnectionsPerChild` (was `MaxRequestsPerChild`)

The timeout options affect how long Apache keeps connections open.

ServerSignature Directive	
Description:	Configures the footer on server-generated documents
Syntax:	<code>ServerSignature On Off Email</code>
Default:	<code>ServerSignature Off</code>
Context:	server config, virtual host, directory, <code>.htaccess</code>
Override:	All
Status:	Core
Module:	core

Figure 1: Excerpt from the Apache HTTP documentation that also shows the context of the directive.

Listing 3: HTTP Header Configs

```
Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
Header always set Strict-Transport-Security "max-age=63072000"
Header always append X-Frame-Options SAMEORIGIN
Header always set X-XSS-Protection "1; mode=block"
Header always set X-Content-Type-Options nosniff
Header always set Content-Security-Policy "default-src 'mailto:' 'self'"
```

Current Apache 2.4 versions usually use sensible defaults (e.g., in version 2.2, `RequestReadTimeout` was still set to 0).

`LimitRequestBody` also has a value of zero in the current Apache version, which means that a client is always allowed to transmit unlimited volumes of data. `MaxRequestWorkers` can be used to control how many simultaneous HTTP connections are allowed and should always be set to reflect

the available RAM. `MaxConnectionsPerChild` does not have a limit by default. A limit can be useful to tell processes to release RAM in case of memory leaks.

HTTP Headers

By defining HTTP headers, you can enhance website security in several ways. On the one hand, you can do this in the HTML code and with server-side scripting languages. On the other hand, you could opt for a centralized approach through the Apache configuration. The examples in [Listing 3](#) show possible application scenarios.

The `HttpOnly` flag prevents cookies being read with scripting languages such as Java or VBScript. The `Secure` flag transmits the cookie only if an encrypted HTTPS connection exists. The `Strict-Transport-Security` header enables HTTP Strict Transport Security (HSTS) for a domain and ensures that, on future visits, the browser will always automatically call the specific domain by HTTPS and deny HTTP access. The remaining options are aimed at providing protection against cross-site scripting (XSS) and similar attacks on the browser.

Mozilla HTTP Observatory [\[4\]](#) provides a good overview of security measures already implemented on a website and suggests further recommendations ([Figure 2](#)).

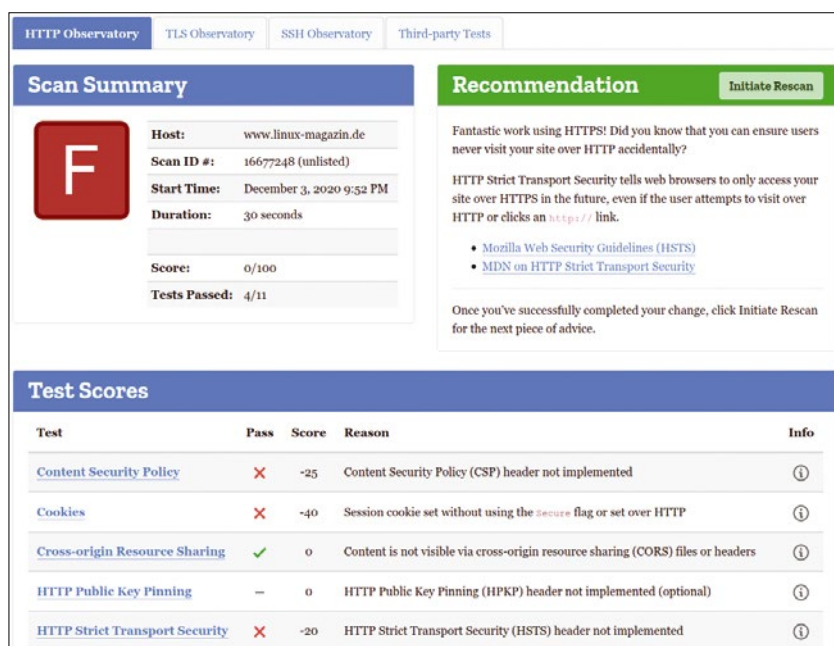


Figure 2: Mozilla HTTP Observatory shows that the German *Linux Magazin* website still has potential for optimization.

TLS/SSL Configuration

With most browsers now warning users against unencrypted HTTP connections, HTTPS connections are already part of a web server's standard configuration.

Mozilla SSL Configuration Generator [\[5\]](#) is a good starting point for a sensible configuration: You specify the server software, including the exact Apache and OpenSSL versions, and then select the desired security level ([Figure 3](#)). The default is *Intermediate*, which is fine in most cases. If you want particularly high security, select *Modern*, which means that only TLS 1.3 is supported and it will surely lock out some visitors on busy sites. Optionally, you can enable HSTS and Online Certificate Status Protocol (OCSP) stacking; both are recommended.

Figure 3: Mozilla SSL Configuration Generator helps find the best possible Apache TLS configuration.

The result is a pre-built configuration ([Figure 4](#)) you can adopt into your own Apache settings. Before restarting to enable the configuration, you again need to check the syntax by typing `apache2ctl -t`. The TLS Checklist Inspector [\[6\]](#) and

SSL Labs [7] can help you check the TLS configuration. The German Check-list Inspector follows Technical Guidelines 03116, Part 4, of the German Federal Office for Information Security [8]. These guidelines are also a good source when it comes to serious recommendations on the subject of TLS/SSL.

Let's Encrypt with mod_md

The managed domain module, `mod_md`, entered Apache in version 2.4.30 and is available as of Ubuntu 20.04 and Debian 10. Additionally, the `mod_watchdog` module handles periodic tasks. You can use `mod_md` to integrate the Automatic Certificate Management Environment (ACME), which is responsible for issuing certificates directly in the Apache configuration. The simplest case will have syntax as in Listing 4. This configuration tells the server to obtain a new certificate automatically from Let's Encrypt, which the virtual host then uses. The renewal process is also automated. Which Let's Encrypt challenge takes effect depends on the port on which Apache is listening. For port 80, you can use the HTTP-01 challenge; otherwise, use TLS-ALPN-01 on port 443.

A wildcard certificate requires a DNS-01 challenge, for which you need to specify a script (i.e., in `/usr/bin/acme-setup-dns`) that matches the DNS records:

```
MDChallengeDns01 /usr/bin/acme-setup-dns
```

This script has to support two calls:

```
acme-setup-dns setup example.com 2
  <challenge-data>
acme-setup-dns teardown example.com
```

apache 2.4.41, intermediate config, OpenSSL 1.1.1d

Supports Firefox 27, Android 4.4.2, Chrome 31, Edge, IE 11 on Windows 7, Java 8u31, OpenSSL 1.0.1, Opera 20, and Safari 9

```
# generated 2020-12-03, Mozilla Guideline v5.6, Apache 2.4.41, OpenSSL 1.1.1d, intermediate configuration
# https://ssl-config.mozilla.org/#server=apache&version=2.4.41&config=intermediate&openssl=1.1.1d&guideline=5.6

# this configuration requires mod_ssl, mod_socache_shmcb, mod_rewrite, and mod_headers
<VirtualHost *:80>
  RewriteEngine On
  RewriteRule ^(.*)$ https://%{HTTP_HOST}%$1 [R=301,L]
</VirtualHost>

<VirtualHost *:443>
  SSLEngine on

  # curl https://ssl-config.mozilla.org/fdhe2048.txt >> /path/to/signed_cert_and_intermediate_certs_and_dhparams
  SSLCertificateFile /path/to/signed_cert_and_intermediate_certs_and_dhparams
  SSLCertificateKeyFile /path/to/private_key

  # enable HTTP/2, if available
  Protocols h2 http/1.1

  # HTTP Strict Transport Security (mod_headers is required) (63072000 seconds)
  Header always set Strict-Transport-Security "max-age=63072000"
</VirtualHost>

# intermediate configuration
SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
SSLHonorCipherOrder off
SSLSessionTickets off

SSLUseStapling On
SSLStaplingCache "shmcb:logs/ssl_stapling(32768)"
```

Copy

Figure 4: Above the configuration, the Mozilla generator indicates from which web browser versions HTTPS access will be possible.

You also need to provide a valid email address in the `ServerAdmin` directive, which is stipulated as a mandatory requirement by Let's Encrypt and which the certification authority uses for important notifications. The Apache log reveals that a new certificate has been obtained:

```
[Fri Jan 01 13:41:06.880247 2020] 2
[md:notice] 2
[pid 776:tid 139942708430592] 2
AH10059: The Managed Domain 2
www.example.com has been setup and 2
changes will be activated on next 2
(graceful) server restart.
```

Afterward, at least a graceful restart is required. When a certificate is renewed, an automatic restart with cronjob makes sense.

The `mod_md` module extends `mod_status` to include the `md-status` section, which means you get JSON output for a specific domain. To query this, use `https://www.example.com/md-status/example.com`. The special status page can be enabled with the configuration:

```
<Location "/md-status">
  SetHandler md-status
```

```
Require ip 192.0.2.0/24
</Location>
```

In the default setting, a certificate status can also be retrieved with `https://www.example.com/.httpd/certificate-status`. You can disable access to it with `MDCertificateStatus off`.

If you don't want to use Let's Encrypt certificates, you can use the `mod_md` module for commercial certificate authorities. Some providers now support the standardized ACME protocol.

Web Application Firewall

The Apache `mod_security2` module implements a full-fledged web application firewall (WAF). Because it is integrated directly into Apache,

Listing 4: mod_md Configuration

```
MDomain example.org
MDCertificateAgreement accepted
<VirtualHost *:443>
  ServerName www.example.com
  ServerAdmin hostmaster@example.com
  SSLEngine on
</VirtualHost>
```

it also works independently regardless of whether a connection is encrypted with HTTPS. The WAF can manipulate the request or the web server's response directly. The module can therefore intervene and provide protection against an attack before the request reaches the web application.

On Ubuntu and Debian, the installation is very simple, thanks to a pre-built module in the repository. To install, run:

```
apt install libapache2-mod-security2
```

You then need a configuration file named `/etc/modsecurity/modsecurity.conf`. However, that's it for the easier part of the setup. The `mod_security2` module is extremely powerful, so I can only provide a brief overview at this point.

By default, the module starts in detection mode (`SecRuleEngine DetectionOnly`), which allows the web application to continue working as usual so you can check ruleset matches. On Debian and Ubuntu, you can find the logfile in `/var/log/apache2/modsec_audit.log`.

You can create all the rules and allow and block lists. However, you also can find free and well-maintained default rules, such as the OWASP ModSecurity Core Rule Set. According to its own specifications, it offers protection against many different types of attacks [9].

Chroot Versus Containers

The `mod_unixd` module and the `ChrootDir` directive let you run Apache in a chroot environment. For a basic static web server, this is quite simple to achieve. Things get more complex if you want to use scripting languages in the chroot, too, and will work with `mod_php`, even in the chroot environment. However, you will face restrictions with certain functions (e.g., the `mail` command), for which you then need additional libraries and configuration files in the chroot directory, quickly making the whole project more error-prone.

As an alternative to the PHP module, the PHP-FPM FastCGI variant comes with its own chroot option. The Apache `mod_security2` module offers a `SecChrootDir` directive for the same purpose, but the overhead remains similar. Independent of the chroot option, isolation in containers is also a good idea. Technology-wise, you can use either Docker containers or Linux containers (LXC/LXD), which allow separate Apache instances for individual applications. If you have to save IP addresses with virtual hosts, it can make sense to connect an upstream proxy that can handle Server Name Indication (SNI) for HTTPS connections in such a scenario.

Conclusions

Sophisticated configuration, timely updates, and well-thought-out security

concepts cannot be implemented in the blink of an eye; instead, they require the admin's constant attention. If you follow all the tips in this article on securing your web server, you will not have to worry too much – the floodgates will be leak-tight. ■

Info

- [1] Announce mailing list: <http://httpd.apache.org/lists.html#http-announce>
- [2] Apache Core functions: <https://httpd.apache.org/docs/2.4/en/mod/core.html>
- [3] Upgrading to 2.4 from 2.2: <https://httpd.apache.org/docs/2.4/upgrading.html>
- [4] Mozilla HTTP Observatory: <https://observatory.mozilla.org/>
- [5] Mozilla SSL Configuration Generator: <https://ssl-config.mozilla.org>
- [6] TLS Checklist Inspector: <https://tls-check.de>
- [7] SSL Labs: <https://www.ssllabs.com>
- [8] German cryptographic guidelines: https://www.bsi.bund.de/EN/Topics/Cryptography/CryptoGuidelines/crypto_guidelines_node.html
- [9] Core Rules protections: [https://github.com/SpiderLabs/ModSecurity/wiki/ModSecurity-Frequently-Asked-Questions-\(FAQ\)#what-attacks-do-the-core-rules-protect-against](https://github.com/SpiderLabs/ModSecurity/wiki/ModSecurity-Frequently-Asked-Questions-(FAQ)#what-attacks-do-the-core-rules-protect-against)

The Author

Christoph Mitasch studied computer and media security at the School of Informatics, Communications, and Media, University of Applied Sciences, FH Upper Austria (Hagenberg), and has worked for Thomas-Krenn AG for 15 years.

Shop the Shop
shop.linuxnewmedia.com

Become a LibreOffice Expert



Explore the **FREE** office suite used by busy professionals around the world!


Create Professional:

- Text Documents
- Spreadsheets
- Presentations
- Databases

Whether you work on a Windows PC, a Mac, or a Linux system, you have all you need to get started with LibreOffice today. This single-volume special edition will serve as your guide!

Order online:
shop.linuxnewmedia.com/specials

For Windows, macOS, and Linux users!



Reducing the Windows 10 attack surface

Digging In

Windows attack surface reduction policies make significant progress in protecting your entire IT infrastructure. By Matthias Wübbeling

Microsoft has been providing tools to administrators to prevent attacks against Windows systems for several years now. The Attack Surface Analyzer introduced in Windows Vista was replaced by Attack Surface Reduction in Windows 10. In this article, we highlight the available protection mechanisms and show you how to use them effectively.

An attacker's goal is to exploit application and computer vulnerabilities (especially their operating systems) alike. In the process, not only servers and workstations attract the attention of attackers, but network devices such as routers, switches, and access points have become targets, especially in recent years. Security researchers are increasingly detecting malware on peripheral devices [1]. If you issue smartphones to your employees with access to the internal network, these devices are also potential gateways for attackers.

Several hundred different attack vectors are known in the literature. Of these, some are well researched and well known to both attackers and system owners, which makes it easy

to provide protection against exploits. Brute force attacks on SSH servers, the lack of encryption in communications, and distributed denial of service (DDoS), for example, can be well managed by tools such as Fail2Ban, a public key infrastructure, and load balancing service providers such as Cloudflare. Although CEO fraud has been very successful in recent years, it can often be averted through awareness campaigns. Zero-day exploits targeting unpublished vulnerabilities in hardware or software have virtually no effective countermeasures.

Ransomware, Phishing, and Insiders

Three different attack vectors have been the subject of recent public discussion. Blackmail trojans, or ransomware, often enter organizational networks through forged email or manipulated email attachments. If the recipient opens the supposedly harmless office file in the attachment and enables the macros it contains, the malware can embed itself in the system. Once in place, the malicious

programs first wait and analyze access patterns, working hours, shares on the local network, or existing backup systems. Once they have collected enough information, the race against time begins.

The malware encrypts as much data as possible without attracting attention, preferably also the backups if they are not access protected. By the time the data owner notices the damage, it is often already too late. Important files are encrypted and a message is sent with an option to receive the code to unlock and decrypt the data by paying a ransom. Often, the programs even offer to decrypt individual files on a test basis. This function is specifically offered by the criminals to strengthen the users' trust in the functionality. The victims are then often much more willing to pay the demanded ransom.

Stolen identity data is another attack vector that has caused more and more damage in recent years. Account hijacking is a problem that should not be underestimated, especially in the area of online stores. In particular, criminals take advantage of the fact that many users use the same passwords to access different services. However, the identity data are not always captured from the service

Photo by Dane Deaner on Unsplash

providers themselves. Although many large online services have had to admit to illegal access to customer data because of a vulnerability in their systems, criminals are unfortunately still very successfully sending very realistic looking phishing email to users. In this way, they can usually rely on obtaining valid access credentials for online accounts directly from the hands of the users themselves. Often enough, insiders are part of successful attack vectors. Although many companies restrict remote access through firewalls, VPN, and special access rights to the local network, the same is not true for physical access to resources on site. Therefore technicians, cleaning staff, or other employees with extensive access to offices and printer and server rooms can also access systems directly. Unlocked user sessions, the ability to boot systems with USB devices, or installed hardware keyloggers that tap user passwords pose a major risk to the integrity of workstations, servers, and printers.

