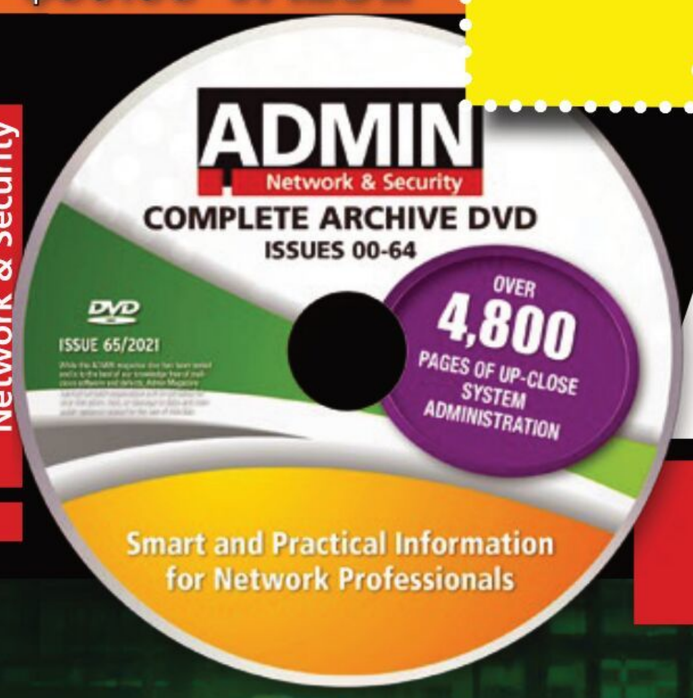
ARCHIVE DVD

HUGE SAVINGS! $39.90 VALUE

COMPLETE ARCHIVE

12 Years of ADMIN at Your Fingertips

4,800 PAGES ON A SEARCHABLE DISC



ADMIN Network & Security

ADMIN

Network & Security

ISSUE 65

7 Email Clients

Ergonomics, security, and extensibility

MULTICLOUD ANSIBLE ROLLOUTS

Independence from your cloud provider

Kubernetes persistent storage management

App Proxy

Flexible working environments with RDS

acme.sh

A lightweight client for the ACME protocol

StackStorm

Automate complex IT infrastructures

QUMBU

Backup and maintenance for MS SQL Server

Darshan

I/O analysis for DL frameworks

MicroK8s

Zero-ops K8s

LINUX NEW MEDIA
The Pulse of Open Source

Meet Me in St. Louis.

HPC is fueling breakthroughs across industries in science and beyond, from academia to commercial enterprises, by harnessing data to unlock insights faster and more accurately than ever before. HPC powers new capabilities in artificial intelligence and machine learning that, combined with modeling and simulation, accelerate discovery in solving our toughest challenges.

Join us in St. Louis to learn how HPC is empowering innovations that were never before possible to improve everyday life across the globe. Register early and save!

Program	Exhibits
14–19	15–18
NOVEMBER	NOVEMBER

The International Conference for High Performance Computing, Networking, Storage, and Analysis

SC21
St. Louis, MO | science & beyond.

Register Today!

sc21.supercomputing.org

Sponsored by: acm | sighpc | IEEE COMPUTER SOCIETY | TCHPC

Keeping Up with the Times

A recipe for relevancy: Sharpen your skills, gain expertise, and practice efficiency.

Every week that goes by, I read about some new technology, new device, new app, new security patch, or new something else for me to try. The speed at which developers create new or improved products is at such a pace that I find it difficult to keep up. I just don't have enough hours in a week to try out every new thing that grabs my attention. It's important to maintain one's "edge" by reading, installing, breaking – I mean testing, and exploring the newest gadgets and the latest and greatest things that someone dishes out. Plus, I still need to devote time to my actual day job, a few games, watching Jeopardy, a bit of cooking, family time, and enjoying a few hobbies such as filmmaking and podcasting.

So, my question is: "How does one keep up with this rapidly changing landscape of apps, gadgets, and life?"

My short answer is to focus on a few specific things yourself and then rely on experts for everything else. Is it the perfect plan? No. Is it manageable? Yes, for me, it is. I must draw the line somewhere because I can't spend every waking hour studying peripheral technologies, and by *peripheral* I mean something that's not in my direct line of sight. I enjoy knowing a little something about a lot of things, but sometimes I must apply a filter on the number of what's covered by "a lot of things."

I admit that knowing a lot of different things kept me employed through habitual layoffs during the time period between 2001 and 2016. I didn't love seeing hundreds of my coworkers walking out the door time and time again over that stretch. I was always happy when my manager would announce to our team that, "There was a layoff event today, but now it's over. If you're still here, you still have a job – for the moment." Not comforting, but I could take a deep breath of relief for having a job for another month.

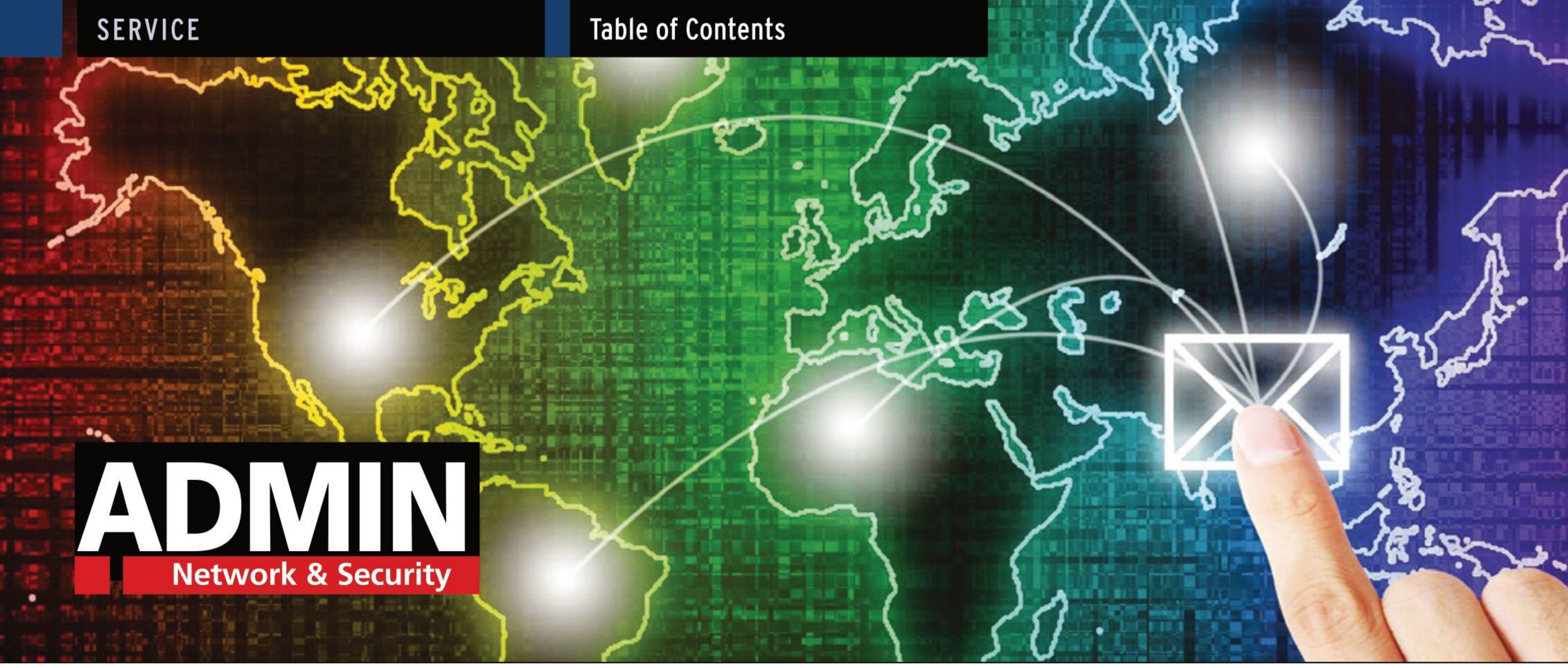
During these "events," I kept my skills sharp, expanded my value by learning new things, and I even started focusing on making suggestions to streamline, cut back, save money, and do more with less. It must have helped. I stayed employed until I left by my own choice. I spent a lot of time hustling, learning, growing, and trying to keep up with every new technology, every security vulnerability, and every new piece of software that I could download, install, and explore.

During one of the "salary leveling" events, I was glad I had many skills to offer my employer. Looking back, I don't know if I was lucky or foolish, because my salary didn't change for the worse or the better, when many of my colleagues took as much as a 33 percent pay cut to retain their jobs. I decided to ignore everything that was going on around me and focus on taking all the online training for which I was eligible. I felt increasing pressure to stay ahead of the curve and to avoid "pigeonholing" myself by becoming too narrow in my career scope.

My perseverance and expanded training paid off. I began to reap the rewards of a decade of hard work in keeping pace with new technologies by receiving raises, bonuses, and other perks. My career finally began to blossom into something I could be proud of and and garner the salary I thought I'd deserved for years. I was also given more responsibility as a technical team leader, and I had my own high-profile projects to manage. I managed to turn things around for myself by extending my learning and keeping up with the times. Remember that you are not always going to see a payoff right away when you do something in technology, but eventually you will.

Ken Hess • ADMIN Senior Editor

Lead Image © Somchai Suppalertporn, 123RF.com



ADMIN Network & Security

Features

The features in this issue tackle digital certificates, email clients, and HP backup strategies.

10 acme.sh ACME Client
The ACME protocol facilitates digital certificates for secure TLS communication channels.

12 Seven Email Clients Tested
We explore seven graphical email clients by investigating ergonomics, security, and extensibility.

20 High-Performance Backups
A secure backup strategy ensures you can back up and restore your data rapidly and reliably.

News

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

8 News
- CloudLinux rescues CentOS 8 from vanishing support
- CentOS replacement, AlmaLinux, available on Azure
- Ubuntu Linux certified for secure and regulated workloads
- Kubernetes 1.22 released with 56 enhancements

Tools

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

24 Ceph Dashboard
A visual overview of cluster health and baseline maintenance tasks; an alerting function can be added, too.

30 Darshan
Characterizing and understanding I/O patterns in the TensorFlow machine learning framework.

36 QUMBU for SQL Server
Even database administrators with little experience can perform straightforward SQL Server backups and maintenance checks.

42 Automation with StackStorm
StackStorm is an open source, event-based platform for runbook automation.

48 MicroK8s
A zero-ops installation of Kubernetes operates on almost no compute capacity and roughly 700MB of RAM.

Containers and Virtualization

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

52 OPA and Gatekeeper
Enforce container compliance in Kubernetes.

58 Persistent Container Storage
CSI-compliant plugins connect their systems to Kubernetes and other orchestrated container environments.

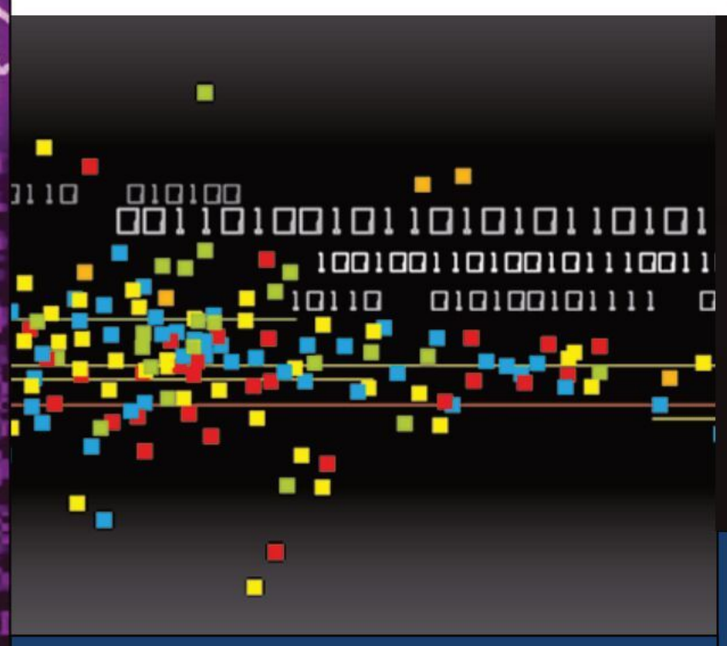
Security

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

64 PKI in the Cloud
Secure digital communication maintains the security of an on-premises solution and reduces complexity.

66 Microsoft Security Boundaries
We look at security boundaries and protection goals and their interpretation of the different security areas of Windows operating systems and components.

68 Watchtower
Automatically update software in the Docker universe.



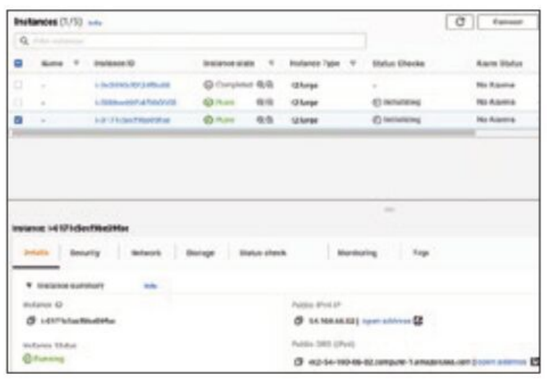
20 **High-Performance Backup**
Back up and restore data quickly and reliably with a coherent backup strategy, coupled with a good combination of hardware and software.

52 **Open Policy Agent and Gatekeeper**
Create compliance rules for applications in Kubernetes with OPA or the customizable Gatekeeper webhook.

Management

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

70 **Multicloud Ansible Rollouts**
Remain independent of your cloud provider by automatically rolling out virtual machines and applications with Ansible neutral inventory files.

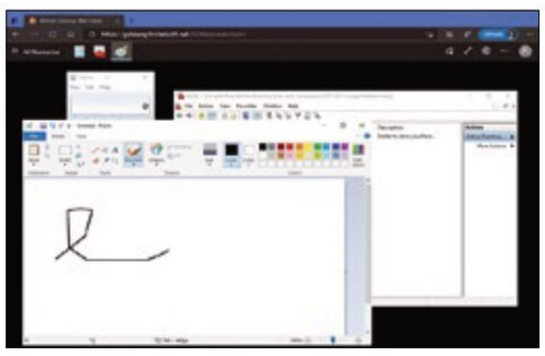


76 **U-Move for AD**
U-Tools Software promises significantly simplified backups and restores of Microsoft's directory service in the event of a disaster, during migrations, and when setting up test environments.

Nuts and Bolts

Timely tutorials on fundamental techniques for systems administrators.

80 **Azure AD App Proxy**
Support flexible working environments with Remote Desktop Services and Azure AD Application Proxy.



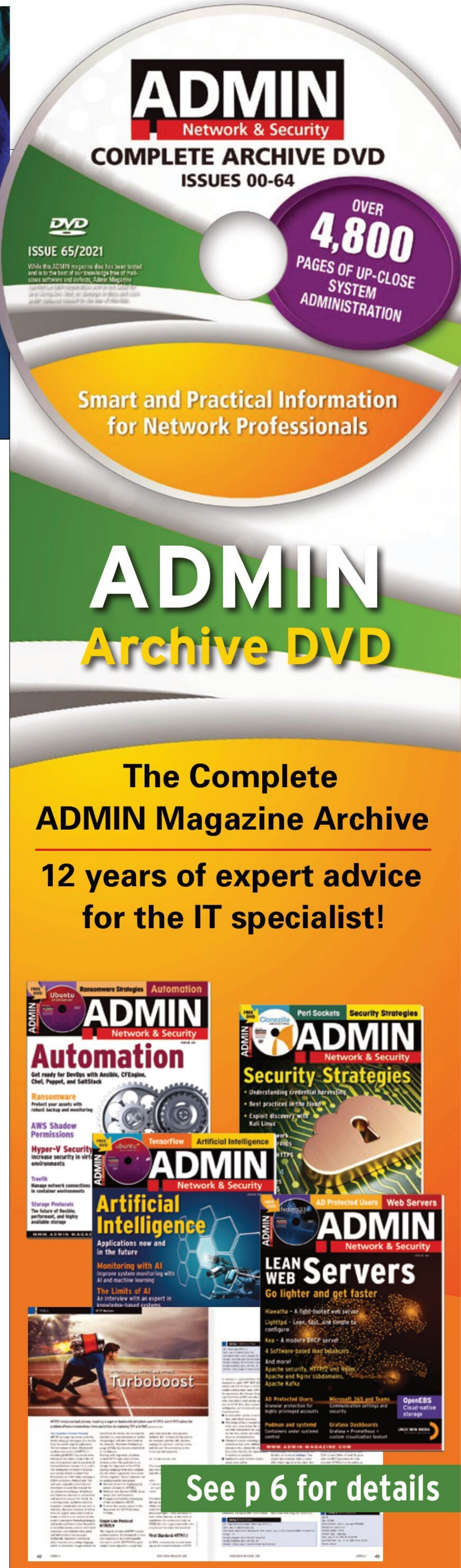
84 **PowerShell for Microsoft 365**
Manage the various components of Microsoft 365 with PowerShell scripts that use modules culled from various Microsoft products.

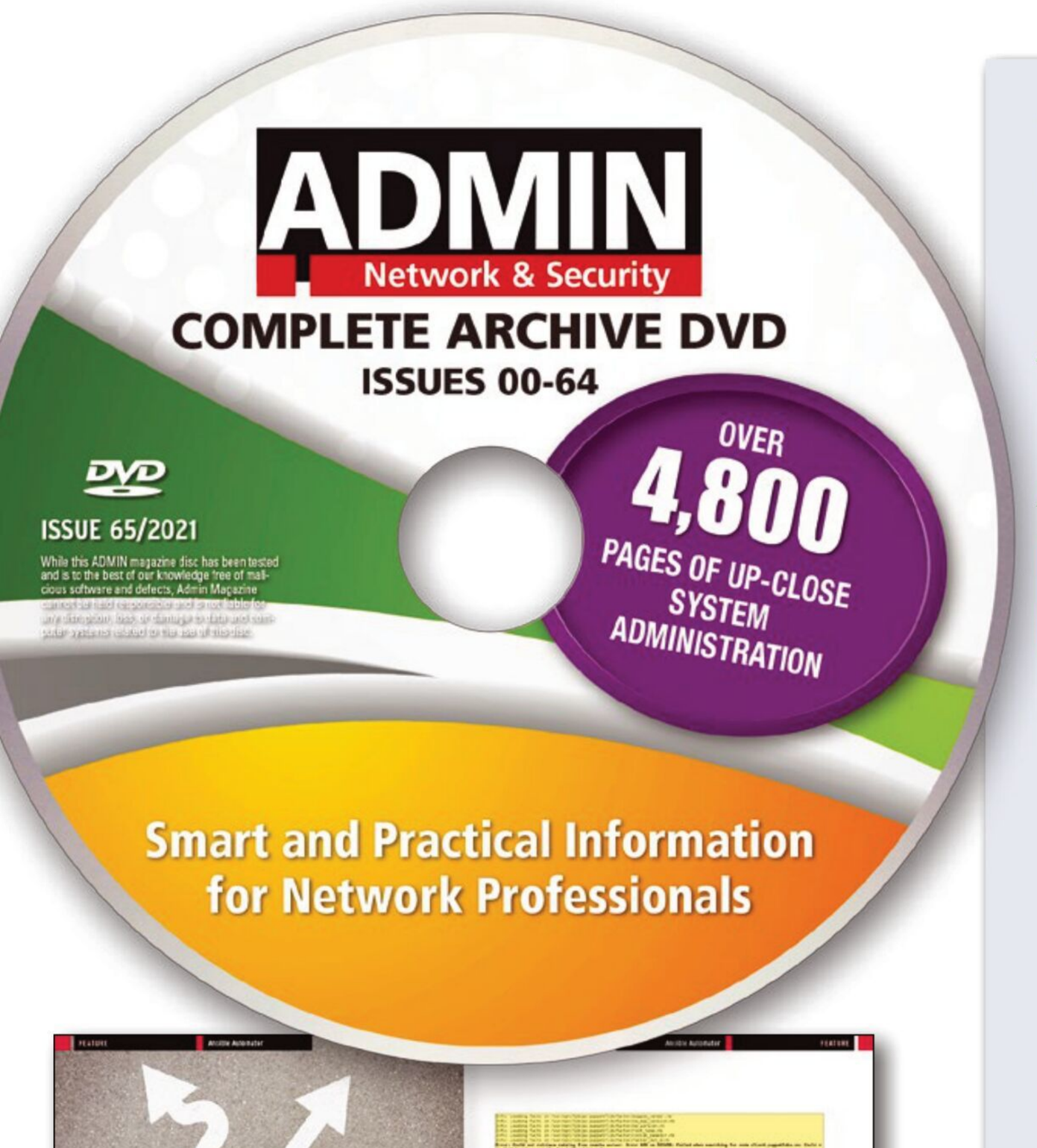
88 **Network Routing with FRR**
The FRR open routing stack can be integrated into many networks because it supports a large number of routing protocols, though its strong dependence on the underlying kernel means it requires some manual configuration.

94 **Performance Tuning Dojo**
A cloud speed test pits Linux distributions against one another.

Service

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





The Complete ADMIN Magazine Archive: 12 years of expert advice for the IT specialist!

ADMIN Archive DVD

The ADMIN Magazine Archive DVD is a comprehensive, searchable collection of ALL previous articles from ADMIN magazine – 65 issues, including the special pilot edition – with more than 4,800 pages of inspired IT.

You'll find practical, hands-on tutorials on the tools and technologies of today's networks. ADMIN patrols the realm of the modern enterprise, with articles from the experts on automation, security, optimization, virtualization, containers, and cloud computing.

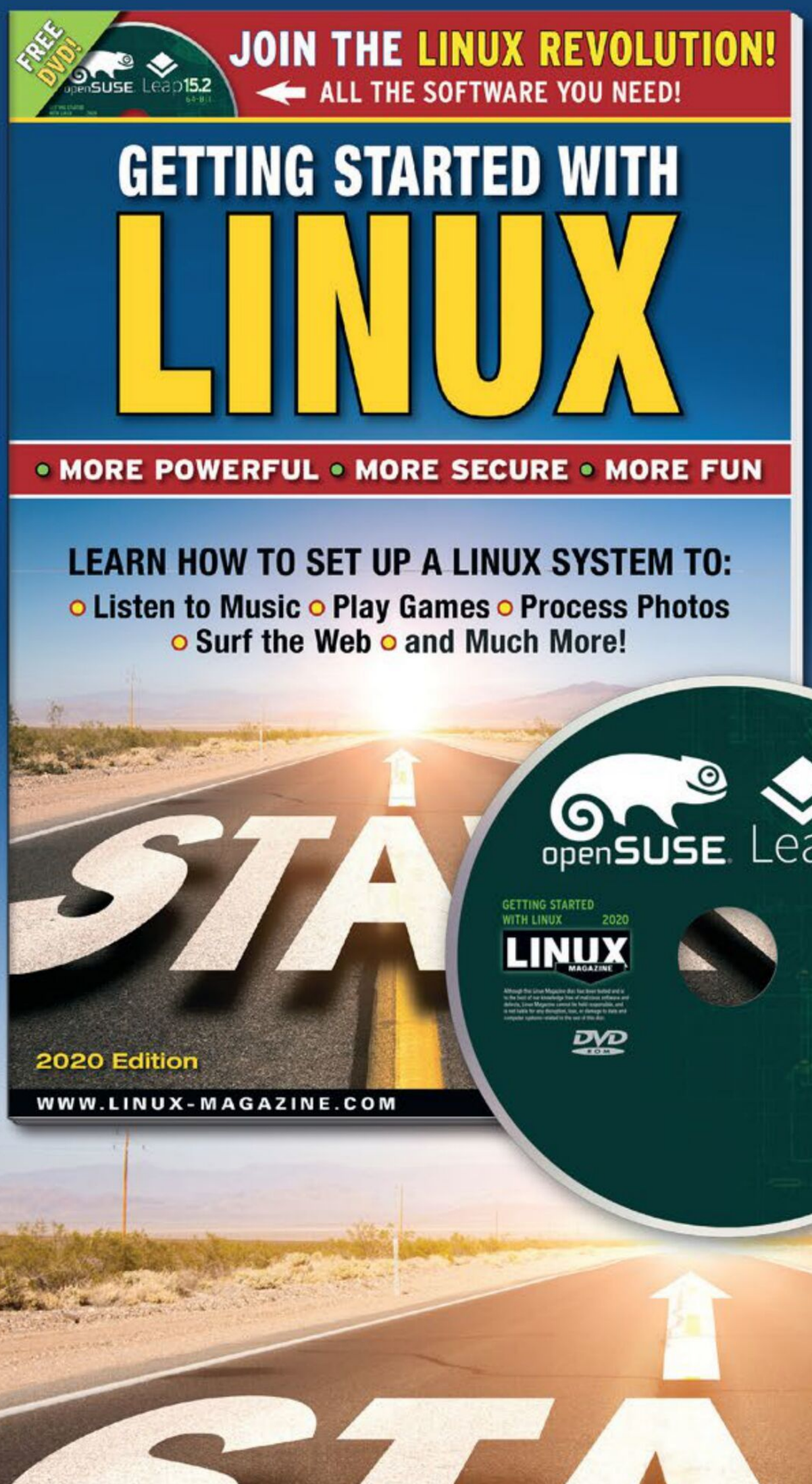
Discover the latest tools, tips, and best practices for scripting, monitoring, and network troubleshooting in Linux and Windows environments.

DEFECTIVE DVD?

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

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

Hit the ground running with Linux



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

This single issue shows beginners how to:

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

ORDER ONLINE: shop.linuxnewmedia.com/specials

News for Admins

Tech News

CloudLinux Rescues CentOS 8 From Vanishing Support

CentOS 8 users were pretty much cast aside when Red Hat shifted the focus of the operating system into a rolling release structure. This left many users and companies effectively on their own. For instance, cPanel is no longer supporting CentOS, which means admins of that platform have been forced to look elsewhere.

The problem is, there are a lot of CentOS 8 deployments running smoothly in the wild. What are those admins to do when the EOL comes for that server operating system? Clearly, they could migrate over to CloudLinux's own AlmaLinux (https://almalinux.org/) or Rocky Linux (https://rockylinux.org/) (which was created by the original CentOS developer). Both of these options have simple-to-use commands to handle the migration from CentOS 8. But if you don't want to risk that migration, you now have another option. Said option comes via TuxCare Extended Lifecycle Support (https://tuxcare.com/extended-lifecycle-support/), which covers out-of-date Linux distributions, such as Ubuntu 16.04, CentOS 8 and 6, and Oracle 6.

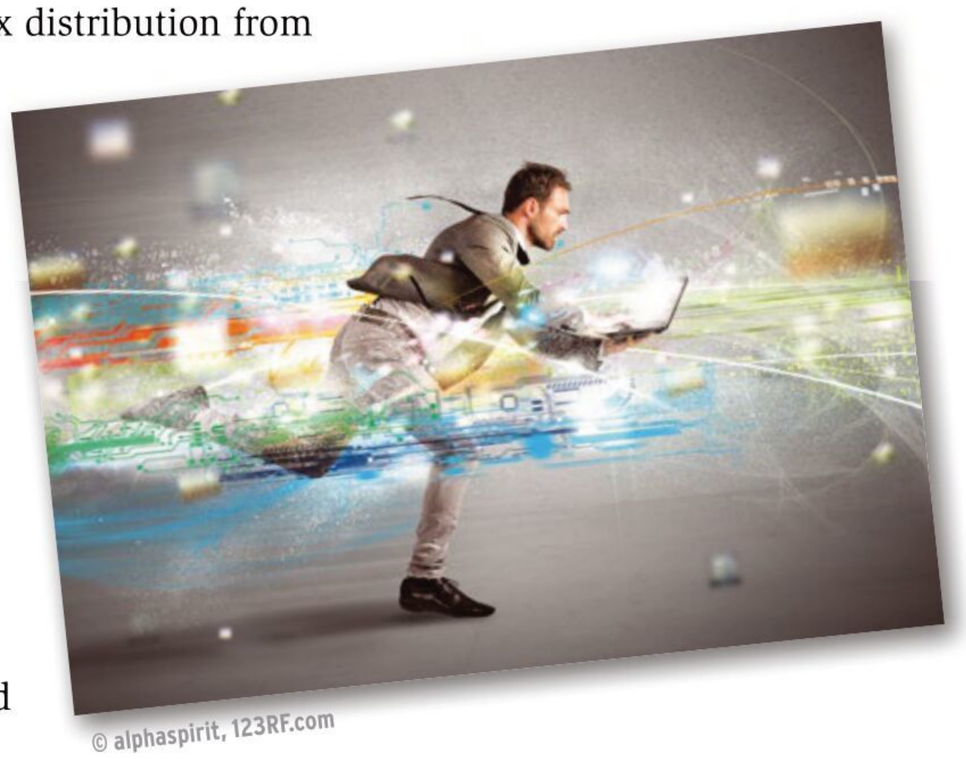
This Extended Lifecycle support will cover updates, including security patches, and general support for CentOS 8 until the close of 2025.

The cost of the TuxCare support for CentOS 6 is $4.25 per instance per month, so you should expect the cost for supporting CentOS 8 to be about the same.

Lead Image © vlastas, 123RF.com

CentOS Replacement, AlmaLinux, Available on Azure

AlmaLinux hit the ground running. As one of the first 1:1 RHEL binary compatible replacements, after CentOS shifted to a rolling release, the Linux distribution from the developers of CloudLinux has gained serious ground over the competition. And now, AlmaLinux has made its way to the Azure Marketplace (https://azuremarketplace.microsoft.com/en-us/marketplace/apps/almalinux.almalinux?tab=Overview), giving it even more credibility as an enterprise-ready operating system. AlmaLinux is one of the first of the new RHEL clones to arrive on Azure, alongside various iterations of Rocky Linux, which were created by third parties (such as Rocky Linux supporter Procomputers.com).



© alphaspirit, 123RF.com

This particular version of AlmaLinux is optimized

Get the latest IT and HPC news in your inbox

Subscribe free to ADMIN Update and HPC Update

bit.ly/HPC-ADMIN-Update

for 64-bit architecture and is considered a general-purpose release, meaning it is suitable for most use cases. And because of its 1:1 binary compatibility with Red Hat Enterprise Linux, if a task can be handled by RHEL, AlmaLinux is equally suited.

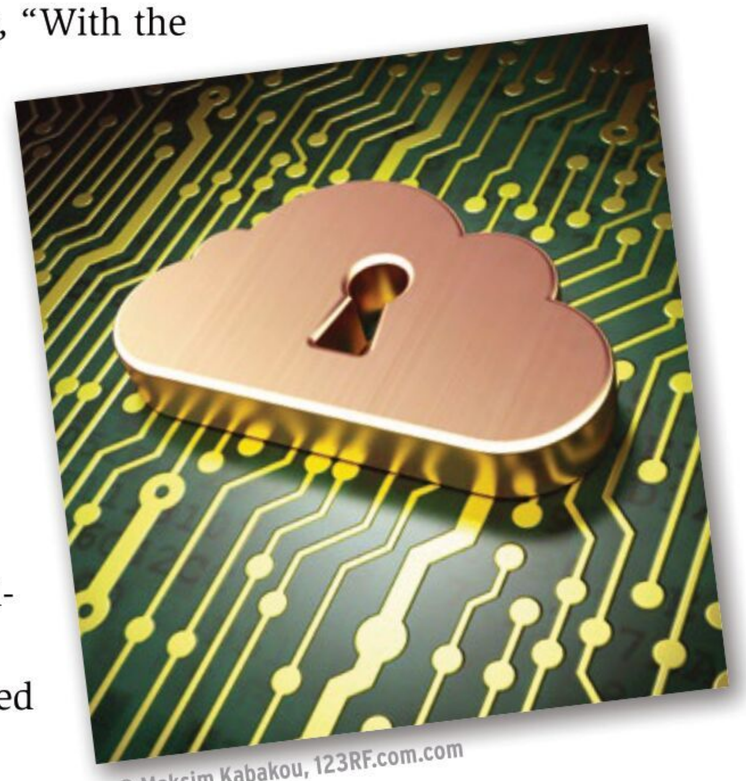
One very important thing of note is that pricing of AlmaLinux on Azure is currently listed as $0.00/hour. You will also find a number of other instances of AlmaLinux on Azure (many of which are purpose-built) that start anywhere from $0.008/hour and go up to $0.034/hour.

AlmaLinux is a 100 percent community-driven distribution with the goal of ensuring it will never follow the same path as CentOS.

Ubuntu Linux Certified for Secure and Regulated Workloads

The world's most widely deployed operating system in the cloud has officially been certified for highly secure and regulated workloads (such as those for US government agencies, prime contractors, service providers, and organizations in healthcare and finance).

According to Nikos Mavrogiannopoulos, Canonical's product manager for security, “With the new FIPS 140-2 validation, we can continue to deliver the security requirements that our government, finance, and healthcare clients trust to implement the most secure open-source software to power their infrastructure.”


© Maksim Kabakou, 123RF.com.com

FIPS 140 is a US and Canadian data-protection standard that defines security requirements for the design and implementation of cryptographic modules. This new standard ensures that only secure cryptographic algorithms are used for data protection and that all algorithms are thoroughly tested by a third party.

The FIPS 140-2 requirements state that any hardware or software cryptographic module implements algorithms from an approved list. The FIPS validated algorithms cover symmetric and asymmetric encryption techniques as well as the use of hash standards and message authentication. For that, Canonical has made available special releases of Ubuntu (Ubuntu Pro and Ubuntu Advantage) that include the new FIPS 140 validated module. With these new releases, you can run regulated workloads, reduce compliance costs, and get NIST-certified compliance.

To find out more about getting Ubuntu with the FIPS 140 validated module, contact Canonical via this form (https://ubuntu.com/security/fips#get-in-touch).

Kubernetes 1.22 Released with 56 Enhancements

Kubernetes 1.22 (http://kubernetes.io/) has been released and there's plenty to talk about. With several new features, Kubernetes admins will certainly find something that impacts their day-to-day dealings with the container technology.

Some of the bigger changes include the addition of the Admission Controller, which takes the place of the now-deprecated Pod Security Policies. Rootless Mode Containers are another very exciting feature, which make it possible to run the entire Kubernetes stack without having to use admin privileges to do so.

© Sascha Burkard, 123RF.com

Security is at the forefront of this new release, as is shown with the addition of an extra layer, named Seccomp. This new profile helps to prevent CVE and zero-day vulnerabilities and is enabled with the SeccompDefault option.

Another new addition is the ability to leave swap on. Prior to 1.22, you had to disable swap in order to run a Kubernetes cluster. This addition should make deploying Kubernetes even easier for admins.

There have also been a few removals from Kubernetes. Many of these are beta APIs (some of which have now been moved to Stable) and include Ingress, CustomResourceDefinition, ValidatingWebhookConfiguration, MutatingWebhookConfiguration, and CertificateSigningRequest.

Read more about what's in Kubernetes 1.22 in the official release notes (https://kubernetes.io/blog/2021/08/04/kubernetes-1-22-release-announcement/).

Obtain certificates with acme.sh

Simply Certified

We take a close look at acme.sh, a lightweight client for the ACME protocol that facilitates digital certificates for secure TLS communication channels. By Thorsten Scherf

The Automatic Certificate Management Environment (ACME) protocol is mostly mentioned in connection with the Let's Encrypt certification authority because it can be used to facilitate the process of issuing digital certificates for TLS encryption. In the meantime, more and more systems have started to support ACME.

Data transmitted on the Internet ideally should be encrypted. The Let's Encrypt organization [1] has played a significant role in making this good idea a reality. Until a few years ago, obtaining an X.509 certificate was a fairly complex process, but this workflow has been greatly simplified by the Let's Encrypt certification authority in combination with the ACME protocol. Anyone can now obtain a certificate for their own web service – or even other services – to ensure secure TLS communication channels.

Basically, two components are indispensable when using ACME: an ACME server and an ACME client. The protocol requires the client to prove that it has control over the domain for which the server is to issue a certificate. If the client can provide evidence, the server issues what is known as a Domain Validated Certificate (DV) and sends it to the client. Unlike the Organization Validation (OV) or Extended Validation (EV) certificate types, for example, no validation of the applicant is necessary, so the conditions are ideal for automating the process from application through the issuing of the certificate.

Different Challenge Types

The client proves control over a domain when it responds appropriately to a challenge sent by the server. The HTTP-01 and DNS-01 challenges have been part of the ACME protocol from the outset and are therefore documented in RFC8555 [2]; the TLS-ALPN-01 challenge was only added last year as an extension to the protocol. This challenge type is described in RFC8737 [3]. Most ACME clients default to the HTTP-01 challenge because it has the lowest requirements. The requester must have a web server that can be reached from the Internet on port 80 and is configured for the domain for which the certificate is to be issued. For test purposes, the ACME client itself can also start a temporary web server.

If the requirement is not met (e.g., because access to port 80 is not possible), either the DNS-01 or TLS-ALPN-01 challenge type can be used. For DNS-01, you must be able to provision a DNS TXT record within your own domain. Alternatively, for the TLS-ALPN-01 challenge type, the client uses Application Layer Protocol Negotiation (ALPN) and generates a temporary certificate used for the period of provisioning and later replaced by the certificate issued by the ACME server. In this case, communication between the ACME server and client takes place over port 443.

Verification of Control

Regardless of the challenge type used, it is always important to al-

Lead Image © Stuart Miles, 123RF.com

low the ACME server access to a specific resource, which it recreates for each challenge and then sends to the client for provisioning. This resource is available on the client as a file with the HTTP-01 challenge type, which the server then tries to retrieve. If, on the other hand, the DNS-01 challenge type is used, the server attempts to verify the resource with a DNS query.

Multilevel Workflow

JSON messages are used for communication between the ACME client and server. The workflow involves a client first registering with the server and then requesting the desired certificate. The client then uses the desired challenge type to prove that it has control over the domain used in the certificate. Before enrollment, the client must generate an asymmetric key pair to sign or verify the messages exchanged between the client and the server. Each ACME server provides a Directory JSON object that ACME clients can use to query the services offered by the server, or you can also accomplish this with the use of `curl` or a similar tool:

```
curl -s https://server.example.com/➥
  acme/directory | ➥
  python -m json.tool
```

The resource addressed earlier comprises a token that the server sends to the client and a hash generated from your public key. If you use the HTTP-01 challenge type, the ACME client must ensure that the server can request this resource under the path `/.well-known/acme-challenge/` over HTTP. If you use the DNS-01 challenge type, the server expects the string in a DNS TXT record, such as:

```
_acme-challenge.www.example.org. ➥
  300 IN TXT "Y5YvkzC_4qh9gKj6...➥
  jxAjEuX1"
```

Additionally, the protocol uses nonces to protect against replay attacks and provides a workflow for revoking issued certificates, if necessary. More information can be found in RFC8555 [2]. Although you do not need to know all the protocol details for day-to-day operation, it often helps with troubleshooting.

Obtaining a Certificate with acme.sh

The Certbot client [4] is well documented on the Internet, so I will instead look at the easiest way to get a certificate from an ACME server by introducing the `acme.sh` shell script tool [5]. Unlike Certbot, it has only a few dependencies on other software packages. Nevertheless, it is almost identical in terms of functionality.
Instead of using the tool with the Let's Encrypt certification authority, you can of course use any other ACME-compliant server. For example, Dogtag [6] or the FreeIPA [6] identity management framework supports the ACME protocol. In these cases, however, you must make sure that you explicitly designate the ACME server with the `--server` option.
To begin, either download the ACME client from the official GitHub site [5], or simply install it with:

```
curl https://get.acme.sh | ➥
  sh -s email=user@example.com
```

The configuration is already set up in `~/.acme.sh/`. To start a registration for your account on the ACME server, call the tool and create the certificate request:

```
acme.sh --register-account
acme.sh --issue --standalone ➥
        -d www.example.com
```

The `socat` tool starts a simple web server on port 80, through which the ACME server can communicate with the client. If everything works, you will be able to view the details of the certificate you just issued with `openssl` (Listing 1).

Listing 1: Viewing Certificate

```
openssl x509 -in /home/tscherf/.acme.sh/www.example.com/www.example.com.cer -noout
  -issuer -subject -dates -serial

issuer= /C=US/O=Let's Encrypt/CN=R3
subject= /CN=www.example.com
notBefore=Feb 21 13:00:28 2021 GMT
notAfter=May 22 13:00:28 2021 GMT
serial=03B46ADF0F26B94C19443669ABD0C5100356
```

In the next step, you only have to include the complete certificate chain in the desired service. To use the tool with other challenge types or in more complex setups, as always, I recommend taking a look at the software documentation [5].

Conclusion

The ACME protocol is becoming increasingly popular. A whole range of products now use it, helping to spread the use of X.509 even further. The `acme.sh` shell script is a very lightweight ACME client that compares well with better known clients such as Certbot. ■

Info

[1] Let's Encrypt project: [https://letsencrypt.org]
[2] ACME RFC8555: [https://datatracker.ietf.org/doc/html/rfc8555]
[3] ACME extension RFC8737: [https://datatracker.ietf.org/doc/html/rfc8737]
[4] ACME Certbot client: [https://certbot.eff.org]
[5] ACME shell script: [https://github.com/acmesh-official/acme.sh]
[6] Dogtag ACME Responder: [https://github.com/dogtagpki/pki/wiki/ACME-Responder]

The Author

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

Ergonomics and security of graphical email clients

Inbox

We look at the ease of finding a way around current graphical email clients by investigating ergonomics, security, and extensibility. By Erik Bärwaldt

For most use cases, email has long since replaced conventional letter mail as a means of communication. Companies in particular handle a large part of their correspondence by email, because it speeds up processes compared with snail mail; also, you can send arbitrary attachments. However, modern email programs can do more than read, write, and send messages. They can also integrate the messages into corporate workflows. The email program often serves as a personal assistant, because it usually also manages contacts and appointments and can forward data to enterprise software through various interfaces.

Additionally, email communication can be automated, if required (e.g., allowing employees to send a vacation message to their communication partners). In this article, I take a closer look at what the common graphical email clients do in terms of user ergonomics, security, and extensibility.

Functionality

The heart of all email clients is the connection to the mail server. The client uses POP3 or IMAP to retrieve incoming messages from the server. Whereas the POP3 protocol retrieves messages from the server and moves them to the client, the IMAP protocol keeps the messages on the server, which allows mail to be viewed and edited without being tied to a specific computer. The email client also archives incoming email, so that it can be accessed, even without a connection to the server. Email is sent over the SMTP protocol, with convenient features such as a queue for outgoing mail.

Email clients can usually manage several accounts simultaneously and independently of each other. Additional functions, such as address books or conversion routines provided by the program, can be used with all accounts. Some programs offer prioritization of outgoing email and can send unimportant items with a time delay if you have a large volume of messages.

Another important criterion when using an email client is security. Usually, the message transport is encrypted, for which the TLS protocol is usually used. The authenticity of messages can be guaranteed by signing; however, complete end-to-end encryption of messages requires a corresponding infrastructure with private and public keys, which is the only way to guarantee that confidential information really remains confidential.

Professional email clients additionally implement various filter and search routines, which can be used to track down any email related to a specific keyword or phrase. It is often possible to combine several terms with each other, which enables a more targeted search. Moreover, the programs help filter out unsolicited email, known as junk or spam, from the Inbox. Specialized back ends such as SpamAssassin, Bogofilter, or Bsfilter are usually used for this purpose. Their filter lists and detection routines are constantly updated, which ensures a high hit rate.

Claws Mail

Claws Mail [1] was created in 2005 as a fork of Sylpheed [2]. The cross-platform software uses the GTK+ toolkit and can be found in the software repositories of most common distributions. Additionally, a Flatpak package allows cross-distribution installation. The modular program, written in the C programming language, can be extended with plugins. After the first start, the software calls up a wizard to carry out a basic configuration. The primary window then opens, looking a bit old-fashioned (**Figure 1**). The most important controls are found at the top in a horizontal buttonbar. A folder tree on the left lists the various mail categories, and a large workspace on the right lets you read incoming email. Additionally, a conventional menubar appears at the top of the screen.

Clumsy

Claws Mail has some weaknesses in its configuration routines. The account settings are entered under *Configuration*, with quite extensive dialogs for each action. Sometimes settings are unnecessarily spread over several dialogs, so, for example, the options you need for a new account cannot be configured in a single operation. A separate category for today's standard transport encryption and port numbers that deviate from

Lead Image © Vlad Kochelaevskiy, Fotolia.com

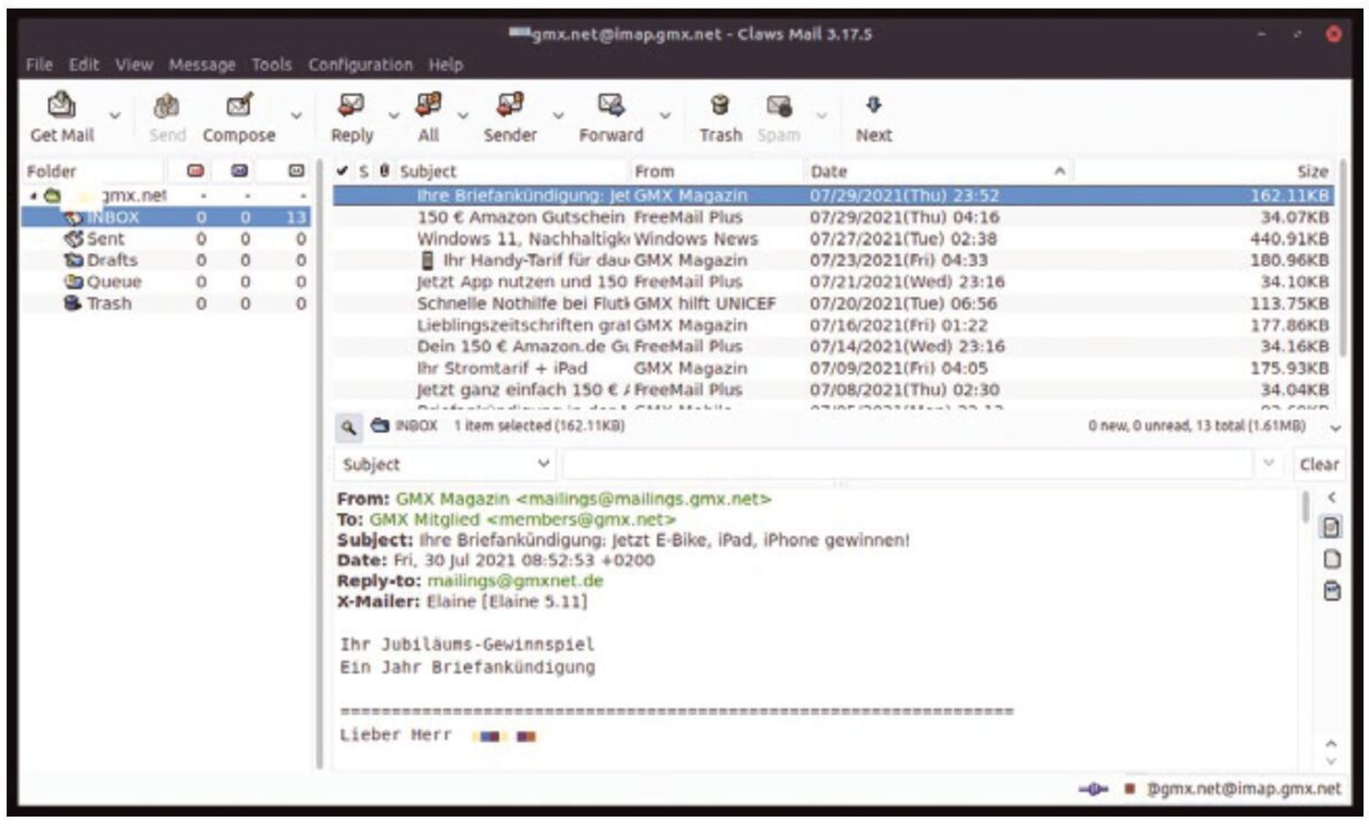


Figure 1: The program window in Claws Mail is visually rustic, but functional.

the default ports must also be set in a separate dialog.

Like most modern email clients, Claws Mail has an integrated database of predefined providers. With its help, you create a new account by clicking *Configuration* | *Create new account*; after entering the email address of the new account under *Server information*, you should click *Auto-configure*. The program uses the database to determine the valid server data for incoming and outgoing mail and configures the details accordingly. It also creates a folder structure. In the test, automated creation of a new account with various freemail providers such as GMX and Web.de did not work. To integrate such accounts into Claws Mail, you have to enter the respective server addresses for incoming and outgoing mail manually, anyway (Figure 2). The same applies to the associated port numbers and the respective method of transport encryption. For some providers, you need to enable the desired account manually in their web interface for use with external email clients with the IMAP and POP3 protocols.

Security

Besides the common transport encryptions, Claws Mail also supports digital signatures and encryption of sent email. The corresponding settings dialogs can be found in the *Privacy* tab of the respective account. Also, you can use plugins to integrate various spam filters and ClamAV, an antivirus solution, into the email program. A list of available extensions can be found on the project's website. Numerous single plugins, but also metapackages with several extensions, are available in the software archives of many distributions. To enable them, you have to integrate them into Claws Mail from the *Configuration* | *Plugins* dialog. In the window that opens, you load the plugins into the program and activate them (Figure 3). Afterwards, the plugins can be configured individually under *Configuration* | *Preferences* | *Plugins*, where you can specify, for example, the paths to the integrated third-party programs that Claws Mail uses when receiving messages or triggering a function.

Besides various antispam and antivirus solutions, some plugins add additional features to the email client that let you integrate various HTML and PDF viewers for direct viewing of corresponding files from received email.

Evolution

The Evolution [3] personal information manager (PIM) with its more than 20 years of development time is one of the dinosaurs among email clients. The software, maintained by the Gnome project, is one of the components of the current Gnome desktop and is therefore either automatically installed on your disk during the installation of a distribution or can at least be set up subsequently from the software archives. Evolution also makes it easy to create email accounts thanks to a wizard that transfers the account to the email client with just a few mouse clicks the first time it is started (Figure 4). An automated recognition routine for various email providers is provided, allowing the software to adapt quickly to different server addresses, port numbers,

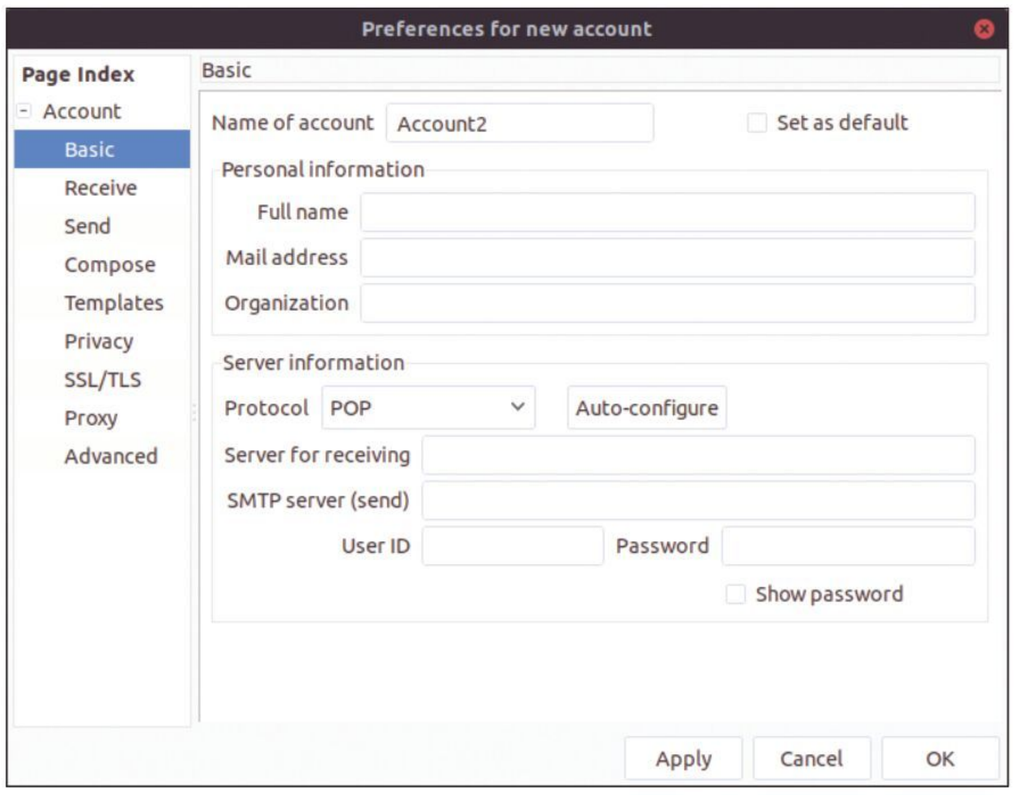


Figure 2: To set up a new account in Claws Mail, you need to work your way through several dialogs.

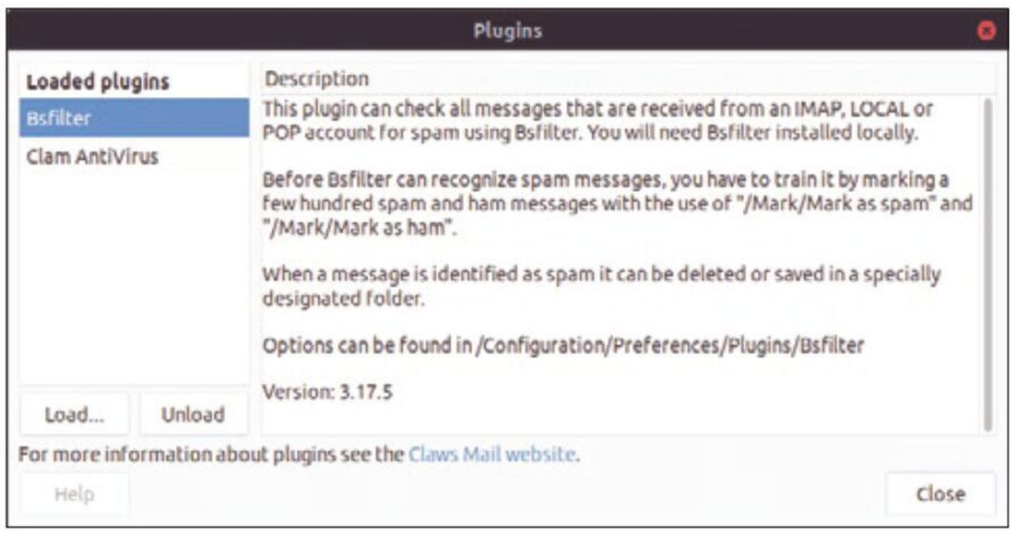


Figure 3: Plugins let you integrate additional functions into Claws Mail.

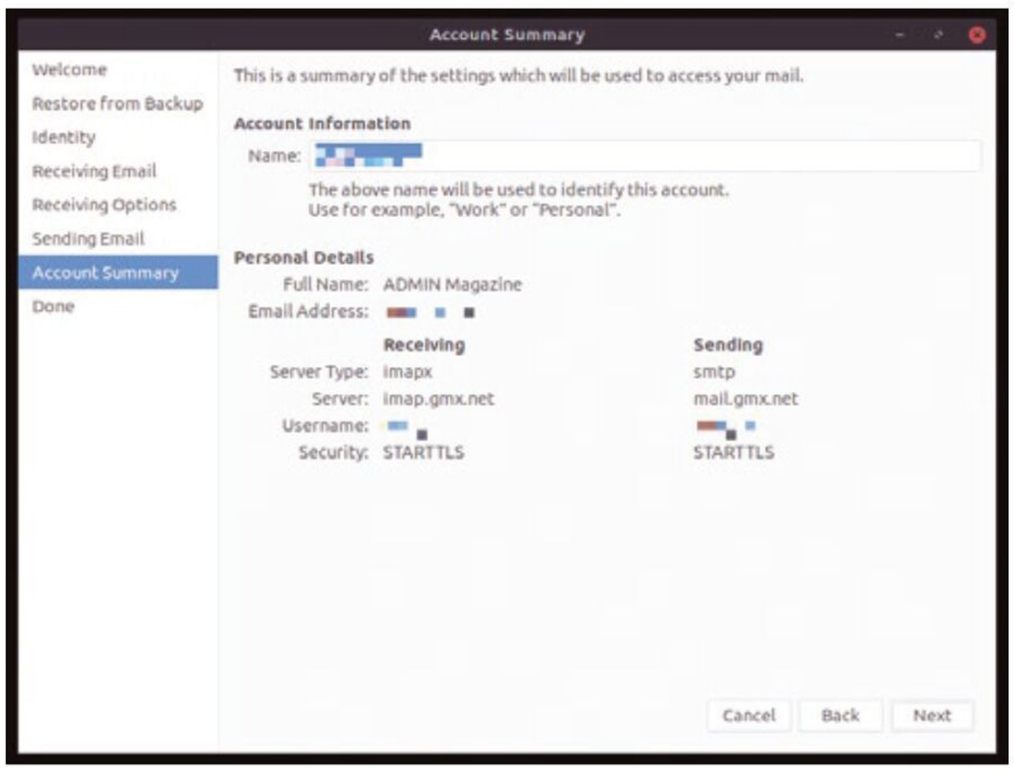


Figure 4: The startup wizard sets up the first account almost fully automatically in Evolution.

and transport encryptions. For some providers, you then have to enable message reception over the POP3 or IMAP protocol in the provider's web-based configuration interface.

Evolution's program window offers no surprises compared with other email clients. The PIM's significantly greater range of functions compared with a plain vanilla email client is noticeable when you look at the *Contacts*, *Calendars*, *Tasks*, and *Memos* buttons at bottom left in the program window. A virtual appointment calendar also shows up on the right side of the window, displaying several days.

Centralist

Evolution summarizes all relevant account-specific setting options in the Account Editor, which you reach by right-clicking on the name of the desired email account in the folder tree of the primary window on the left, selecting *Properties* in the context menu, and then selecting one of the configuration groups on the left in the Account Editor dialog.

The *Receiving Email* and *Sending Email* options allow all necessary adjustments of the server settings. Also in the Account Editor, you will find all the important options for cryptographic handling of email in the *Security* tab (Figure 5), which is where you can manage S/MIME certificates and encrypt the contents with OpenPGP. However, you have to generate keys and certificates separately with GnuPG. The options are summarized in a comprehensible manner and can be activated in part simply by placing a check mark.

Undesirables

Besides a manually adjustable message filter for incoming and outgoing email, Evolution also lets you integrate a professional spam filter into the system. It supports Bogofilter and SpamAssassin, both of which can be found in the package sources of common distributions as Evolution plugins. After their installation, you activate the respective spam filter from *Edit* | *Preferences*, where you can access the configuration dialog of the software and make all the necessary adjustments in the *Mail Preferences* | *Junk* tab (Figure 6).

Once you have added both spam filters to the system, you can select one from a selection field in the dialog and modify specific options. In a table, you also define flags in the email headers, which trigger sorting of the corresponding email, if the flags are present. The messages then end up in the *Junk* folder.

Geary

The very lean Geary [4], developed in the Vala programming language, is under the auspices of the Gnome project. The program can be found in the software archives of most distributions and can also be installed under other GTK+ -based interfaces.

Geary starts a wizard when first launched, offering import options for existing email accounts. You can integrate accounts from Gmail, Yahoo, and Outlook.com at the push of a button. For other providers, you enter the access data manually in a separate dialog. Geary only integrates IMAP accounts and does not support POP3 mail retrieval (Figure 7).

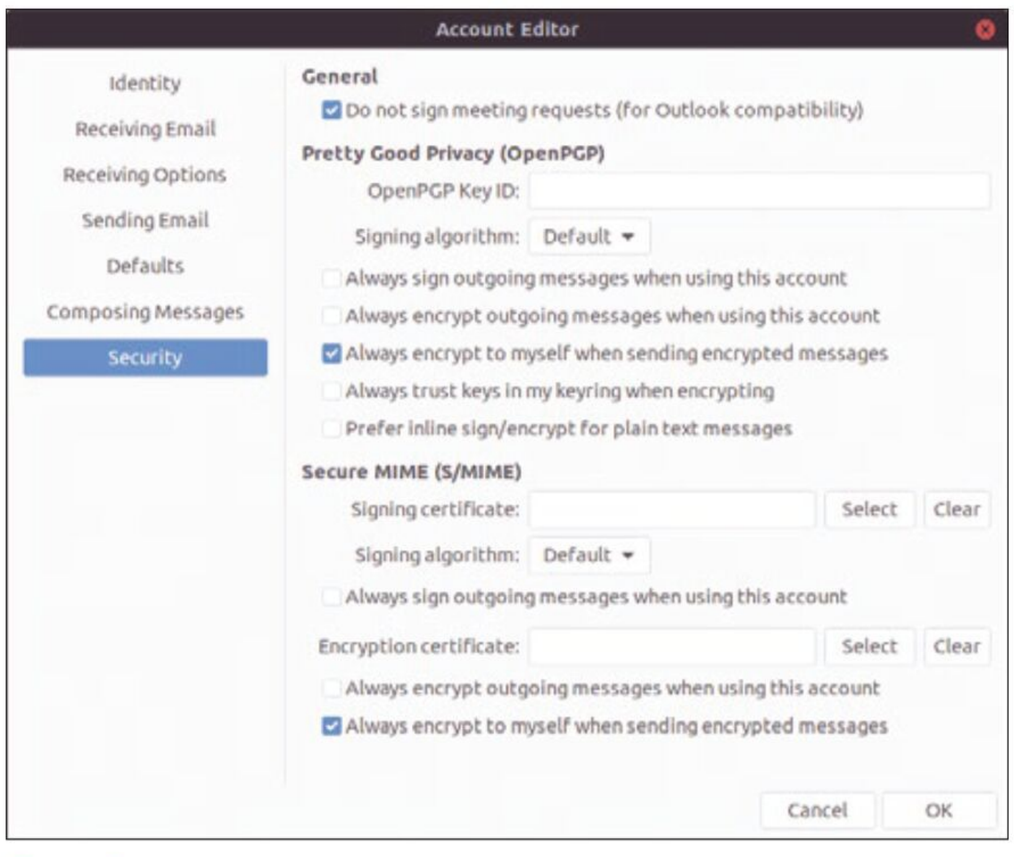


Figure 5: Evolution offers end-to-end encryption and digital signatures.

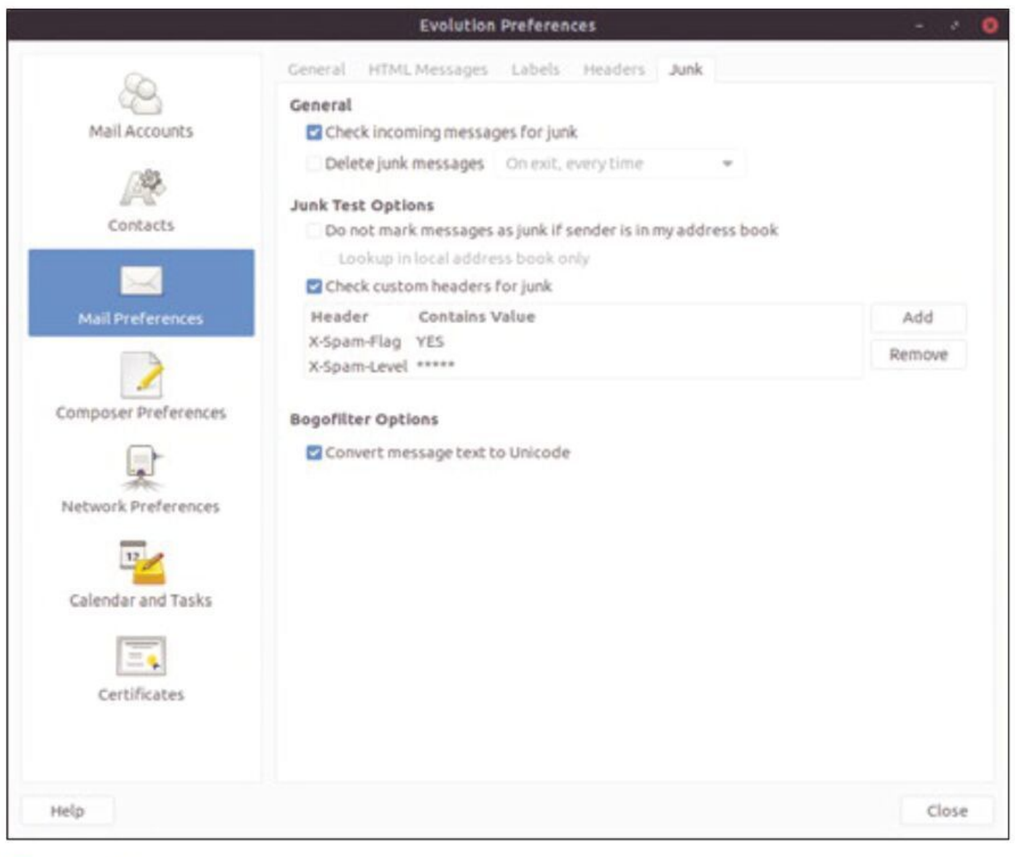


Figure 6: Evolution can include external spam filters.