Reducing the Attack Surface

Simple methods can help you reduce the attack surface of a system. The example in this article is a database system. In the standard use case, the database system opens a network socket and listens for incoming connections on all IP addresses used by the operating system – including connections from the Internet, of course. However, if you only use local database access, you will want to select the network interface for the server process or configure the local IP address to rule out access from the Internet.

Incidentally, a firewall installed at the perimeter has a similar effect, at least as far as access from the Internet is concerned. However, if an attacker is already on your network, this firewall can no longer protect your company, but you can restrict local attackers. Many distributions offer passwordless access to the database administrator account for locally logged-in users.

You need to restrict this account, too. With just a simple few steps, you have now reduced your database server's attack surface.

Protection with Windows Defender

Introduced in Windows 10 version 1709, Microsoft Windows Defender Exploit Guard comprises four components: attack surface reduction (ASR), network protection, controlled folder access, and exploit protection. (The abbreviation ASR is used in other ways by Microsoft; for example, it can also mean Azure Site Recovery.) ASR uses various measures to provide protection against malware looking to attack your system through installed Office programs, scripts, or email. Network protection protects your system by expanding the scope of Defender SmartScreen, adding further rules that prevent outgoing HTTP traffic to destinations whose domain names or hostnames are untrustworthy. Controlled folder access protects your data in certain folders from access by less trusted programs on your system. You can configure this feature in the Windows Defender Security Center. When a program from an untrusted

source accesses one of your protected folders, the user sees an *Access Denied* message from Windows Defender. Exploit protection is the logical successor to the Enhanced Mitigation Experience Toolkit (EMET) and offers additional features such as Code Integrity Guard, blocking Win32k system calls, or the ability to check the integrity of the heap memory area. You can try out all the security features of Windows Defender Exploit Guard after enabling it on Microsoft's specially set up demo website [2].

Setting Up ASR

The full scope of ASR is only available with a license for Windows 10 Enterprise. However, some of the ASR rules can also be used in other versions. You have different options for enabling and configuring rules on your systems: Microsoft Intune, Mobile Device Management (MDM), Microsoft Endpoint Configuration Manager, Group Policy, and PowerShell. Depending on which tool you typically use to configure your machines, the convenience of activating the rules varies.

Figure 1 shows ASR rules in the Endpoint Configuration Manager.

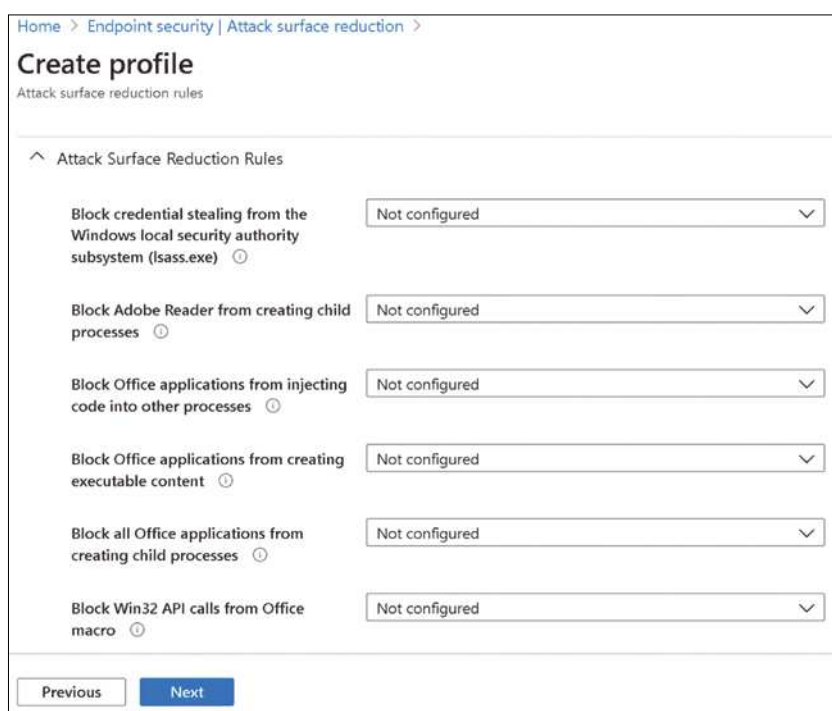


Figure 1: Setting up security rules in Endpoint Configuration Manager.

You will find more information about the effects of the different rules there. Even if it seems to make sense at first glance to activate all rules, you will want to define different rulesets depending on the activity of your employees on the affected system. You can choose different actions for each rule. To check the effects of different rules, select audit mode. You can then first check the effects of the individual rules in the event log.

PowerShell for Quick Tests

If you want to try out ASR rules on individual systems without the available device management tools, PowerShell is a quick alternative for testing. Unlike the graphical interfaces, you need the GUID assigned to each rule when selecting rules in PowerShell. You will already be familiar with the PowerShell `Get-`, `Set-`, and `Add-MpPreference` cmdlets from Windows Defender, which you can use to define, for example, exceptions in monitoring for individual folders or processes.

With the introduction of attack surface reduction, you can also use it to change the actions for ASR rules. The `AttackSurfaceReductionRules_Ids` parameter lets you specify one or more of the GUIDs for which you want to change the status, and you pass in the new status with the `AttackSurfaceReductionRules_Actions` parameter. Given three different rules, you could then configure the actions as follows if you want to disable the first rule, enable the second rule, and disable checking for the third rule:

```
Set-MpPreference -AttackSurfaceReductionRules_Ids -
  <GUID1>, <GUID2>, <GUID3> -
  -AttackSurfaceReductionRules_Actions -
    Disabled, Enabled, AuditMode
```

Keep in mind that you need to specify the action for each rule separated by a comma, even if you want to apply the same action for all rules.

Defining Exceptions

If you want to disable protection with ASR rules for individual objects, you can specify them with the `AttackSurfaceReductionOnlyExclusions` parameter. As the argument, specify the absolute path to a directory or a program. Note the difference between the calls to `Set-MpPreference` and `Add-MpPreference`. The first call completely deletes the existing list of exceptions and creates a new exception list with the values you pass in. However, if you want to add individual objects as exceptions to an existing list, choose the second cmdlet.

15 ASR Rules

In total, Microsoft defines 15 ASR rules for the execution of content from received email or from Office and script files. The following sections look at the GUID and the effect of applying each rule.

The rule with GUID `BE9BA2D9-53EA-4CDC-84E5-9B1EEEE46550` prevents the execution of programs and scripts, JavaScript, and Visual Basic or PowerShell if they are saved from Outlook or Outlook.com. Microsoft promises that email attachments from other webmail providers are also protected but supply no overview of supported providers in the documentation. Outlook itself can be restricted significantly by applying rule `26190899-1602-49E8-8B27-EB1D0A1CE869` when creating child processes.

Different rules protect your system from misbehavior from within Office applications. Enabling the GUID `D4F940AB-401B-4EFC-AADC-AD-5F3C50688A` lets you prevent Office applications from starting additional child processes. This rule contains macros that retroactively load and run files from the Internet and sometimes restricts the ability of legitimate programs and macros that you use as an extension of your Office package to work correctly. However, as described before, exceptions can be defined for this purpose; you should use them carefully, of course.

Keeping Malware Out

To prevent malware from being stored on your system's hard drive from within Office (e.g., for manual execution by one of your employees later on), you need to enable the rule with GUID `3B576869-A4EC-4529-8536-B80A7769E899`. Doing so automatically prevents a very common method of permanently embedding malware on a system. You can block calls to the Win32 API from within Office applications with rule `92E97FA1-2EDF-4476-BDD6-9DD0B4DDDC7B`, which prevents the direct execution of malware, without it having been written to the hard drive first.

You can prevent access to working memory areas used by third-party processes and thus the ability of malware to infiltrate other active processes from within Microsoft Office by activating GUID rule `75668C1F-73B5-4CF0-BB93-3ECF5CB7CC84`. For the most part, you can use this rule safely. Virtually no legitimate enterprise software implements this kind of functionality.

To prevent scripts that are used to download and then install malware on a system, activate two rules: GUID `D3E037E1-3EB8-44C8-A917-57927947596D` blocks JavaScript such that no content downloaded from the Internet or another internal system can be executed, and GUID `5BEB7EFE-FD9A-4556-801D-275E5FF-C04CC` prevents scripts from executing if source code obfuscation methods are detected in them.

Blocking Unknown Software

To prevent the execution of unknown or previously unused software on your systems, use GUID rule `01443614-CD74-433A-B99E-2ECD07BFC25`. However, this is only enabled if you also enable Microsoft Defender cloud delivery protection at the same time with:

```
Set-MpPreference -MAPSReporting Advanced
Set-MpPreference -SubmitSamplesConsent -
  SendAllSamples
```

You also need cloud delivery protection enabled for the next rule, GUID *C1DB55AB-C21A-4637-BB3F-A12568109D35*, which prevents executable files that Microsoft classifies as potential ransomware from entering your system. When it comes to execution prevention rules, you must always keep in mind that the list of exceptions, which you may have already expanded in one of the previous rules, will apply to all rules when they are checked.

The use of analysis tools such as Mimikatz [3] to grab valid login or session data through the Local Security Authority Subsystem Service (LSASS) are restricted with the rule *9E6C4E1F-7D60-472F-BA1A-A39EF669E4B2*. Microsoft mentions in the documentation for this rule that it can generate many false positives under certain circumstances, because the legitimate use of the LSASS interface (e.g., when creating a child process) is also restricted when you apply the rule.

Restricting Software Execution

As an administrator, you know the benefits of remote execution with tools like PsExec or Windows Management Instrumentation (WMI). Because these techniques are also used regularly by malware authors, you can restrict remote execution with the ASR rule *D1E49AAC-8F56-4280-B9BA-993A6D77406C*. The same applies to the execution of programs launched from external storage media such as a USB stick. The rule with GUID *B2B3F03D-6A65-4F7B-A9C7-1C7EF74A9BA4* restricts the execution rights of programs or scripts stored on external devices.

To prevent the permanent installation of malware with WMI event

notifications – or appropriately configured trigger functions for specific events – use rule *E6DB77E5-3DF2-4CF1-B95A-636979351E5B*. However, this rule is currently only available if you use Group Policy or PowerShell, because the graphical tools do not yet have a name, and thus no access, to the configuration of this rule. Officially, it is therefore listed as unsupported in the Microsoft documentation.

As the only non-Microsoft program, you can also restrict the functionality of Adobe Reader with rule *7674BA52-37EB-4A4F-A9A1-F0F9A1619A2C*. Particularly in the case of targeted social engineering attacks by email, but also in the case of classic spam, the malware is often hidden in PDF files that can find their way to users in compressed archives and thus get past the central firewall. Therefore, it may well make sense for you to restrict the ability to launch child processes for Adobe Reader.

If entering the GUIDs for calling the cmdlet is too complicated, you will find useful open source tools on GitLab that can help you select and enable rules [4]. You can download and execute the executable file or the PowerShell script from the repository if you have not already prevented the execution of downloaded content with ASR.

Practice Without Risk

To check the effectiveness of the available ASR rules, you should visit the web page with examples [2] provided by Microsoft, as mentioned earlier. As a registered user, for each ASR rule, you can pick up test files that you should not be able to open or start on your system with the ASR ruleset in place. In this way, you will

also get to know the entries in the Event Viewer and the warnings output by Microsoft Defender.

For use on production systems, you should first configure all rules in audit mode. You can then quickly determine which of your employees' activities conflict with the selected ASR rules, customize the rulesets for distribution with your device management, and distribute the rulesets to the systems that need them.

Conclusions

Over the past few years, Microsoft has repeatedly developed new malware protection approaches. Not all of them have been successful at the end of the day. Basically, you will also want to keep an eye on the attack surface of the entire corporate network beyond your Windows systems. The attack surface reduction of Windows Defender Exploit Guard investigated here covers only a small area of potential points of attack, although they are very common methods used by attackers. Additionally, ASR offers an easy way to get started and, when deployed, can take meaningful steps to protect your entire IT infrastructure. ■

Info

- [1] Xiao, K., D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. Hardware trojans: lessons learned after one decade of research. *ACM Transactions on Design Automation of Electronic Systems*, 2016;22(1). (<http://jin.ece.ufl.edu/papers/TODAES16.pdf>)
- [2] Microsoft demo website: (<https://demo.wd.microsoft.com>)
- [3] Mimikatz: (<https://github.com/gentilkiwi/mimikatz/>)
- [4] Rule selection and activation tool: (https://github.com/hemaureur/MDATP_PoSh_Scripts)

Secure passwordless logins
with FIDO2 and LDAP

Access Granted

Log in to your account securely without a password with LDAP and a schema to establish the objects and attributes required for FIDO2 authentication. By Matthias Wübbeling

Recently, FIDO2 and the passwordless authentication that goes with it have been in focus. As long as you base your login on a flexible database system, you can simply add the required fields for one or more public keys. To store the required information in Lightweight Directory Access Protocol (LDAP), as well, you need to extend the schema and define your own object and attribute types. FIDO2 is a milestone of passwordless authentication. When logging in, the user's browser receives a challenge and has to sign it with the user's private key so that the service provider can validate the signature against the stored public keys. If validation is successful, the login is considered complete.

One advantage of public key procedures such as FIDO2 is that service providers and users no longer have to share a secret, including secrets that are used for two-factor authentication. Therefore, these secrets can no

longer be lost on either side. In case of an attack, all the attacker gets is a user's public key, and as the name suggests, this key can be widely known. Logging in to the service itself or to other services in which the user has deposited the same key remains impossible. To make sure a user is not left without access if a private key is lost, most FIDO2 implementations allow the direct storage of several public keys or different security tokens.

Extending the LDAP Schema

The LDAP schemas normally available (e.g., from the OpenLDAP distribution) do not provide the objects needed to store the information required for FIDO2 authentication directly within the directory. Like other database systems, LDAP lets you extend the set of storable objects by adding an appropriate schema. Once you have successfully loaded a schema, you can create the objects to

match directly afterward. To prevent confusion, each schema is assigned an individual identification number. Globally unique object identifiers (OIDs) have become established for this purpose.

If you have ever worked with the Simple Network Management Protocol (SNMP), you are probably familiar with OIDs. In SNMP, they are used intensively for the queries and responses of the network management protocol. However, OIDs are also used elsewhere, such as, in technical standardization or in requests for comments (RFCs). In particular, Abstract Syntax Notation One (ASN.1), which is often used in such documents, also uses OIDs. To define your own objects and attributes in an LDAP schema, you can now request a personal OID range from the Internet Assigned Numbers Authority (IANA) [1]. The allocation of a subtree for your own OIDs is very convenient and can be accomplished within a matter of days.

For the FIDO2 schema in this article, I have already assigned OIDs from my assigned namespace. You can use these, of course, as long as you do not have to adapt the schema again, which is the only way to ensure global uniqueness.

FIDO2 Schema Structure

To map FIDO2 to an LDAP directory, you need different objects and attributes. First, you need to define the public key as an attribute:

```
attributetype ( 1.3.6.1.4.1.56227.1.1.1
NAME 'fido2pubkey'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
SINGLE-VALUE )
```

The OID in the first line is the unique ID of this attribute type. The name of the attribute is arbitrary; in this scenario, I am prefixing all assigned names with *fido2* for ease of recognition. The syntax specification defines the type of data that will be stored in this object. The value specified is from the namespace belonging to Mark Wahl [2], one of the authors of the LDAPv3 standard, and indicates that binary data is stored in this attribute. The *SINGLE-VALUE* specification means that this attribute can be contained only once in the parent object. The *SINGLE-VALUE* in the definition is then used to define the identifier of the public key that the security tokens specifically use when selecting the private key:

```
attributetype ( 1.3.6.1.4.1.56227.1.1.2
NAME 'fido2credentialId'
EQUALITY caseExactIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
SINGLE-VALUE )
```

This identifier identifies the entire key pair. The specified syntax OID defines an IA5 string, an ASCII-compatible character set, as required by the FIDO2 specification. This attribute can also only exist once in the parent object. Now you need to combine the public key with the identifier in the resulting object type to define this object for FIDO2 authentication and specify a parent object type, in the sense of an inheritance hierarchy, with *SUP top*:

```
objectclass ( 1.3.6.1.4.1.56227.1.1.4
NAME 'fido2credential'
SUP top STRUCTURAL
MUST ( fido2credentialId $ fido2pubkey)
MAY ( fido2credentialDescription ) )
```

Objects defined as *STRUCTURAL* are combined from other object and attribute types. The specifications of *MUST* and *MAY* define the mandatory and optional attributes of your object. The finished *fido2credential* object thus has to consist of the two attributes *fido2credentialId* and *fido2pubkey* that were defined earlier. Optionally, an attribute of the type *fido2credentialDescription* is specified, which can be used to distinguish the different public keys:

```
attributetype ( 1.3.6.1.4.1.56227.1.1.3
NAME 'fido2credentialDescription'
EQUALITY caseExactMatch
ORDERING caseExactOrderingMatch
SUBSTR caseExactSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE )
```

This method lets users manage different key pairs themselves. The user can freely choose the names and use them to remove lost or unused key pairs from their account.