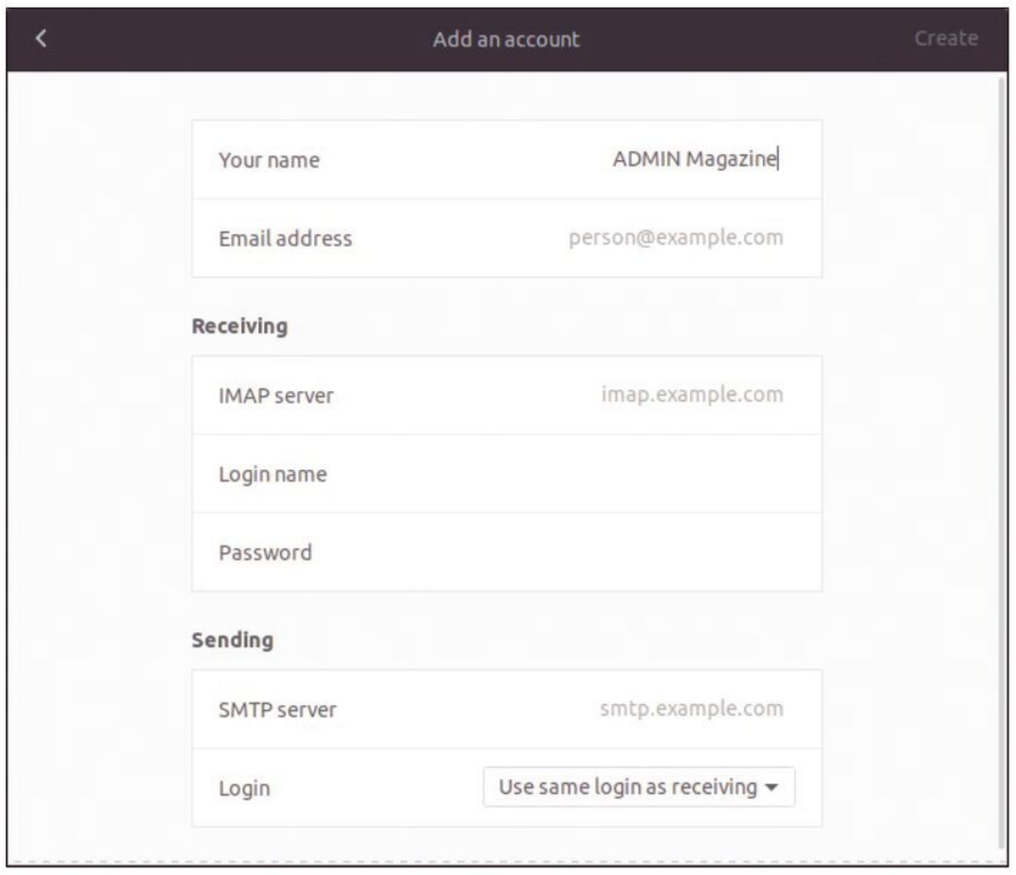


Figure 7: The wizard for creating email accounts in Geary is very spartan.

Geary does not have a function for automatically determining the access credentials for email accounts with third-party providers, so you must first determine the server data on the Internet for a manual configuration. Moreover, you cannot enter port numbers in the configuration wizard. If your provider uses ports that deviate from the norm for IMAP or SMTP access, you cannot use Geary. At least the transport encryption can be changed in the wizard, and in a separate dialog you will also find an option for entering a personal signature that appears under your email.

After finishing the configuration, the main window of the application, which is adapted to the Gnome operating conventions, opens. All the controls are located in the titlebar. The only available menu is reached through a hamburger icon.

When displaying messages, Geary follows the usual conventions: On the left you find the folder tree subdivided by email accounts; the middle contains the incoming mail in list form with sender, subject, and the content of the first line; on the right you can see the text of the active email. Mailboxes are managed from the action menu and the small buttons in the menubar.

The dialog for composing a new message, which you can access on the far left by pressing the *Compose Message* button, opens a corresponding input area in the right window segment. Geary stands out compared with other email clients in that the editor allows quite extensive formatting. By default, Geary generates HTML email. To compose simple text messages, switch the editor to plain text with the *More Settings* button bottom right in the window.

Additions?

Geary consistently follows the controversial Gnome strategy of being as easy to use as possible, which means that encryption and signature mechanisms such as GPG or S/MIME are missing, as is a spam filter to remove advertising and other messages with malicious code. Although the client does have a spam folder, it is completely aligned with the corresponding folders at the providers. Also, Geary cannot be extended with plugins.

KMail

The KMail [5] email client forms an integral part of the KDE Plasma interface. The program offers a conventional user interface and a wizard that helps integrate new accounts into the application. The latter sets up the first account largely automatically. If the routine finds another email program in the system, it also asks whether KMail should adopt its data. When creating a new account, the client also accesses the Mozilla databases after entering the email address, to determine the corresponding access data of the respective provider (Figure 8). It supports both IMAP and POP3 accounts.

In the further course of an account setup, the wizard lets you set up strong encryption with OpenPGP and S/MIME certificates. Special dialogs are provided for this purpose, with which you can also create your own keys, if required, making setting up a cryptographic infrastructure a very convenient process. For subsequent modifications or to integrate more accounts later on, call the dialog with *Settings* | *Configure KMail*. There you will find all account-specific options divided into categories in a single dialog, so that you can configure all the settings in one go without having to click through menu hierarchies. Additional convenient functions, such as time-delayed sending of email and settings for read receipts for received email, are available in this dialog. Additional dialogs allow for the visual design of outgoing messages.

Filters

KMail supports internal and external filters. The internal message filters support filtering of the incoming messages according to defined criteria. To create such filters, use the *Message* | *Create Filter* dialog and select in a context menu the message

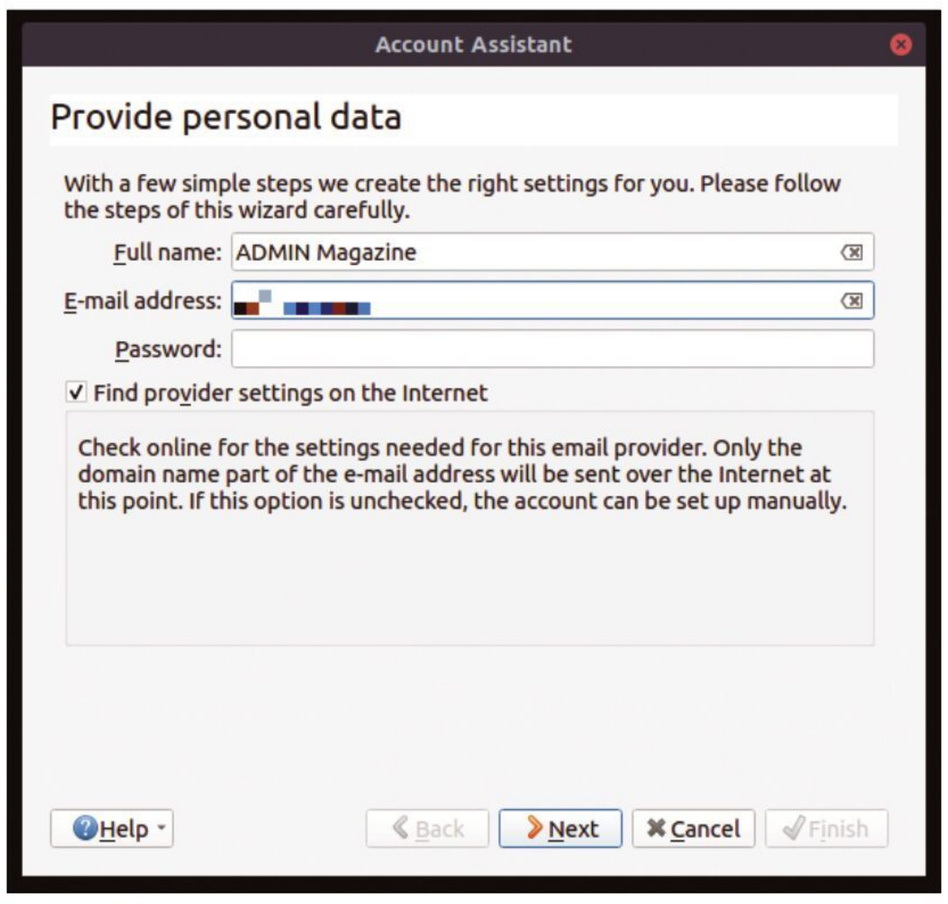


Figure 8: The KMail assistant creates accounts almost fully automatically.

criterion on which KMail will apply the filter. You can choose between *Subject*, *Sender*, *Blind Copy*, and *Recipient*. After selecting one of the criteria, a dialog opens where you define the rules and specify the folders on which KMail applies the filters. Several rules can be combined. Alternatively, you can create filters from the *Settings* | *Configure Filters* menu. The dialogs correspond to those you reach from the context menu. After saving the filters, right-click on incoming email and select *Apply Filter* to let KMail manage the messages according to the defined filters.

The application itself does not provide its own spam filter. However, to weed out spam from the messages you receive, external filtering programs such as Bogofilter can be integrated into KMail. Once installed, open *Tools* | *Anti-Spam Wizard*. KMail finds the spam filters installed on the system and displays them. It also takes into account filters that providers already use on their servers to filter spam (Figure 9). You then select one of the filters for use with KMail and specify in further dialogs how the client should proceed with the classified messages.

Similarly, if needed, you can install antivirus filters that filter out email with malware from the message files. To do this, use *Tools* | *Anti-Virus Wizard*. The software now determines the anti-virus applications installed on the computer and lets you select the desired tool. After that, you need to specify in a separate configuration dialog how KMail should handle messages that potentially contain malware (Figure 10).

Figure 9: You need to include spam filters as external programs; KMail works with all popular solutions.

Figure 10: Antivirus software also embeds KMail from external sources.

Mailspring

Mailspring [6], a fork of Nylas Mail discontinued in 2017, is a still a fairly young, largely unknown project. The application is missing from the package sources of the popular Linux derivatives thus far, but you pick it up directly from the project's website as an RPM or DEB package for 64-bit systems. You can also find a Snap package.

The application integrates with the menu hierarchies of the common desktop environments without any problems and launches a wizard when first run. To begin, it introduces you to some of the program's functions and then offers a Mailspring ID. You will need one, for example, to keep data synchronized between multiple instances on different computers. Some additional functions, such as the use of plugins, also require a valid subscription, which costs $8 per month. If you do not want these additional features, skip the Mailspring ID creation step, and you will be taken to the configuration dialog where you can click to select a mail provider from a list to connect an existing account (Figure 11). If you use an account of a provider that is not listed, select the *IMAP/SMTP* option. Mailspring does not support POP3 accounts. After that, configure all the required settings in another dialog. After saving the entries, the wizard closes.

You now call the mail program from the desktop menu to open a window with several panels. Depending on the working environment, a horizontal menubar at the top of the screen supplements the conventional display. Below this are several buttons for quick access, which are used to manage communication. The arrangement of the three or four panels, depending on your selection, and the button bar is based on other popular email clients – no training is required.

Attitude Thing

The Edit menu gives you access to a visually modern-looking, unusually detailed configuration dialog that is divided into several categories accessed

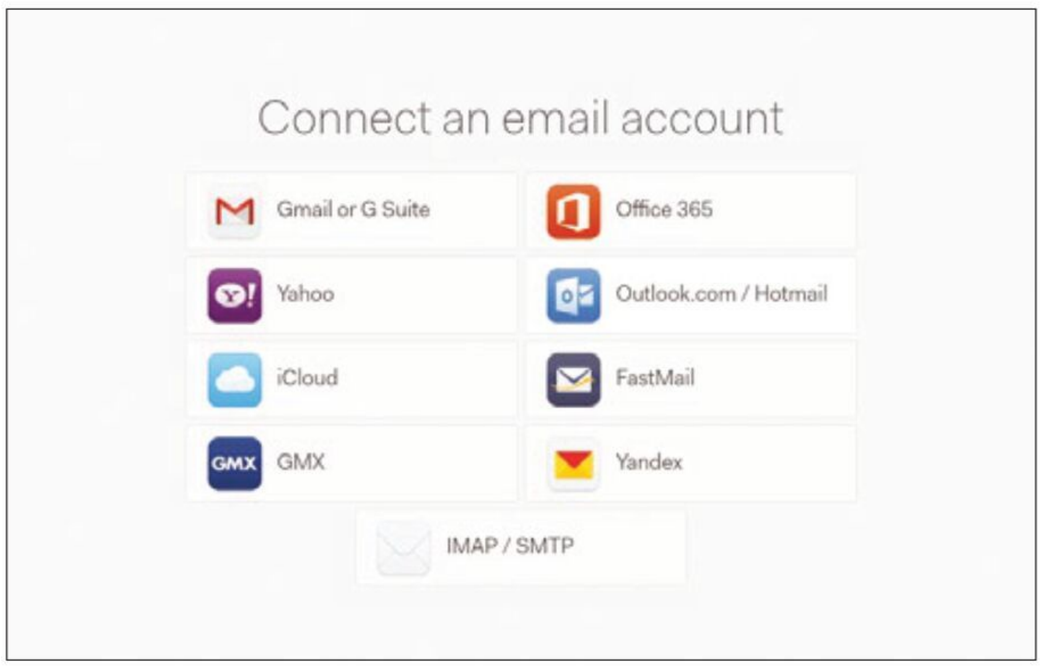


Figure 11: Mailspring has preconfigured some major mail providers.

from a buttonbar at the top of the window (Figure 12). You can change the appearance, modify key combinations for quick access, create signatures for outgoing email, or make general settings. In the *Accounts* group, you can also integrate additional mail accounts into the client.

Security

Mailspring currently offers neither an option for encryption with OpenPGP nor an option for S/MIME cryptosystems. However, for transport encryption of content, the common TLS specifications are implemented in the mail program.

You can filter out unwanted messages by manually tagging them as spam, after which the client moves them to the spam folder. No rules can be defined, nor does Mailspring perform any genuine spam filtering. For email providers that operate spam filters on the server side, this shortcoming is not significant. However, if you operate your own IMAP server that delivers email to clients, you will need to add a spam filter there to avoid time-consuming manual sorting of spam on the clients.

Plugins

The modular Mailspring supports extensions. The developers explicitly invite users to write their own plugins and publish them on a community platform. The project provides detailed documentation for developing such extensions. Themes for customizing the appearance of the application can also be downloaded here and integrated into Mailspring. If needed, you can even design your own theme and make it available for other users to download. The corresponding documentation is available for this, too.

Thunderbird

As the top dog among email clients, Thunderbird [7] can now look back on some 18 years of development. Originally developed by the Mozilla Foundation, the program now resides under the umbrella of MZLA Technologies Corporation, a subsidiary of the Mozilla Foundation.

Thunderbird not only offers a mail client, but also the complete feature set of a PIM with a newsreader, a chat and messaging client, calendar and appointments management, and contacts. The package can thus be used as a communications center, removing the need for numerous individual tools.

Thunderbird can be found in the package sources of almost all distributions but can also be obtained directly from the project's website. Many distributions also preinstall the application as standard software for email management.

Wizard

Thunderbird opens a wizard to help configure an email account when it is first launched. Next to it, the program displays an information page in the main window that contains links to several wizards for setting up the different communication services supported by Thunderbird (Figure 13). After entering the email address, the wizard for creating the first email account automatically determines the access data from a Mozilla database for incoming and outgoing mail. The application supports IMAP and POP3 accounts and sets the port numbers and encryption methods accordingly. If the routine does not find an entry in the database for the provider used, you can enter the required details manually. After saving the configuration, the main window of the application opens with several conventionally arranged segments. A vertical bar on the left edge of the window shows the folder tree for the individual accounts. Top right is a list of incoming messages, including the subject, sender, and date. The area below is for displaying the message. Above this are a few buttons for quick access to the most important functions; a menubar is missing.

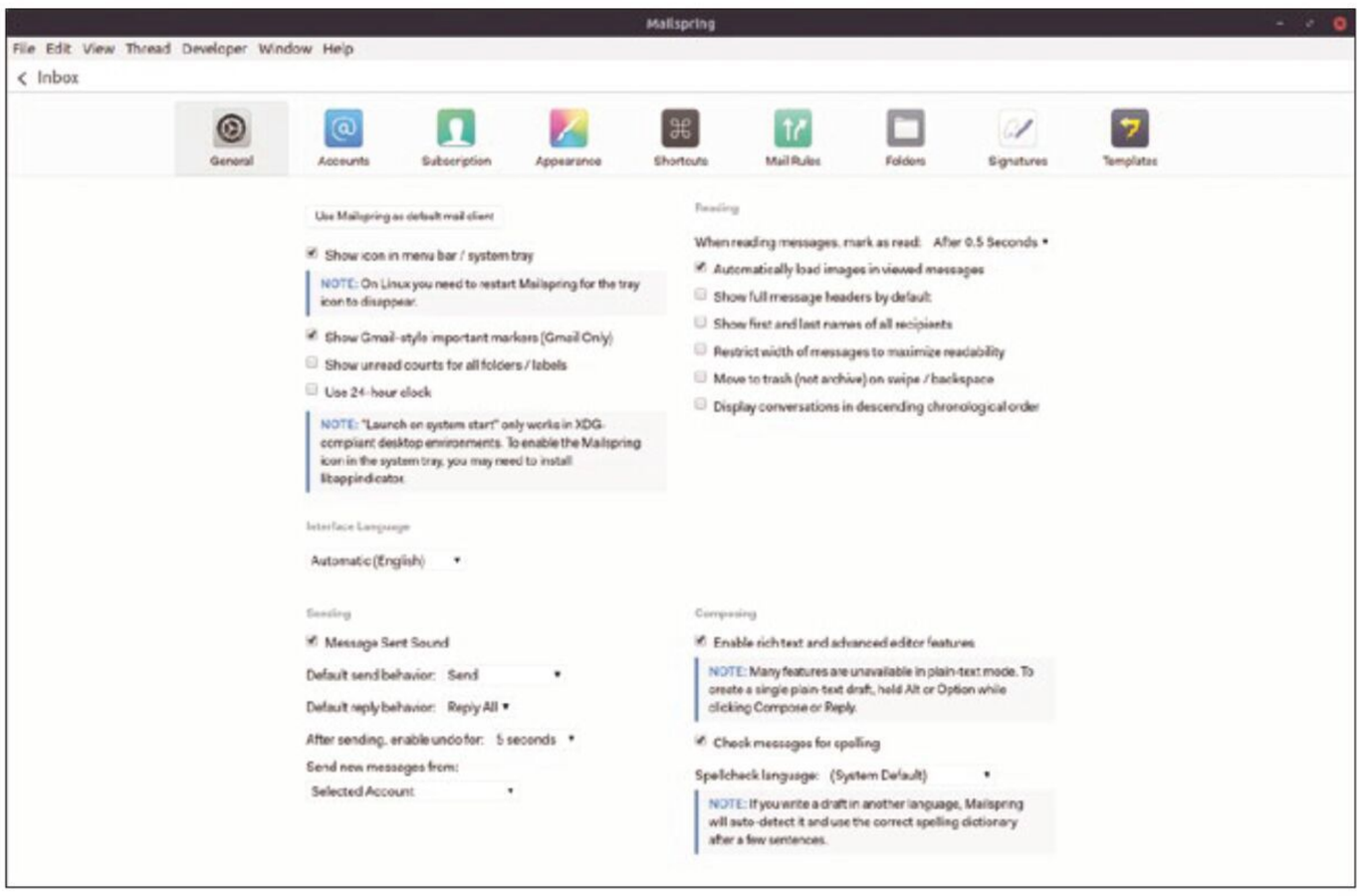


Figure 12: In Mailspring, you complete the configuration from numerous dialog boxes.

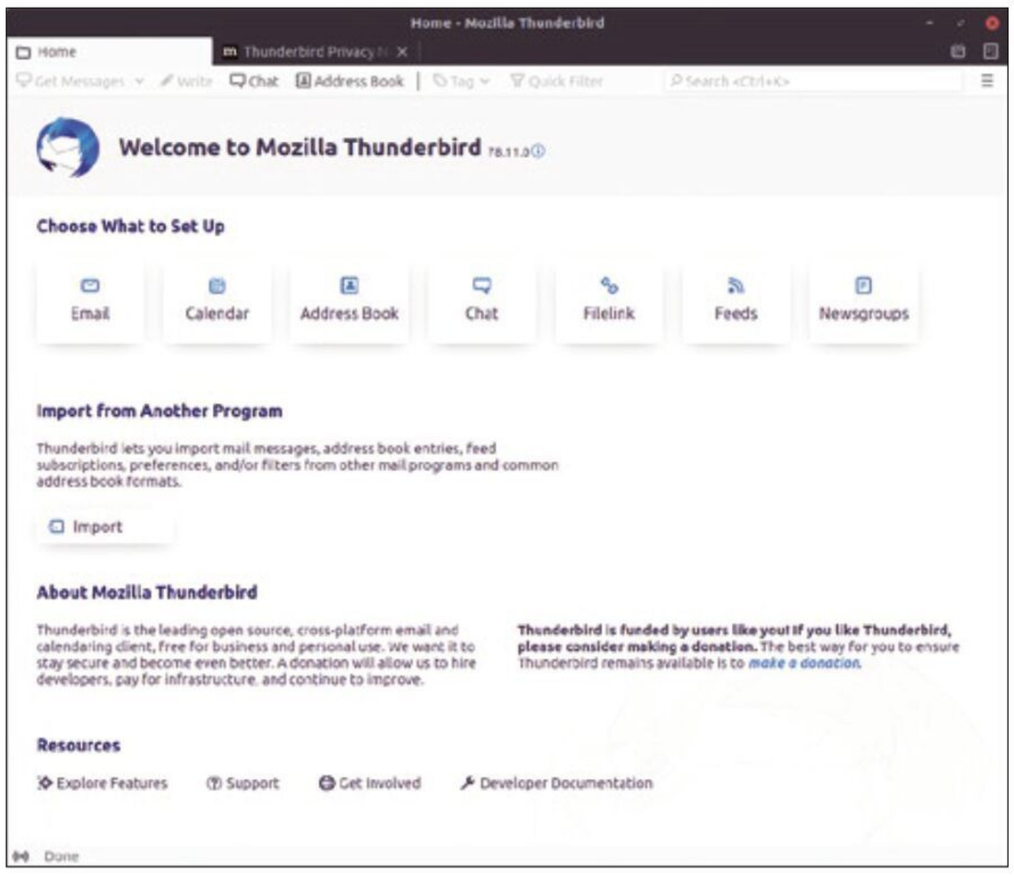


Figure 13: Thunderbird offers a matching wizard for each service.

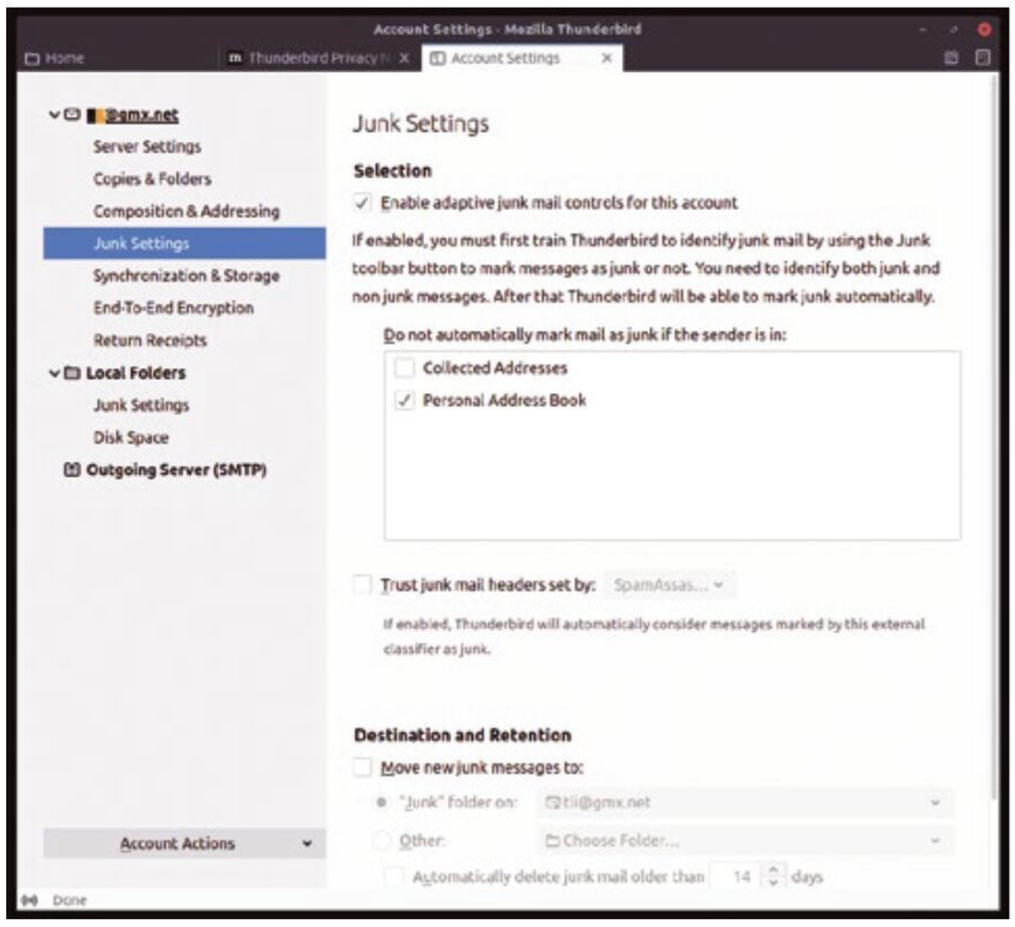


Figure 14: Thunderbird embeds spam filters both internally and from external sources.

Thunderbird, like most popular web browsers, supports a tab structure. Clicking on the icons for the calendar and tasks in the top right corner opens a new tab in each case. The hamburger menu located to the right of the search field gives you access to the configuration dialogs for the general *Preferences*, as well as the *Account Settings*.

The Preferences dialog primarily comprises of the general settings relevant for other modules, as well. Under *Account Settings*, you can tweak the email accounts by setting up end-to-end encryption with OpenPGP and S/MIME or configuring junk filters.

The software comes with its own spam filter, which does not require any additional configuration. However, it needs some time to learn how to distinguish useful messages from spam from the headers. In addition to this filter, you can also integrate external filters by placing a check mark in front of the *Trust junk mail headers set by* option in the Junk Settings configuration dialog and selecting the desired external filter in the selection field beside it (Figure 14).

Thunderbird already supports numerous third-party spam filters by default. You can apply them to local folders, too, by checking the corresponding box in the dialog for this category. In both dialogs, you also specify the folder to which Thunderbird will move messages marked as spam.

To enable end-to-end encryption, configure your keys and certificates in the *End-To-End Encryption* option. You can use OpenPGP to generate keys or import existing ones. Note that you should continue to use existing keys, because any newly generated keys will no longer support access to older messages. You can also use external keys, such as those stored on a smartcard, with GnuPG.

Thunderbird also cooperates with externally installed antivirus solutions, if required, by letting them check incoming messages for malware and moving them to a quarantine folder, without storing them in the Inbox. This function is primarily recommended for POP3 accounts.

Trojitá

The Qt-based email program Trojitá [8] primarily targets less powerful hardware and users who only need an email client with a basic feature set. The software is only capable of handling a single IMAP account. The program can be found in the software archives of almost all distributions.

Configuration

Like other common mail clients, Trojitá has a configuration wizard (Figure 15), but it lacks a database connection to determine access data automatically, so when creating an account, you need to know the data of the IMAP and SMTP servers. You can also manually assign port numbers that deviate from the standard, choose from various authentication methods, and set options for transport encryption.

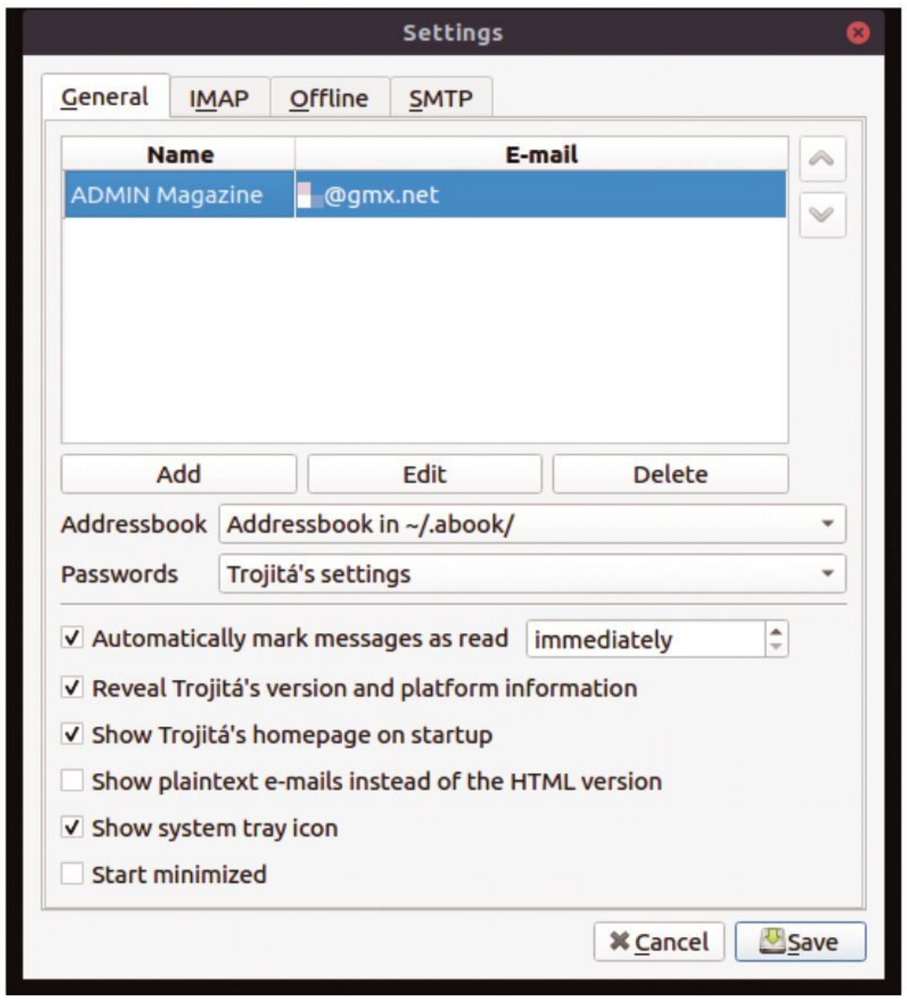


Figure 15: Trojitá comes with a simple but effective configuration dialog.

Conventional

After creating the account, the application opens a conventional window with the usual division into segments (Figure 16). A menubar and a button-bar give you quick access to frequently used functions. You can manage contacts with the help of an integrated address book. Trojitá also displays email in either plain text or HTML format.
A special feature under *IMAP | Network Access* lets you choose between the options *Offline*, *Expensive Connection*, and *Free Access*, which make it possible to minimize the volume (and cost) of data to be transferred in the event of expensive paid access to the Internet with a UMTS/LTE connection.

Problematic

Because the application does not have a recycle bin, email disappears irrevocably into a black hole when deleted, so special care is required. For messages saved offline, you can also specify whether Trojitá should delete them after a defined period of time. This action is also irrevocable, so it is a good idea to save the messages permanently, which you can also set as an option.

Missing

Trojitá's feature set covers only the basic scope of message management. Because of the lack of a modular structure, the application cannot be extended with plugins. You cannot use OpenPGP and S/MIME with Trojitá, nor can you integrate common spam filters into the application. You can only manually tag spam as such, which prompts the program to move the messages to the *Spam* folder. A print function is completely missing.

Conclusions

The email clients I looked at in this article cover the full range of functions for every individual need (Table 1). Users who only need a plain mail client for a single account, with which they only read, write, and manage mail, are well served by Trojitá. The program is particularly suitable for mobile use.
The solid all-rounders Claws Mail, Evolution, and Geary facilitate the daily handling of email. However, Geary neither supports POP3 accounts nor has strong encryption mechanisms. KMail is not only best suited for daily use, but also stands out with a very well thought out, catchy operating concept. Mailspring is well suited for users who want to use a state-of-the-art interface, whereas Thunderbird unites a wide variety of programs – from mail clients to feed readers – under a single interface. Therefore, it is particularly well suited for users who are looking for an integrated environment for their complete work organization in the office.
Because an almost uniform interface has become established for graphical email clients, you will not need any training for any of the programs, so you can immediately concentrate on the essential functions. ■

Info

[1] Claws Mail: [https://www.claws-mail.org]
[2] Sylpheed: [https://sylpheed.sraoss.jp/en/]
[3] Evolution: [https://wiki.gnome.org/Apps/Evolution]
[4] Geary: [https://wiki.gnome.org/Apps/Geary]
[5] KMail: [https://apps.kde.org/kmail2/]
[6] Mailspring: [https://getmailspring.com]
[7] Thunderbird: [https://www.thunderbird.net]
[8] Trojitá: [http://trojita.flaska.net]

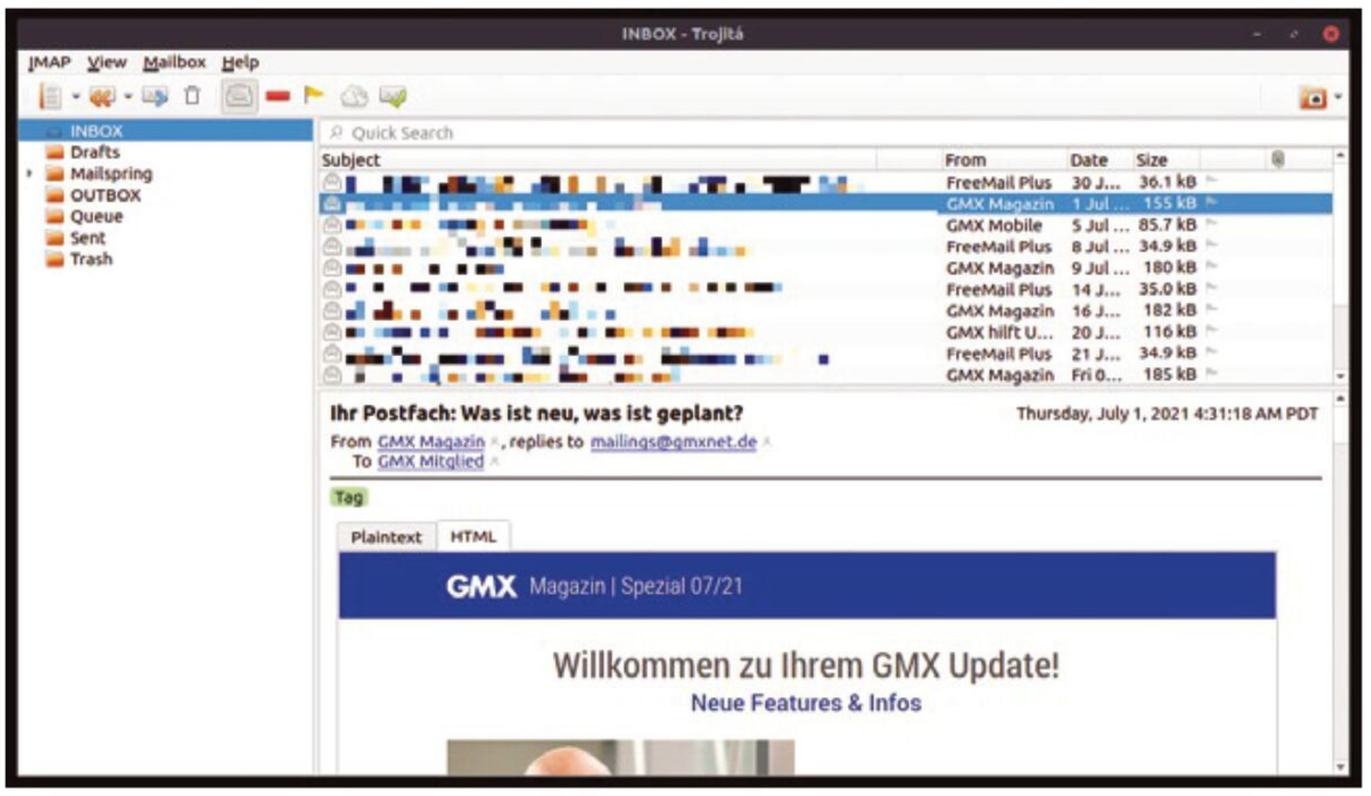


Figure 16: The Trojitá program window follows conventional concepts.

Table 1: Email Clients

Feature	Claws Mail	Evolution	Geary	KMail	Mailspring	Thunderbird	Trojitá
License	GPL	LGPLv2	LGPLv2.1	GPLv2	GPLv3	MPL, GPL, LGPL	GPL
Setup wizard	Yes	Yes	Yes(1)	Yes	Yes(1)	Yes	Yes
Provider database	Yes	Yes	No	Yes	No	Yes	No
Manual configuration	Yes	Yes	Yes	Yes	Yes	Yes	Yes
IMAP accounts	Yes	Yes	Yes	Yes	Yes	Yes	Yes
POP3 accounts	Yes	Yes	No	Yes	No	Yes	No
Mail encryption	Yes	Yes	No	Yes	No	Yes	No
Digital signatures	Yes	Yes	No	Yes	No	Yes	No
Spam filter integration	Yes	Yes	No	Yes	No	Yes	No
Antivirus software integration	Yes	No	No	Yes	No	Yes	No
Plugins	Yes	Yes	No	Yes	Yes(2)	Yes	No
Integrated PIM	Yes(1)	Yes	No	No	No	Yes	No

(1) With restrictions
(2) Commercial option in part

High-performance backup strategies

Keep More Data

A sound backup strategy with appropriate hardware and software ensures you can backup and restore your data rapidly and reliably. By Jan Kappen

With increases in data growth and larger and larger servers, the need to back up systems efficiently is becoming increasingly urgent. A backup needs to be available and also be restorable in a timely manner. Establishing suitable strategies that look in detail at storage, networks, and the software used is important. In this way, peak performance in the backup process can be ensured, even for very large data volumes.

Data backup is not a new topic, of course. In fact, it has already been discussed so often that it is encountered with a certain apathy in some places, which makes it all the more important to take a look at the current situation and talk about new possible types of data loss. A well-thought-out backup infrastructure can not only save your data but ensure the continued existence of your entire company in the event of an incident.

In addition to common IT hardware failures due to technical defects or damage caused by external factors such as fire or water, a new field of threat has been emerging for some time: ransomware, which attacks companies through malware and, in the event of a successful infection, encrypts the existing data and demands cash to free it.

Regardless of the type of failure that affects you, the data must be restored as quickly as possible, and a backup must be as up to date as possible. Ensuring that these requirements can be met even with multiterabyte systems requires an appropriate strategy.

Fast Backup Storage

A commonality in current IT landscapes is massive growth in data. Even companies with fewer than 20 employees can have data on the order of 5TB or more. Medium-sized companies commonly need to have 30-100TB constantly available. In other cases, companies have long since reached petabyte dimensions. The data needs to be backed up continuously and made available again as quickly as possible in the event of loss.

Backing up data the first time is a huge job because all the files have to be moved to the backup storage medium once. After that, the backup time required decreases significantly through the use of technologies such as changed block tracking, wherein the current backup only needs to include blocks that have been changed since the previous backup. In the event of a restore, however, the IT manager must take into account the available bandwidth, the size of the virtual machines (VMs) or data, and the time required for such a process.

I/O Storage Performance

Besides a good connection, the type of backup storage you have also matters. Traditional hard disk drives (HDDs) still play an important role because they provide a large

Lead Image © Kian Hwi Lim, 123RF.com

amount of space for little money. However, this also means that you are limited by the performance of these data carriers. The throughput is not necessarily the problem, but I/O performance is.

To speed up the recovery of VMs, many manufacturers now have built into their software the option of executing backups directly from the backup memory, which makes it possible to start virtual systems even though the data is not yet back on the storage space originally used. This technique speeds up a restore to the extent that you can sometimes bring systems back online within a few minutes, regardless of the storage space used.

However, you have to keep in mind that the I/O load has now shifted to your backup storage. Depending on the equipment, it can be slower in some cases and unusable in others. Additionally, you have the extra load from the process of copying the data back to production storage. If you want to use these functions, you have to consider these aspects during the planning phase.

Hard drives offer a performance of around 100-250 input/output operations per second (IOPS), depending on the model and class. Classic solid-state drives (SSDs) approved for use in a server raise these values to between 20,000 and 60,000 IOPS. If even these values are not sufficient, the use of NVMe storage is an option. Here, flash memory is not addressed over SATA or SAS bus but over PCIe, which unleashes maximum performance and offers values of up to 750,000 IOPS per data medium, depending on the model. I will go into that in more detail later to show you how to speed up your backup without having to invest vast sums in flash storage.

The Thing About the Network

The connection between the backup server and your infrastructure can often be optimized. If you use a 1Gbps connection for the backup, you can theoretically transfer just under 450GB per hour. In reality, this value is somewhat lower, but you can reckon on 400GB per hour. Restoring 5TB of data will take just under 12 hours, and 10TB will take a day or more.

For better transfer times, you should start increasing the usable bandwidth. Hardware for 10 or 25Gbps is quite affordable today and directly eliminates a major potential bottleneck with a shorter backup window and significantly reduced recovery times. Running the backup on a dedicated network also relieves the load on your production network so the bandwidth is available for other things.

In some environments, even connections with 100Gbps are now used, and this hardware is no longer a budget-buster. If you use Ethernet as the storage protocol in your infrastructure (e.g., with Microsoft technologies such as Storage Spaces Direct (S2D) or Azure Stack HCI, i.e., hyperconverged infrastructure), you can integrate the backup infrastructure and might not even need additional network hardware.

Multilayer Backup Strategy

Storage area network (SAN) storage has long been capable of using a combination of data carriers with different attributes in order to use the best properties of each. The combination of SSDs and HDDs, for example, ensures that the fast but more expensive SSDs can be used for the part of the data on which you are currently working (hot data). The other part of the data, which is not really in use but still has to be available (cold data), is then stored on the large and slow, but inexpensive, HDDs.

To achieve a performance boost for your backup through this combination, you define two different storage locations. The first location (referred to as tier 1 in the remainder of this document) comprises flash storage and is capable of holding the backup data for a few days (**Figure 1**). You can define how long this period is yourself. Because a large part of data restores relate to the most recent backup, a few days are usually enough in this case. The longer you want to store data on tier 1, the more expensive this storage area becomes. Restores are very fast, function like instant recovery (i.e., starting VMs from backup storage), and benefit strongly from the high performance.

Some backup products also support manual or automated restoration of backups for testing purposes and to verify data integrity, which not only ensures that the backup is present but also demonstrates that recovery is possible and your backup is usable. This feature of restoring data to a sandbox enhances security and often satisfies various compliance requirements.

Many audits or insurance policies now ask whether restores are performed regularly for validation purposes. Backup storage with sufficient performance enables such tests within a very short time, even when using multiple VMs at the same time. By restoring in a special sandbox, your production is not disturbed at any time (e.g., the tests can even run during the daytime). Another advantage of this shielded environment is that you can run scenarios such as updates, upgrades, and so on with temporarily started VMs.

Once the time period during which the backup data is allowed to reside in tier 1 has expired, the data automatically moves to backend storage (tier 2). For this storage, too, you define how long it stores the data. Through strategic timing (e.g., weekly, monthly, or annually), you can achieve periods of several years. Because you will be using HDDs, you can't expect too much in the way of performance (compared with tier 1 storage), but you will get significantly more storage at a lower cost. Each day the backup data ages decreases the likelihood that you will need to restore the data again. However, if a coworker only notices months later that, say, important

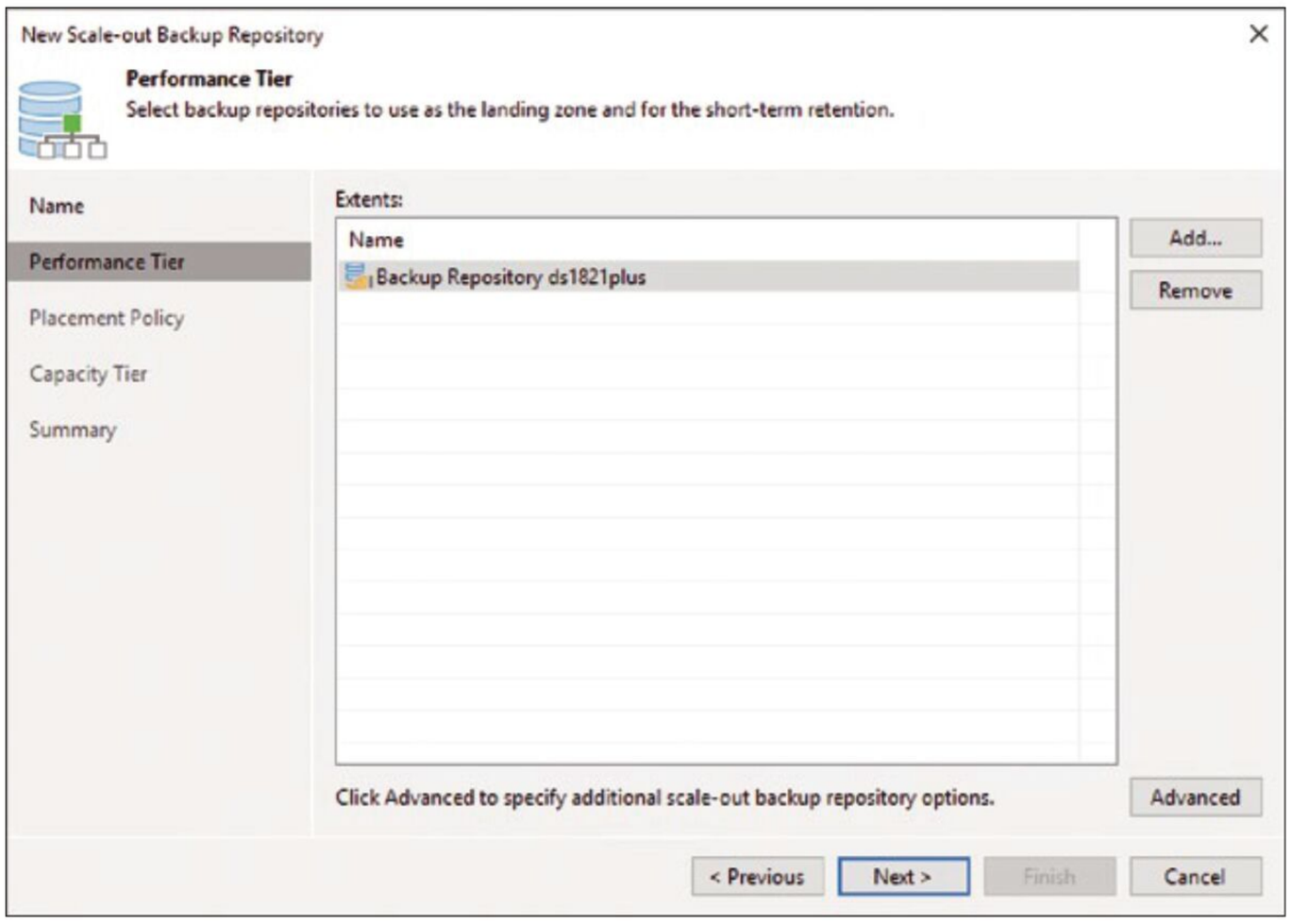


Figure 1: The structure of storage tiers (here, a Synology DS1821+ in the demo lab with Veeam) ensures that backups are distributed over time to storage media of different costs.

files on the file server have been mistakenly overwritten or are completely missing, they will be glad the data is still available somewhere, even if a restore takes longer than a few minutes.

The process of moving data between the different backup tiers should happen automatically so that no manual intervention is necessary. Professional backup software supports you in these steps and offers this feature by default. Be sure to pay attention to the editions available: Depending on the software, these functions are more likely to be included in the more expensive editions and may have to be licensed subsequently or additionally.

Optimal Hardware and Software Combination

The Veeam Backup & Replication product [1] has achieved a considerable market share in just a few years and enjoys a very good reputation. Coupled with a well-thought-out hardware configuration, you can build a backup infrastructure that is highly scalable and benefits from the hardware performance already mentioned (Figure 1).

The smallest and simplest setup combines software and storage directly in a single system. Here, the data carriers are either operated directly in the server, or you can connect one or more external JBOD (just a bunch of disks) enclosures and thus greatly increase the number of data carriers. Depending on the model, between 12 and 72 data carriers will fit and can then be combined into one or more pools in a classic approach with a RAID controller.

Alternatively, you can use S2D, wherein each hard disk, connected by a non-RAID controller (host bus adapter, HBA) and individually visible on the Windows Server operating system, is included in a storage pool; on the basis of this pool, you can then create virtual disks. The advantage of this approach is that you are not tied to the performance of the RAID controller. Furthermore, with a Windows Storage Spaces pool, you can combine several data carrier types to use the flash memory strategically as a cache.

If you size the server adequately in terms of CPU, RAM, and network, it can also act as a VMware backup proxy when using Veeam. On Hyper-V, this would be an alternative to using the resources of the respective Hyper-V host.

If a single system is too risky for you or you need more storage space, one option is to operate a failover cluster that is responsible for storing the backup data. Since Windows Server 2016, this has been available in S2D. Note that you need the Datacenter Edition to run it because the Standard Edition does not offer this feature. With ordinary server systems and locally attached disks, you can build highly available and scalable storage. The setup requires a minimum of two nodes with a maximum of up to 16 servers per cluster, which supports a storage capacity of up to 4PB per cluster that is available exclusively for your backup data, if so desired.

If this setup is not enough, or if you want to set up two storage clusters in different locations or fire zones, you can add the clusters to a scale-out repository in the Veeam software. This technique assembles individual storage locations and devices to create a logical storage target from which you can add or remove any number of backup stores in the background, without changing anything for the backup jobs. This setup gives you a vector for flexible growth without having to check all backup jobs every time you expand or, in the worst case, move multiple terabytes of data.

ReFS is the filesystem used in an S2D cluster. Veeam works with this filesystem, and this shared use offers you some advantages in your daily work. With support for metadata operations, lengthy copy processes on the drive are not required; instead, new pointers are directed at the data blocks and are especially noticeable in backup chains. For example, if you create an incremental backup for 14 days, the change compared with day 14 must be saved on day 15, and the data from day 1 must be moved to the file for day 2. Depending on the size and performance of the storage, this process can sometimes take hours on an NTFS volume. On ReFS, the data on the volume is not moved; references to the location already in use are used. This process is very fast and usually completed within one to two minutes.

Another advantage is that when making weekly, monthly, or yearly backups that create a complete backup file in memory each time, the data is not completely rewritten but references the existing blocks – saving time and storage space.

Money or Data Loss!

Backups also have options to protect your data against ransomware by storing the backup on additional storage (tier 3), which can be on-site, in another building, another city, another country, or outsourced to a cloud provider. Depending on the type and nature of the storage, it might not be possible to modify data after the fact. As a result, malware cannot delete or encrypt the stored backups (Figure 2).
Examples of this type of storage include a deduplication appliance or object storage (e.g., Amazon Simple Storage Service (AWS S3) storage), where each object is given an attribute that specifies how long it cannot be modified.
If you use this type of storage, either the data can be written directly from tier 1 to tier 2 and tier 3, or the data from tier 2 can be moved downstream to tier 3. However, you do need to take into account how old the backup data was at the moment of removal and whether it is already too old for disaster recovery.

Always Encrypt

Always encrypt your backups, whether they are stored on-site or with a third-party provider or in cloud storage. Encryption means you do not have to worry about who else has access to the data. If the data is lost or stolen without authorization, a password is required for decryption. The password must be unique and as long as possible, and at least one copy must be available offline (e.g., sealed in a safe or deposited with a public notary).
Many companies still rely on tape backup to achieve the "air gap." Because such data can only be actively deleted or manipulated when the tape is inserted, you have a very high level of protection against external and internal attacks. The disadvantage of such a backup is that you have to take care of the tapes regularly and manually move monthly or annual backups to a safe or locker.
If you are considering outsourcing your data to a cloud provider, you need to clarify a few things in advance and include them in your backup strategy, including the available bandwidth, among other things. If an asymmetrical line is used, the upload capacities are usually very limited, and it is technically impossible to store the backup data quickly on the provider's facilities.
Cloud storage is often used to archive backup data that needs to be stored for years or even decades. Do not underestimate the costs for this time, and research your options carefully. The choice of provider also plays a major role. In some cases, plain vanilla data storage is very inexpensive, but retrieving the data then costs several times the storage fee.
Once you have chosen a provider, migrating the backups to another provider after the fact usually involves high costs. Also bear in mind that outsourcing data may oblige you to continue the original storage subscription, even if you have long since changed the primary provider used.
Cloud storage also offers some advantages. Even small companies without a large IT budget can store their data quickly and easily at a second location without investing in additional hardware and software. If an incident occurs on-site and the local systems are compromised, destroyed, or stolen, the data is safe and still accessible in one or more data centers.

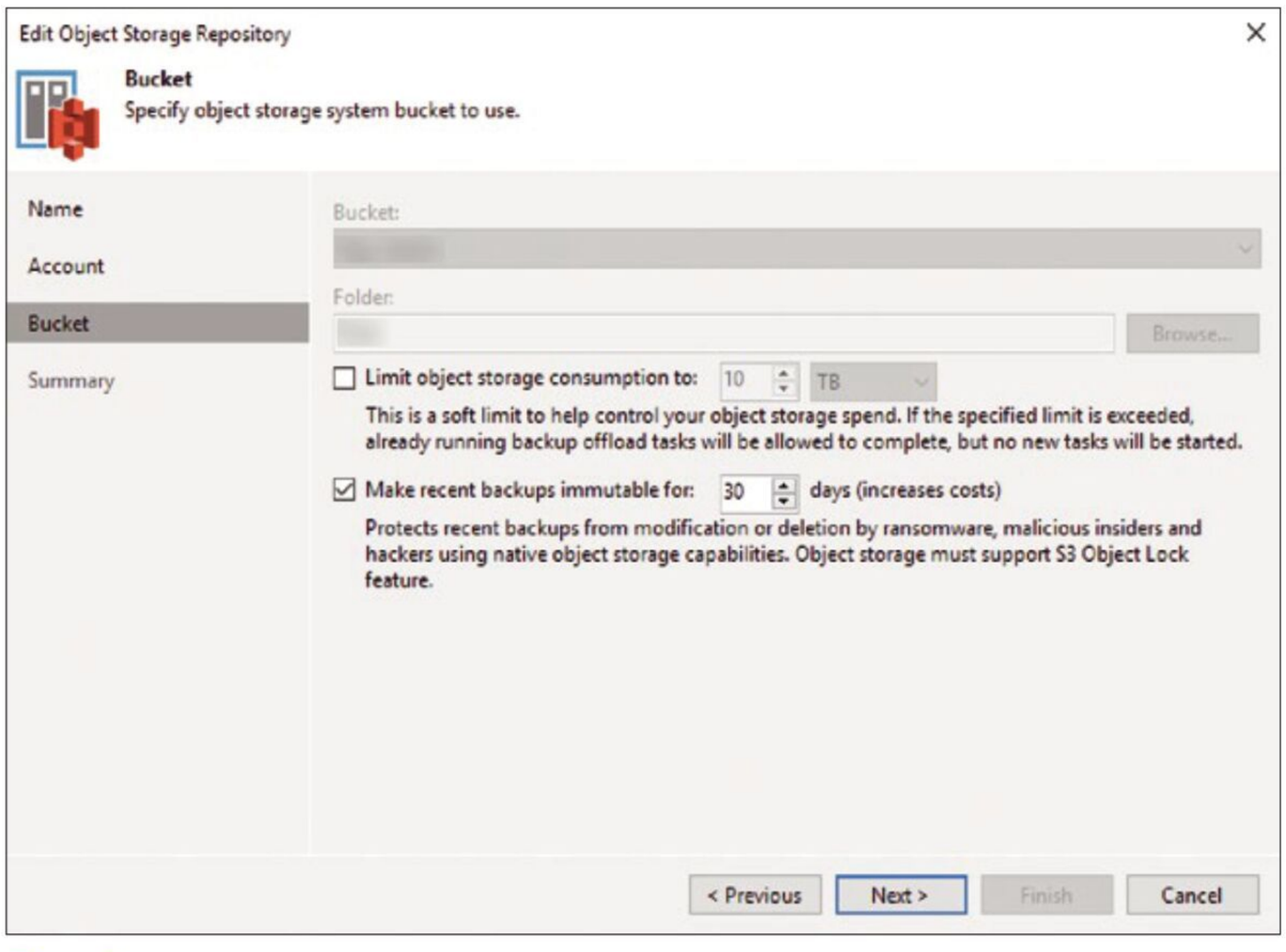


Figure 2: A third storage tier with an immutable attribute can protect against ransomware.

Conclusions

A coherent backup strategy, coupled with a good combination of hardware and software, enables your data to be restored quickly and reliably. Outsourcing to another location additionally ensures that operations can be resumed on-site, even in the event of a major disruption.
High-performance backup not only relieves the burden on the infrastructure in production environments but often also empowers you to create backups at times when this was previously not possible. In companies for which IT must be available 24/7, this can have an extremely positive effect. ■

Info

[1] Veeam: [https://www.veeam.com]

Manage cluster state with Ceph dashboard

Not Just a Pretty Face

The Ceph dashboard offers a visual overview of cluster health and handles baseline maintenance tasks; with some manual work, an alerting function can also be added. By Martin Loschwitz

One criticism directed at software-based storage solutions is that they lack functional management tools. On the other hand, if you have ever dealt with typical storage area network (SAN) or network attached storage (NAS) appliances from the established manufacturers, you know you can get a web interface with a few virtual traffic lights that show the data status as a signal color. If the light is green, you can sleep soundly knowing that your data is fine. These management tools by the established manufacturers not only provide information about the data status, they let you carry out certain operations in a safe way. For example, if you want to set up a logical unit number (LUN) in a SAN, the graphical wizard will guide you through this process without any glitches.

To solve the problem for Ceph, the developers have been working for several years on the Ceph Dashboard, which is now an integral part of Ceph and has undergone a great deal of development since it was once launched as a fork of the openATTIC [1] storage management system. That said, the tool is still largely unknown to many administrators. In this article, I introduce the Ceph Dashboard, show how to activate it, discuss the information to be gleaned from it, and demonstrate the maintenance tasks the dashboard performs on demand.

Well Integrated

The good news right away is that the dashboard is very well integrated with Ceph. If you are using a recent Ceph cluster, you may already be running the dashboard without realizing it because, over the years, Ceph has undergone several radical changes to its own toolchain. The latest development is the management framework for Ceph, which is a kind of orchestration service specifically tailored to Ceph and its needs. The new deployment tool, `cephadm` [2], is also based on the management framework in the background, as is the dashboard, `ceph-mgr` [3]. The somewhat unwieldy short form of the framework is now part of the standard installation in a Ceph deployment, and most Ceph products install the dashboard at the same time.

However, you are not completely ready to go yet because, depending on the local specifications, it may be necessary to execute a few additional commands relating to the Ceph dashboard. In the following sections, we look at how you can get the dashboard started in the configuration that is ideal for you.

Finding the Ideal Configuration

To begin, you face the task of finding out which hosts are running an instance of the Ceph Manager daemon (`ceph-mgr`). These are the hosts that are running an instance of the dashboard. Contrary to what you might expect, the Ceph dashboard does not come as a clustered service. Consequently, it is also necessary to configure the Ceph dashboard per manager instance, not globally for the cluster. To identify the hosts running the manager component, just run the `ceph-mgr` command on each host where the `ceph` command works. The MON servers are generally the safest bet. A MON server is a kind of cluster watchdog in the Ceph context. It enforces a quorum for cluster partitions to prevent split-brain situations and keeps track of all existing MON and object storage daemon (OSD) services. OSD is a Ceph-owned service that turns any block storage device into a volume usable by Ceph.

For each instance of the Ceph dashboard, you then configure the IP address,

```
ceph config set mgr ❢
  mgr/dashboard/<NAME>/server_addr
```

replacing `<NAME>` with the name of the Ceph Manager daemon instance and,

Lead Image © damedeeso,123rf.com

ideally, leaving the ports for the connection over HTTP(S) untouched – unless you are dealing with a complicated firewall configuration. Next, run the command

```
ceph config set mgr ➥
 mgr/dashboard/<NAME>/ssl_server_port
```

If you want to access the individual Ceph dashboards, you have to use their respective IP addresses.

Get Your Own SSL Certificate

Because you log in to the dashboard with a username and password combination, it is obvious that any communication between the browser and the dashboard needs to be encrypted. The Ceph Manager daemon sets this up out of the box, but it uses a self-issued and self-signed SSL certificate. If you need an SSL certificate issued by the in-house certificate authority (CA) or even an official CA, you have to replace the Ceph Manager daemon certificate. The example below assumes that a wildcard certificate for `*.example.net` exists in the `cert.pem` file and that its unprotected key exists in `key.pem`. If an intermediate certificate is required for the SSL CA, this must also be available in `cert.pem`. The installation is then simple:

```
ceph config-key set mgr ➥
  mgr/dashboard/crt -i cert.pem
ceph config-key set mgr ➥
  mgr/dashboard/key -i key.pem
```

The commands

```
ceph mgr module disable dashboard
ceph mgr module enable dashboard
```

restart the `mgr` component and the dashboard. After that, the dashboard is available with an official SSL certificate if you call it using the correct hostname.

Dashboard Overview

Because the dashboard has seen many new features over the past few years, it can be a little difficult to get started. It won't hurt to familiarize yourself with the various menu items after the initial dashboard login. A key new feature in the Octopus release of Ceph was the introduction of the navigation bar on the left side of the screen. This feature helps with organization because it divides the most important menu items into clusters and displays them coherently.

The *Cluster* item (Figure 1), for example, hides entries that refer to the components of the RADOS object storage, including the MON and OSD services; all disks used; the logfiles generated by the services; and the configuration of the Ceph Manager daemon component. Additionally, it will also give you some insight into the continuous replication under scalable hashing (CRUSH) map, which describes the algorithm Ceph uses to distribute data to the available disks on the basis of specific rules and creates replicas of the disks. The CRUSH map lets you determine which logical structure the cluster follows (i.e., which hard drives belong to which servers, which servers belong to which firewall zones, etc.).

Navigating to the *Pools* entry in the side menu on the left takes you to the setup of pools. They are a kind of logical division: Each binary object in the Ceph cluster belongs to a pool. Pools act kind of like name tags; they also allow Ceph to implement much of its internal replication logic. In fact, it is not the objects that belong to a pool but the placement groups. A pool is therefore always a specific set of placement groups with a defined number of objects. By creating, deleting, and configuring pools, which is possible from this entry in the dashboard, you can define several details of the cluster, including the total number of placement groups available, which affects the performance of the storage, as well as the number of replicas per pool.