The schema attributes *EQUALITY*, *ORDERING*, and *SUBSTR* describe rules for finding, comparing, and sorting the values of this type. In this example, to distinguish between upper- and lowercase descriptions. The specified OID syntax defines a UTF-8-encoded string.

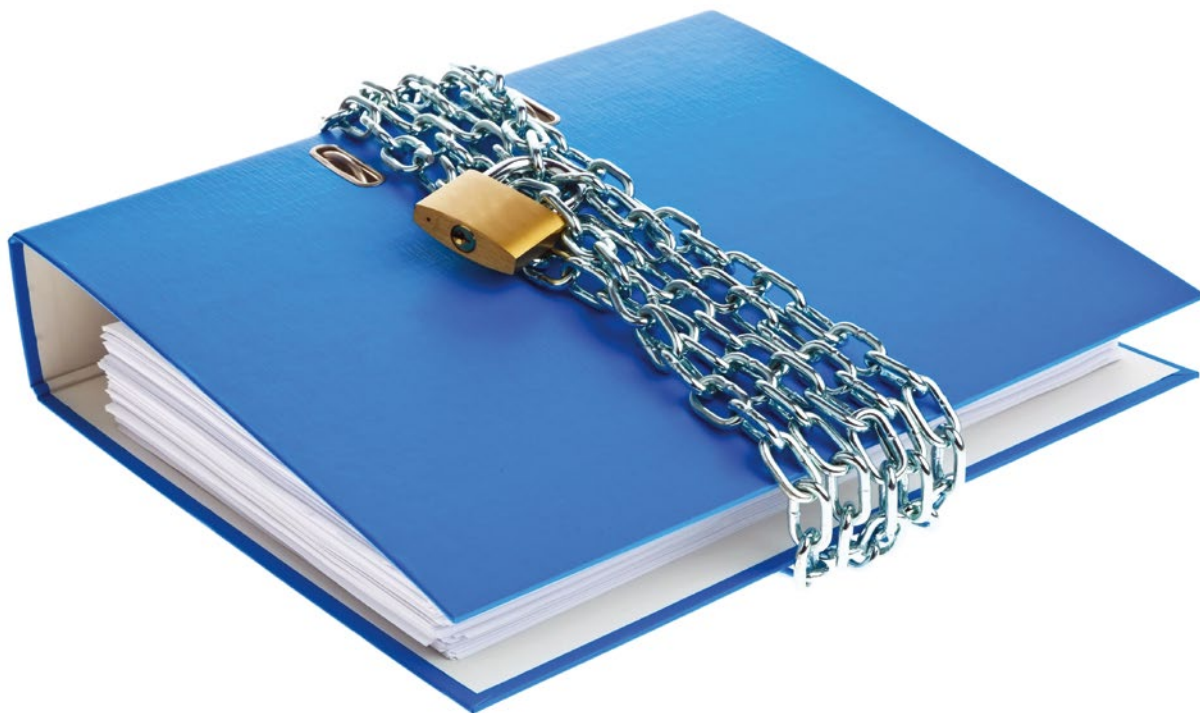
To use the types developed here, you need to group them in a schema file and import them to your LDAP server with *ldapadd*; then, you can assign objects of type *fido2credential* to your users. Remember to adapt the access authorizations of your LDAP server. In particular, you need to make sure that users can only write their own public keys, whereas read access to them may be less restrictive and may even have to be, depending on the implementation of the login process.

Conclusions

FIDO2 lets users log in to their accounts securely without a password. If you use LDAP for authentication in your environment and want to use FIDO2, you can use the schema created in this article to establish the objects and attributes required for FIDO2 authentication in your directory. If you need additional elements in your schema, just apply for a separate section in the IANA OID tree and create additional unique object IDs there. ■

Info

- [1] Request OID area: [\[https://pen.iana.org/pen/PenApplication.page\]](https://pen.iana.org/pen/PenApplication.page)
- [2] Mark Wahl namespace: [\[https://www.alvestrand.no/objectid/1.3.6.1.4.1.1466.115.121.1.html\]](https://www.alvestrand.no/objectid/1.3.6.1.4.1.1466.115.121.1.html)



Protect privileged accounts in AD

Highly Confidential

Granular protection for highly privileged accounts is granted by the Protected Users group in Active Directory and Kerberos authentication policies. By Evgenij Smirnov

In environments characterized by great complexity or the crucial importance of the connected systems, authentication must be clearly regulated. The need for protection is particularly great for privileged accounts such as domain or organization administrators. Active Directory (AD) offers, among other things, the Protected Users group and authentication policies for this purpose. AD entered the market about 20 years ago with Windows 2000 Server. However, it was to be 13 years before Server 2012 R2 introduced one of the biggest security enhancements in the form of Kerberos authentication for highly privileged accounts. Part of the new functionality, the special treatment of the Protected Users group, was automatically introduced on upgrading the domain function level and caused irritation in some cases. Other features such as authentication policies are still unknown to many administrators.

Protecting Highly Privileged Accounts

For the normal user who logs on to their workstation or a shared computer without system admin rights to perform their tasks with the help of application programs, a standard login by Kerberos is quite good enough. The ticket-granting ticket (TGT) issued at login time is valid

for 10 hours by default and is silently renewed when it expires. In principle, you can log in to any client system in the forest, provided the default configuration has not been changed. If older devices are in use, NT LAN Manager (NTLM) authentication is also possible, if necessary, and in most cases, perfectly acceptable. However, the situation is different for a highly privileged account (**Figure 1**).

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\admin>whoami /groups | findstr Protected
OFFICE\Protected Users
Mandatory group, enabled by default, Enabled group

C:\Users\admin>klist
Current LogonId is 0:0x4c3cfd
Cached Tickets: (1)
#0> Client: admin @ OFFICE.EXAMPLETECH.COM
Server: krbtgt/OFFICE.EXAMPLETECH.COM @ OFFICE.EXAMPLETECH.COM
Kerberos Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags: 0x410000 -> canewahla_initial_pre_authent_name_canonicalize
Start Time: 2/15/2021 12:54:12 (local)
End Time: 2/15/2021 16:54:12 (local)
Renew Time: 2/15/2021 16:54:12 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x1 -> PRIMARY
Kdc Called: VMSERVER1

C:\Users\admin>

```

Figure 1: The service life of the TGT is limited to four hours for Protected Users.

“Highly privileged” does not necessarily mean a domain or schema admin. IT managers also need to protect an account adequately that has wide-ranging administrative and data access privileges on a system other than AD itself. An SQL admin has the potential to steal or modify important business transaction data. A file server admin could possibly access confidential drafts from product development, and so on. Service accounts can also be a valid attack vector. These accounts often have deeper access to resources – both on the computer running the service and on the network – than regular users. What often complicates matters is that the passwords for these accounts do not expire. If the password-protecting data protection application programming interface (DPAPI) in Service Control Manager can be stolen, the attacker can exploit the rights of the account for a long time.

The problem of having to update the service account on many servers at the same time when the password changes can be partially countered by administrators who use group Managed Service Accounts (gMSA): Their password is regularly changed by authorized computers and automatically updated in the Service Control Manager [1]. However, the logon capability of such an account is not limited to certain computers per se, and if the automatically set password is known, it can be used anywhere.

Therefore, it is hugely important for the security of your IT environment to address the following issues for highly privileged accounts:

- On which systems can the account log in? A domain admin, Exchange admin, or SQL admin has no business using the PC in the warehouse, where malware or hackers can sniff their credentials.
- How long can an account use the TGT issued at login time? Is re-authentication necessary when it expires? To mitigate the risk, if a highly privileged Kerberos ticket is stolen, the ability to use it for authentication should be time-bombed.

- For service accounts, who is allowed to request a service ticket against services running under this account? Where is this account allowed to log in? For example, if the service account belongs to the database layer of a three-tier application, the only computers the account needs to log on to are the database servers, and the only computers and users that need to communicate with the database service are the application layer servers and their respective service accounts, as well as database administrators and their workstations. If necessary, it is still important that backup and monitoring systems can connect. Normal workstation PCs, on the other hand, have no reason to communicate directly with the back end of an application.

- For computer accounts, some services run in the security context of the executing machine. Here, too, it is important to be able to limit who is allowed to authenticate against these services and whether NTLM is allowed or Kerberos is enforced in the process.

In the early years of AD, security teams built very complex and nearly unmanageable constructs of permissions, policies, and firewall rulesets to enable this kind of granular segmentation. Server 2012 R2 made this work far easier.

Better Protection Through Protected Users

For highly privileged accounts, an easy way to tighten authentication was introduced with Server 2012 R2. When the Primary Domain Controller (PDC) Emulator flexible single master operation (FSMO) role is transferred to a domain controller running Server 2012 R2 (or newer), Windows creates a new global group – Protected Users – with a unique object ID within the domain (RID) 525. This group is not assigned permissions in AD or on individual systems. That said, the membership of a user account in this group causes

some important changes to the behavior of logins for that account:

- None of the login methods (NTLM, Digest, CredSSP, Kerberos) store credentials in the clear or use the highly insecure NT one-way function, independent of the policy regarding delegation of default credentials.
- The login confirmation is not cached, so offline login with a protected account is not possible.
- Kerberos does not generate RC4 or DES keys.

The Protected Users group is available on all server operating systems from 2012 and on all client operating systems from Windows 8. Systems as of Windows 7/Server 2008 R2 were given a security update that also activated this feature in May 2014. However, the protection is only genuinely effective as of Server 2012 R2, which means the following restrictions apply to members of the group:

- NTLM authentication is not allowed.
- Delegation for the account is not allowed, whether limited or unlimited.
- DES or RC4 encryption types for Kerberos pre-authentication cannot be used.
- The TGT is valid for four hours and cannot be renewed. After four hours at the latest, the logged-in user must re-authenticate against a domain controller.

No exceptions exist for specific user groups such as domain or enterprise admins. Completely preventing an account from logging in is theoretically possible just by adding it to the Protected Users group. However, to do this, the domain must have been created before Server 2008, and the user password must have remained the same. As of Server 2008, an AES key is also generated each time the password is changed.

If an account was already created under Server 2000 or 2003 and the password was not changed since then, this account has no AES key and can no longer authenticate. This case is not that unlikely for “emergency admin accounts” that rarely log in. This

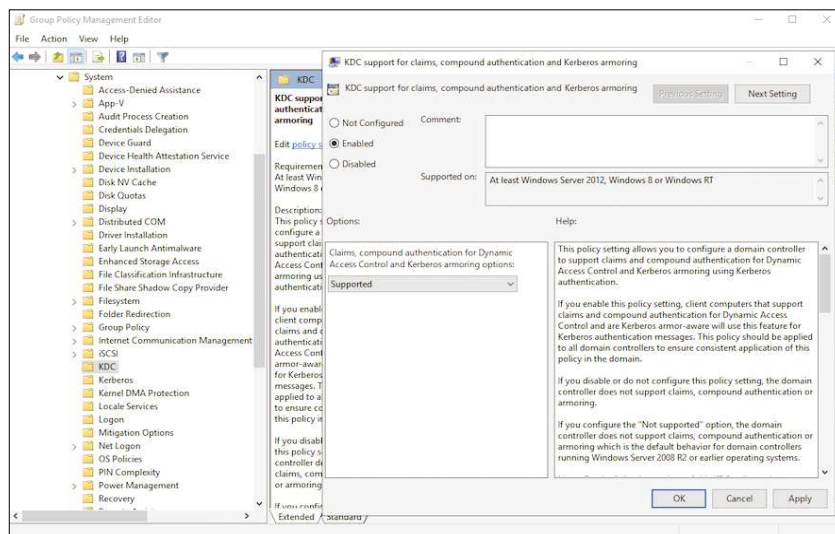


Figure 2: The value *Supported* is the most compatible setting for interoperating with other systems that use AD authentication.

type of account should not become a member of Protected Users, nor should computer or service accounts. For all its hardening, the Protected Users group cannot prevent highly privileged accounts from logging on to machines where they have no business doing so. The granularity of the TGT lifetime or NTLM login authorization also leaves something to be desired: The only central policies are for the domain and the hard-wired configuration of the Protected Users group. The authentication policies provide for finer granularity, which is often necessary, and make the limits even more binding.

Enabling Authentication Policies

Authentication policies only become available after you upgrade AD to the 2012 R2 domain functional level or higher; then, you can design policies and assign them to different user and computer accounts. To take effect, you need to set two group policy settings. On domain controllers, it's the setting with the long name *KDC support for claims, compound authentication and Kerberos armoring* in the *Administrative Templates | System | Kerberos* computer configuration branch (Figure 2). The possible values are *Supported*, *Not Supported*, *Always provide claims*,

and *Fail unarmored authentication requests*. Select *Supported* for the optimal combination of protection and compatibility with other systems that use AD authentication. To the clients – at least those running Server 2012 or Windows 8 or later – assign the *Kerberos client support for claims, compound authentication and Kerberos armoring*. It is located in the *Administrative Templates | System | Kerberos* computer branch. It can only be enabled or disabled; thus, compatibility is

controlled solely on the domain controller side. To check whether the policy is enabled for a particular device, use `whoami /claims`. If you have *Fine Grained Password Policies* (FGPP) in use in your AD environment, you should fast become familiar with how authentication policies work.

On the one hand, this setup allows deviating values to be assigned to individual accounts or groups of accounts for parameters that are normally regulated globally per domain – in this case, the TGT lifetime and NTLM options. On the other hand, these settings are managed from the Active Directory Administration Center (ADAC), which also provides access to the AD recycle bin and dynamic access control.

The authentication policy comprises five sections. Depending on the purpose, entries are not always necessary in all sections:

- User login: TGT lifetime, NTLM login capability.
- Service ticket for user accounts: Which users or computers can request a service ticket for a service running under this user?
- Service registration: TGT life for gMSA.
- Service ticket for service accounts.

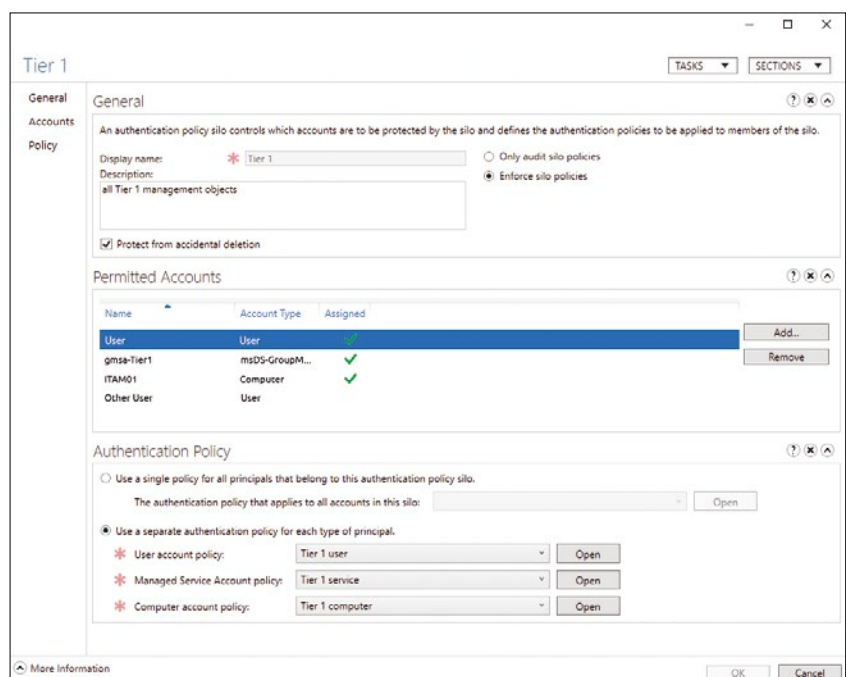


Figure 3: A silo provides the link between accounts and policies.

■ Computer: TGT lifetime, limitations of the application.

For a policy to work for users or gMSA, you need to configure the TGT lifetime for that account type. If in doubt, use the value inherited from the domain or Protected Users membership.

If you enter a different value here, it applies. This way you can also set a longer TGT lifetime for the members of the Protected Users than the hard default of 240 minutes.

Group and Conquer: Policy Silos

One of the most common use cases for authentication policies is that of creating administrative tiers or application silos, where a well-defined set of user and service accounts are allowed to log on to a well-defined set of computers. You map such silos with authentication policy silos, to which you can assign users, managed

service accounts (gMSA), and computers. The silo then determines which authentication policies apply to the silo members: Either one policy for all types of accounts or separate policies per account type.

Policies and silos can be assigned to managed service accounts by PowerShell only. The `Set-ADAccountAuthenticationPolicySilo` cmdlet can be used for both operations. In both cases, you have to use the `-Identity` argument to specify the account to assign. The second parameter is either `-AuthenticationPolicy` or `-AuthenticationPolicySilo`, depending on the desired assignment. Authentication policies can be assigned to the same security principal both directly and through policy silos (Figure 3). Policies from the silo always take precedence over the directly assigned policies.

Another useful aspect of silos is the option to use them as a criterion in selecting allowed users and computers (e.g.,

for issuing service tickets). For example, you can authorize a user account within its own silo to log in interactively. However, if a service is running under that account, only devices from another specific silo can authenticate to that service.

Build an Island

The simplest deployment scenario for authentication policies is to build an authentication silo – that is, a group of computers and a group of users who are the only ones who can log in to those computers (and only there). The following examples show how you can achieve this:

- Create an authentication *Island* policy.
- Create an authentication policy silo *Island silo* and set the Island policy for all account types.
- Assign the designated computers and users to the Island silo.

Shop the Shop → shop.linuxnewmedia.com

Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

Want to subscribe?

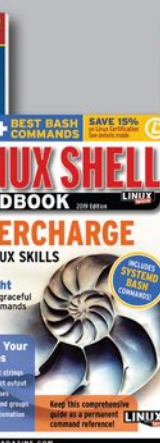
Searching for that back issue you really wish you'd picked up at the newsstand?

➤ shop.linuxnewmedia.com

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



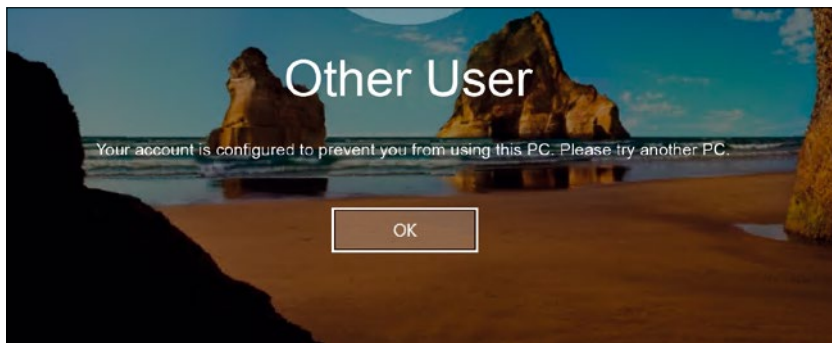


Figure 4: Users cannot access computers outside the authentication policy silo.

- Edit the Island policy and set the TGT lifetime for user accounts to an appropriate value (600 minutes is the AD default; 240 minutes is the default for Protected Users).
- In the Computer section, create the Access Control Conditions *User – Authentication Silo – equals – Island silo*.
- Create the same condition in the User section.

As of now, only the Island users can log in to Island computers; all others will receive an error message indicating the authentication firewall. When Island users try to access machines outside their island, they get an error message (Figure 4). For the settings to take effect, you have to restart the Island computers.

Monitoring and Troubleshooting

Microsoft has provided a special event log channel *Microsoft\Windows\Authentication* for monitoring the Kerberos protection functionality. On the domain controllers, the authentication attempts of the protected users are recorded in the log channels *Microsoft\Windows\Authentication\ProtectedUserSuccesses-DomainController* and *Microsoft\Windows\Authentication\ProtectedUserFailures-DomainController*. Failed authentication policy activity can be found in the *Microsoft\Windows\Authentication\AuthenticationPolicyFailures-DomainController*

channel. Regardless of whether the Protected Users group contains members or you have created authentication policies, these protocols are disabled for now. You should also enable them for troubleshooting purposes only by selecting *Enable Log* from the context menu of the respective event log. Alternatively, you can issue the following PowerShell command on each domain controller:

```
$logname = "
"Microsoft-Windows-Authentication/
ProtectedUserSuccesses-DomainController"
$log = Get-WinEvent -ListLog $logname
$log.IsEnabled = $true
$log.SaveChanges()
```

All three channels can be enabled similarly. On the affected member machines, you need to enable the client event log. Again, the fastest way to do this is with PowerShell:

```
$logname = "
"Microsoft-Windows-Authentication/
ProtectedUser-Client"
$log = Get-WinEvent -ListLog $logname
$log.IsEnabled = $true
$log.SaveChanges()
```

The logs are very verbose, so their size can quickly reach the default configured limit of 1MB if you have a large number of protected accounts. You can set the size immediately on enabling by adding the `$log.MaximumSizeInBytes = 128MB` line before `SaveChanges()` in

the PowerShell code above. This addition increases the maximum log size to 128MB. Other useful tools for troubleshooting Kerberos authentication include `klist` [2] and `whoami` [3] – specifically, `whoami /claims` in this case.

When troubleshooting, always keep in mind that authentication policies, as the name suggests, control authentication. All other mechanisms evaluated after successful authentication, such as the revocation of local logon rights or the logon restrictions in AD (attributes `logonHours` and `userWorkstations`), remain in force.

Conclusions

Protected Users and authentication policies allow highly granular control of the user login for highly privileged accounts. Because these mechanisms act directly on the Kerberos protocol, they are more robust against unwanted changes than other approaches used to influence login behavior. Accounts are managed by the AD Management Center or PowerShell and are distinguished between User, Computer, and Service (gMSA) accounts.

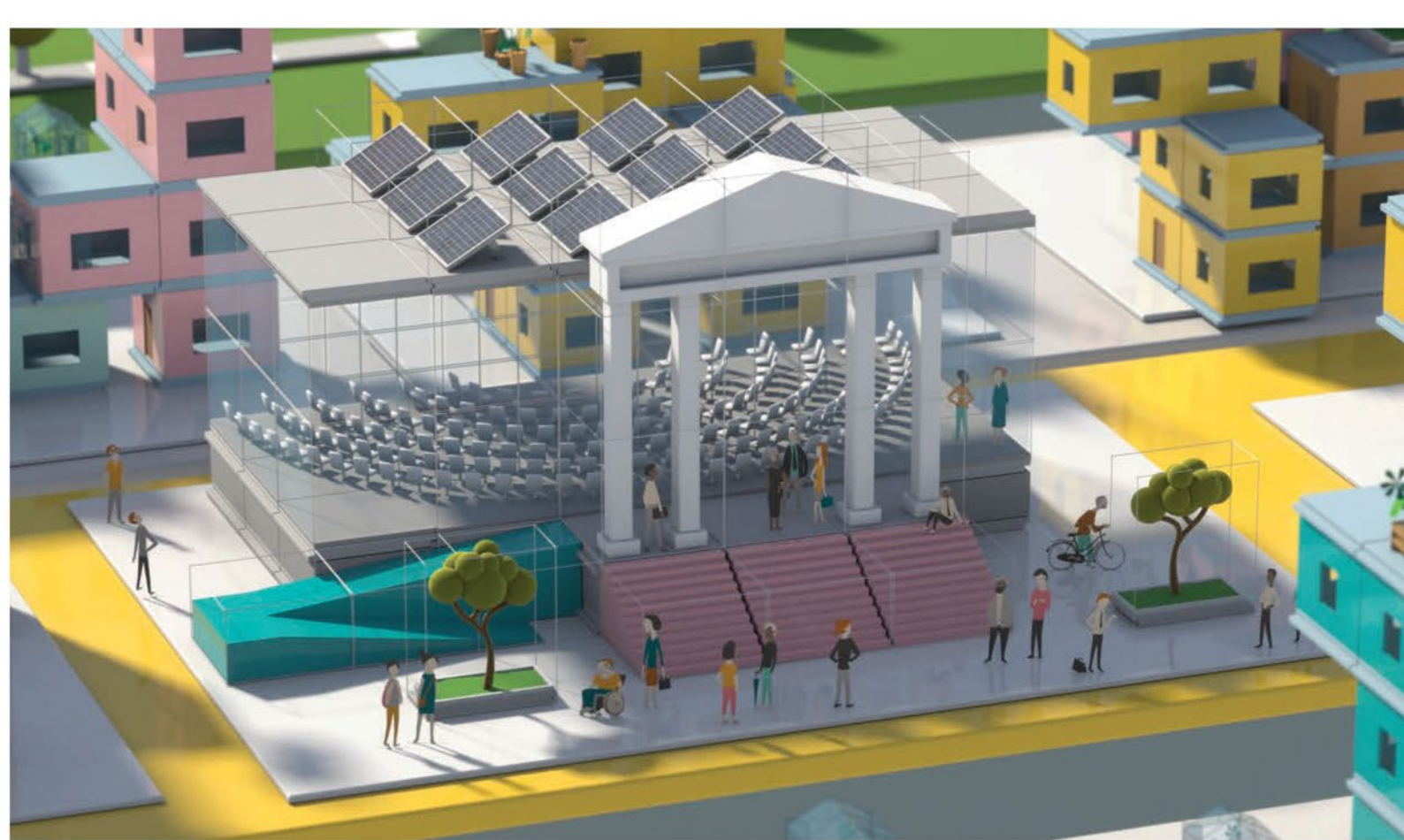
Even these new ways of securing for highly privileged accounts do not relieve you of the responsibility of implementing the least privileges principle and carefully monitoring login behavior in your environment. ■

Info

- [1] gMSA: [<https://docs.microsoft.com/en-us/windows-server/security/group-managed-service-accounts/group-managed-service-accounts-overview>]
- [2] `klist`: [https://web.mit.edu/kerberos/krb5-devel/doc/user/user_commands/klist.html]
- [3] `whoami` documentation: [<https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/whoami>]

Public Money

Public Code



Modernising Public Infrastructure with Free Software



Free Software Foundation Europe

Learn More: <https://publiccode.eu/>



HPC resource monitoring for users

Close Companion

Remora provides per-node and per-job resource utilization data that can be used to understand how an application performs on the system through a combination of profiling and system monitoring. By Jeff Layton

While chatting with some colleagues, we discussed how some applications return unusual results, even though previous runs had produced the expected results. As they were discussing ways to tell whether applications were running correctly or incorrectly, I thought it would be great to get a snapshot of what was happening on all of the nodes involved in a job. I like to think of this as “application telemetry.” A simple search on “telemetry” [1] brings up a definition like “... the process of recording and transmitting the readings of an instrument.” In this case, the instrument is the high-performance computing (HPC) system, and the readings are resource aspects of the system (e.g., CPU, memory, network and storage usage, etc.). Here, the telemetry is used to help the user understand what their application was doing during execution. The keyword in that last sentence is *user*. The case in point: The user has access to resource usage for their application to spot problems, prompting me to revisit Remora. Remora: REsource MONitoring for Remote Applications [2], from the

University of Texas Advanced Computing Center (TACC), combines monitoring and profiling to provide information about your application. Not strictly a profiler and not strictly a monitoring tool in the traditional sense of monitoring the entire cluster, Remora provides per-node and per-job resource utilization data that can be used to understand how an application performs on the system. The user (not just the admin) can go back and examine what was happening on systems while a job was running. The goal of Remora is simplicity, which is achieved by using commonly installed tools that focus on the user, putting data and possibly information in the *user's* hands (and probably the admin's if an issue crops up). The data can also be used by admins in a collective way to understand how the system is being used.

Overview

Before diving into the ins and outs of Remora, keep in mind two things: (1) it is focused on the user, and (2) it is neither a profiling tool nor a

monitoring tool. In essence, Remora is more of a higher level usage reporting tool that tells what resources your job used along with some associated details. It does not dive deep – that's more the function of a profiler – and it doesn't produce fine-grained system monitoring details; rather, it focuses on information that the user can use to understand whether a job seems to perform correctly and, if not, information that can start the “debugging” process. System administrators have access to the same information, so they too can examine the job if the user feels something went wrong. Remora collects several streams of information with simple userspace tools:

- Memory usage, including CPUs, Xeon Phi, and NVidia GPUs
- CPU utilization
- I/O usage – Lustre, data virtualization service (DVS)
- Nonuniform memory access (NUMA) properties
- Network topology
- Message passing interface (MPI) communication statistics (currently you have to use Intel MPI or MVAPICH2)

Lead Image © Andrey Russinkovskiy, 123RF.com

- Power consumption
- CPU temperatures
- Detailed application timing

To capture all this information, Remora uses SSH to connect to all the nodes used in the application by spawning a background task on each of the nodes and regularly capturing the data. However, the I/O data is only captured on the master node of the application.

No Remora-specific applications are used to gather the information. Rather, existing applications are used along with information parsed from the `/proc/` table. A partial list of the tools and data sources used are:

- `numastat`
- `mpstat` (one of my personal favorites)
- `nvidia-smi`
- `ibtracert`
- `ibstatus`
- `xltop`
- `mpip`
- `python`
- `/proc/meminfo`
- `/proc/[pid]/status`
- `/proc/sys/lnet/stats`
- `/sys/class/infiniband`

Remora uses these tools and sources to collect information at a specific interval while the application runs. Because Remora runs in user space, it only collects information associated with the application and can't gather escalated privileged information. In the case of MPI applications, it grabs the `hostnode` list of environment variables and uses that for SSHing into the nodes for data gathering.

As of version 1.8.4, Remora requires either Intel MPI or MVAPICH2. The profiling library `mpiP` [3], is used in conjunction with one of these two MPI libraries to gather MPI profile stats. When Remora is finished, it creates a directory of form `remora_XXX` in the directory in which the application runs. Subdirectories contain the raw data, and you'll find an HTML page you can open to examine and plot the data (this is REALLY amazing!). Remora collects data from as many of the sources as possible. For example, if it detects that Lustre [4] is installed, it will grab data for that. If

it detects the presence of an InfiniBand network, it will collect data for that. If it doesn't detect something, it won't try to gather data for it, and you won't be able to create a chart for the data.

The sources for which it attempts to gather information is controlled by a configuration file. On my test system the path to the file is `/home/<user>/bin/remora-1.8.4/bin/config/modules`. You can edit that file to remove resources you do not want gathered. When installed, that file contains all the sources of information. In the default list below, each line comprises the name of the module and the directory in which the metric belongs.