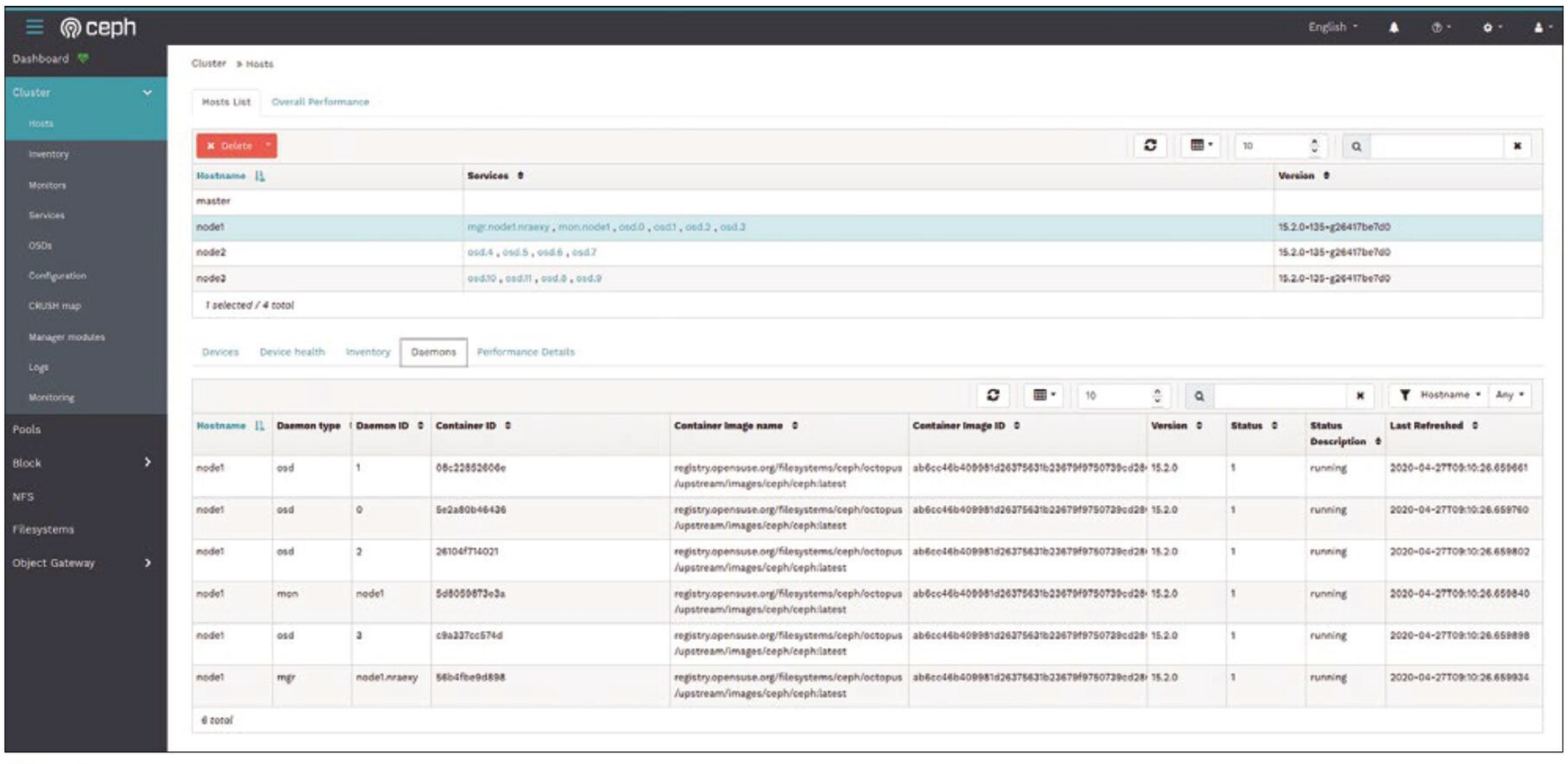


Figure 1: The Ceph dashboard shows the running services and their configuration for each host.

Menu Settings

The next set of menu items deals with the front ends for Ceph that the user needs for access. Under *Block*, you can access an overview of the configured virtual RADOS block devices (RBDs), although this is more for statistical purposes because RBD volumes are usually set up autonomously within the framework of their specifications without you having to do anything with them. Similar restrictions apply to the menu items *NFS* and *Filesystems*, which provide insights into the statistics relating to Ganesha, a userspace NFS server, and CephFS, a POSIX-compliant filesystem built on top of RADOS. Here, too, you will tend to just watch rather than touch.

The *Object Gateway* item, on the other hand, allows a bit more interaction. Here, you can access the configuration of the Ceph Object Gateway, which is probably still known to many admins under its old name, RADOS Object Gateway (RGW). Depending on the deployment scenario, the object gateway comes with its own user database, which you can influence through the menu.

You can see the current status of the cluster in the dashboard. You can also handle a large number of basic maintenance tasks without having to delve into the depths of the command line and the `ceph` tool. Because Ceph has gained more and more functions in recent years, its commands have also had to become more comprehensive. Newcomers might find it difficult to make sense of the individual commands that Ceph now handles. The dashboard provides a much-needed bridge for the most basic tasks.

Monitoring Hard Disks

Although most administrators today want a flash-only world for their storage, this is not yet the reality in most setups. If you want a large amount of mass storage, you will still find a lot of spinning metal. In the experience of most IT professionals, these are the components most likely to fail. Anyone running a Ceph cluster with many hard disks will therefore be confronted with the fact that hard drives break down sooner or later. The dashboard helps you in several places. Although the self-monitoring, analysis, and reporting technology (S.M.A.R.T.) does not work 100 percent for all devices, certain trends can be read from the disks' self-monitoring. For this reason, the developers of the Ceph dashboard integrate S.M.A.R.T. data into the GUI and prominently display any of its warnings in the dashboard. You can access the overview by first selecting the respective host and the respective disk and then clicking on *SMART* in the *Device health* tab (Figure 2). S.M.A.R.T. information is immediately recognizable in another place in the dashboard: If a hard disk is in a questionable state of health according to the S.M.A.R.T. information, Ceph evaluates the data in the background and outputs a corresponding warning. This information appears prominently on the start page in the Cluster Health section. By the way, you can get the same output at the command line by typing `ceph -w` or `ceph health`.

Creating Graphics with Grafana

In terms of the dashboard, the Ceph developers didn't start from nothing. Instead they rely on existing functionality. The Ceph Manager daemon also rolls out several Docker containers along with the Ceph dashboard: one for the Prometheus time series database and one for Grafana.

Ceph has its own interface that outputs metrics data in a Prometheus-compatible format. The Prometheus container rolled out by the Ceph Manager daemon taps into this, and the Grafana container, also rolled out, then draws graphics from these metrics on the basis of preset values.

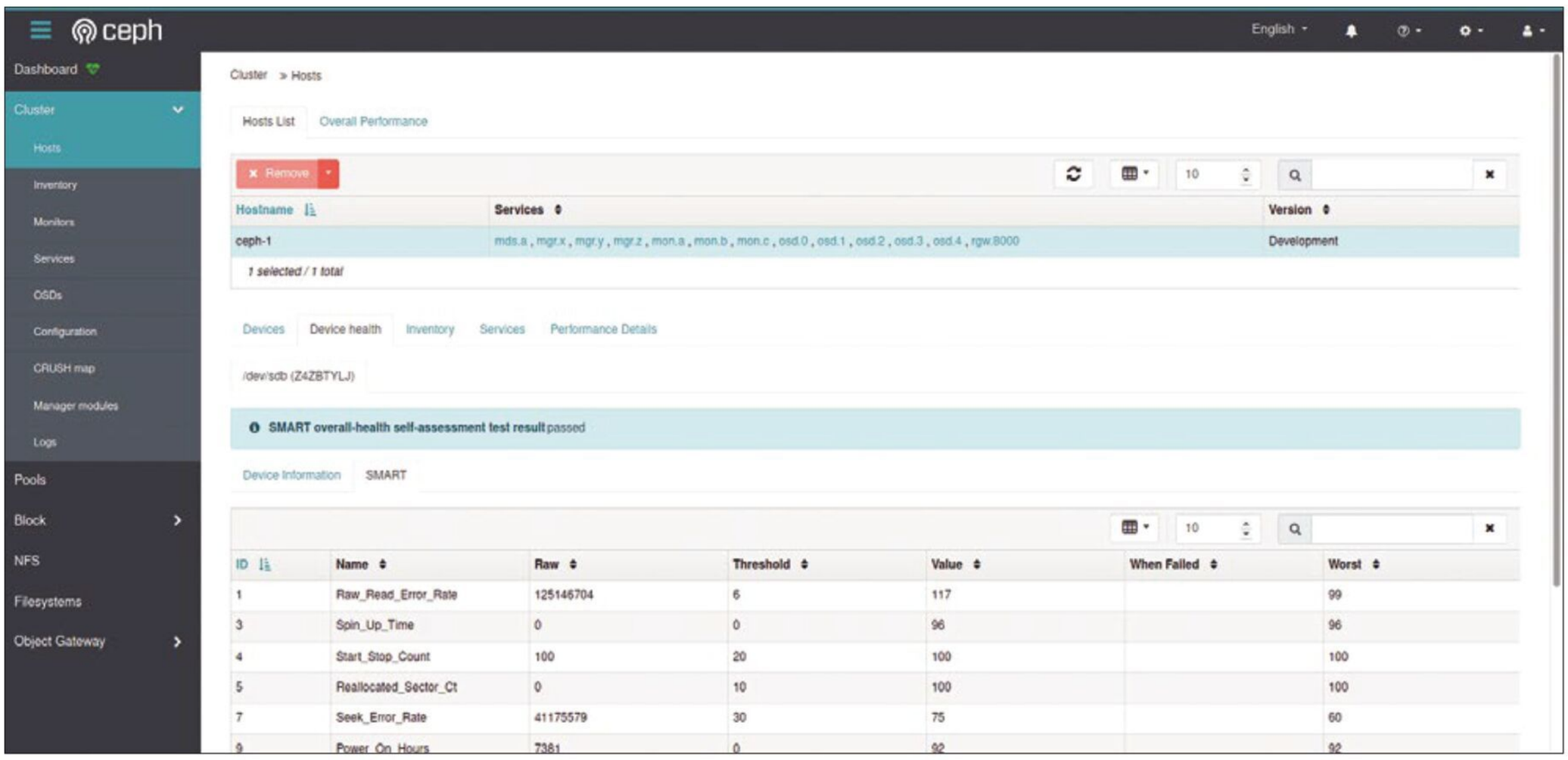


Figure 2: Ceph can read S.M.A.R.T. information from recent devices and, if necessary, issue an alert, which the dashboard then displays.

The dashboard ultimately embeds the graphics from Grafana with an iFrame, and the drawing job is done. In any case, you shouldn't panic if you suddenly find Docker instances running on your Ceph hosts – that's quite normal. More than that, the latest trend is, after all, to roll out Ceph itself in a containerized form. Consequently, not only do Prometheus and Grafana run on the servers but so do the containers for MONs, OSDs, and the other Ceph services.

The original focus of the dashboard was to display various details relating to the Ceph cluster. In recent years, however, this focus has shifted. Today, the dashboard also needs to support basic tasks relating to cluster maintenance, facilitated by levers and switches in various places on the dashboard that can be used to influence the state of the cluster actively, as the next section demonstrates.

Creating OSDs, Adding Storage Pools

Consider a situation in which you want to scale a cluster horizontally – a task that Ceph can easily handle. The first step (at least for now) is to go back to the command line to integrate the host into Ceph's own orchestration. On a system where Ceph is already running, the command

```
ceph orch host add <hostname>
```

will work. The host will now show up on the dashboard, along with a list of storage devices that can become OSDs. In the next step, you then add the OSDs to the cluster from the *Cluster* | *OSDs* menu. To do this, select the devices on each host that will become OSDs (Figure 3) and confirm the selection. The remaining background work is again taken care of by the Ceph Manager daemon. Unlike creating OSDs, you do not need to add hardware when creating a pool in Ceph. You can add an additional pool to the existing storage at any time; likewise, you can remove existing pools at any time from the previously mentioned *Pools* menu in the left overview menu.

Displaying Logs

Anyone who deals with scalable software will be familiar with the problem that, in the worst case, an error on one host is directly related to an error on another host. To follow the thread of error messages, you comb through the various logfiles on the different systems one by one, moving from one file to the next, a tedious and time-consuming effort. In large environments, it is therefore normal to collect logfiles centrally, index them, and so make them searchable. The Ceph dashboard at least implements a small subset of this functionality. In the *Cluster* | *Logs* menu item you will find the log messages from the various daemons involved in the cluster, as well as status messages from Ceph itself. Here you can efficiently search for an error message without having to go through all the servers in your setup.

Creating Alerts

Generating alerts directly from the dashboard (e.g., for the admin on standby who is responsible for the health of a Ceph cluster) is also possible. As mentioned earlier, most of the dashboard's monitoring functionality is based on containerized instances of

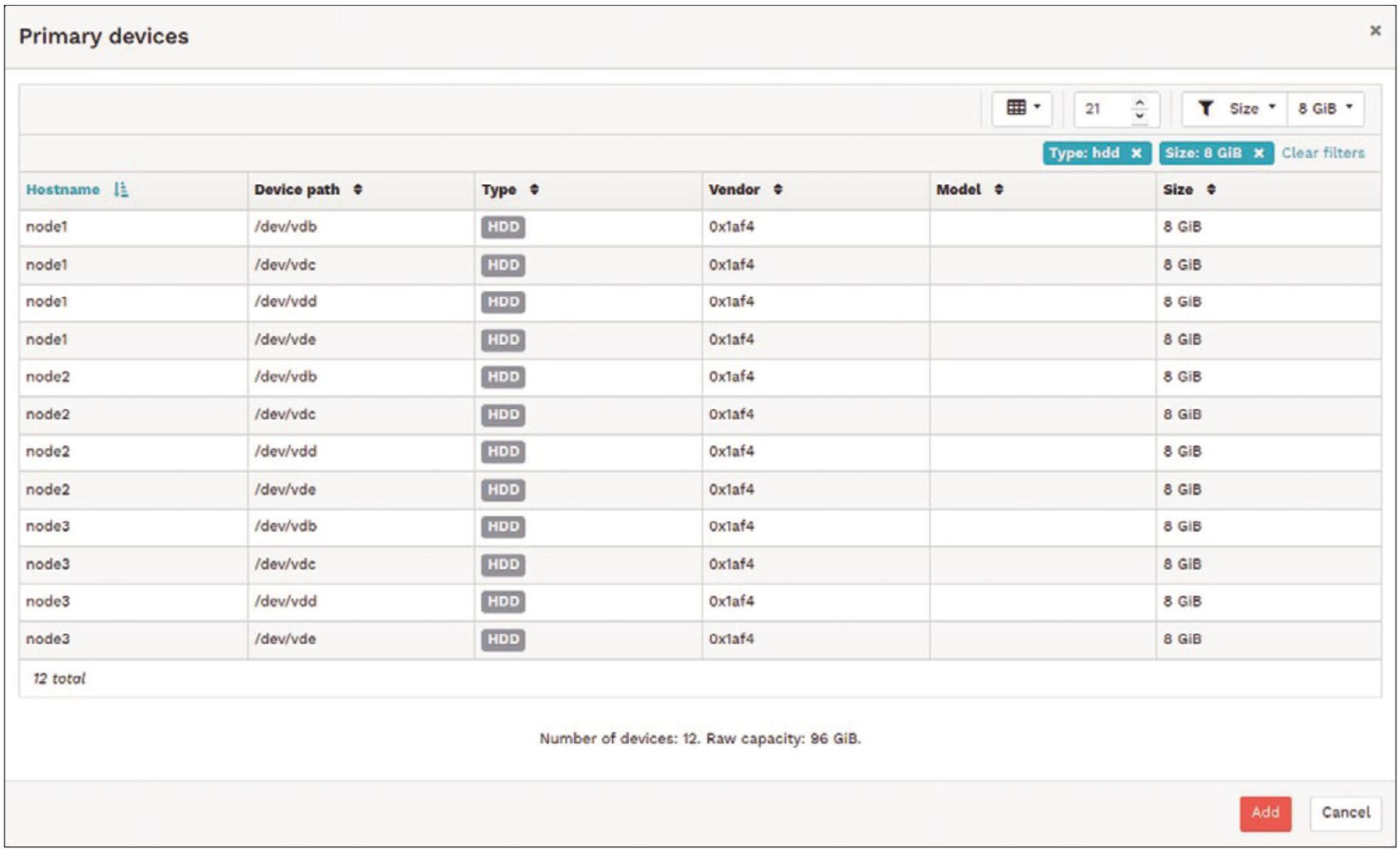


Figure 3: The dashboard is no longer considered just a graphical view but also an admin tool, as seen here when creating OSDs.

Prometheus and Grafana in the background. The fact that Ceph comes with a built-in interface to provide metrics data in Prometheus format is, of course, extremely convenient. What the dashboard does not deliver so far, however, is the Prometheus component to generate and send alerts – the Alertmanager [4]. With a little manual work, you can quickly retrofit this element.

Because the Prometheus developers also offer Alertmanager as a container, this technique even works on servers that are already running the Prometheus and Grafana containers from the `ceph-mgr` component. Instructions are provided by the Prometheus developers online [5]. Predefined alerts for Ceph clusters can also be found online [6]. The rest then just involves putting the puzzle together: In the Alertmanager configuration, you need to add the alerting targets and store the alerts that wake up the Alertmanager in its configuration. Finally, you need to enable the ability to generate alerts through the dashboard by telling it the URL on which it can reach the Alertmanager:

```
ceph dashboard ➥
  set-alertmanager-api-host ➥
  'http://localhost:9093'
```

The rest is then quite simple. The Alertmanager receives alerts from Prometheus directly from the Ceph dashboard and forwards them over the configured channels. Admittedly, such a construct has the disadvantage that it is an isolated solution because it only works for Ceph. In return, however, you get a very granular, powerful monitoring and alerting tool for Ceph.

Conclusions

The former openATTIC module has evolved into a comprehensive Ceph-monitoring environment, which the developers are continuously developing. People who deride the dashboard as nothing more than a colorful appendage are doing it an injustice: The ability to get a quick, visual overview of the cluster's status is particularly helpful in emergency situations.

By the way, the dashboard can certainly change its visual appearance depending on the product with which it is rolled out. The developers have also made sure that the Ceph dashboard can be visually adapted to a manufacturer's specifications with a theme. On SUSE, it accordingly presents itself in green (Figure 4), whereas the standard version uses the classic Ceph colors instead. Whatever the color, though, the functionality always remains the same. ■

Info

[1] openATTIC: [https://documentation.suse.com/en-us/ses/5.5/html/ses-all/ceph-oa.html]

[2] cephadm: [https://docs.ceph.com/en/latest/cephadm/index.html]

[3] ceph-mgr: [https://docs.ceph.com/en/latest/mgr/index.html]

[4] Prometheus Alertmanager: [https://prometheus.io/docs/alerting/latest/alertmanager/]

[5] Roll out Alertmanager: [https://prometheus.io/docs/alerting/latest/alertmanager/]

[6] Preconfigured alerts for Ceph in Prometheus: [https://awesome-prometheus-alerts.grep.to/rules#ceph]

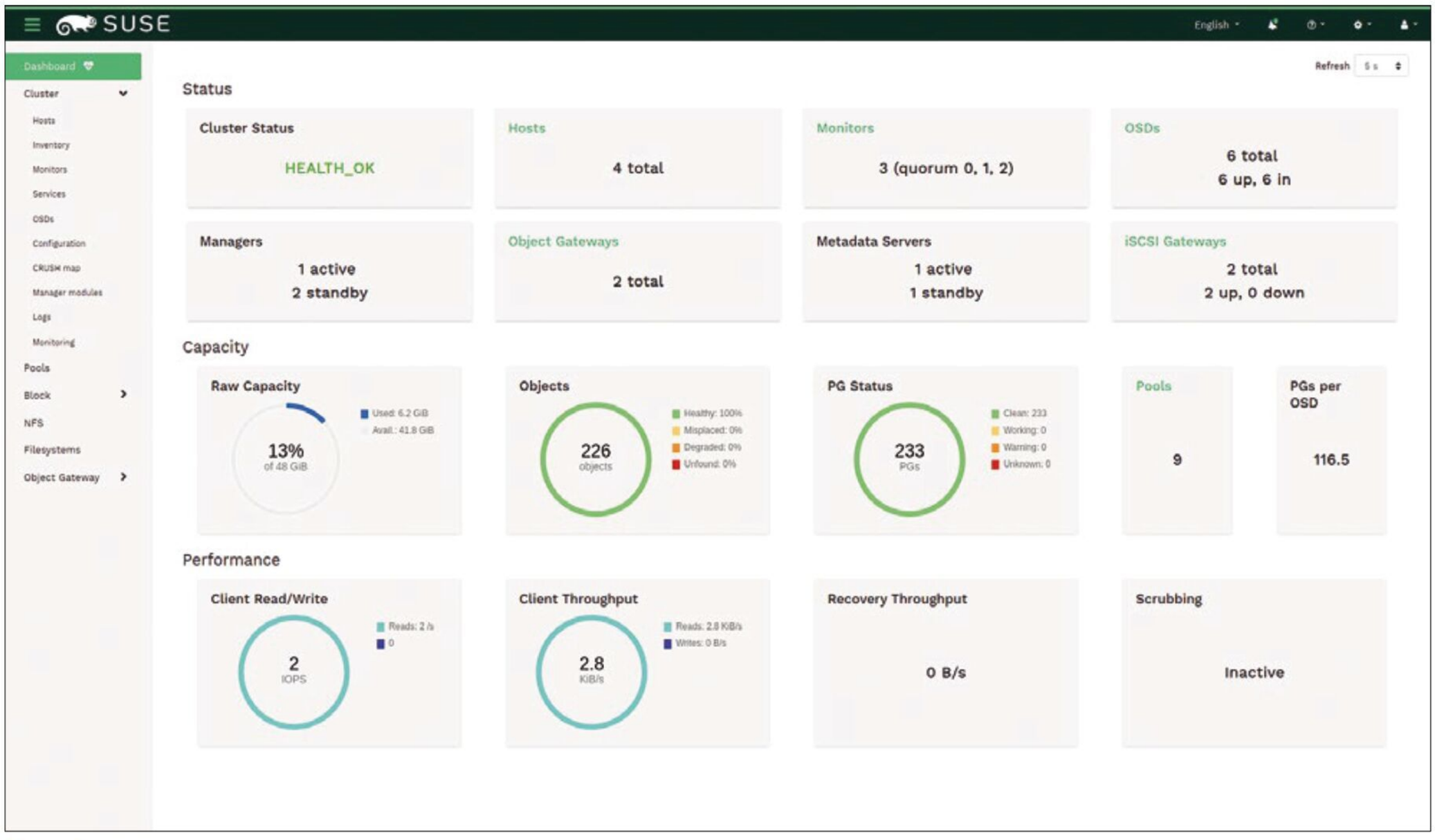


Figure 4: The dashboard supports different themes. It looks a bit different on SUSE than in the original version, but the functionality remains the same.

SeaGL 2021 JOIN US!

Seattle GNU/Linux Conference returns for the ninth year!

We're an annual community-focused free/libre/open source software, hardware, and culture event in Seattle, and since last year, *virtually* all over the world! Join us for free, no registration required!

This year, alongside the keynotes and many presentations, will see the return of the **Career Expo** - providing resume reviews and career guidance, **TeaGL** - our virtual tea time and online tea swap, and many more socialization opportunities!

Interested in helping make SeaGL happen? Lend a wing to our *all volunteer effort*! Connected to a company who wants to sponsor? Contributions will have a *real and lasting impact* on the FLOSS community!

November 5th & 6th, 2021
Virtual Conference

Visit SeaGL.org for more information!

Darshan I/O analysis for Deep Learning frameworks

Looking and Seeing

Characterizing and understanding I/O patterns in the TensorFlow machine learning framework with Darshan. By Jeff Layton

The Darshan [1] userspace tool is often used for I/O profiling of HPC applications. It is broken into two parts: The first part, *darshan-runtime*, gathers, compresses, and stores the data. The second part, *darshan-util*, postprocesses the data.

Darshan gathers its data either by compile-time wrappers or dynamic library preloading. For message passing interface (MPI) applications, you can use the provided wrappers (Perl scripts) to create an instrumented binary. Darshan uses the MPI profiling interface of MPI applications for gathering information about I/O patterns. It does this by "… injecting additional libraries and options into the linker command line to intercept relevant I/O calls" [2] (section 5.1).

For MPI applications, you can also profile pre-compiled binaries. It uses the `LD_PRELOAD` environment variable to point to the Darshan shared library. This approach allows you to run uninstrumented binaries for which you don't have the source code (perhaps independent software vendor applications) or applications for which you don't want to rebuild the binary.

For non-MPI applications you have to use the `LD_PRELOAD` environment variable and the Darshan shared library.

Deep Learning (DL) frameworks such as TensorFlow are becoming an increasingly big part of HPC workloads. Because one of the tenets of DL is using as much data as possible, understanding the I/O patterns of these applications is important. Terabyte datasets are quite common. In this article, I take Darshan, a tool based on HPC and MPI, and use it to examine the I/O pattern of TensorFlow on a small problem – one that I can run on my home workstation.

Installation

To build Darshan for non-MPI applications, you should set a few options when building the Darshan runtime (`darshan-runtime`). I used the `autoconf` command:

```
./configure ➥
  --with-log-path=/home/laytonjb/➥
                  darshan-logs ➥
  --with-jobid-env=NONE ➥
  --enable-mmap-logs ➥
  --enable-group-readable-logs ➥
  --without-mpi CC=gcc ➥
  --prefix=[binary location]
```

Because I'm the only one using the system, I put the Darshan logs (the output from Darshan) in a directory in my home directory (`/home/laytonjb/darshan-log`), and I installed the binaries into my `/home` directory; however, this is not a good idea on multiuser systems. The script preps the environment and creates a directory hierarchy in the log directory (the one specified when configuring Darshan). The organization of the hierarchy is simple. The topmost directory is the year. Below that is the month. Below that is the day.

After the usual, `make; make install`, you should run the command

```
darshan-mk-log.pl
```

which preps the log directory. For multiuser systems, you should read the documentation [2] (section 3.2).

Next, I built the Darshan utilities (`darshan-util`) with the command:

```
./configure CC=gcc ➥

  --prefix=[binary location]
```

Because I'm running these tests on an Ubuntu 20.04 system, I had to install some packages for the postprocessing (`darshan-util`) tools to work:

```
texlive-latex-extra
libpod-latex-perl
```

Photo by Michaela Murphy on Unsplash

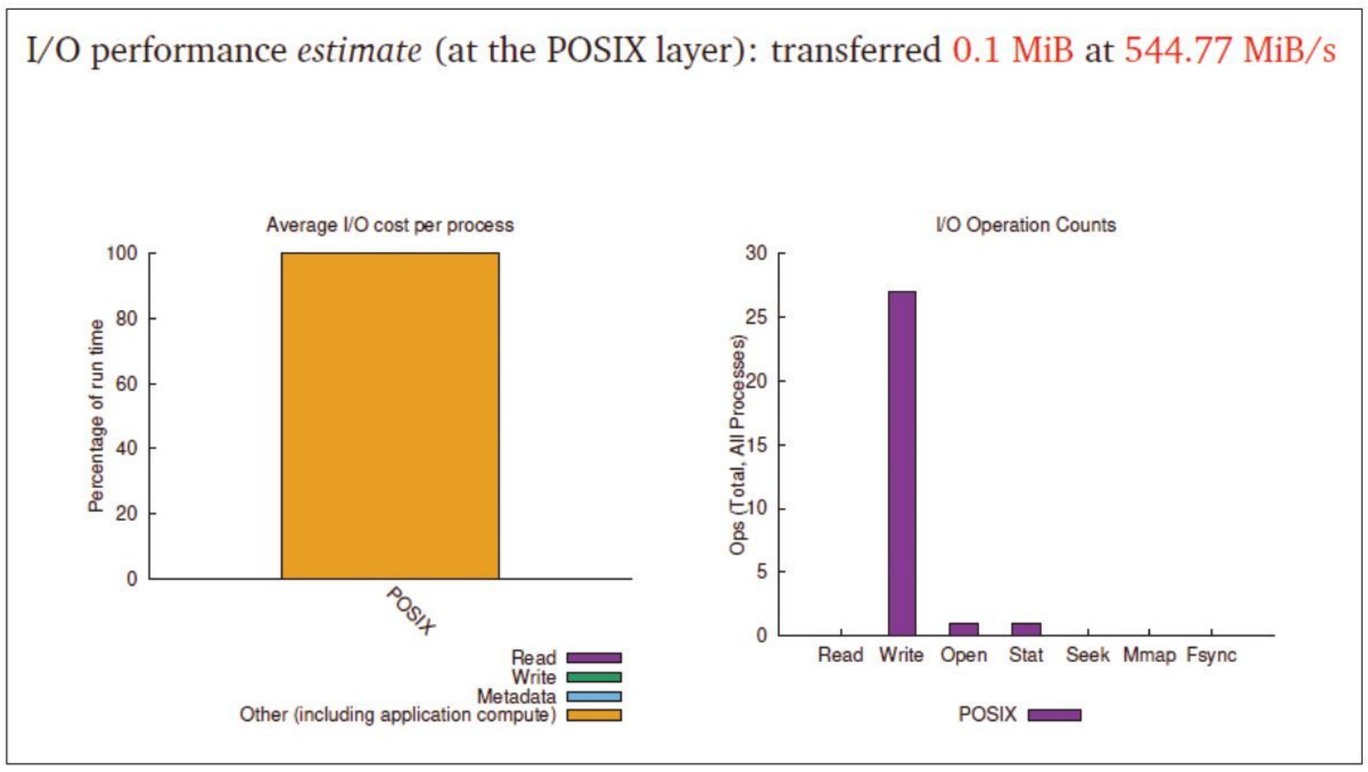


Figure 1: I/O performance from `darshan-job-summary.pl` output PDF.

Different distributions may require different packages. If you have trouble, the Darshan mailing list [3] is awesome. (You'll see my posts where I got some help when I was doing the postprocessing.)

Simple Example

Before jumping into Darshan with a DL example, I want to test a simple example, so I can get a feel for the postprocessing output. I grabbed an example from a previous article [4] (Listing 1). Although this example doesn't produce much I/O, I was curious to see whether Darshan could profile the I/O the application does create.

For this example, the command I used for Darshan to gather I/O statistics on the application was:

```
env LD_PRELOAD=/home/laytonjb/bin/➥
  darshan-3.3.1/lib/libdarshan.so ./ex1
```

I copied the Darshan file from the log location to a local directory and ran the Darshan utility `darshan-job-summary.pl` against the output file. The result is a PDF file that summarizes the I/O of the application.

Rather than include the entire PDF file in this article, I grabbed some of the plots and tables and present them here. Figure 1 shows a quick summary at the beginning of the output and says that it measured 0.1MiB of I/O. It also estimates the I/O rate at 544.77MiBps.

The left-hand chart presents the percentage of the run time for read, write, and metadata I/O and computation. For this case, it shows that the runtime is entirely dominated by computation. The read, write, and metadata bars are negligible and really can't even be seen.

The chart on the right presents the number of some specific I/O operations, with one open operation and 27 write operations. It also shows that all of the I/O is done by POSIX I/O functions [5].

The next snippet of the summary PDF is shown in the histogram in Figure 2, a plot of the typical payloads for read and write function calls (i.e., how many function calls use how much data per read or write per I/O function call). This chart shows that all of the write payloads are between 1 and 10KiB and that this example had no reads.

The job summary from Darshan also creates some very useful tables (Figure 3). The table on the left presents the most common payload sizes for

Listing 1: I/O Example Code

```
01 program ex1
02
03     type rec
04         integer :: x, y, z
05         real :: value
06     end type rec
07
08     integer :: counter
09     integer :: counter_limit
10     integer :: ierr
11
12     type(rec) :: my_record
13
14     counter_limit = 2000
15
16     ierr = -1
17     open(unit=8,file="test.bin", status="replace", &
18          action="readwrite", &
19          iostat = ierr)
20     if (ierr > 0) then
21         write(*,*) "error in opening file Stopping"
22         stop
23     else
24         do counter = 1,counter_limit
25             my_record%x = counter
26             my_record%y = counter + 1
27             my_record%z = counter + 2
28             my_record%value = counter * 10.0
29             write(8,*) my_record
30         end do
31     end if
32
33     close(8)
34
35 end program ex1
```

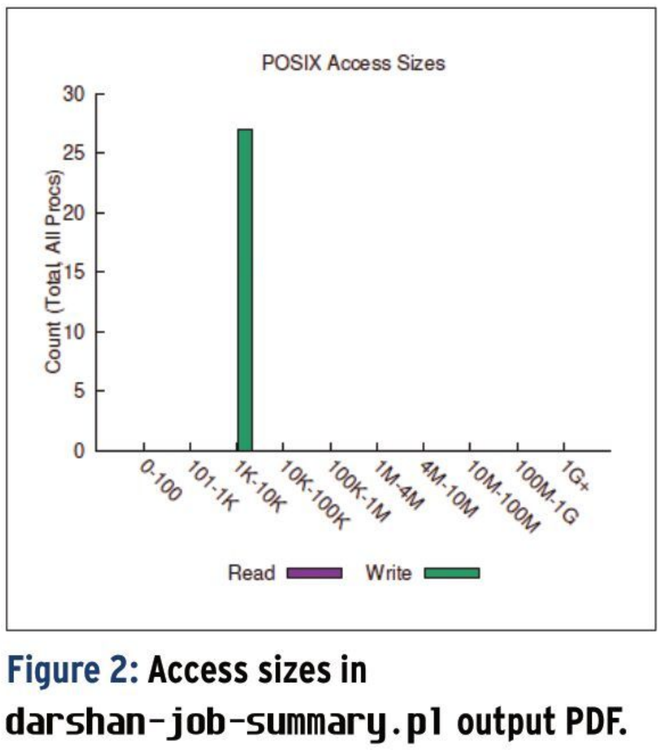


Figure 2: Access sizes in `darshan-job-summary.pl` output PDF.

Most Common Access Sizes (POSIX or MPI-IO)

	access size	count
POSIX	4104	26
	1296	1

File Count Summary (estimated by POSIX I/O access offsets)

type	number of files	avg. size	max size
total opened	1	106K	106K
read-only files	0	0	0
write-only files	1	106K	106K
read/write files	0	0	0
created files	1	106K	106K

Figure 3: Job summary tables from `darshan-job-summary.pl` output PDF.

Average I/O per process (POSIX and STDIO)

	Cumulative time spent in I/O functions (seconds)	Amount of I/O (MB)
Independent reads	0	0
Independent writes	0.000127	0.102996826171875
Independent metadata	6.2e-05	N/A
Shared reads	0	0
Shared writes	0	0
Shared metadata	0	N/A

Data Transfer Per Filesystem (POSIX and STDIO)

File System	Write		Read	
	MiB	Ratio	MiB	Ratio
/home	0.10300	1.00000	0.00000	0.00000

Figure 4: I/O stats from `darshan-job-summary.pl` output PDF.

POSIX I/O functions, with two access sizes, 4104 and 1296 bytes. The table on the right presents file-based I/O stats. Only one file was involved in this really simple example, and it was a write-only file. Note that the Darshan summary matches the source code; the one and only file was opened as write only. The table also shows that the average size of the file was 106KiB (the same as the maximum size).

Another group of useful tables is shown in Figure 4. The top table presents the cumulative time spent in reads and writes for both independent and shared operations. Don't forget that Darshan's origins are in MPI and HPC I/O, where shared files are common. It also presents information on how much I/O was performed for both reads and writes. Notice that the write time was really small (0.000127sec), and the I/O was also very small (0.103MiB).

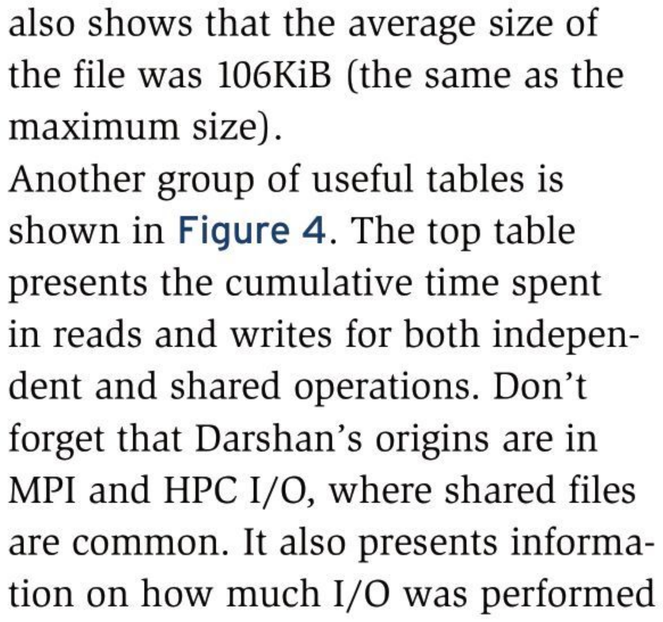


Figure 5: Read/write operations from the `darshan-job-summary.pl` output PDF.

Darshan also shows a great stat in the amount of time spent on metadata. Just focusing on read and write performance is not quite enough for understanding I/O. Metadata I/O can have a big effect, and separating it out from reads and writes is very useful.

The bottom table gives the total I/O for the various filesystems. This table is a bit more useful for HPC applications that use a scratch filesystem for I/O and a filesystem for storing the application binaries. Increasingly, DL applications are using this approach, so examining this table is useful.

The last snippet of the job summary I want to highlight is the number of read/write I/O operations (Figure 5). The figure shows only write operations, as expected. The total operations include all I/O functions.

Darshan with TensorFlow

Darshan has had a number of successes with MPI applications. For this article, I tried it on a TensorFlow framework, with Keras [6] loading the data, creating the model, training the model for only 100 epochs, checkpointing after every epoch, and saving the final model.

The system I'll be using is my home workstation with a single Titan V card, a six-core AMD Ryzen CPU, and 32GB of memory. I plan to use the CIFAR-10 data [7] and use the training code from Jason Brownlee's Machine Learning Mastery website [8]. I'll start the training from the beginning (no pre-trained models) and run it for 100 epochs. I've updated the training script to checkpoint the model weights after every epoch to the same file (it just overwrites it).

The code is written in Python and uses Keras as the interface to TensorFlow. Keras is great for defining models and training. I used the individual edition of Anaconda Python [9] for this training. The specific software versions I used were:

Table 1: Training Model

Layer (type)	Output Shape	No. of Parameters
`conv2d` (Conv2D)	(None, 32, 32, 32)	896
`conv2d_1` (Conv2D)	(None, 32, 32, 32)	9,248
`max_pooling2d` (MaxPooling2D)	(None, 16, 16, 32)	0
`conv2d_2` (Conv2D)	(None, 16, 16, 64)	18,496
`conv2d_3` (Conv2D)	(None, 16, 16, 64)	36,928
`max_pooling2d_1` (MaxPooling2)	(None, 8, 8, 64)	0
`conv2d_4` (Conv2D)	(None, 8, 8, 128)	73,856
`conv2d_5` (Conv2D)	(None, 8, 8, 128)	147,584
`max_pooling2d_2` (MaxPooling2)	(None, 4, 4, 128)	0
`flatten` (Flatten)	(None, 2048)	0
`dense` (Dense)	(None, 128)	262,272
`dense_1` (Dense)	(None, 10)	1,290
Total parameters		550,570
Trainable parameters		550,570
Non-trainable parameters		0

Listing 2: Training Script

```
export DARSHAN_EXCLUDE_DIRS=/proc,/etc,/dev,/sys,/snap,/run,/user,/lib,/bin,/home/laytonjb/anaconda3/lib/python3.8,/home/laytonjb/bin,/tmp
export DARSHAN_MODMEM=20000
env LD_PRELOAD=/home/laytonjb/bin/darshan-3.3.1/lib/libdarshan.so python3 cifar10-4.checkpoint.py
```

- Ubuntu 20.04
- Conda 4.10.3
- Python 3.8.10
- TensorFlow 2.4.1
- cudatoolkit 10.1.243
- System CUDA 11.3
- Nvidia driver 465.19.01

A summary of the model is shown in Table 1, with a total of six convolution layers, a maximum of three pooling layers, and the final flattening layer followed by a fully connected layer (dense) that connects to the final output layer for the 10 classes. The first two convolutional layers have 32 filters each, the second two convolutional layers have 64 filters each, and the final two convolutional layers have 128 filters each. The maximum pooling layers are defined after every two convolution layers with a 2x2 filter. The fully connected layer has 128 neurons. The total number of parameters is 550,570 (a very small model).

The command to run the training script with Darshan is shown in Listing 2. Running the training script takes several lines, so I just put them into a Bash script and run that.

When the training script is run, Python starts and Python modules are loaded, which causes a large number of Python modules to be converted (compiled) to byte code (`.pyc` files). Darshan can currently only monitor 1,024 files during the application run, and running the training script exceeded this limit. Because most of the files being compiled were Python modules, the Python directory (`/home/laytonjb/anaconda3/lib/python3.8`) had to be excluded from the Darshan analysis. Other directories were excluded, too, because they don't contribute much to the overall I/O and could cause Darshan to exceed the 1,024-file limit. The first line of the script excludes specific directories. The second line in the script increases the amount of memory the Darshan instrumentation modules can collectively use at runtime. By default, the amount of memory is 2MiB, but I allowed 19.53GiB (20GB) to make sure I gathered the I/O data.

Remember that the Darshan runtime just collects the I/O information during the run. It does not calculate any statistics or create a summary. After it collects the information and the application is finished, the Darshan utilities can be run. I used the `darshan-job-summary.pl` tool to create a PDF summary of the analysis.

The top of the PDF file (Figure 6) gives you some quick highlights of the analysis. The very top line in the output says that one processor was used, and it took 1,404sec for the application to complete. You can also see that the POSIX interface (the POSIX I/O functions) transferred 770.9MiB of data at 416.94MiBps. The STDIO interface (STDIO I/O functions) transferred 0.0MiB at 33.46MiBps. The amount of data transferred through the STDIO interface is so small that the output shows 0.0MiB, which would indicate less than 0.499MiB of data (the code would round this down to one decimal place, or 0.0).

The first snippet of the job summary is shown in Figure 7. The top left-hand chart shows that virtually all of the POSIX and STDIO runtime is for computation. You can see a little bit of I/O at the very bottom of the

jobid: 10719	uid: 1000	nprocs: 1	runtime: 1404 seconds

I/O performance *estimate* (at the POSIX layer): transferred 770.9 MiB at 416.94 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 0.0 MiB at 33.46 MiB/s

Figure 6: Analysis highlights from the `darshan-job-summary.pl` output PDF.

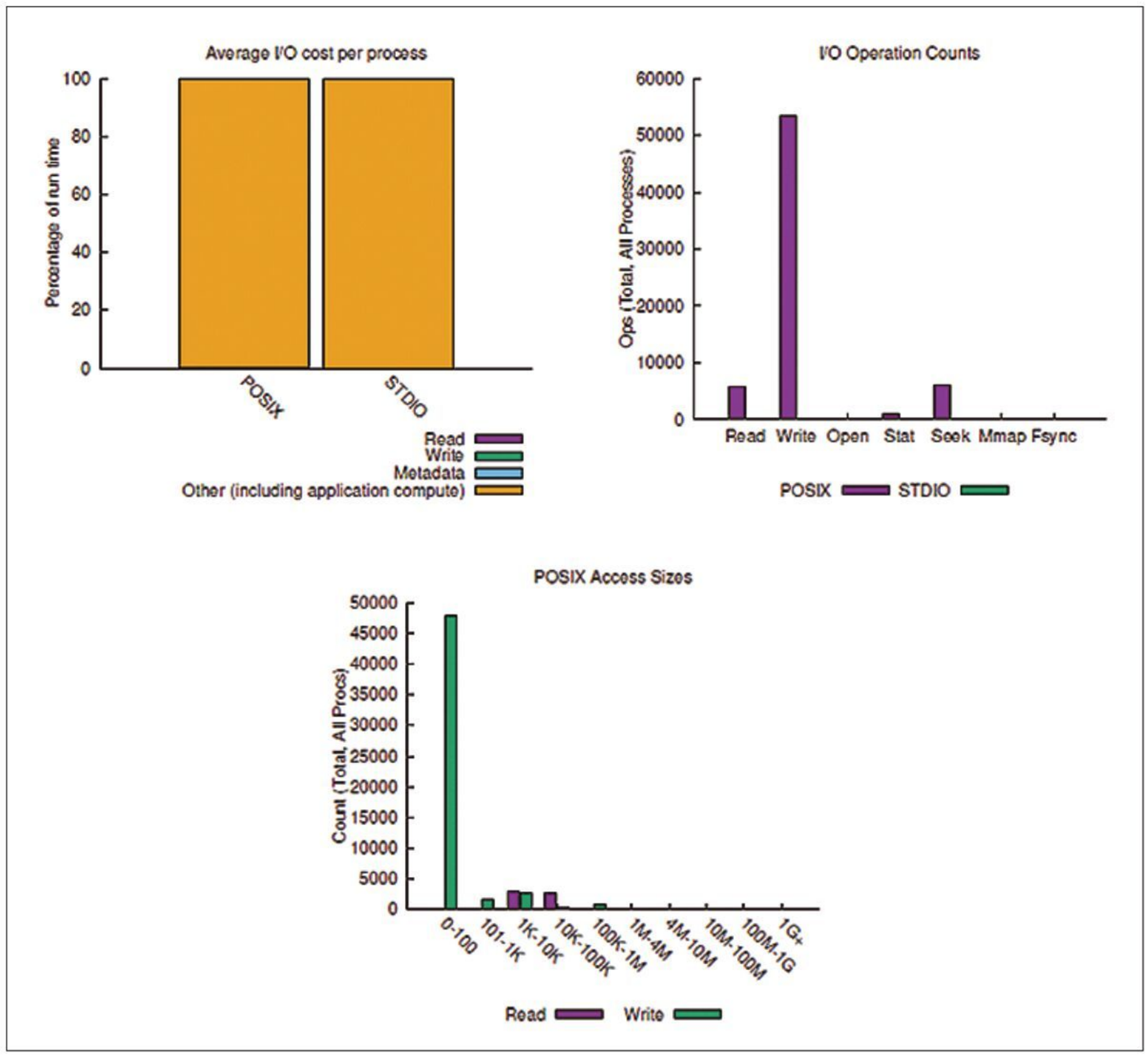


Figure 7: Read/write operations from `darshan-job-summary.pl` output PDF.

Most Common Access Sizes (POSIX or MPI-IO)

	access size	count
POSIX	85	23540
	84	17065
	86	6276
	4096	3069

File Count Summary (estimated by POSIX I/O access offsets)

type	number of files	avg. size	max size
total opened	14	25M	163M
read-only files	10	35M	163M
write-only files	3	2.7M	4.3M
read/write files	0	0	0
created files	3	2.7M	4.3M

Figure 8: Access sizes and file counts from `darshan-job-summary.pl` output PDF.

POSIX bar (the left bar in the chart). The top right-hand chart shows very little STDIO (green bars). The rest is POSIX I/O, dominated by writes.
Recall that Darshan has to be run excluding any I/O in the directory `/home/laytonjb/anaconda3/lib/python3.8` (the location of Python).
When Python converts (compiles) the source, it reads the code and writes the byte code (PYC extension). This I/O is not captured by Darshan. One can argue whether this is appropriate or not because this conversion is part of the total training runtime. On the other hand, excluding that I/O focuses the Darshan I/O analysis on the training and not on Python.
About 54,000 write operations occur during training. (Recall that a checkpoint is written after every epoch.)
About 6,000 read operations appeared to have occurred during the training, which seems to be counterintuitive because DL training involves repeatedly going through the dataset in a different order for each epoch. Two things affect this: (1) TensorFlow has an efficient data interface that minimizes read operations; (2) all of the data fits into GPU memory. Therefore, you don't see as many read operations as you would expect.
A few metadata operations take place during the training. The top-right chart records a very small number of `open` and `stat` operations, as well as a number of `lseek` operations – perhaps 7,500. Although there doesn't appear to be a noticeable number of `mmap` or `fsync` operations, if Darshan puts them in the chart, that means a non-zero number of each occur.
The bottom chart in Figure 7 provides a breakdown of read and write operations for the POSIX I/O functions. The vast majority of write operations are in the range of 0-100 bytes. A much smaller number of write operations occur in other ranges (e.g., some in the 101 bytes to 1KiB and 1-10KiB ranges, and a very small number in the 10-100KiB and 100KiB to 1MiB ranges). All of the ranges greater than 100 bytes have a much smaller number of write operations than the lowest range.
A small number of read operations are captured in that chart as well. Most read operations appear to be in the 1-10KiB range, with a few in the 10-100KiB range.
Notice that for both reads and writes, the data per read or write operation is small. A rule of thumb for good I/O performance is to use the largest possible read or write operations, preferably in the mebibyte and greater range. The largest part of the operations for this DL training is in the 1-100 byte range, which is very small.
Figure 8 presents two tables. The left-hand table presents the most common access sizes for POSIX operations. For reads and writes, this is the average size of the data per I/O function call. The right-hand chart presents data on the files used during the run.
The left-hand table shows that the vast majority of POSIX operations, predominately writes with a small number of reads, are in the 84-86 byte range. These account for 46,881 operations of the roughly 50,000 read and write operations (~94%). This table reinforces the observations from the charts in Figure 7.
The right-hand table shows that 14 files were opened, of which 10 were read-only and three were write-only. The average size of the files was 25MiB.
The next two tables (Figure 9) present more information about the amount of I/O. The top table first shows the amount of time spent doing reads, writes, and metadata. For the DL training problem, 0.063sec was spent on reads, 0.44sec on writes, and 1.35sec on metadata (non-read and -write I/O operations), or a total of 1.8491sec spent on some type of I/O out of 1,404sec of runtime (0.13%), which illustrates that this problem is virtually 100 percent dominated by computation.
The last column of the top table also presents the total amount of I/O. The read I/O operations, although only a very small number, total 340MiB. The write operations, which dominated the reads, total only 430MiB. These results are very interesting considering the roughly nine write I/O operations for every read operation.
Finally, the DL training has no shared I/O, so you don't see any shared reads, writes, or metadata in the top table. Darshan's origins are in the HPC and MPI world, where shared I/O operations are common, so it will present this information in the summary.
The bottom table presents how much I/O was done for the various filesystems. A very small amount of I/O is at-

Average I/O per process (POSIX and STDIO)

	Cumulative time spent in I/O functions (seconds)	Amount of I/O (MB)
Independent reads	0.062605	340.331475257874
Independent writes	0.439001	430.531460762024
Independent metadata	1.347525	N/A
Shared reads	0	0
Shared writes	0	0
Shared metadata	0	N/A

Data Transfer Per Filesystem (POSIX and STDIO)

File System	Write		Read	
	MiB	Ratio	MiB	Ratio
UNKNOWN	3.82019	0.00887	0.00000	0.00000
/home	426.71127	0.99113	340.33148	1.00000

Figure 9: STDIO and POSIX I/O from `darshan-job-summary.pl` output PDF.

tributed to *UNKNOWN* (0.887%), but `/home` (except for `/home/laytonjb/anaconda3`, which was excluded from the analysis) had 99.1 percent of the write I/O and 100 percent of the read I/O.
The final snippet of Darshan output is in Figure 10, where the I/O operation sequences are presented. The chart shows roughly 54,000 total write and perhaps 6,000 total read operations. For the write I/O, most were sequential (about 52,000), with about 47,000 consecutive operations.

Summary

A small amount of work has taken place in the past characterizing or understanding the I/O patterns of DL frameworks. In this article, Darshan, a widely accepted I/O characterization tool rooted in the HPC and MPI world, was used to examine the I/O pattern of TensorFlow running a simple model on the CIFAR-10 dataset.
Deep Learning frameworks that use the Python language for training the model open a large number of files as part of the Python and TensorFlow startup. Currently, Darshan can only accommodate 1,024 files. As a result, the Python directory had to be excluded from the analysis, which could be a good thing, allowing Darshan to focus more on the training. However, it also means that Darshan can't capture all of the I/O used in running the training script.
With the simple CIFAR-10 training script, not much I/O took place overall. The dataset isn't large, so it can fit in GPU memory. The overall runtime was dominated by compute time. The small amount of I/O that was performed was almost all write operations, probably writing the checkpoints after every epoch.
I tried larger problems, but reading the data, even if it fit into GPU memory, led to exceeding the current 1,024-file limit. However, the current version of Darshan has shown that it can be used for I/O characterization of DL frameworks, albeit for small problems.
The developers of Darshan are working on updates to break the 1,024-file limit. Although Python postprocessing exists, the developers are rapidly updating that capability. Both developments will greatly help the DL community in using Darshan. ■

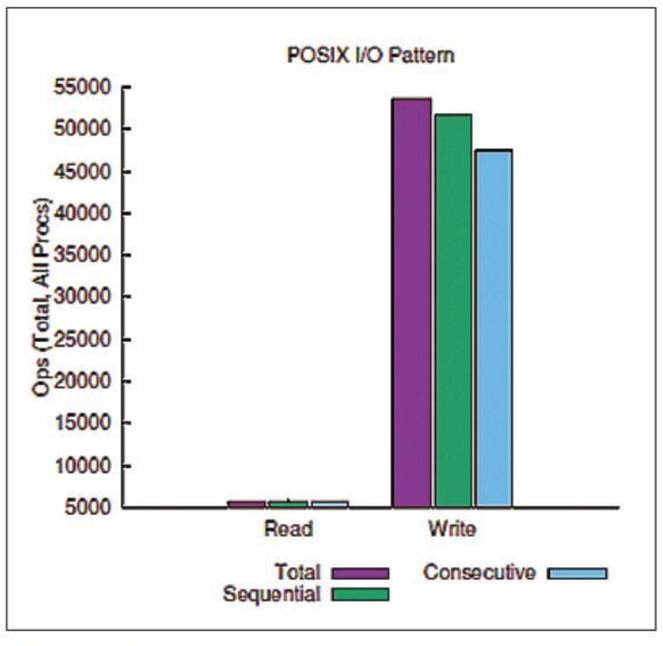
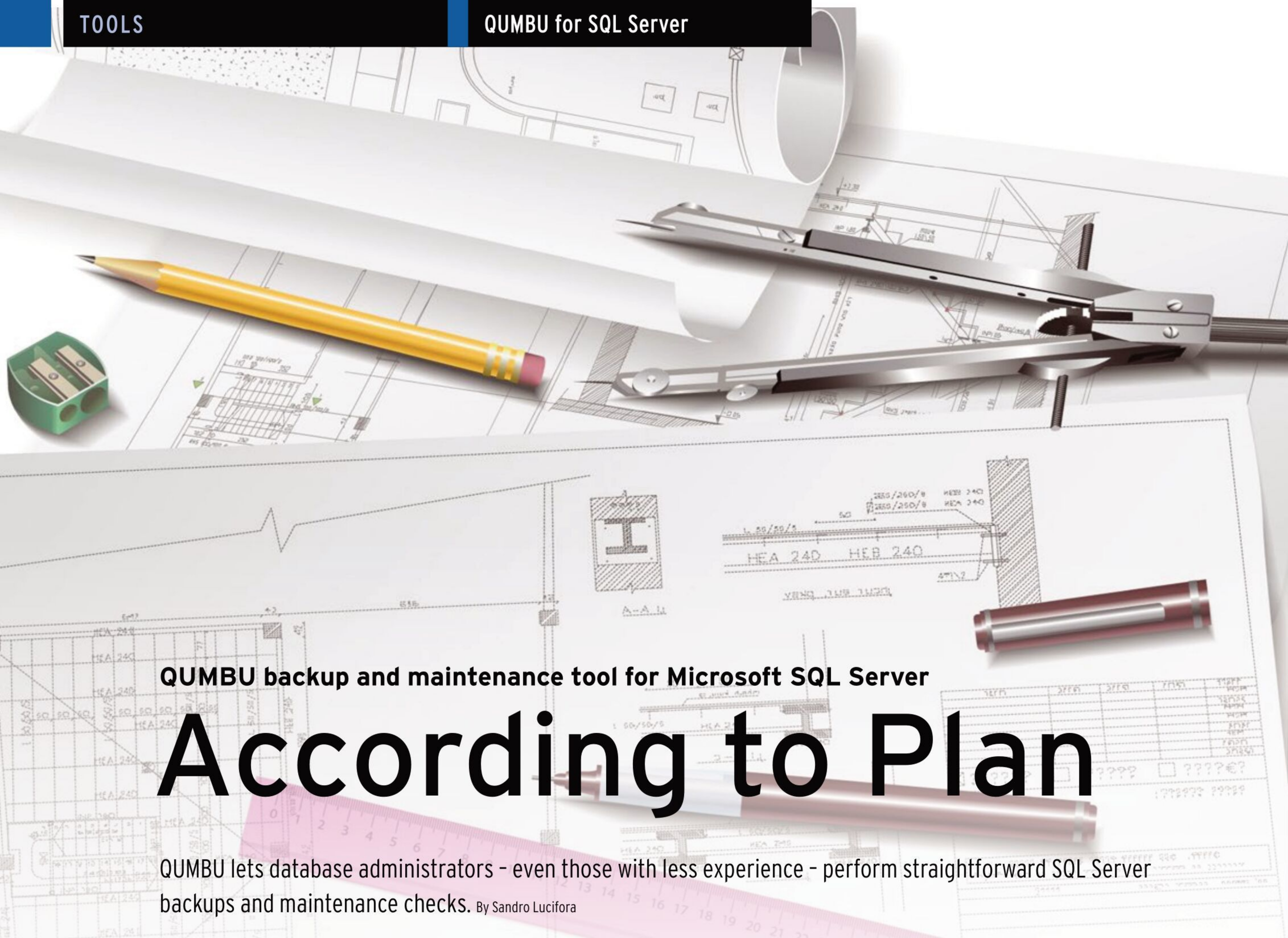


Figure 10: I/O operation sequences from `darshan-job-summary.pl` output PDF.

Info

[1] Darshan: [https://www.mcs.anl.gov/research/projects/darshan/]
[2] Documentation for multiuser systems: [https://www.mcs.anl.gov/research/projects/darshan/docs/darshan-runtime.html#_environment_preparation]
[3] Darshan mailing list: [https://lists.mcs.anl.gov/mailman/listinfo/darshan-users]
[4] "Understanding I/O Patterns with strace, Part II" by Jeff Layton, [https://www.admin-magazine.com/HPC/Articles/Tuning-I-O-Patterns-in-Fortran-90]
[5] POSIX I/O functions: [https://www.mkompf.com/cplus/posixlist.html]
[6] Keras: [https://keras.io/]
[7] CIFAR-10 data: [https://www.cs.toronto.edu/~kriz/cifar.html]
[8] "How to Develop a CNN From Scratch for CIFAR-10 Photo Classification" by Jason Brownlee, accessed July 15, 2021: [https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/]
[9] Anaconda Python: [https://www.anaconda.com/products/individual]



QUMBU backup and maintenance tool for Microsoft SQL Server

According to Plan

QUMBU lets database administrators – even those with less experience – perform straightforward SQL Server backups and maintenance checks. By Sandro Lucifora

Databases are the tech heart of the enterprise. If they stop beating, the entire work process usually grinds to a standstill, which is why it is so important constantly to maintain and secure your databases. I took a look at how this can be done with QUMBU [1] – new software by WSW Software for creating backups and maintaining the SQL Server database management system (Table 1) – and discovered that the developers have put a great deal of thought into ease of use and a planned approach. In the lab, I checked whether the software can measure up to the claim that these processes can be organized in an effortless and secure way.

For this example, the tool was installed on Windows Server 2019; it will run on Windows 7 or Windows Server 2012 with a .NET Framework. As the SQL server, the QUMBU developers assume version 2008 R2 or newer. The lab had SQL server 2019 in place. The well-structured instructions will also help administrators get through the installation and familiarization process, even if you do not work with SQL servers on a daily basis.

The setup wizard gives you the option of installing a single QUMBU client, the QUMBU server, or both components together. Therefore, the server component and the client can be installed separately (e.g., on the server and an administrative desktop computer). For this example, the full install was chosen. The QUMBU server runs in the background as a Windows service; it requires a logged in user account to run, which you specify during the installation.

After the software is installed, the wizard expects you to enter a QUMBU account name, which is the user who has access to all SQL Server instances and can be a user who authenticates locally on Windows; in other words, you do not need a domain controller. This process is simple; you can view all the potential users and select the one you want from a list. After entering the password, you can then complete the installation. Note that you do need admin privileges for the

Table 1: QUMBU

Product	Software for Microsoft SQL Server backup and maintenance
Manufacturer	WSW Software GmbH
Price	12 months: EUR59 per month and per server
	36 months: EUR39 per month and per server
Supported operating systems	Windows 7 and later with .NET Framework 3.5; Windows Server 2012, 2016, or 2019
System requirements	• Processor: 2GHz or faster
	• Memory: At least 4GB of RAM
	• Disk space: At least 1GB of free hard drive space
	• Graphics resolution: At least 1024x768
	• Microsoft SQL Server from version 2008 R2 on Windows
	• Internet access for licensing the program

Lead Image © Alexander Tkach, 123RF.com

installation; otherwise, the required services will not start.
During the installation on Windows Server 2019, I initially had problems because the QUMBU service failed to launch. The manufacturer's support team then analyzed the problem and determined that – contrary to the stated system requirements – .NET Framework 3.5 was required. After installing it, the setup completed successfully.

No Connection to Azure Database

After the launch, an uncluttered window came up; this is the QUMBU cockpit. Next to the toolbar is an area for displaying the servers, the tasks, and the task details and notifications. To begin, add a Microsoft SQL server. To do so, right-click in the *Servers* area and select the *Connect to Server* entry. In the dialog that follows, enter the server name and your login details.
After connecting, you will see that the device has been added to the *Servers* area. Additional servers can also be entered at this point, but they must be accessible on the network. Connections with a virtual private network (VPN) are also possible, as are publicly accessible servers, which is already an advantage over Microsoft SQL Server Management Studio. QUMBU also lets you include and manage multiple servers. However, QUMBU refused to connect to an SQL database I was running in Azure, saying that the server did not run on a Windows operating system. The requirement is that you run the SQL server on a dedicated Windows server – physical or virtual – for QUMBU to be able to handle it.
The license model also reflects this requirement. The manufacturer only requires one license per physical or virtual server. The license is then valid for managing any number of database instances and users. The properties of the server instance and database can be seen in the context menu. For example, you can see whether the server version and the user used for the connection had full authorization. For the database, QUMBU shows the current status as well as the *created on* and *last backup* dates.

Cross-Server Tasks

Once the server is set up, you can turn your attention to the central task area. After selecting a server in the corresponding area, you can proceed to configure it and then select *All Servers* in the top level of the tree structure (Figure 1) to apply the task you defined to all servers. The *Job details* area displays the history of executed tasks in a table, where you can filter for specific servers or tasks. Without this restriction, QUMBU shows all tasks for all SQL servers. The lower window shows the current system health notifications that relate to individual tasks or individual SQL server instances. I liked the fact that I could also see the urgency level (i.e., whether the message was just a notification or a warning that needed a response).

Tasks Everywhere

No matter what you want to do with QUMBU, the developers packaged all the functions into tasks. Both backing up and restoring a database are tasks, as are the various maintenance functions.
To begin, create a database backup task by selecting the *Backup* icon in the toolbar, which reveals options to *Backup All* databases, *Backup Selected* databases, or *Backup by Pattern* databases that match a search pattern. Clicking on the respective entry takes you to the corresponding task configuration. The operation is always the same: Right-click in the resulting dialog to open the context menu and select *Add*. The configuration dialog prompts you for parameters relating to the task. To back up all the databases,

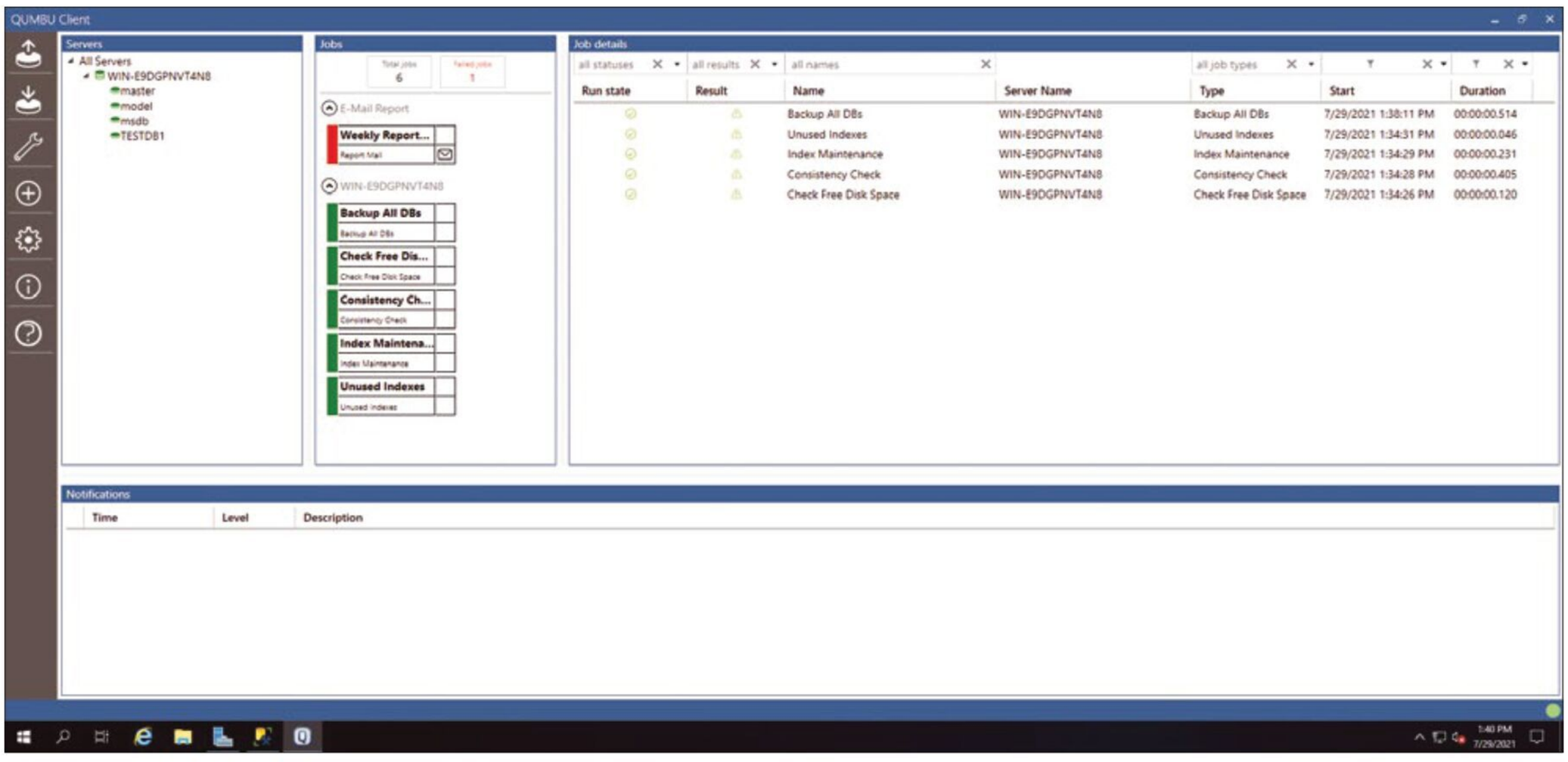


Figure 1: The QUMBU client has a state-of-the-art, clear-cut interface.