```
cpu,CPU
memory,MEMORY
numa,NUMA
dvs,IO
lustre,IO
lnet,IO
ib,NETWORK
gpu,MEMORY
network,NETWORK
power,POWER
temperature,TEMPERATURE
eth,NETWORK
```

For example, if you do not have an InfiniBand network, you remove it from the list and Remora won't attempt to gather that information.

For this article, my config file is:

```
cpu,CPU
memory,MEMORY
eth,NETWORK
```

Installing Remora

Installation is not difficult. The approach is slightly different from the usual `./configure; make; make install`, and you need to be aware that because Remora can provide MPI statistics, you need to build it with the intended version of MPI (i.e., do not cross the MPIs). Of course, you don't have to use MPI tools, and Remora will just continue with the configuration and installation.

For this article, I built Remora into my home account with the command:

```
REMORA_INSTALL_PREFIX=/home/laytonjb/2
bin/remora-1.8.4 ./install.sh
```

You can install it in a common directory if all users are to have access.

(Note that a previous article on Remora [5] originally had a typo (now corrected) in the installation command, in which the letter *P* was missing in *PREFIX*.)

If you use multiple versions of MPI, you need to build Remora for each version. If you are using environment modules (e.g., `Lmod` [6]), you can easily write an environment module for Remora so that it is added to the user environment when the corresponding MPI module is loaded.

Example 1

Remora is very simple to use: Just add `remora` before your usual command. For example, a simple command line for the application `./myapp.exe` would become:

```
$ remora ./myapp.exe
```

In the case of MPI code, the command

```
mpirun [...] ./mpiapp.exe
```

would become:

```
$ remora mpirun [...] ./mpiapp.exe
```

Notice that both commands are run as a user – elevated or root privileges are not required, which goes back to Remora's design of focusing on and providing users with useful information.

The code in this article is for a simple serial Poisson solver for a rectangular grid. Although I used it in the past, the link to it no longer works. If you really want something equivalent, then use the OpenMP version of the code (see Example 2) and just set `OMP_NUM_THREADS` to 1. You won't get the exact same timings as the old serial code, but it is probably a reasonable substitute.

To get a fairly long run time, I adjusted a few of the application parameters

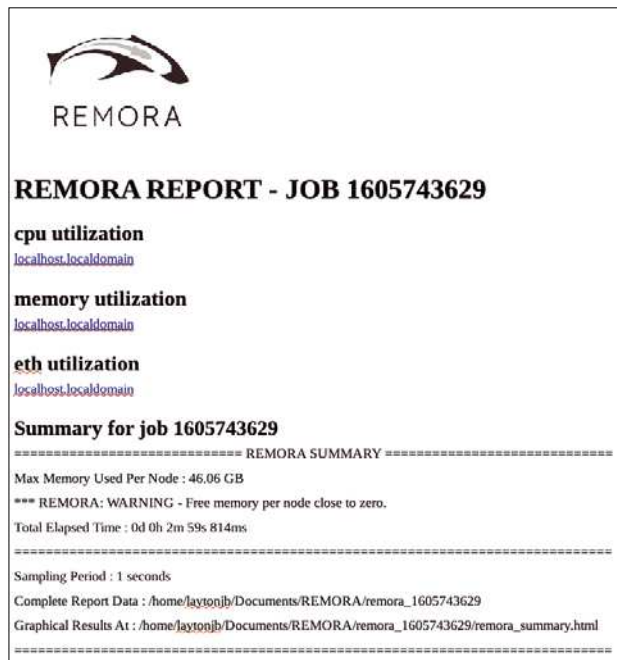


Figure 1: The web page showing the Remora output for the Example 1 code.

```
nx = 8000
ny = 8000
t_max = 10000
tolerance = 0.00004D+00
```

and compiled the code with GFortran on a CentOS 7.8 system.

Remora first creates a subdirectory that contains the system information as a function of time. For this particular test, that subdirectory is `remora_1605743629`. This directory then has several subdirectories with the raw data. You can parse through the data, or Remora creates a web page (HTML) that plots the data for

of the metrics have been monitored. The images are created by Google Charts. To include them in this article, they have been screen captured. **Figure 2** is a plot of CPU usage versus time for Example 1, which is a serial application, so only one core was used. Notice how the kernel moves the application from one core to another. Remora itself uses little CPU time.

The second plot of memory usage during the application run is shown in **Figure 3**. These memory metrics are gathered from `/proc/<pid>/status` and `/dev/shm`.

you, which is the easiest way to get a quick glimpse of what happens during application execution. Just open the web page in your favorite browser (**Figure 1**).

The HTML summary page lists the system metrics that Remora is capable of monitoring. A link below the metric means the corresponding data is available. Notice that for this simple case, only some

The memory stats cover:

- **TMEM (Max):** Total free memory. Considers the memory not being used by the application, the libraries needed by the application, and the OS.
- **SHMEM:** Shared memory. Applications have access to shared memory by means of `/dev/shm`. Any file put there counts toward the memory used by the application.
- **RMEM:** Resident memory. Physical memory used by the application.
- **RMEM (Max):** Maximum resident memory.
- **VMEM:** Virtual memory. This information is important for watching to see if the OOM killer kicks in.
- **VMEM (Max):** Maximum virtual memory.

Finally, **Figure 4** is a plot of Ethernet usage during the run.

Remora uses SSH to gather stats because the application can use MPI; otherwise, the usage is just regular network traffic.

A few environment variables can be used with Remora to control its behavior:

- **REMORA_PERIOD:** How often statistics are collected. The default is 10 seconds. Integer values are acceptable.
- **REMORA_VERBOSE:** If set to 1, this variable tells Remora to send all information to a file. The default is 0 (off).
- **REMORA_MODE:** Specifies how many stats are collected. Possible values

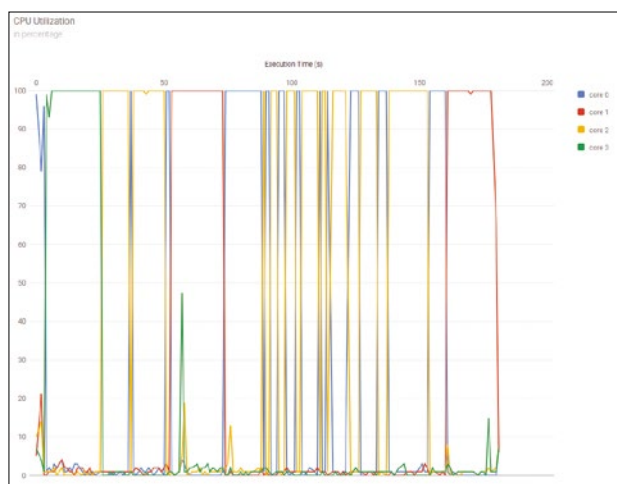


Figure 2: Example 1 CPU utilization plot.

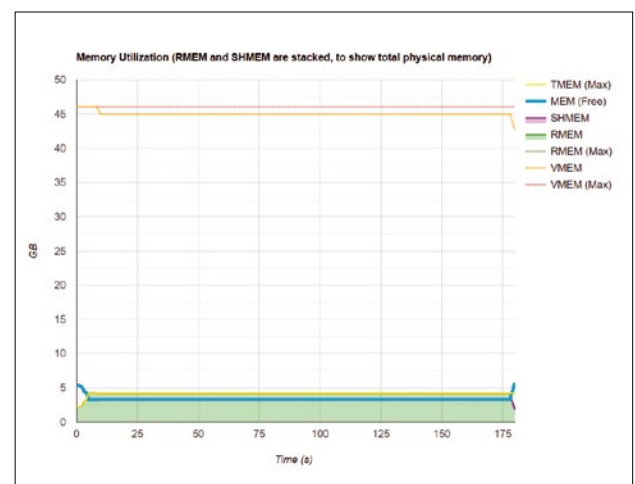


Figure 3: Example 1 memory utilization plot.

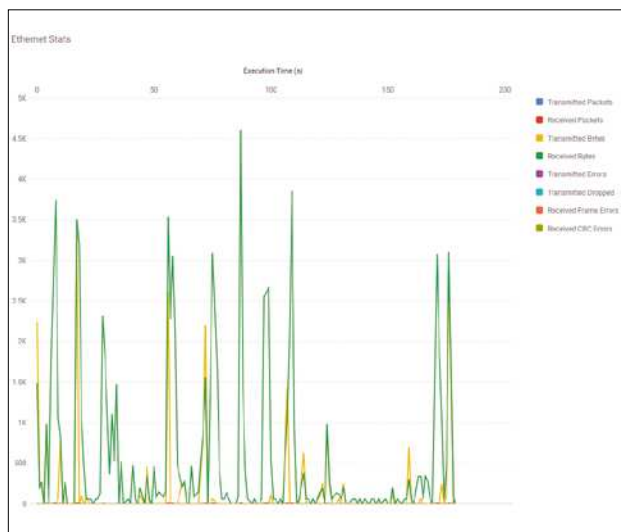


Figure 4: Example 1 Ethernet utilization plot.

include FULL (default) for CPU, memory, network, and Lustre, and BASIC for CPU and memory.

- **REMORA_PLOT_RESULTS:** Controls whether the results are plotted. The default, 1, generates HTML files, and 0 generates plots only if the postprocessing tool (`remora_post`) is involved.
- **REMORA_CUDA:** If set to 0, turns off GPU memory collection when a GPU module is available on the system.

You can set these variables as you like or need. Setting `REMORA_PERIOD` to 1 second could produce a fair amount of data if the application runs over a long period of time. A period that is too short could also affect application performance.

`REMORA_VERBOSE` is a good flag to set if you want to learn more about the code and understand how it measures resource usage. Of course, its obvious use is for when you are having issues with Remora.

I never use `REMORA_MODE` because I can control what I want measured by changing the module config file, as previously mentioned. The same is true for `REMORA_PLOT_RESULTS`, because I always want to see the plots.

Example 2

Example 2 is basically the same as Example 1, but it uses the OpenMP version of the Poisson solver [7]. In

this case, all of the cores in the system are used.

Notice in **Fig-**

ure 5 that the application execution time with OpenMP is much shorter when using four cores than when using one core. **Figure 6** shows CPU usage versus time for Example 2. Remember, this is an OpenMP application running on all cores, which is reflected by 96%-100% core utilization during the entire run. **Figure 7** shows the plot of memory usage and **Figure 8** a plot of Ethernet during the application run.

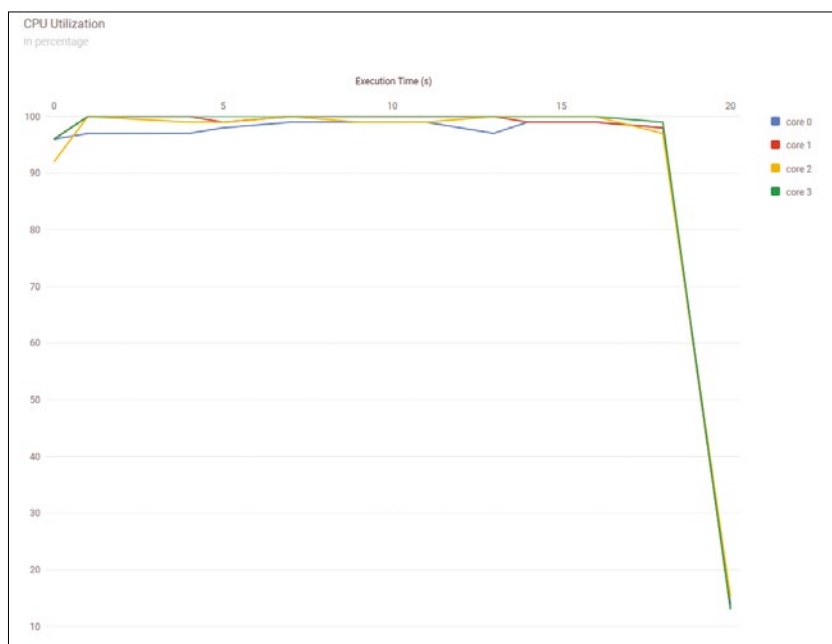


Figure 6: Example 2 CPU utilization plot.



REMORA

REMORA REPORT - JOB 1605743904

cpu utilization

[localhost.localdomain](#)

memory utilization

[localhost.localdomain](#)

eth utilization

[localhost.localdomain](#)

Summary for job 1605743904

```
===== REMORA SUMMARY =====
Max Memory Used Per Node : 42.69 GB
*** REMORA: WARNING - Free memory per node close to zero.
Total Elapsed Time : 0d 0h 0m 20s 268ms
=====
Sampling Period : 1 seconds
Complete Report Data : /home/laytonjb/Documents/REMORA/remora_1605743904
Graphical Results At : /home/laytonjb/Documents/REMORA/remora_1605743904/remora_summary.html
=====
```

Figure 5: Remora output for Example 2.

Summary

HPC admins are always looking for better ways to monitor the systems for which they are responsible by understanding how the hardware is operating and seeing how user applications are performing. Many tools and techniques – both hardware and software – are available for monitoring systems. Even though you can find tools and techniques to coordi-

nate monitoring with resource managers (job schedulers), all of these are administrator-oriented tools. Users have precious few tools to monitor the resources their applications are using. With this “application telemetry” information, users can

understand the *pattern* of their application, whether it seems to be performing correctly or incorrectly, what resources they consumed, and how their application is balanced across several nodes in the system – or even a single node.

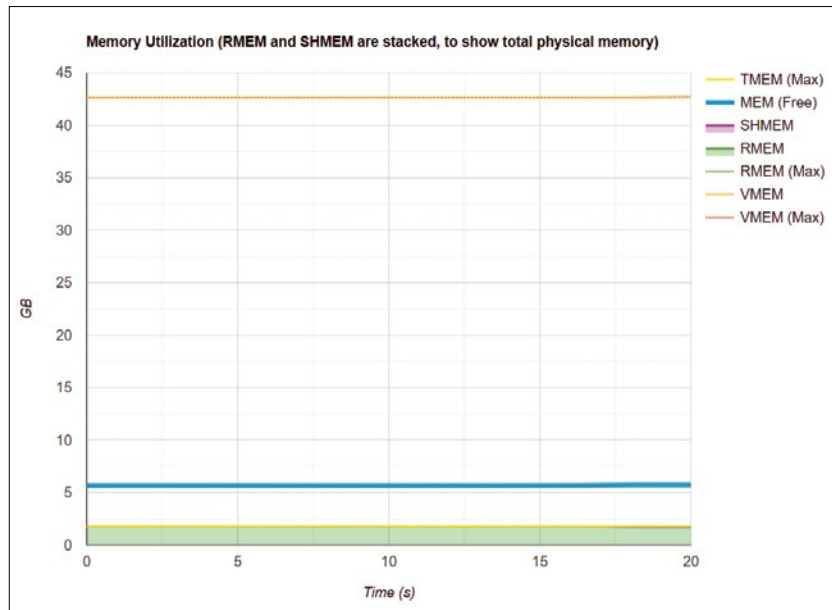


Figure 7: Example 2 memory utilization plot.

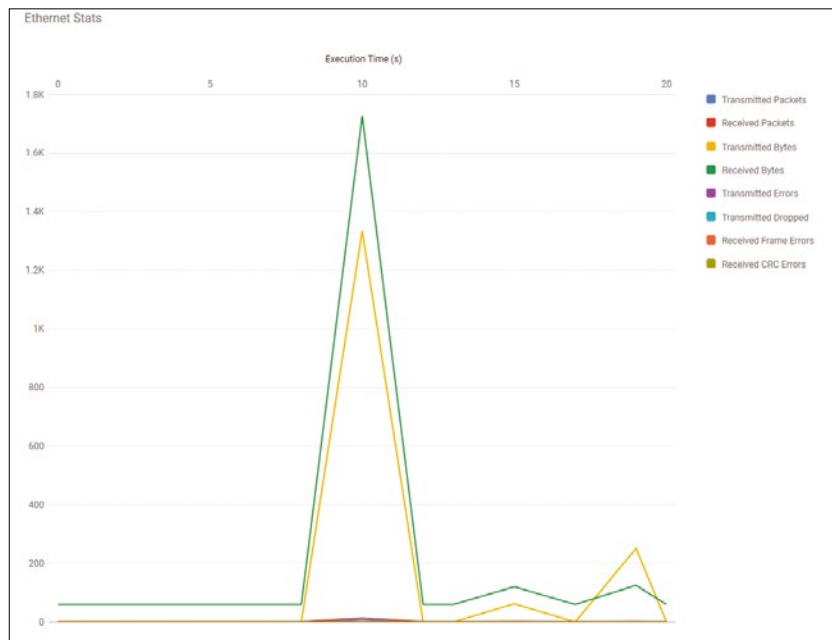


Figure 8: Example 2 Ethernet utilization plot.

Remora from TACC can gather this information for you and create plots to help guide you to a better understanding of your application without affecting its performance. Typically, the system administrator installs Remora, but users can install it in their accounts, as well.

Tuning the Remora installation is possible, particularly around what is monitored. Once installed, you just put the command `remora` before the command that runs the application, and you start gathering information. A few environment variables adjust how Remora gathers the data, but for the most part, it just silently gathers the data for you.

Remora is a great tool for users who want an idea of their application resource usage. Not pure profiling, Remora is really a combination of profiling and system monitoring. Easy to install and fairly light on resource usage, Remora can be a great help to users. ■

Info

- [1] Telemetry: [<https://www.multiformtech.co.uk/telemetry>]
- [2] Remora: [<https://github.com/TACC/remora>]
- [3] mpiP: [<https://github.com/LLNL/mpiP>]
- [4] Lustre: [<https://www.lustre.org>]
- [5] “Resource Monitoring For Remote Applications” by Jeff Layton, *HPC*, September 2017: [<https://www.admin-magazine.com/HPC/Articles/REMORA>]
- [6] Lmod: [<https://github.com/TACC/Lmod>]
- [7] POISSON_OPENMP: [https://people.sc.fsu.edu/~jburkardt/f_src/poisson_openmp/poisson_openmp.html]

The Author

Jeff Layton has been in the HPC business for almost 25 years (starting when he was 4 years old). He can be found lounging around at a nearby Frys enjoying the coffee and waiting for sales.



Linux Magazine is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!

Expand your Linux skills with:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

Subscribe now!

shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY
WITH OUR PDF
EDITION

Kafka: Scaling producers and consumers

Smooth

A guide to 10x scaling in Kafka with real-world metrics for high throughput, low latency, and cross-geographic data movement. By Jesse Yates

According to the [Kafka](#) home page [1], “Apache Kafka is an open-source distributed event streaming platform ... for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.”

A basic Kafka ecosystem has three components – producers, brokers, and consumers. Although much has been written about tuning brokers, reliably configuring producers and consumers is something of a dark art. All systems are dependent on your local setup, but some standard metrics that you can look at and knobs that can be tuned can increase performance 10 times or more. In this article, I walk through a real-life(-ish) example of how to diagnose and then fix a bottlenecked stream-processing system.

Assume you are mirroring data from an edge Kafka cluster into a central Kafka cluster that will feed your analytics data warehouse ([Figure 1](#)).

You’ve set up the edge with 100+ partitions for many of the topics you are consuming (because you had the forethought to expect scale and knew partitions are generally pretty cheap – go you!). That means you could easily be mirroring 1000+ partitions into your central Kafka for each edge cluster. In just this small data flow, you would need to consider a number of possible issues. Not only do you need to ensure your Kafka clusters are configured to scale (i.e., the number of partitions, the right tuning parameters) but also that the mirrors are scaled and tuned correctly. Many articles and posts have been written on tuning Kafka clusters, so I will focus on the client side, getting the producer and consumers (conveniently encapsulated in the data mirroring processing) tuned for high throughput, low latency, and cross-geographic data movement.

Edge Kafka clusters have some notable advantages – in particular, geographic fault isolation and reduced latencies to end users: User data can be sent faster to a secure replicated store (i.e., to the central Kafka cluster), which reduces the risk of critical data being lost. At the same time, you get a fallback for any local catastrophes (e.g., an earthquake in California), so your system remains available. However, you still need to get that data back to a central location for global analysis, and that means each of your remote mirrors has an extra 100ms of latency, roughly, for every mirror request you make. Chances are, this isn’t going to work out of the box. Too bad, so sad. Time to get engineering!

As you scroll through the logs of your misbehaving consumer, you might see something like the output in [Listing 1](#). If you turn on DEBUG logging, you might also see log lines, as in [Listing 2](#). Your consumer is trying to tell you: I didn’t get a response in the time I expected, so I’m giving up and trying again soon.

Here are some quick configurations to check:

- `default.api.timeout.ms`: In older client versions (pre 2.0) this controlled all the connection timeouts, and increasing it might be all you need to do.
- `session.timeout.ms`: The time until your consumer rebalances. Watch the `join-rate` for all consumers in the group – joining is the first step in rebalancing. High rates here indicate lots of rebalances, so increasing this timeout can help.

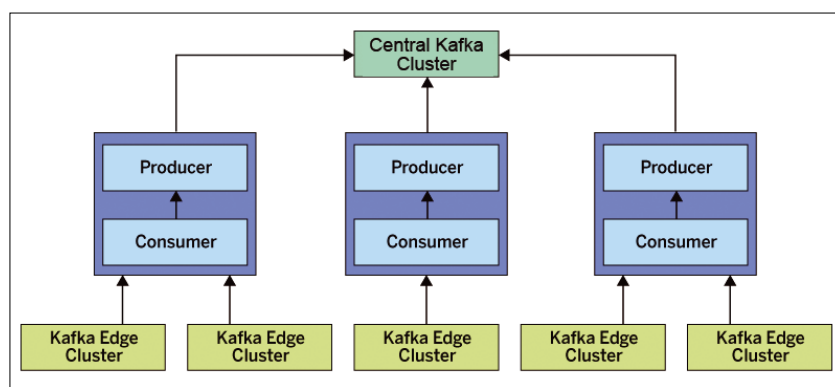


Figure 1: The data generated in the edge clusters travels by mirrors, on which consumers and producers convert the data and feed it into a central Kafka cluster.

- `request.timeout.ms`: The time the consumer will wait for a response (as of version 2.0). Logs are a great place to look to see whether you have a lot of failing fetches. Metrics to watch include the broker (`kafka.server:type=BrokerTopicMetrics,name=FailedFetchRequestPerSec`) for a gut check of fetch statuses and the client (`fetch-latency-avg` and `fetch-latency-max`) for latency when getting data.
- `fetch.max.wait.ms`: The time the server will block waiting for data to fill your response. Metrics to watch on the broker are `kafka.server:type=BrokerTopicMetrics,name=FailedFetchRequestPerSec` for a gut check of fetch statuses and `kafka.server:type=DelayedOperationPurgatory,delayedOperation=Fetch,name=PurgatorySize` for the number of fetch requests that are waiting (aka, stuck in purgatory). Metrics to watch on the client are `fetch-latency-avg` and `fetch-latency-max` for latency when getting data
- `fetch.min.bytes`: The minimum amount of data you want to fill your request. Client requests will wait on the broker up to this many bytes (or `fetch.max.wait.ms`, whichever comes first) before returning. Metrics to watch on the client, both at the consumer level and the topic level, are `fetch-size-avg` and `fetch-size-max` to see your fetch size distribution, `records-per-request-avg` for the number of messages you are getting per request, and `fetch-latency-avg` and `fetch-latency-max` to ensure this configuration is not causing you unexpected latency. Unless otherwise noted, all the metrics above are assumed to be client

(consumer)-side metrics MBeans and have the prefix