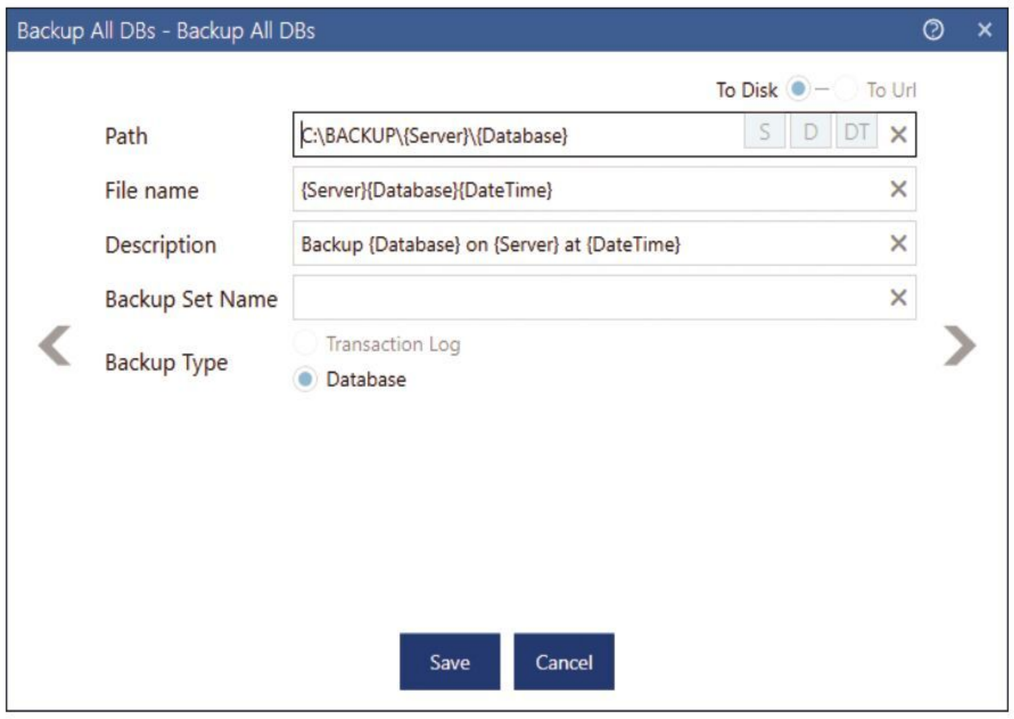


Figure 2: Variables can be used in the specifications for the path and file name of a backup.

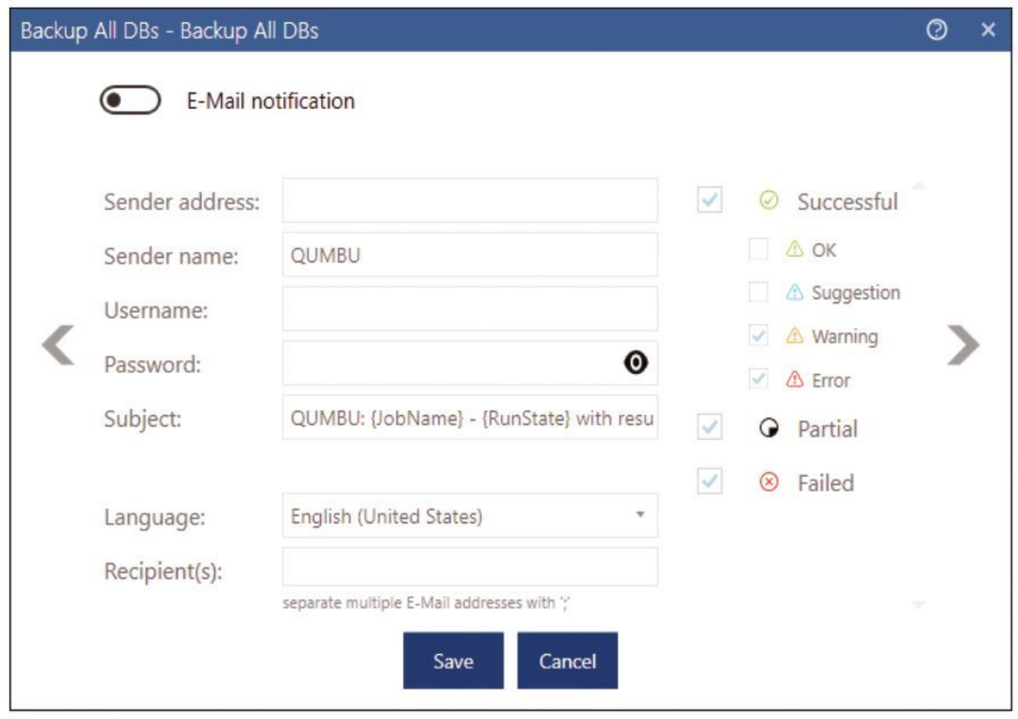


Figure 3: The email notification informs you of the success or failure of a task.

you just need to assign a name to the task and select the SQL server to be backed up.

On the next page of the dialog, you define the path, file name, and description (Figure 2). QUMBU also gives you variables for the server name, database, and backup date. The *Expert* tab has a variety of options regarding the backup itself. For example, you can specify the compression ratio and decide whether you want a differential backup or a full copy. The optional encryption is yet another level of protection for your data. Other options included the number of buffers, the maximum transfer rate, and the block size setting.

Notifications

QUMBU only sends notifications by email. To configure the email settings for backup status notification (Figure 3), you first have to specify an SMTP server, which QUMBU uses to dispatch email. After that, you specify parameters for backup notification, such as deciding on the events about which you want to be notified. What is missing at this point are alternative notification paths, such as a webhook, which you could use to integrate Slack, for example.

In the last window, you define the schedule for the backup (Figure 4). In addition to choosing daily, weekly, or monthly backups, you have options for specifications such as the day of the week, time, and the start and end dates. Configuring a backup for individual databases was identical, except for the option to select one or more databases at the beginning instead of just the SQL server.

One very helpful feature is backing up by pattern matching (i.e., performing an action against all databases that match a certain filter). This option can be helpful if, for example, all databases for production operations start with *Prod* and databases for development start with *Dev*. Perhaps you don't want to back up the Dev databases daily, but you do want to for the production databases. In this scenario, you would filter by database name, starting with *Prod*. This option is especially useful where databases are added or deleted dynamically and a server has large numbers of databases.

You also need a function to restore a backed up database to the same or another Microsoft SQL Server. The *Restore* task in QUMBU does this for you. As options, the developers implemented recovery from the backup history, which is only possible on the same server, and recovery from a file, which lets you recover the database on another SQL server. For both options, the configuration dialog for the task is just as simple as the backup task: Right-click and select

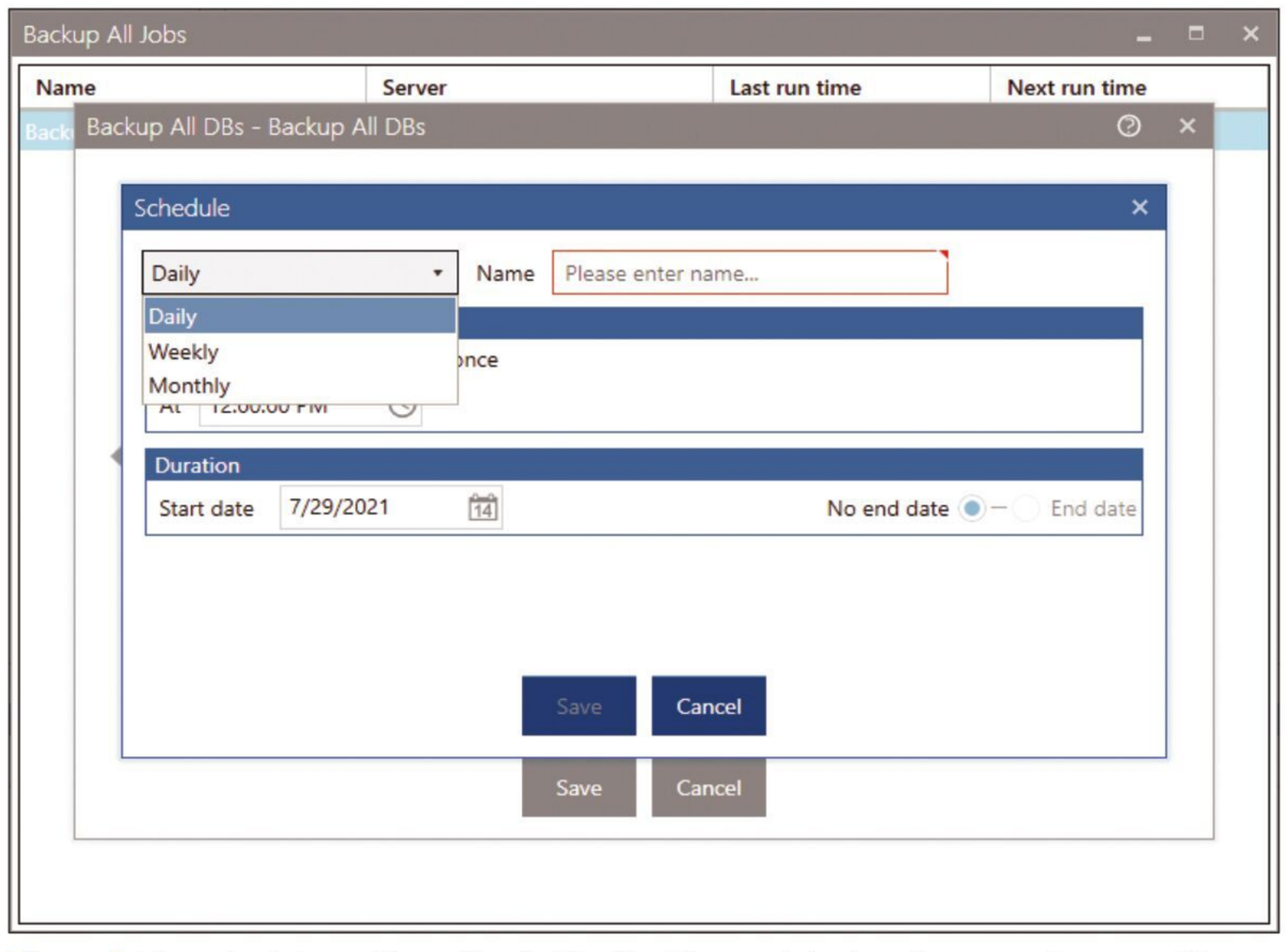


Figure 4: The schedule configuration is identical for each task and covers all reasonable possibilities.

Add to open the dialog; then, select the server and the database if you are restoring from the history. In the next window, the program shows you the history, from which you then select the desired restore start time. After that, check the expert and email settings for further tweaks. The schedule configuration dialog lets you specify when you want the restore to take place. Daily, weekly, and monthly intervals are also possible.

When restoring from a file, the task definition differs only in terms of the first dialog, in which you select the backup file and specify the SQL server and target database for the restore. The remainder of the configuration process is exactly the same as restoring from the history. To clone a database from one server to another, the vendor uses sequential backup and restore tasks. One use case for this example is regularly replicating a database to another server.

Simple Recovery

To check the consistency of a backup that you created, you have the option of simulating a restore. In this case, QUMBU reads and processes the data exactly as if the real restore were taking place, the only difference being that the software does not write any data at the end. In this way, you can check whether the backup from another database server is compatible with the target server if the two SQL servers are running different versions. The tasks for the simulated restore are identical to the processes of the actual restore described earlier.

In addition to the main tasks of backing up and restoring databases, the developers also offer the ability to perform some maintenance tasks, including consistency checks, index maintenance, finding unused indexes, and checking for free hard disk capacity.

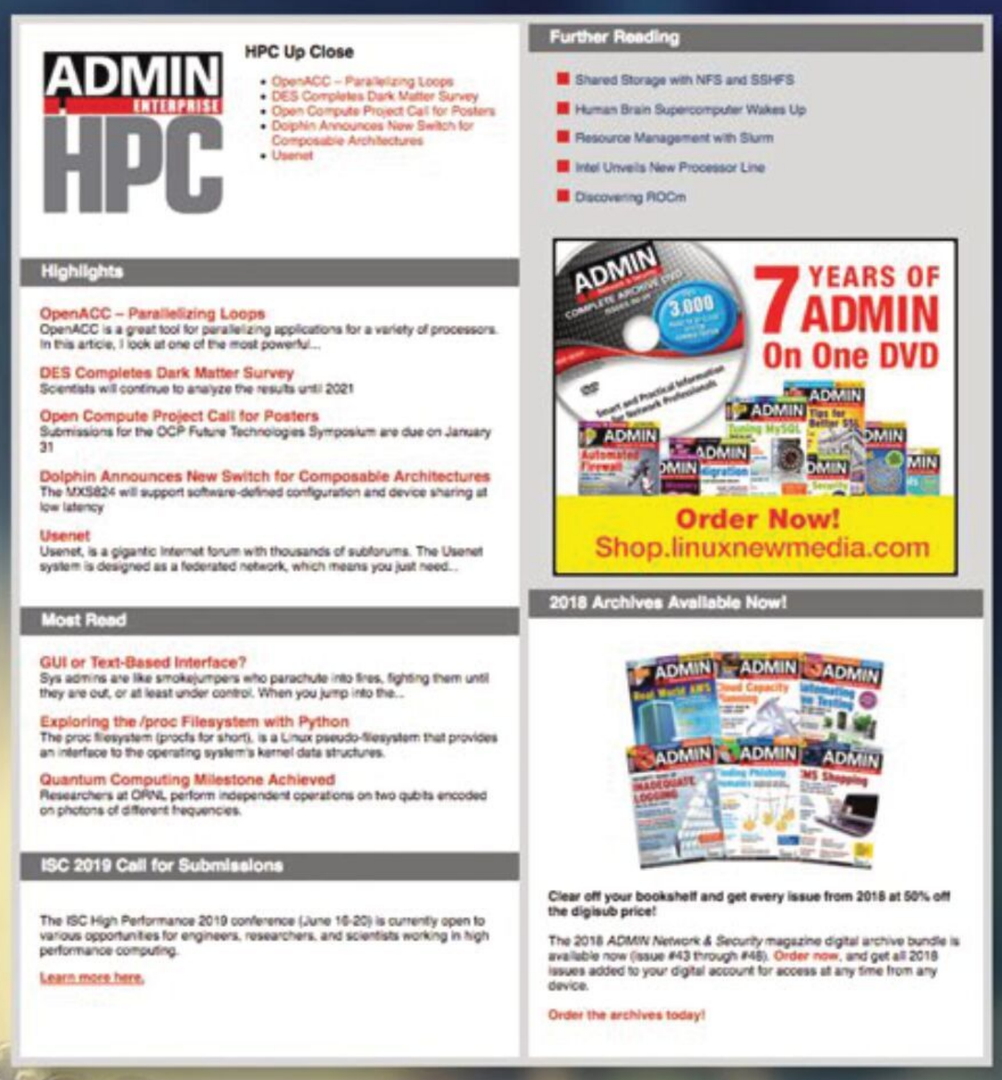
Here, too, I liked the identical process of creating tasks as was used in previously described procedures. I added a new task from the context menu and made the task-specific settings. For the consistency check, I selected the server and the database. In the expert settings, I determined the database console command (DBCC) I wanted to run for the check. In addition to the familiar email notification, I also specified the schedule for the check.

For regular index maintenance, I again created a task for which I selected the server and the database. In the expert settings this time, I was able to set the thresholds for the reorganization and rebuild as percentages. The developers also include an option for the minimum number of index pages. According to the manufacturer, however, index maintenance only makes sense after seven days of operation, because no reliable state-

GOT CLUSTER?

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

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



ments about outdated or unused indexes can be made before that. However, if necessary, you can adjust this time in the configuration.
To make sure the database server has enough disk space, you need to set up the appropriate check. In the configuration dialog, select the SQL server and chose the drives you want to monitor from the list of available drives, setting the threshold values in gigabytes for warnings and notifications in case of errors individually for each drive.
The subsequent configuration for the email notification and the process of creating the schedule are familiar. For this task, however, you do not select one-off execution per day, weekday, or month but specify a daily check in a given time frame and a certain interval. In the test, this was the weekday test, without weekends, during production hours from 8am to 10pm with an interval of 15 minutes. This was a way of ensuring no downtime because of a lack of hard disk space, especially during peak hours when the database is in use.

User Management

It is worth mentioning that QUMBU has no user management with regard to the operation of the software. The target group of users is limited to the administrator, who is allowed to do everything, or users who just have read-only rights. These settings also showed up in the list of users under the *Settings* function. Here, QUMBU outputs all the users that have been saved for access to the databases.
An additional read-only checkbox restricts their authorizations.
To round off the feature set, the manufacturer integrated – again, as a task – a reporting function, which made it possible to receive a weekly report by email (e.g., that clearly shows the results of the defined tasks). In addition to selecting the server and the possible tasks, you can also set the reporting period.

Conclusions

The software did a great job of handling the core tasks of backing up and restoring SQL databases (Table 2). The additional maintenance tasks even allow less experienced database administrators to perform preemptive checks. Grouping all the functions as a single task and the consistent, easy-to-understand user interface simplify integration into an existing infrastructure.
In contrast to Microsoft SQL Server Management Studio, QUMBU can also be used to manage multiple Microsoft SQL Servers and to configure the tasks identically across servers. In terms of functionality, the developers have taken care in the configuration of the tasks of ensuring that the dialogs only differ where they expect function-specific input. The dialogs are not overloaded, and the configuration options are very well thought out.
For time-controlled cloning of databases, the developers rely on a backup with a subsequent restore. The additional maintenance functions for consistency checks, index maintenance, finding unused indexes, and the hard disk check set QUMBU apart from most other tools in this category. The maintenance functions might only be "little things," but they do help to optimize a database continuously.
Not to be overlooked were teething problems, especially during the install, but these were solved quickly and competently by the support team. For me, QUMBU is a very useful tool with a well-thought-out and modern user interface, simple operation, and, above all, useful and reliable functions. ■

Table 2: Review

Rating (out of 10)	Backup databases: 7
	Database recovery: 8
	Maintenance functions: 8
	Configuration options: 7
	Notifications: 6
This product is suitable	• Great for backing up and maintaining multiple Microsoft SQL Servers; the tool is also designed for less experienced administrators
	• With restrictions as a plain vanilla maintenance tool for SQL Server
	• Not as a backup solution for databases that do not run on a Windows operating system, such as Azure or AWS

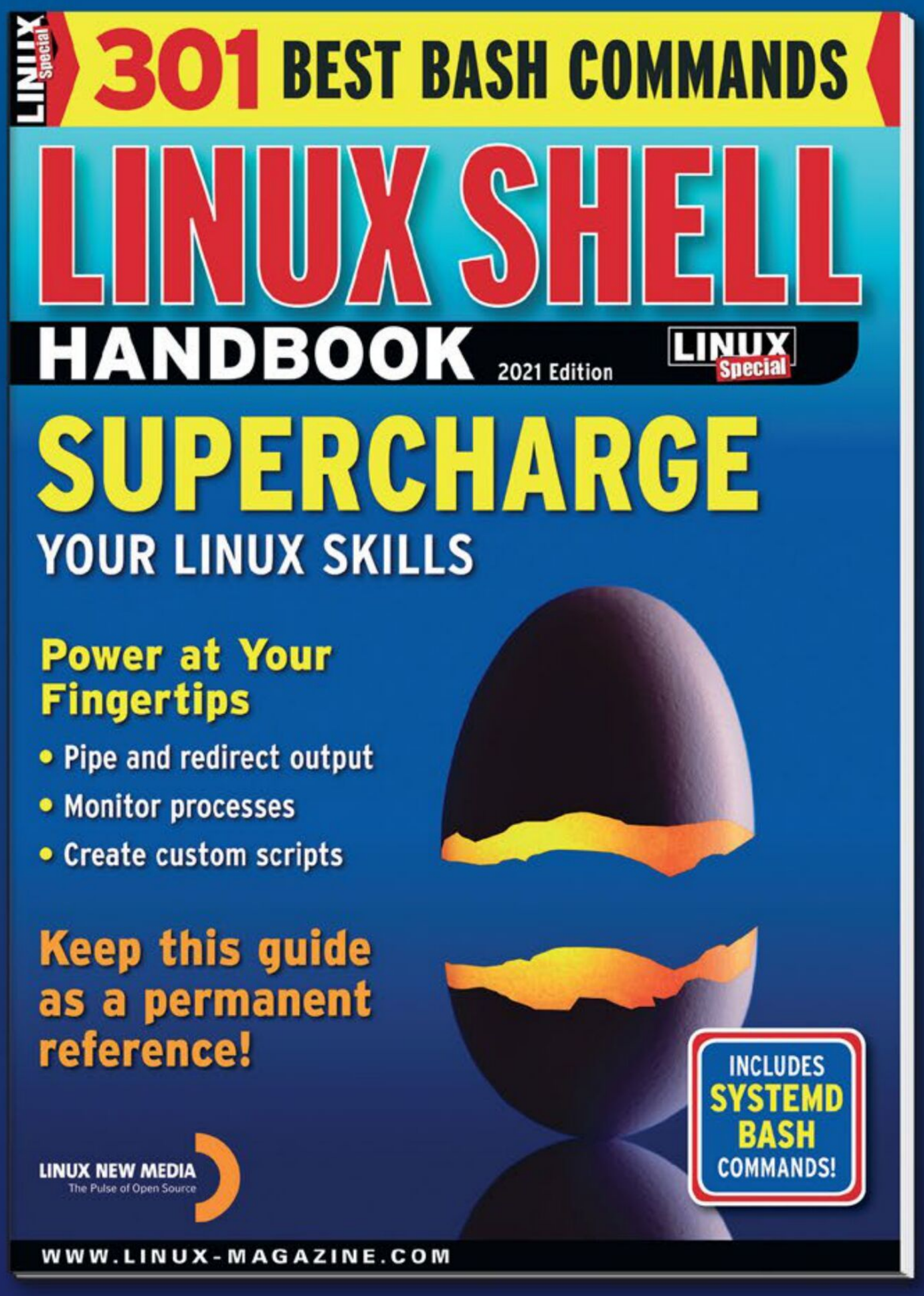
Info

[1] QUMBU: [https://www.qumbu.de/en/]

THINK LIKE THE EXPERTS

Linux Shell Handbook 2021 Edition

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

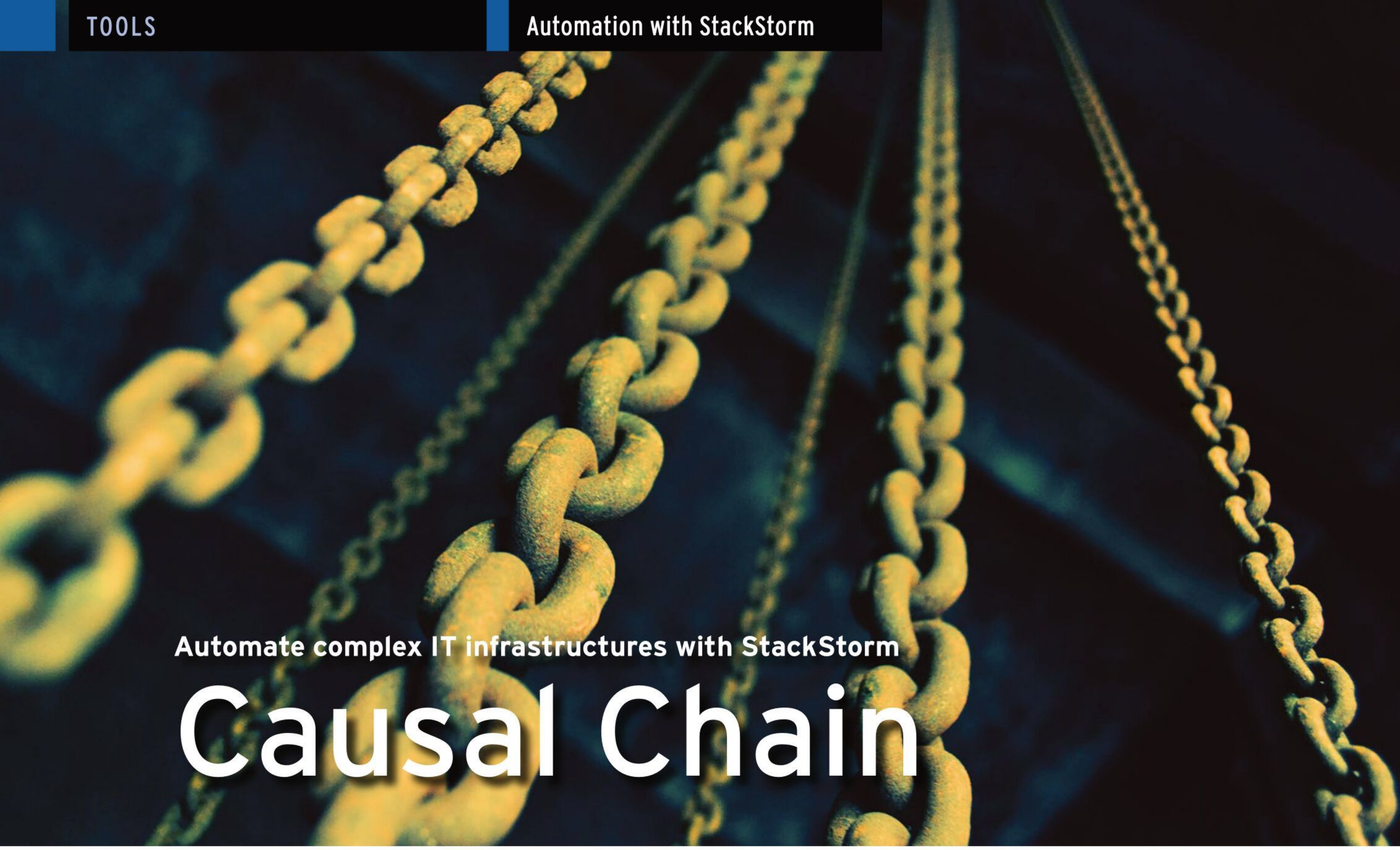


Here's a look at some of what you'll find inside:

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

ORDER ONLINE:

shop.linuxnewmedia.com/specials



Automate complex IT infrastructures with StackStorm

Causal Chain

StackStorm is an open source, event-based platform for runbook automation. By Holger Reibold

If you want to take home a message from the coronavirus pandemic, one would be that it is acting as a catalyst for digitization in all areas of life. However, it is precisely this phenomenon that presents IT administrators with new challenges because environments are not only becoming more complex but also more diverse, which goes hand in hand with an increase in administrative overhead. Gone are the days when infrastructures could be managed manually. Modern infrastructure management environments open an opportunity for avoiding error-prone manual adjustments. Insiders have bandied about Infrastructure as Code (IaC) as the key to tomorrow's infrastructure management for some time now. IaC is to be understood as an abstraction solution for managing the hardware and software components of an IT infrastructure. Machine-readable definition files are used instead of a physical hardware configuration or special configuration tools. In this article, I describe the basic structure and practical use of StackStorm, "a platform for integration and automation across services and tools [that] ties together your existing infrastructure and application environment so you can more easily automate that environment" [1].

StackStorm at a Glance

StackStorm [2] is often mentioned in the same context as SaltStack, which was acquired by VMware, and Ansible; however, the comparison is misleading because StackStorm focuses on running management tasks or workflows on an event-driven basis. In particular, the tool defines triggers and events, which it then reacts to when those triggers or status changes occur. StackStorm supports automatic correction of system settings, security reactions, rules-based troubleshooting, and deployment. The tool also has a rules engine and a workflow manager.

In the major leagues, StackStorm is still a fairly unknown player that targets the integration and automation of services and tools. The goal is to capture an existing infrastructure and application environment and react automatically across the infrastructure when certain events occur. A few examples illustrate the potential of using StackStorm. For example, assume you rely on Nagios for infrastructure monitoring; you could use StackStorm to trigger further diagnostic checks and make the results available to third-party applications. That's not all, though: You can also use the tool for automated remediation by monitoring critical systems and initiating follow-up actions when errors are identified. Finally, StackStorm supports you during deployment. For example, you can use the tool to deploy a new AWS cluster or activate a load balancer for essential load distribution in the event of an imminent system overload.

From Event to Action

To initiate various actions, the system needs to know about the corresponding states or events. To do so, StackStorm draws on various sensors, which are Python plugins for the integration of the various infrastructure components (**Figure 1**). When a sensor registers a defined event, it issues a StackStorm trigger. The management environment distinguishes between generic and integration triggers. You can define your own trigger types in the form of a sensor plugin, should Stack-

Photo by Fré Sonneveld on Unsplash

Storm itself not provide the desired trigger type.

Another important element of the solution is actions, which subsume all actions the management environment can perform on infrastructure components. Most actions are executed automatically, but it is also possible to run commands from the integrated command-line interface (CLI) or with an application programming interface (API). The most common use case, however, is the use of actions in rules and by triggers.

StackStorm also has workflows that bundle different actions into what it dubs "uber-actions"; they define the execution order and the transition conditions and take care of the data transfer. Because most automation tasks are a sequence of two or more actions, workflows should be considered the defining element in the StackStorm environment. To simplify the execution of recurring tasks, the tool has a workflow library.

In the form of "packs," the management environment provides another function for bundling tasks that supports grouping of integration functions (triggers, actions) and automation mechanisms (rules, workflows). A growing number of integration modules for Active Directory, AWS, Icinga, Jenkins, Exchange, MySQL, Nagios, and OpenStack are available from the StackStorm Exchange platform [3].

StackStorm also has an auditing mechanism that records all relevant details for manual or automatic execution. The core function of the tool can be described by the cycle of trigger, rule, workflow, action, results. According to the StackStorm website, several well-known companies (e.g., Cisco, NASA, and Netflix) are already using these environments in their IT infrastructures.