```
kafka.consumer:type=consumer-fetch-  
manager-metrics,client-id=(-.w)+)
```

or have `topic=(-.w)+`` for topic-scoped metrics.

These configurations can interact in interesting ways. For instance, if you tell the server to wait to fill the request (`fetch.max.wait.ms`) and have the timeout set short, you will have more retries but potentially better throughput and will have likely saved bandwidth for those high-volume topics/partitions. If you are running clusters in cloud providers, this can save you bandwidth exit costs and, thus, non-trivial amounts of money over time. Don't forget that whatever you were using when connecting to a geographically "more local" source cluster (i.e., not across the country) will probably stop working if you use the same settings to connect to a more distant cluster, because now you have an extra 50-100ms of roundtrip latency to bear. The default settings with 50ms timeouts for responses mean you will start to disconnect early all the time. Sadly, I cannot offer any exact advice that will always work for these settings. Instead, along with the associated metrics, these configurations are a good place to start fiddling and can give you a good understanding when reading the standard Kafka documentation.

Now say that you have tuned your timeouts up, made sure you are fetching at least 1 byte, and you are still getting these errors in the logs. It can be a challenge to pinpoint the exact problem; you might have 100+ consumer instances, and because they

work as a team, just one bad apple could tip you into perpetual rebalance storms.

To simplify the problem, turn down the number of instances and select only some of the topics you need to mirror. Eventually, you probably will get to a set of topics that suddenly starts to work. Hooray! Things are working magically. Maybe it was those tweaks you made to get the topics working? Time to scale it back up ... and it's broken again. Crapola. (This is exactly what happened to me recently.)

Did you remember to check your garbage collection (GC)? I bet you are going to find that your consumers are doing stop-the-world (STW) GC pauses for near or over your timeouts. Your one (or two or three) little mirrors are GCing themselves to death; every time they disconnect, they generate a bunch more objects, which then add GC pressure. Even if your mirror starts working, it can quickly churn garbage and spiral into a GC hole from which it never recovers. This case is even more frustrating because it can look like the mirror is running fine for 10 or 20 minutes, and then suddenly – BOOM! – it stops working.

I've found that using the Java GC command-line options

```
-server  
-XX:+UseParallelGC  
-XX:ParallelGCThreads=4
```

are more than sufficient to keep up. It doesn't use the fancy garbage first GC (G1GC), but for a simple mirror application, you don't need complex garbage collection – most objects are highly transient and the rest are small and very long lived. This setup is quite a nice fit for the "old" Java GC.

Listing 1: Misbehaving Consumer

```
2019-06-28 20:24:43 INFO [KafkaMirror-7] o.a.k.c.FetchSessionHandler:438 - [Consumer clientId=consumer-1, groupId=jesseyates.kafka.mirror] Error  
sending fetch request (sessionId=INVALID, epoch=INITIAL) to node 3: org.apache.kafka.common.errors.DisconnectException
```

Listing 2: DEBUG Logging

```
2019-06-27 20:43:06 DEBUG [KafkaMirror-11] o.a.k.c.c.i.Fetcher:244 - [Consumer clientId=consumer-1, groupId=jesseyates.kafka.mirror] Fetch READ_  
UNCOMMITTED at offset 26974 for partition source_topic-7 returned fetch data (error=NONE, highWaterMark=26974, lastStableOffset = -1, logStartOffset =  
0, abortedTransactions = null, recordsSizeInBytes=0)
```

Unbalanced Consumers

Consumers can become “unbalanced” and have some instances in the group with many partitions and others with none. But isn’t Kafka supposed to distribute partitions across consumers? A quick reading of the documentation would have you think that it should just evenly assign partitions across all the consumers, and it does – as long as you have the same number of partitions for all topics. As recently as Kafka 2.1 + (the latest stable release I’ve tested), as soon as you stop having the same number of partitions, the topic with the lowest number of partitions is used to determine the buckets, and then those buckets are distributed across nodes. For example, say you have two topics, one with 10 and another with 100 partitions, and 10 consumer instances. You start getting a lot of data coming into the 100-partition topic, so you turn up the number of consumers to 100, expecting to get 90 consumers with one partition and 10 consumers with two partitions (about as even as you can distribute 110 partitions). That is, one partition on 10 instances for each of topic 1 and then an even distribution of topic 2.

Unfortunately, this distribution is not what you see. Instead, you will end up with 10 consumers, each with 11 partitions and 90 consumers sitting idle. That’s the same distribution you had before, but now with extra overhead to manage the idle consumers. The client configuration you need is:

```
partitioner.class = org.apache.kafka.
clients.producer.RoundRobinPartitioner
```

Now the consumer group will assign the partitions by round robin across the entire consumer group, which will get you back to the distribution you expected, allowing you to balance load nicely and increase your overall throughput.

Tuning Producers

Now that you have the consumer side of your mirror pushing data quickly, you need to tune up the producing

side, as well, to push data to the central Kafka cluster as quickly as possible. Already you are in the 95th percentile of users, because generally the default client configurations are more than enough to work for most producers. To learn more about the internals of the producers, I recommend you look at a talk by Jiangie Qin [2]. Not only does his presentation walk you through how the producer works, it can give you some first-pass tuning recommendations. Still, I personally prefer a more empirical approach that is based on what clients say in terms of their metrics, which can then be used to optimize your particular use case.

Back to Basics

First, you need to understand why your producer is going slow, so the first question you need to ask is: Is it Kafka or is it me? Maybe it’s Kafka. Some metrics to check to ensure that the cluster is happy are:

- The network handler idle time,

```
kafka.network:type=SocketServer,name=
NetworkProcessorAvgIdlePercent
```

which is the idle time for the pool of threads that answer network requests and then pass them onto the request handler pool. I recommend folks to generally not go below 60%, with an average above 80%.

- The request handler idle time,

```
kafka.server:type=
KafkaRequestHandlerPool,name=
RequestHandlerAvgIdlePercent
```

which is the idle time for the pool of threads that process the network requests, doing everything from writing data to disk to serving read requests from disk. I recommend folks to generally not go below 60%, with the global average consistently above 70%.

- Disk utilization. If it’s high, the cluster likely is maxed out and cannot write/read any more data.
- CPU usage should not be high if the above metrics are happy, unless you are running other processes on the

server. Kafka uses a lot of thread pools, so that is likely to cause CPU contention and slow down the broker. Generally, try to avoid co-locating processes (especially processes with high CPU use) with brokers.

Unfortunately, in this convenient tuning investigation story, Kafka seems to be idling happily along, so you are left with tuning your client.

The obvious first starting place is ensuring that you have `compression.type` set on your producer. Compression on the producer side is seriously worth considering, especially if you have even a little bit of extra CPU available. Producer-side compression will help Kafka store more data quickly because the broker just writes the data to disk directly out of the socket (and vice versa for the consumer path), making it much more efficient for the whole pipeline (writing, storing, and reading) if the producer can just handle the compression up front. If you are running Kafka 2.X +, you should have access to `zstd` compression. Some tests I’ve seen show a marked improvement over the alternatives. It got close to `gzip` compression, but with the CPU overhead of `lz4`. Other tests I’ve run on my data show it can be significantly worse than other, already quite compressed formats. Your mileage may vary; be sure to test on your data.

The next thing you should check is the state of your batches. The easiest configuration to tweak is `linger.ms`. You can think of this as batching by time. By increasing latency, you can then increase your throughput by eliminating the overhead of extra network calls. Therefore, you should check out the `record-queue-time-avg` metrics, or the average time a batch waits in the send buffer (aka, how long it takes to fill a batch). If you are consistently below your `linger.ms`, you are filling your batch sizes. The first tweak is to increase latency so that you can (no surprise) increase the throughput, too, by increasing `linger.ms` (Note that Kafka defaults to not waiting for batches, leaning toward producing lower latency at the risk of more remote procedure calls). I find 5ms is a nice sweet spot.

The configurations used so far are then

```
compression.type=zstd
linger.ms = 5
```

Unfortunately in this convenient example, even after setting compression and tuning `linger.ms`, you are still not getting the throughput you need.

Going Deeper

Once you get further into producer tuning, the configurations start to get more interrelated, with some important non-linear and sometimes unexpected effects on performance. It pays to be extra patient – and scientific – about combinations of different parameters. Remember, you should be continually going back to understanding the root bottleneck while keeping an eye on optimizing the rate of records flowing through the producer. The next questions to ask are: How big are your records (as Kafka sees them, not as you think they are)? Are you making “good” batches? The size of the batch is determined by the `batch.size` configuration, which is the number of bytes after which the producer will send the request to the brokers, regardless of the `linger.ms` value. Requests sent to brokers will contain multiple batches – one for each partition.

A few other things you need to check include the number of records per batch and their size. Here is where you can start really digging into the `kafka.producer` MBean. The `batch-size-[avg|max]` can give you a good idea of the distribution of the number of bytes per batch, and `record-size-[avg|max]` can give you a sense of the size of each record. Divide the two and you have a rough rate of records per batch. Match this to the `batch.size` configuration and determine approximately how many records should be flowing through your producer. You should also sanity check this against the `record-send-rate` – the number of records per second – reported by your producer. You might be a bit surprised if you occasionally have very large mes-

Table 1: Producer Tuning Summary

Config/Metric	Comment
<code>compression.type</code>	Test on your data.
<code>linger.ms</code>	Check the average time a batch waits in the send buffer (how long it takes to fill a batch) with <code>record-queue-time-avg</code> .
<code>batch.size</code>	Determine records per batch, bytes per batch (<code>batch-size-avg</code> , <code>batch-size-max</code>), and records per topic per second (<code>record-send-rate</code>) and check your bytes per topic.
<code>max.request.size</code>	Limit the number and size of batches (<code>record-size-max</code>).
Time spent waiting for I/O	Are you really waiting (<code>io-wait-ratio</code>)?
<code>buffer.memory + queued requests</code>	32MB default (roughly total memory by producer) allocated to buffer records for sending (see <code>buffer-available-bytes</code>).

sages, for which you should check `record-size-max`, because the `max.request.size` configuration will limit the maximum size (in bytes) of a request and therefore inherently limit the number of record batches, as well. What about the time you are waiting for I/O? Check out the `io-wait-ratio` metrics to see where you are spending time. Is the I/O thread waiting or are your producers processing? Next, you need to make sure that the client buffer is not getting filled. Each producer has a fairly large buffer to collect data that then gets batched and sent to the brokers. In practice, I have never seen this to be a problem, but it often pays to be methodical. Here, the metric `buffer-available-bytes` is your friend, allowing you to ensure that your `buffer.memory` size is not being exhausted by your record sizes, batching configurations from earlier, or both. Producing too many different topics can affect the quality of compression, because you can’t compress well across topics. In that case, you might need some application changes so that you can batch more aggressively per destination topic, rather than relying on Kafka to just do the right thing. An advanced tactic would be to check the bytes-per-topic metrics from the producer, so you should only consider doing so after benchmarking and making sure other adjustments are not helping.

Wrap-Up

The configurations and metrics to tweak on the producer to get high throughput are summarized in [Table 1](#). At this point, you should have all

the tools you need to scale up your client instances. You know the most important optimization switches, some guidelines for adjusting garbage collection, and the nice round robin trick for balancing consumer groups when the consumers encounter differently partitioned topics. For slow producers, apply standard optimizations for compression and idle time or dive into the depths of the producer configuration to find out what really happens to entries and stacks. The tips in this article should give you a bit more guidance beyond the raw documentation in the Kafka manual for how to go about removing bottlenecks and getting the performance out of all parts of your streaming data pipelines that you know you should be getting. ■

Info

- [1] Apache Kafka: <https://kafka.apache.org/>
- [2] Producer performance tuning for Apache Kafka: <https://www.slideshare.net/JiangjieQin/producer-performance-tuning-for-apache-kafka-63147600>

Author

Jesse Yates is a Staff Engineer at Tesla and leads the development of real-time stream processing on the Data Plat-



form Team. Before joining Tesla, Jesse created a real-time IoT data startup, helped Salesforce build the first enterprise-grade Apache HBase installation, and consulted on healthcare, defense, and cross-cloud analytic projects. He is also a committer on Apache HBase and a Project Management Committee member of Apache Phoenix, for both big data storage and query projects. You can follow his writing at [\[jesseyates.com\]](https://jesseyates.com)

Rethinking RAID (on Linux)

Madam, I'm mdadm

Configure redundant storage arrays to boost overall data access throughput while maintaining fault tolerance. By Petros Koutoupis

Often, you find yourself attempting to eke out a bit more performance from the computer system you are attempting to either build or recycle, usually with limited funds at your disposal. Sure, you can tamper with the CPU or even memory settings, but if the I/O hitting the system needs to touch the underlying storage devices, those CPU tunings will make little to no difference.

In previous articles, I have shared methods by which one can boost write and read performance to

slower disk devices by leveraging both solid state drives (SSD) and dynamic random access memory (DRAM) as a cache [1]. This time, I will instead shift focus to a unique way you can configure redundant storage arrays so that you not only boost overall data access throughput but also maintain fault tolerance. The following examples center around a multiple-device redundant array of inexpensive (or independent) disks (MD RAID) in Linux and its userland utility mdadm [2].

Conventional wisdom has always dictated that spreading I/O load across more disk drives instead of bottlenecking a single drive does help significantly when increasing one's workload. For instance, if instead of writing to a single disk drive you split the I/O requests and write that same amount of data in a stripe across multiple drives (e.g., RAID0), you are reducing the amount of work that a single drive must perform to accomplish the same task. For magnetic spinning disks (i.e., hard disk drives, HDDs), the advantages should be more noticeable. The time it takes to seek across a medium introduces latency, and with randomly accessed

Listing 1: Random Write Test

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1 --runtime=60 --filename=/dev/sdc1 --rw=randwrite --numjobs=1 --name=test
test: (groupid=0, jobs=1): err=0: pid=3377: Sat Jan 9 15:31:04 2021
write: IOPS=352, BW=1410KiB/s (1444kB/s)(82.9MiB/60173msec); 0 zone resets
[ ... ]
Run status group 0 (all jobs):
WRITE: bw=1410KiB/s (1444kB/s), 1410KiB/s-1410KiB/s (1444kB/s-1444kB/s), io=82.9MiB (86.9MB), run=60173-60173msec

Disk stats (read/write):
sdc1: ios=114/21208, merge=0/0, ticks=61/1920063, in_queue=1877884, util=98.96%
```

I/O patterns, the I/O throughput suffers as a result on a single drive. A striped approach does not solve all the problems, but it does help a bit. In this article, I look at something entirely different. I spend more time focusing on increasing read throughput by way of RAID1 mirrors. In the first example, I discuss the traditional read balance in mirrored volumes (where read operations are balanced across both volumes in the mirrored set). The next examples are of read-preferred (or write-mostly) drives in a mirrored volume incorporating non-volatile media such as SSD or volatile media such as a ramdisk.

In my system, I have identified the following physical drives that I will be using in my examples:

```
$ cat /proc/partitions |grep 2
    -e sd[c,d] -e nvme
259      0  244198584 nvme0n1
259      2  244197543 nvme0n1p1
      8   32 6836191232 sdc
      8   33 244197544 sdc1
      8   48 6836191232 sdd
      8   49 244197544 sdd1
```

Notice that I have one non-volatile memory express (NVMe) drive and

Listing 3: Create a Mirrored Volume

```
$ sudo mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdc1 /dev/sdd1
mdadm: Note: this array has metadata at the start and
may not be suitable as a boot device. If you plan to
store '/boot' on this device please ensure that
your boot-loader understands md/v1.x metadata, or use
--metadata=0.90
Continue creating array? y
mdadm: Fail create md0 when using /sys/module/md_mod/parameters/new_array
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Listing 4: View RAID Status

```
$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5]
[raid4] [raid10]
md0 : active raid1 sdd1[1] sdc1[0]
      244065408 blocks super 1.2 [2/2] [UU]
      [=>.....] resync = 6.4% (15812032/244065408)
      finish=19.1min speed=198449K/sec
      bitmap: 2/2 pages [8KB], 65536KB chunk

unused devices: <none>
```

two serial-attached SCSI (SAS) drives. Later, I will introduce a ramdisk. Also notice that single partitions have been carved out in each drive that are approximately equal in size, which you will see is necessary when working with the RAID logic.

A quick random write test benchmark of one of the SAS volumes with the fio performance benchmarking utility can establish a baseline of both random write (Listing 1) and random read operations (Listing 2). The results show

Listing 2: Random Read Test

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1 --runtime=60 --filename=/dev/sdc1
--rw=randread --numjobs=1 --name=test
test: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=32
fio-3.12
Starting 1 process
Jobs: 1 (f=1): [r(1)][100.0%][r=1896KiB/s][r=474 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=3443: Sat Jan  9 15:32:51 2021
  read: IOPS=464, BW=1858KiB/s (1903kB/s)(109MiB/60099msec)
    [ ... ]
Run status group 0 (all jobs):
  READ: bw=1858KiB/s (1903kB/s), 1858KiB/s-1858KiB/s (1903kB/s-1903kB/s), io=109MiB (114MB),
        run=60099-60099msec

Disk stats (read/write):
sdc1: ios=27830/0, merge=0/0, ticks=1912861/0, in_queue=1856892, util=98.07%
```

that the single HDD has a throughput of 1.4MBps for random writes and 1.9MBps for random reads. To test a RAID1 mirror's read balance performance, I create a mirrored volume with the two HDDs identified earlier (Listing 3) and then view the status (Listing 4) and details (Listing 5) of the RAID volume.

Listing 5: View RAID Details

```
$ sudo mdadm --detail /dev/md0
/dev/md0:
    Version : 1.2
    Creation Time : Sat Jan  9 15:22:29 2021
    Raid Level : raid1
    Array Size : 244065408 (232.76 GiB 249.92 GB)
    Used Dev Size : 244065408 (232.76 GiB 249.92 GB)
    Raid Devices : 2
    Total Devices : 2
    Persistence : Superblock is persistent

    Intent Bitmap : Internal

    Update Time : Sat Jan  9 15:24:20 2021
    State : clean, resyncing
    Active Devices : 2
    Working Devices : 2
    Failed Devices : 0
    Spare Devices : 0

    Consistency Policy : bitmap

    Resync Status : 9% complete

    Name : dev-machine:0 (local to host dev-machine)
    UUID : a84b0db5:8a716c6d:ce1e9ca6:8265de17
    Events : 22

    Number Major Minor RaidDevice State
      0      8     33        0  active sync  /dev/sdc1
      1      8     49        1  active sync  /dev/sdd1
```

You will immediately notice that the array is initializing the disks and zeroing out the data on each to bring it all to a good state. You can definitely use

it in this state, but it will affect overall performance. Also, you probably do not want to disable the initial resync of the array with the `--assume-clean`

option. Even if the drives are new out of the box, it is better to know that your array is in a proper state before writing important data to it. This process will definitely take a while, and the bigger the array, the longer the initialization process. Before proceeding with follow-up benchmarking tests, you should wait until volume synchronization completes.

Next, verify that the mirror initialization has completed,

```
$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid1 sdd1[1] sdc1[0]
      244065408 blocks super 1.2 [2/2] [UU]
      bitmap: 1/2 pages [4KB], 65536KB chunk

unused devices: <none>
```

and repeat the random write and read tests from before, but this time to the RAID volume (e.g., `/dev/md0`). Remember, the first random writes were 1.4MBps and random reads 1.9MBps. The good news is that whereas random writes dropped a tiny bit to 1.2MBps ([Listing 6](#)), random reads increased to almost double the throughput with a rate of 3.3MBps ([Listing 7](#)).

NVMe

Now I will introduce NVMe into the mix – that is, two drives: one NVMe and one HDD in the same mirror. The `mdadm` utility offers a neat little feature that can be leveraged with the `--write-mostly` argument, which translates to: Use the following drives for write operations only and not read operations (unless a drive failure were to occur on the volumes designated for read operations).

To begin, create the RAID volume ([Listing 8](#)). Next, view the RAID volume's details and pay particular attention to the drive labeled *write-mostly* ([Listing 9](#)).

Then, repeat the same `fio` tests by executing the random write test

Listing 6: Random Write to RAID

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1 --runtime=60 --filename=/dev/md0
--rw=randwrite --numjobs=1 --name=test
test: (groupid=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=32
fio-3.12
Starting 1 process
Jobs: 1 (f=1): [w(1)][100.0%][w=1200KiB/s][w=320 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=3478: Sat Jan 9 15:46:11 2021
write: IOPS=308, BW=1236KiB/s (1266kB/s)(72.5MiB/60102msec); 0 zone resets
[ ... ]
Run status group 0 (all jobs):
WRITE: bw=1236KiB/s (1266kB/s), 1236KiB/s-1236KiB/s (1266kB/s-1266kB/s), io=72.5MiB (76.1MB),
run=60102-60102msec

Disk stats (read/write):
md0: ios=53/18535, merge=0/0, ticks=0/0, in_queue=0, util=0.00%, aggrrios=33/18732, aggrmerge=0/0,
aggrticks=143/1173174, aggrin_queue=1135748, aggrutil=96.50%
sdd: ios=13/18732, merge=0/0, ticks=93/1123482, in_queue=1086112, util=96.09%
sdc: ios=54/18732, merge=0/0, ticks=194/1222866, in_queue=1185384, util=96.50%
```

Listing 7: Random Read from RAID

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1 --runtime=60 --filename=/dev/md0
--rw=randread --numjobs=1 --name=test
test: (groupid=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=32
fio-3.12
Starting 1 process
Jobs: 1 (f=1): [r(1)][100.0%][r=3184KiB/s][r=796 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=3467: Sat Jan 9 15:44:42 2021
read: IOPS=806, BW=3226KiB/s (3303kB/s)(189MiB/60061msec)
[ ... ]
Run status group 0 (all jobs):
READ: bw=3226KiB/s (3303kB/s), 3226KiB/s-3226KiB/s (3303kB/s-3303kB/s), io=189MiB (198MB),
run=60061-60061msec

Disk stats (read/write):
md0: ios=48344/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%, aggrrios=24217/0, aggrmerge=0/0,
aggrticks=959472/0, aggrin_queue=910458, aggrutil=96.15%
sdd: ios=24117/0, merge=0/0, ticks=976308/0, in_queue=927464, util=96.09%
sdc: ios=24318/0, merge=0/0, ticks=942637/0, in_queue=893452, util=96.15%
```

Listing 8: Create the RAID Volume

```
$ sudo mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/nvme0n1p1 --write-mostly /dev/sdc1
mdadm: Note: this array has metadata at the start and
may not be suitable as a boot device. If you plan to
store '/boot' on this device please ensure that
your boot-loader understands md/v1.x metadata, or use
--metadata=0.90
mdadm: /dev/sdc1 appears to be part of a raid array:
level=raid1 devices=2 ctime=Sat Jan 9 15:22:29 2021
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Listing 9: View the RAID Volume

```
$ sudo mdadm --detail /dev/md0
/dev/md0:
    Version : 1.2
    Creation Time : Sat Jan  9 15:52:00 2021
    Raid Level : raid1
    Array Size : 244065408 (232.76 GiB 249.92 GB)
    Used Dev Size : 244065408 (232.76 GiB 249.92 GB)
    Raid Devices : 2
    Total Devices : 2
    Persistence : Superblock is persistent

    Intent Bitmap : Internal

    Update Time : Sat Jan  9 15:52:21 2021
    State : clean, resyncing
    Active Devices : 2
    Working Devices : 2
    Failed Devices : 0
    Spare Devices : 0

    Consistency Policy : bitmap

    Resync Status : 1% complete

    Name : dev-machine:0 (local to host dev-machine)
    UUID : 833033c5:cd9b78de:992202ee:cb1bf77f
    Events : 4

    Number Major Minor RaidDevice State
       0   259     2     0      active sync  /dev/nvme0n1p1
       1     8    33     1      active sync writemostly /dev/sdc1
```

(Listing 10) and the random read test (Listing 11).

Wow. Although the result is a very small increase in write operations because of the NVMe (up to 1.4MBps), look at the random reads. In the case of RAID, you are only as fast as your slowest drive, which is why speeds are hovering around the original baseline of 1.3MBps. The original single HDD benchmark for random reads was 1.9MBps, and the read-balanced mirrored HDDs saw 3.3MBps. Here, with the NVMe volume set as the drive from which to read, speeds are a whopping 715MBps! I wonder if it can be better?

Ramdisk

What would happen if I introduced a ramdisk into the picture? That is, I want to boost read operations but also persist the data after reboots of the system. This process should not be confused with caching. The data is *not* being staged on the ramdisk temporarily before being persisted into a backing store.

In the next example, the ramdisk will be treated like a backing store, even though the volatile medium technically isn't one. Before I proceed, I will need to carve out a partition roughly the same size as the ramdisk. I have chosen a meager 2GB because the older system I am currently using does not have much installed to begin with:

Listing 10: Random Write with NVMe

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1 --runtime=60 --filename=/dev/md0
--rw=randwrite --numjobs=1 --name=test
test: (g=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=32
fio-3.12
Starting 1 process
Jobs: 1 (f=1): [w(1)][100.0%][w=1441KiB/s][w=360 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=3602: Sat Jan  9 16:14:10 2021
    write: IOPS=342, BW=1371KiB/s (1404kB/s)(80.5MiB/60145msec); 0 zone resets
    [ ... ]
Run status group 0 (all jobs):
    WRITE: bw=1371KiB/s (1404kB/s), 1371KiB/s-1371KiB/s (1404kB/s-1404kB/s), io=80.5MiB (84.4MB),
    run=60145-60145msec

Disk stats (read/write):
    md0: ios=100/20614, merge=0/0, ticks=0/0, in_queue=0, util=0.00%, aggrrios=103/20776, aggrmerge=0/0,
    aggrticks=12/920862, aggrin_queue=899774, aggrutil=97.47%
    nvme0n1: ios=206/20776, merge=0/0, ticks=24/981, in_queue=40, util=95.01%
    sdc: ios=0/20776, merge=0/0, ticks=0/1840743, in_queue=1799508, util=97.47%
```

Listing 11: Random Read with NVMe

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1 --runtime=60 --filename=/dev/
md0 --rw=randread --numjobs=1 --name=test
test: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio,
    iodepth=32
fio-3.12
Starting 1 process
Jobs: 1 (f=1): [r(1)][100.0%][r=678MiB/s][r=173k IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=3619: Sat Jan  9 16:14:53 2021
    read: IOPS=174k, BW=682MiB/s (715MB/s)(10.0GiB/15023msec)
    [ ... ]
Run status group 0 (all jobs):
    READ: bw=682MiB/s (715MB/s), 682MiB/s-682MiB/s (715MB/s-715MB/s), io=10.0GiB (10.7GB),
    run=15023-15023msec

Disk stats (read/write):
    md0: ios=2598587/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%, aggrrios=1310720/0, aggrmerge=0/0,
    aggrticks=25127/0, aggrin_queue=64, aggrutil=99.13%
    nvme0n1: ios=2621440/0, merge=0/0, ticks=50255/0, in_queue=120, util=99.13%
    sdc: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
```

```
$ cat /proc/partitions |grep -e sd[c,d]
8      48 6836191232 sdd
8      49 244197544  sdd1
8      50   2097152  sdd2
8      32 6836191232 sdc
8      33 244197544  sdc1
8      34   2097152  sdc2
```

Now to add the ramdisk. As a prerequisite, you need to ensure that the jansson development library is installed on your local machine. Clone the rapiddisk Git repository [3], build the package, install it,

```
$ git clone https://github.com/2
pkoutoupis/rapiddisk.git
$ cd rapiddisk/
$ make
$ sudo make install
```

insert the kernel modules,

```
$ sudo modprobe rapiddisk
$ sudo modprobe rapiddisk-cache
```

verify that the modules are installed,

```
$ lsmod|grep rapiddisk
rapiddisk_cache      20480  0
rapiddisk            20480  0
```

create a single 2GB ramdisk,

```
$ sudo rapiddisk --attach 2048
rapiddisk 6.0
Copyright 2011 - 2019 Petros Koutoupis
```

Attached device rd0 of size 2048 Mbytes

verify that the ramdisk has been created,

```
$ sudo rapiddisk --list
rapiddisk 6.0
Copyright 2011 - 2019 Petros Koutoupis
List of RapidDisk device(s):
  RapidDisk Device 1: rd0 Size (KB): 2097152
List of RapidDisk-Cache mapping(s):
  None
```

and create a mirrored volume with the ramdisk as the primary and one of the smaller HDD partitions set to write-mostly (Listing 12). Now, verify the RAID1 mirror state:

```
$ cat /proc/mdstat
Personalities : [linear] [multipath] 2
  [raid0] [raid1] [raid6] [raid5] 2
  [raid4] [raid10]
md0 : active raid1 sdc2[1](W) rd0[0]
      2094080 blocks super 1.2 [2/2] [UU]
```

unused devices: <none>

The initialization time should be relatively quick here. Also, verify the RAID1 mirror details (Listing 13) and rerun the random write I/O test (Listing 14). As you saw with the NVMe drive earlier, you also see a small bump in random write operations at approximately 1.6MBps. Remember that you are only as fast as your slowest disk (i.e., the HDD paired with the ramdisk in the mirrored set). Now, run the random read test (Listing 15). I know I said wow before, but, Wow. Random reads are achieving greater than 1GBps throughput because it is literally only hitting RAM. On faster systems with faster memory, this number should be much larger.

Listing 12: Create Mirrored Volume

```
$ sudo mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/rd0 --write-mostly /dev/sdc2
mdadm: /dev/rd0 appears to be part of a raid array:
      level=raid1 devices=2 ctime=Sat Jan  9 16:32:35 2021
mdadm: Note: this array has metadata at the start and
      may not be suitable as a boot device.  If you plan to
      store '/boot' on this device please ensure that
      your boot-loader understands md/v1.x metadata, or use
      --metadata=0.90
mdadm: /dev/sdc2 appears to be part of a raid array:
      level=raid1 devices=2 ctime=Sat Jan  9 16:32:35 2021
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Listing 13: Verify RAID1 Mirror Details

```
$ sudo mdadm --detail /dev/md0
/dev/md0:
      Version : 1.2
  Creation Time : Sat Jan  9 16:32:43 2021
    Raid Level : raid1
    Array Size : 2094080 (2045.00 MiB 2144.34 MB)
  Used Dev Size : 2094080 (2045.00 MiB 2144.34 MB)
    Raid Devices : 2
   Total Devices : 2
 Persistence : Superblock is persistent

 Update Time : Sat Jan  9 16:32:54 2021
   State : clean
 Active Devices : 2
 Working Devices : 2
 Failed Devices : 0
 Spare Devices : 0

Consistency Policy : resync

      Name : dev-machine:0 (local to host dev-machine)
     UUID : 79387934:aaaad032:f56c6261:de230a86
    Events : 17

   Number Major Minor RaidDevice State
      0     252      0        0 active sync  /dev/rd0
      1       8     34        1 active sync writemostly  /dev/sdc2
```

Listing 14: Ramdisk Random Write

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1
--runtime=60 --filename=/dev/md0 --rw=randwrite --numjobs=1 --name=test
test: (g=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T)
4096B-4096B, ioengine=libaio, iodepth=32
fio-3.12
Starting 1 process
Jobs: 1 (f=1): [w(1)][100.0%][w=1480KiB/s][w=370 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=5854: Sat Jan 9 16:34:44 2021
write: IOPS=395, BW=1581KiB/s (1618kB/s)(92.9MiB/60175msec); 0 zone
resets
[ ... ]
Run status group 0 (all jobs):
WRITE: bw=1581KiB/s (1618kB/s), 1581KiB/s-1581KiB/s (1618kB/s-1618kB/s),
io=92.9MiB (97.4MB), run=60175-60175msec

Disk stats (read/write):
md0: ios=81/23777, merge=0/0, ticks=0/0, in_queue=0, util=0.00%,
aggrios=0/11889, aggrmerge=0/0, aggrticks=0/958991, aggrin_
queue=935342, aggrutil=99.13%
sdc: ios=0/23778, merge=0/0, ticks=0/1917982, in_queue=1870684,
util=99.13%
rd0: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
```

Listing 15: Ramdisk Random Read

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1
--runtime=60 --filename=/dev/md0 --rw=randread --numjobs=1 --name=test
test: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T)
4096B-4096B, ioengine=libaio, iodepth=32
fio-3.12
Starting 1 process
Jobs: 1 (f=1)
test: (groupid=0, jobs=1): err= 0: pid=5872: Sat Jan 9 16:35:08 2021
read: IOPS=251k, BW=979MiB/s (1026MB/s)(2045MiB/2089msec)
[ ... ]
Run status group 0 (all jobs):
READ: bw=979MiB/s (1026MB/s), 979MiB/s-979MiB/s (1026MB/s-1026MB/s),
io=2045MiB (2144MB), run=2089-2089msec

Disk stats (read/write):
md0: ios=475015/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%,
aggrios=0/0, aggrmerge=0/0, aggrticks=0/0, aggrin_queue=0,
aggrutil=0.00%
sdc: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
rd0: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
```

This setup has a problem, though. It has only a single persistent (or non-volatile) volume in the mirror, and if that drive were to fail, only the volatile memory volume would be left. Also, if you reboot the system, you are in a degraded mode and reading solely from the HDD – until you recreate the ramdisk and rebuild the mirror, that is (which can be accomplished with simple Bash scripts on bootup). How do you address this problem? A simple solution would be to add a second persistent volume into the mirror, creating a three-copy RAID1 array.

If you recall in my earlier example, I created a 2GB partition on the second volumes that can be configured with `mdadm` (Listing 16). When you verify the details (Listing 17), notice that both the HDDs are set to *writemostly*. Once the volume completes its initialization, do another run of `fio`

benchmarks and execute the random write test (Listing 18) and the random read test (Listing 19). The random writes are back down a bit

Listing 16: Config 2GB Partition

```
$ sudo mdadm --create /dev/md0 --level=1 --raid-devices=3
/dev/rd0 --write-mostly /dev/sdc2 /dev/sdd2
mdadm: /dev/rd0 appears to be part of a raid array:
level=raid1 devices=2 ctime=Sat Jan 9 16:32:43 2021
mdadm: Note: this array has metadata at the start and
may not be suitable as a boot device. If you plan to
store '/boot' on this device please ensure that
your boot-loader understands md/v1.x metadata, or use
--metadata=0.90
mdadm: /dev/sdc2 appears to be part of a raid array:
level=raid1 devices=2 ctime=Sat Jan 9 16:32:43 2021
mdadm: /dev/sdd2 appears to be part of a raid array:
level=raid1 devices=2 ctime=Sat Jan 9 16:32:43 2021
Continue creating array? (y/n) y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Listing 17: Verify 2GB Details

```
$ sudo mdadm --detail /dev/md0
/dev/md0:
    Version : 1.2
  Creation Time : Sat Jan 9 16:36:18 2021
    Raid Level : raid1
    Array Size : 2094080 (2045.00 MiB 2144.34 MB)
  Used Dev Size : 2094080 (2045.00 MiB 2144.34 MB)
    Raid Devices : 3
    Total Devices : 3
 Persistence : Superblock is persistent

    Update Time : Sat Jan 9 16:36:21 2021
      State : clean, resyncing
    Active Devices : 3
    Working Devices : 3
    Failed Devices : 0
    Spare Devices : 0

Consistency Policy : resync

    Resync Status : 23% complete

    Name : dev-machine:0 (local to host dev-machine)
    UUID : e0e5d514:d2294825:45d9f09c:db485a0c
    Events : 3

   Number Major Minor RaidDevice State
     0     252      0      0     active sync  /dev/rd0
     1       8     34      1     active sync writemostly /dev/sdc2
     2       8     50      2     active sync writemostly /dev/sdd2
```

Listing 18: 2GB Random Write

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1 --runtime=60 --filename=/dev/
md0 --rw=randwrite --numjobs=1 --name=test
test: (g=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio,
iodepth=32
fio-3.12
Starting 1 process
Jobs: 1 (f=1): [w(1)][100.0%][w=1305KiB/s][w=326 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=5941: Sat Jan 9 16:38:30 2021
write: IOPS=325, BW=1301KiB/s (1333kB/s)(76.4MiB/60156msec); 0 zone resets
[ ... ]
Run status group 0 (all jobs):
WRITE: bw=1301KiB/s (1333kB/s), 1301KiB/s-1301KiB/s (1333kB/s-1333kB/s), io=76.4MiB (80.2MB),
run=60156-60156msec

Disk stats (read/write):
md0: ios=82/19571, merge=0/0, ticks=0/0, in_queue=0, util=0.00%, aggrrios=0/13048, aggrmerge=0/0,
aggrticks=0/797297, aggrin_queue=771080, aggrutil=97.84%
sdd: ios=0/19572, merge=0/0, ticks=0/1658959, in_queue=1619688, util=93.01%
sdc: ios=0/19572, merge=0/0, ticks=0/732934, in_queue=693552, util=97.84%
rd0: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
```

Listing 19: 2GB Random Read

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=10g --direct=1 --runtime=60 --filename=/dev/
md0 --rw=randread --numjobs=1 --name=test
test: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio,
iodepth=32
fio-3.12
Starting 1 process
Jobs: 1 (f=1)
test: (groupid=0, jobs=1): err= 0: pid=5956: Sat Jan 9 16:38:53 2021
read: IOPS=256k, BW=998MiB/s (1047MB/s)(2045MiB/2049msec)
[ ... ]
Run status group 0 (all jobs):
READ: bw=998MiB/s (1047MB/s), 998MiB/s-998MiB/s (1047MB/s-1047MB/s), io=2045MiB (2144MB),
run=2049-2049msec

Disk stats (read/write):
md0: ios=484146/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%, aggrrios=0/0, aggrmerge=0/0,
aggrticks=0/0, aggrin_queue=0, aggrutil=0.00%
sdd: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
sdc: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
rd0: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
```

to 1.3Mbps as a result of writing to the extra HDD and the additional latencies introduced by the mechanical drive.

Notice that the 1GBps random read throughput is still maintained, now with the security of an extra volume for protection in the event of a drive failure. However, you will still need to recreate the ramdisk and rebuild the mirrored set on every reboot.

Conclusion

As you can see, you can rely on age-old concepts such as RAID technologies to give you a boost of performance in your computing environments – and without relying on a temporary cache. In some cases, you can breathe new life into older hardware.

Info

- [1] “Tuning ZFS for Speed on Linux” by Petros Koutoupis, *ADMIN*, 57, 2020, pp. 44-46
- [2] mdadm(8): [<https://www.man7.org/linux/man-pages/man8/mdadm.8.html>]
- [3] The RapidDisk Project: [<https://github.com/pkoutoupis/rapiddisk>]

The Author

Petros Koutoupis is a senior performance software engineer at Cray (now HPE) for its Lustre High Performance File System division. He is also the creator and maintainer of the Rapid-Disk Project (www.rapiddisk.org). Petros has worked in the data storage industry for well over a decade and has helped pioneer many technologies unleashed in the wild today.

ADMIN

Network & Security

NEWSSTAND

Order online:
bit.ly/ADMIN-Newsstand

ADMIN is your source for technical solutions to real-world problems. Every issue is packed with practical articles on the topics you need, such as: security, cloud computing, DevOps, HPC, storage, and more! Explore our full catalog of back issues for specific topics or to complete your collection.

#61/January/February 2021

Secure Containers

Security is the watchword this issue, and we begin with eliminating container security concerns.

On the DVD: Clonezilla Live 2.7.0

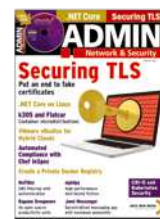


#60/November/December 2020

Securing TLS

In this issue, we look at ASP.NET Core, a web-development framework that works across OS boundaries.

On the DVD: Ubuntu Server Edition 20.10



#59/September/October 2020

Custom MIBs

In this issue, learn how to create a Management Information Base module for hardware and software.

On the DVD: CentOS 8.2.2004

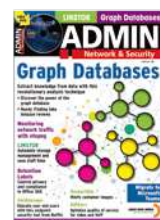


#58/July/August 2020

Graph Databases

Discover the strengths of graph databases and how they work, and follow along with a Neo4j example.

On the DVD: Fedora 32 Server (Install Only)



#57/May/June 2020

Artificial Intelligence

We look at the progress and application of artificial intelligence, deep and machine learning, and neural networks.

On the DVD: Ubuntu Server 20.04 LTS



#56/March/April 2020

Secure DNS

In this issue, we look at solutions for encrypted DNS, so admins can block domains that distribute malware.

On the DVD: Kali Linux 2020.1 (Live)



WRITE FOR US

Admin: Network and Security is looking for good, practical articles on system administration topics. We love to hear from IT professionals who have discovered innovative tools or techniques for solving real-world problems.

Tell us about your favorite:

- interoperability solutions
- practical tools for cloud environments
- security problems and how you solved them
- ingenious custom scripts

- unheralded open source utilities
- Windows networking techniques that aren't explained (or aren't explained well) in the standard documentation.

We need concrete, fully developed solutions: installation steps, configuration files, examples – we are looking for a complete discussion, not just a “hot tip” that leaves the details to the reader.

If you have an idea for an article, send a 1-2 paragraph proposal describing your topic to: edit@admin-magazine.com.



Authors

Chris Binnie	30
Oliver Frommel	54
Oliver Gutperl	16
Ken Hess	3
Thomas Joos	50
Petros Koutoupis	90
Jeff Layton	80
Martin Loschwitz	10, 44
Christoph Mitasch	62
Benjamin Pfister	38
Thorsten Scherf	58
Tim Schürmann	22
Evgenij Smirnov	74
Jack Wallen	8
Stefan Wintermeyer	26
Matthias Wübbeling	68, 72
Jesse Yates	86

Contact Info

Editor in Chief

Joe Casad, jcasad@linuxnewmedia.com

Managing Editors

Rita L Sooby, rsooby@linuxnewmedia.com
Lori White, lwhite@linuxnewmedia.com

Senior Editor

Ken Hess

Localization & Translation

Ian Travis

News Editor

Jack Wallen

Copy Editors

Amy Pettie, Megan Phelps

Layout

Dena Friesen, Lori White

Cover Design

Lori White, Illustration based on graphics by
vn19ko93, 123RF.com

Advertising

Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Publisher

Brian Osborn

Marketing Communications

Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription

For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2021 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by hofmann infocore GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN (ISSN 2045-0702) is published bimonthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. March/April 2021. Periodicals Postage paid at Lawrence, KS. Ride-Along Enclosed. POSTMASTER: Please send address changes to ADMIN, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Published in Europe by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.



Powerful business workhorse

TUXEDO Aura 15



AMD Ryzen 7 4700U
8 Cores | 15 Watt



USB-C 3.2 Gen2
DisplayPort & PD



2 cm | 1,65 kg
slim & light



4G / LTE
Cellular Modem



100%
Linux

5

Year
Warranty



Lifetime
Support



Built in
Germany



German
Privacy



Local
Support

TUXEDO
COMPUTERS

 tuxedocomputers.com

T A U R I A

Stop Your Work From Being Spied On



Unlike the other apps, only you have the encryption keys. Tauria has true and full end-to-end encryption.



The Only Encrypted All-In-One Collaboration Platform



Scan to learn more about Tauria



Calls

Encrypted Video Conferencing



Chats

Confidential Messaging



Files

Fully Secured Cloud Storage



Calendar

Truly Private Scheduling



Tauria meets & exceeds worldwide security and privacy standards.



A Dutch-Canadian Company

info@tauria.com
WWW.TAURIA.COM