Getting Started Quickly

StackStorm was developed for Linux-based operating systems and cooperates especially well with Ubuntu, Red Hat Enterprise Linux, and CentOS. The installation is particularly easy on a new Linux installation. Make sure that Curl is present; then, working as an administrator, run the command to install:

```
curl -sSL https://stackstorm.com/➥
  packages/install.sh | ➥
  bash -s -- --user=st2admin ➥
  --password='<secret>'
```

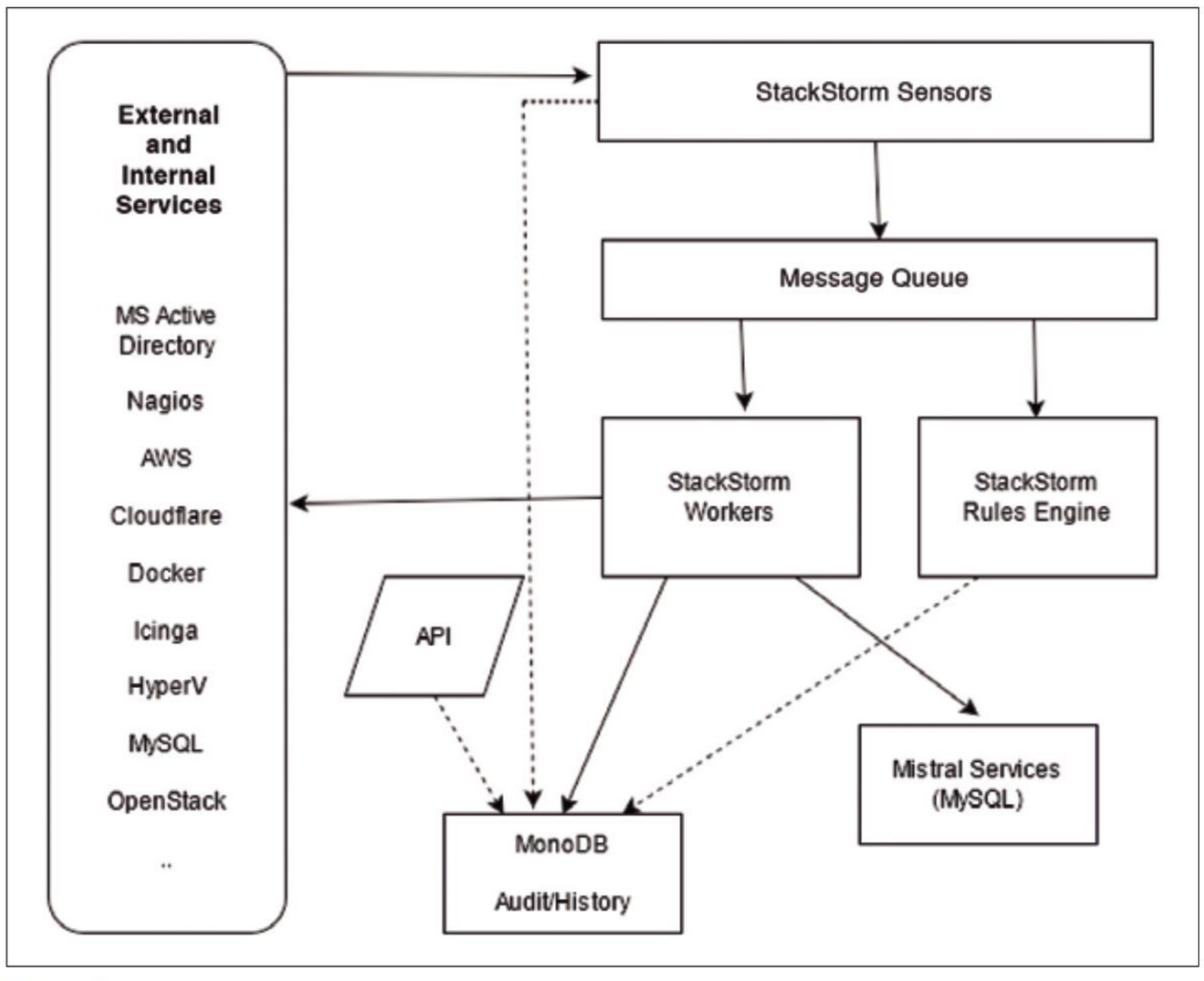


Figure 1: In StackStorm, events from the sensors are aggregated then matched with triggers; if necessary, actions are then triggered. Workflows are optional.

This command installs a full StackStorm version. The developers explicitly point out that problems are bound to occur on Linux systems with enterprise applications already installed. If you are running the installation behind a proxy server, export the proxy environment variables `http_proxy`, `https_proxy`, and `no_proxy` before running the script:

```
export http_proxy = ➥
  http://proxy.server.com:port
export https_proxy = ➥
  http://proxy.server.com:port
export no_proxy = localhost, 127.0.0.1
```

Firewall settings might need to be adjusted to access the web GUI.

The core function of StackStorm is provided by the `st2` service, which you will find in `/opt/stackstorm/st2`. This service is configured by the associated configuration file `/etc/st2/st2.conf`. StackStorm has its own web GUI, which can be found in the directory `/opt/stackstorm/static/webui` and is configured by the JavaScript-based configuration file `webui/config.js`.

The developers prefer to use the CLI with StackStorm. Some basic commands will help you familiarize yourself with the environment. For example, to output the version in use and view the available triggers, actions, and rules, use the commands:

```
st2 -version
st2 action list --pack=core
st2 trigger list
st2 rule list
```

StackStorm not only has some default triggers and rules but also various predefined actions. You can retrieve these with the same scheme. To retrieve the list of all actions in the library, get the metadata, view the details and available parameters, and initiate an action from the CLI, use the respective commands:

```
st2 action list
st2 action get core.http
st2 run core.http -help
st2 run key=value '<arguments>'
```

Listing 1: Sample rule with webhook

```
name: "sample_rule_with_webhook"
pack: "examples"
description: "Sample rule dumping webhook payload to a file."
enabled: true
trigger:
      type: "core.st2.webhook"
      parameters:
           url: "sample"
criteria:
      trigger.body.name:
           pattern: "st2"
           type: "equals"
action:
      ref: "core.local"
      parameters:
           cmd: "echo \"{{trigger.body}}\" > ~/st2.webhook_sample.out ; sync"
```

To execute a Linux command on multiple hosts over SSH, you can use the `core.remote` action. All that is required is that passwordless SSH access is configured on the various hosts. Execution is according to the scheme:

```
st2 run core.remote ➥
  hosts='<www.examplehost1.com>, ➥
        <www.examplehost2.com>' ➥
  username='<SSH user>' -- ls -l
```

You can view the action history and execution details and list executions with:

```
st2 execution
st2 execution list
```

To limit the output to the last 10 executions, use

```
st2 execution list -n 10
```

Rules are an essential tool of the StackStorm concept. The tool uses rules to execute actions or workflows when specific events have occurred in the IT infrastructure. Events are usually registered by sensors. When a sensor detects an event, it fires a trigger, which itself triggers the execution of a rule again. The conditions of such a rule determine which actions take place.

By default, a StackStorm installation has a sample pack that includes various sample rules. One of them is the *Sample rule with webhook* (Listing 1). The rule definition is a YAML file that includes three sections: `trigger`, `criteria`, and `action`. This sample is designed to respond to a webhook trigger and apply filter criteria to the contents of the trigger.

Predefined Routine Packages

Dealing with StackStorm is simplified by using the packs already mentioned. You can think of them as deployment units for integrating and automating established services and applications. Thanks to this approach, it is easy to integrate AWS, Docker, GitHub, or similar systems into the management environment. Actions, workflows, rules, sensors, and aliases are bundled in such a pack. Different packs capture automation patterns – the developers also refer to automation packages.

Predefined integration packages are available, in particular, through the StackStorm Exchange platform but can also be created independently. The

```
st2 pack <package name>
```

command manages StackStorm packages. The default installation already has some pre-installed packages that you retrieve with the `list` command. By default, they are located in the `/opt/stackstorm/packs` directory. StackStorm Exchange doesn't inform you at first glance that it has well over a hundred StackStorm packages. You can conveniently browse the inventory from within the CLI. To do so, use the commands:

```
st2 pack search
st2 pack show
```

Once you have identified one or more packages of interest, install them from the CLI on the basis of the Exchange designation:

```
st2 pack install <package1> <package2>
```

Various packages are dependent on the existence of others. Corresponding information is stored in the `dependencies` section of the `pack.yaml` file. You do not have to deal with these dependencies any further because StackStorm automatically installs the relevant packages.

However, just installing is not usually enough: You need to adapt the setup to your framework conditions. If a package supports sending notification email, for example, you will need to configure the SMTP server. Another typical requirement is specifying access credentials for a service. Mostly, the package configuration is interactive – for example, by typing

```
st2 pack config cloudflare
```

when setting up the Cloudflare package.

In this example, the CLI presents you with an interactive dialog with default values, suggestions, and input fields for your own settings. The package configuration is stored in the file `/opt/ stackstorm/configs/<package name>.yaml`.

Dealing with Actions

Actions are one of the core concepts of the StackStorm environment. These code snippets can be written in any programming language and basically perform any automation or correction tasks in your environment. Actions can implement a wide variety of management tasks. For example, you can restart services on a server, set up a new cloud server, verify a Nagios alert, use email or SMS notification, start a Docker container, take a snapshot of a virtual machine, or initiate a Nagios check. Actions start

running when a rule with matching criteria triggers them. Execution can be from the CLI or with an API.
To perform an action manually, use the commands:

```
st2 run <action with parameter>
st2 action execute <action with parameter>
```

To create your own action, you again need to create a YAML-based metafile with the relevant information and a script that implements the action logic. Getting started is simplified with the use of predefined actions that are part of the core package.
You can use the `core.local` action to execute arbitrary shell commands. A simple example is:

```
st2 run core.local cmd='ls -l'
```

The `core.remote` action supports the execution of commands on remote systems and `core.http` lets you execute an HTTP request, as in:

```
st2 run core.remote cmd='ls -l' ➥
  hosts='host1,host2' username='user1'
st2 run core.http ➥
  url="http://www.server.de/get" ➥
  method="GET"
```

The following action is similar to `curl` and allows authentication on a remote system with credentials:

```
st2 run core.http ➥
  url="http://www. server.com/get" ➥
  method="GET" ➥
  username=user1 password=pass1
```

These actions not only cover the standard tasks but are also great for finding your way around the StackStorm environment. For the full list of core packages, use the command:

```
st2 action list --pack=core
```

StackStorm has another special feature to offer: If you already have scripts in any programming or scripting language, StackStorm can convert them into actions. First, you need to make sure that the script conforms to the conventions laid down by the StackStorm developers. In particular, it is important that it returns a status code of zero after execution is complete and that it terminates on a non-zero error. The tool uses the exit codes to determine whether the script completed successfully. You also need to generate a metadata file that lists the script name, a description, the entry point, the runner to use, and the script parameters.

Sensors and Triggers

The purpose of sensors is to integrate external systems and events into a StackStorm environment. These sensors either query external systems for specific system and environment variables or wait until they register a conspicuous or critical event that, if detected on a sensor configuration, causes StackStorm to fire a trigger, with possible actions executed according to a set of rules. The tool has a sensor interface that queries or collects the data. In most cases, sensors and triggers interact, but some triggers do not require sensors, such as the webhook trigger (Figure 2).
Out of the box, StackStorm comes with several internal triggers that you can apply to your ruleset when you configure a new installation. These can be distinguished from non-system triggers by their `st2` prefix:

- `core.st2.generic.actiontrigger` is a generic trigger for action execution.
- `core.st2.generic.notifytrigger` triggers a notification.
- `core.st2.action.file_written` writes to files on the target system.
- `core.st2.generic.inquiry` executes a new status query after being assigned the *pending* state.

You will find several sensor-specific triggers in a StackStorm installation. For example, the `core.st2.sensor.process_spawn` trigger indicates that a sensor process has been activated. The `core.st2.sensor.process_exit` trigger tells you that a sensor process has been stopped. Sensors run as separate processes and can execute different operations. Before these can be executed, registration with `strctl` is necessary. After successful registration, the sensor starts automatically.

Generating Workflows

Automation and management tasks are usually characterized by a sequence of actions. In practice, it makes little sense to initiate only one action and launch a subsequent one based on it at a later time. To bundle different tasks, StackStorm uses workflows that assign actions to a higher level of automation and coordinate their execution by running the right action at the right time

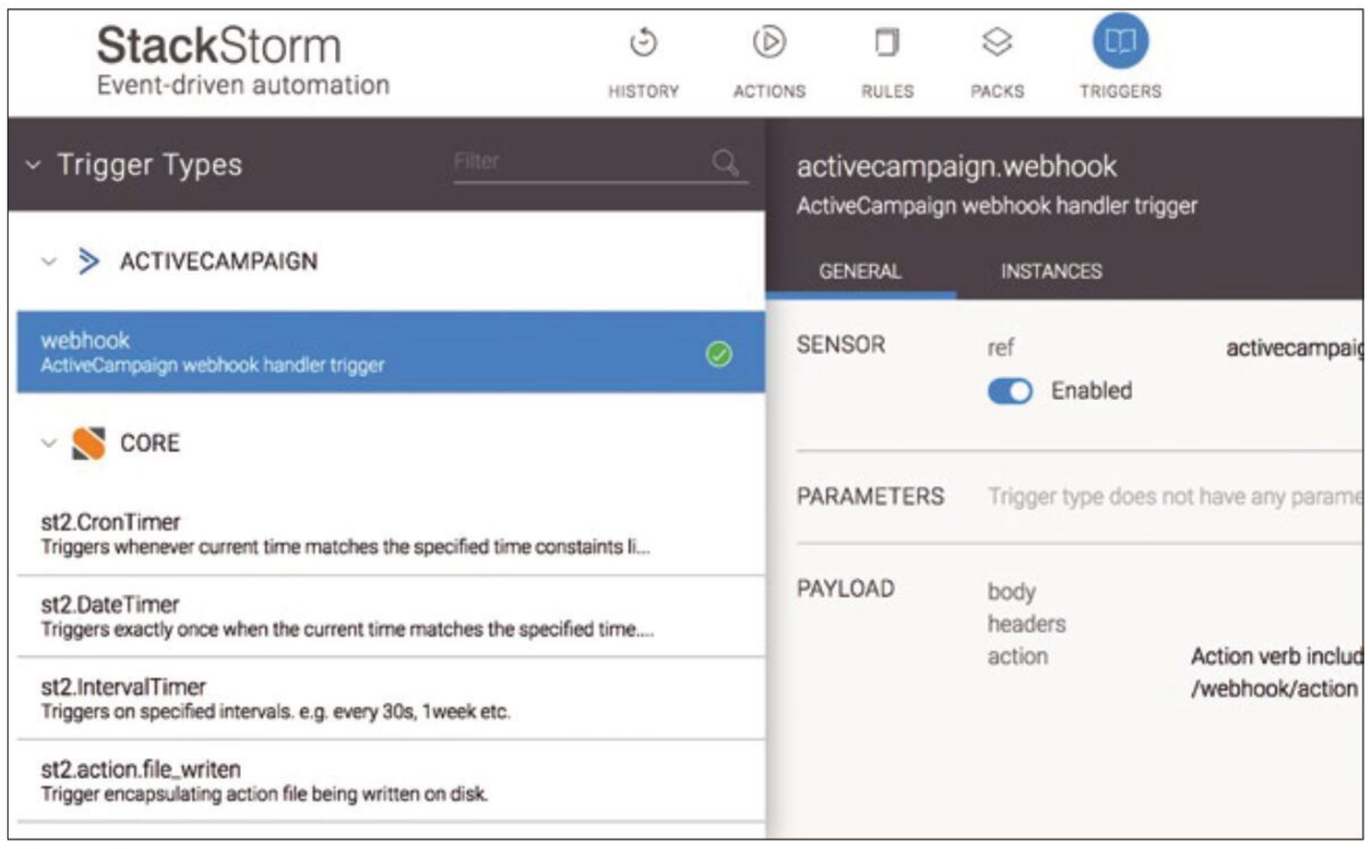


Figure 2: The automation environment has a clear-cut web interface; however, the developers recommend using the CLI.

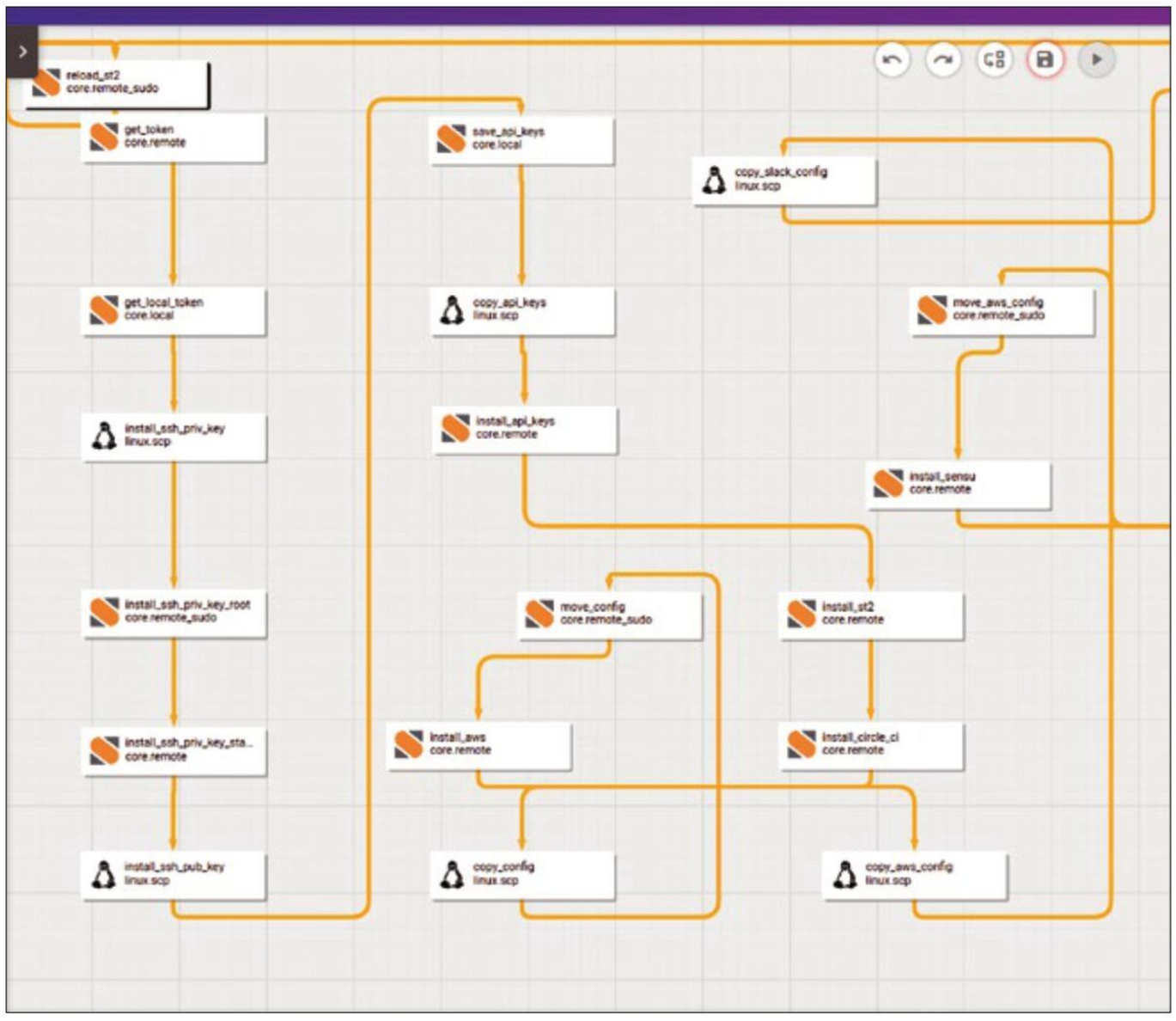


Figure 3: The commercial StackStorm variant Extreme Workflow Composer provides a visual editor that simplifies the configuration and management of workflows.

with the right input. Information can be passed into and processed in such an execution thread. Like actions, you manage workflows in the automation library and fall back on the configurations stored there, if necessary. In principle, a workflow can even be made up of other workflows.

StackStorm supports two workflow variants: ActionChain and Orquesta. ActionChain is the older variant that uses simple syntax to define a chain of actions. The disadvantage of this variant is that complex workflows are not possible. Orquesta is a new workflow engine that recognizes sequential workflows or complex workflows with forks, links, and sophisticated data transformations and queries. The developers advise that you use Orquesta.

StackStorm provides various tools for creating workflows. In the open source variant, you have to make use of console-based development. The commercial StackStorm variant is known as Extreme Workflow Composer [4] (Figure 3), which has an integrated visual editor you can use to design actions and their sequence in a drag-and-drop process. StackStorm is under active development and will see further continuous development. The roadmap [5] summarizes planned innovations. However, advance announcements at press time did not go beyond those of the current version 3.3.0.

Conclusions

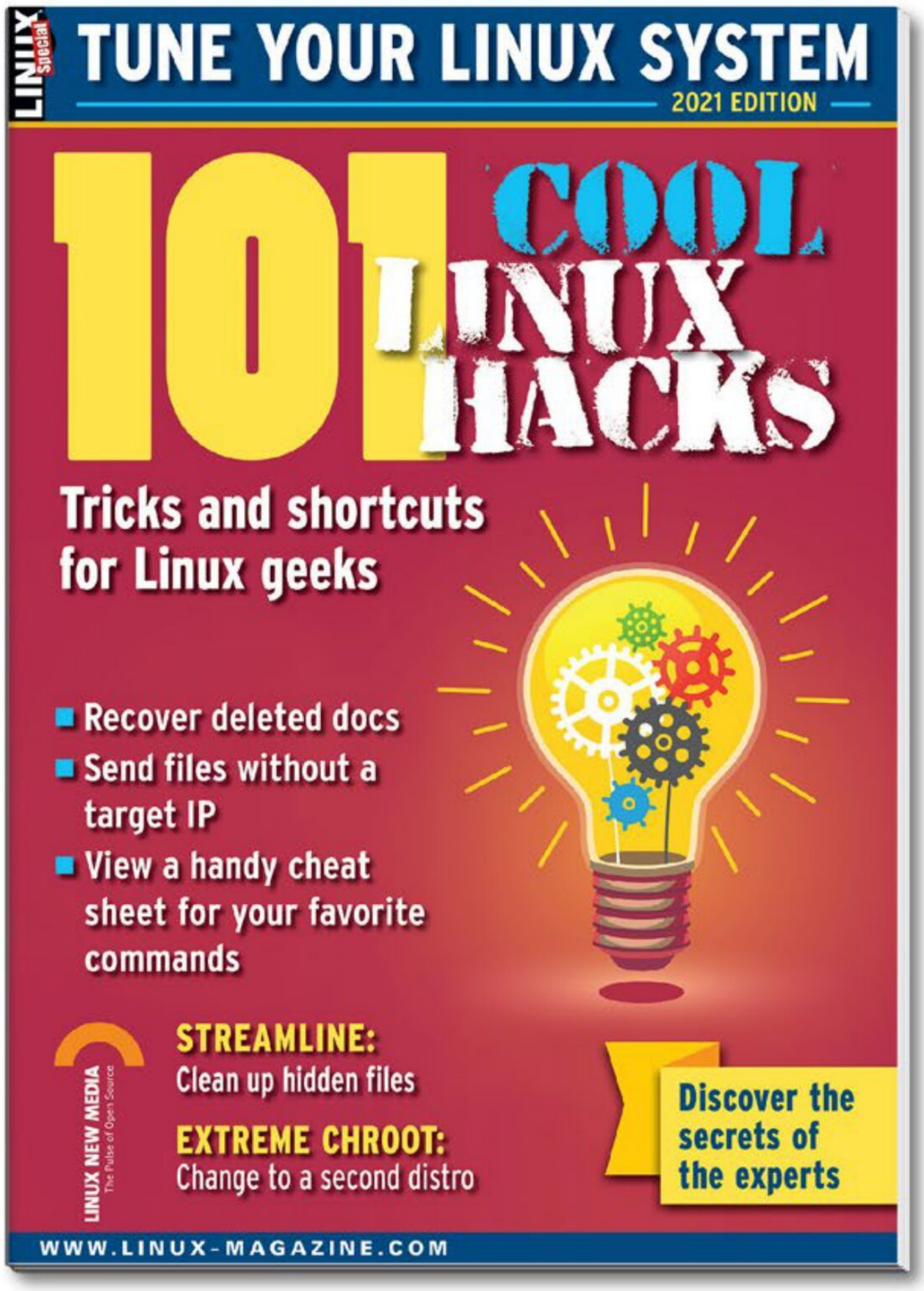
StackStorm is pursuing a highly interesting approach that significantly simplifies the automation of complex IT infrastructures. The limitation to events and service states could prove to be a disadvantage. Approaches that can act in event-dependent and -independent ways might be a better solution. ■

Info

[1] StackStorm overview: [https://docs.stackstorm.com/overview.html]
[2] StackStorm homepage: [https://stackstorm.com]
[3] StackStorm Exchange: [https://exchange.stackstorm.org]
[4] Extreme Workflow Composer: [https://stackstorm.com/stackstorm-6/]
[5] StackStorm roadmap: [https://docs.stackstorm.com/roadmap.html]

SHOP THE SHOP
shop.linuxnewmedia.com

GET PRODUCTIVE WITH 101 LINUX HACKS



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

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

ORDER ONLINE:
shop.linuxnewmedia.com/specials

Zero-Ops Kubernetes with MicroK8s

Small Packages

A zero-ops installation of Kubernetes with MicroK8s operates on almost no compute capacity and roughly 700MB of RAM. By Chris Binnie

Among the number of burgeoning Kubernetes distributions available today is the excellent production-ready K3s [1], which squeezes into a tiny footprint and is suitable for Internet of Things (IoT), thanks to a binary of just 100MB. The perfect laboratory companion that offers immediate access to Kubernetes is the clever minikube [2].

Another distribution caught my eye recently when I was arriving, very late to the party, and tinkering with some fascinating Raspberry Pi tech: MicroK8s [3]. Two things jumped out when I spotted Ubuntu's mini Kubernetes documentation. First, as with K3s and minikube, a software build suitable for ARM64 processors (e.g., which a Raspberry Pi uses) is available; second, the documentation prominently notes zero-ops infrastructure.

I hadn't seen terminology about hands-off operations software since Amazon Linux 2 extolled its virtues about live kernel patching [4], which I started using on critical production servers about a decade ago. I can attest to the fact that it's a great feature that saved many 4:00am reboots (after kernel security updates were applied to a critical running system) and was an exceptionally welcome addition to my toolbox because it reduced maintenance windows and downtime considerably.

In this article, I look at how a Raspberry Pi 4 (Model B) with 4GB of RAM stands the tests of an installed Ubuntu MicroK8s (minimal production Kubernetes), courtesy of Canonical. Note that 4GB of RAM is definitely recommended; although, as you will see, you might possibly get away with 2GB.

Pie-O-My

As with other Ubuntu documentation, the different routes of getting started with MicroK8s are clearly written in welcome detail. The marketing strapline offers the high-availability banner along with the description: "Low-ops, minimal production Kubernetes, for devs, cloud, clusters, workstations, Edge and IoT" [3].

The documentation talks about how MicroK8s doesn't have any of the standard Kubernetes APIs removed, and you are encouraged to enter your email address for a research whitepaper [5] that walks through security, operations, and where IoT workloads make the most demands from a Kubernetes cluster. You are then gently reminded that enterprise support is available, should it be required.

The zero-ops claim is explained with the statement: "MicroK8s will apply security updates automatically by default, defer them if you want" [3]. Apparently you can upgrade MicroK8s with just one command – an impressive statement in terms of reducing downtime and admin overhead.

For the Raspberry Pi ARM64 support, the MicroK8s website reminds you where IoT devices are deployed these days: "Under the cell tower. On the racecar. On satellites or everyday appliances, MicroK8s delivers the full Kubernetes experience on IoT and micro clouds" [3].

Blueberry Pie

Clearly, you have to have an Ubuntu version, whether a desktop build or a server, to test MicroK8s. In my case, I had the latest Long Term Support (LTS) version, Ubuntu Server 20.04. Older LTS releases are equally suitable (i.e., 18.04 and 16.04).

Fret not if you can't meet these requirements, however. In addition to offering some welcome troubleshooting advice, the Alternative Installs

Photo by Viktor Forgacs on Unsplash

page [6] is well worth a look and offers information about offline cluster creation and Windows installations, among other things.

On the Raspberry Pi, I was running Docker containers for an automation project, so to prevent any breakages of that build, I cloned the SD card running that project.

As you will see in a second, Ubuntu is a big proponent of Snappy package management [7], commonly known as "snap" (the package is `snapd`), which cleverly packages up software so that it can run on almost any device [8].

To install snap on the Raspberry Pi, you should run as root the commands:

```
$ apt update; apt install -y snapd
```

The packages pulled down for installation (in this case) were `snapd` and `squashfs-tools`.

The Raspberry Pi whirs away for a minute or two during this process. Remember the hardware specification that is responsible for running the installation and be warned that a modicum of patience is needed.

According to the docs [9], you need to reboot:

```
$ reboot
```

Although I hadn't asked snap to install or run anything yet, I took a quick check of available RAM (Listing 1). Even with Docker running, the Raspberry Pi still had about 2.9GB of free RAM.

To start the MicroK8s installation, simply enter the command:

```
$ snap install microk8s ➥
  --classic --channel=1.19
```

The docs explain exactly what `channel` refers to [10]. If you intend to continue using MicroK8s beyond testing, you should definitely understand a little more about the release cycle, which you can find on that page, as well. The stable release version, for example, wouldn't follow that channel but would instead be installed with the command:

```
$ snap install microk8s --classic
```

That's not always the version you want, so you could use a specific version:

```
$ snap install microk8s ➥
  --classic --channel=1.18/stable
```

I continued with the 1.19 channel (`--classic` without mentioning a channel, unless you run into problems, which in my case pulled down the 1.20 version). Again, having run the `snap install` command, you will need a little patience, so water the plants and polish your shoes while you wait. The download takes a few minutes and then the setup of the Snap Core follows before MicroK8s is downloaded and then installed. Once the process has finished, expect to see output something like:

```
microk8s (1.19/stable) v1.19.7 from ➥
  Canonical\u00e2 installed
```

In true snap style, you don't get much information.

The docs then follow a route similar to what a standard Docker installation might. I won't follow those steps here but will continue using the root user. The docs suggest adding your own login user to the `microk8s` system group:

```
$ usermod -a -G microk8s chris
$ chown -R chris ~/.kube
```

The docs are not very clear about the `~/.kube` cache access for the less privileged user, so I stuck with the root user and not `chris` (which you should replace with your own username if you followed the docs).

At this stage, you should open another terminal, log in to your Raspberry Pi, and check the installation progress in the original window:

```
$ microk8s status --wait-ready
```

With the `top` command in the other terminal window, you can watch the MicroK8s processes running. My device showed a one-minute average load of between 3.0 and 3.5, even with Docker running dutifully in the background. Nearer the end of the installation, the one-minute load jumped to about 5.0. The first time I tried MicroK8s's own version of the `kubectl` command not much happened, even after waiting an age (Listing 2).

To make the MicroK8s build friendlier to Raspberry Pi, I followed some `cgroups` (control groups) instructions on the Alternative Installs page [6]. Control groups help manage, typically on Linux containers, the limits on resources consumed in terms of RAM and I/O and are a feature of the kernel. It seems that the Raspberry Pi needs a configuration change to play nicely with MicroK8s. The fix is to add

```
cgroup_enable=memory cgroup_memory=1
```

to the end of the first line already present in the `/boot/firmware/cmdline.txt` file and reboot. The docs point out that some Raspberry Pi versions use the file `/boot/firmware/nobtcmd.txt` instead. Use that if it's required.

Next I removed the existing iptables rules I had installed and made sure that after a reboot they weren't applied. You can put the following entries in a file and run it as a script to flush iptables rules:

```
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
```

Listing 1: Free RAM on Rasp Pi

```
Popo ~ # free -m
              total        used        free      shared  buff/cache   available
Mem:           3827         548        2620         173         658        2977
Swap:            99           0          99
```

Listing 2: Not Much Happening

```
$ microk8s kubectl get pods -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   calico-kube-controllers-[snip]-pnnkc       0/1     Pending   0          66m
kube-system   calico-node-m79sg                          0/1     Pending   0          66m
```

```
iptables -P OUTPUT ACCEPT
iptables -t nat -F
iptables -t mangle -F
iptables -F
iptables -X
```

To check your rules are cleared, run the command:

```
$ iptables -nvL
```

A handy tip (because Kubernetes uses iptables extensively) is to check that MicroK8s is starting up as hoped by re-running that command every 30 seconds or so during start-up to see multiple Kubernetes and Calico [11] rules filling up your iptables chains. You can also use the `watch` command in a new terminal:

```
$ watch -n1 iptables -nvL
```

The next thing I realized was that Docker wasn't necessarily playing that nicely with MicroK8s, so I stopped the service and removed it completely with the command:

```
$ systemctl purge docker-ce
```

Your package name might be `docker.io` instead of `docker-ce`.
To make sure MicroK8s was completely happy, I then ran an exceptionally useful command,

```
$ microk8s inspect
```

which, as noted on the MicroK8s Troubleshooting page [12], you can run at any time during the installation. (It might just tell you, for example, that you haven't started the cluster yet, more on that below.)
The abbreviated output from the `inspect` command is shown in Listing 3. Note that this command also gives a warning if you don't get the control groups file entry quite right, which is really handy. Even after removing Docker from the system, this command offers advice about how Docker should be configured to access the sophisticated built-in image registry that is available for MicroK8s on *localhost:32000*.

Listing 3: `microk8s inspect` Output

```
Inspecting services
  Service snap.microk8s.daemon-cluster-agent is running
  Service snap.microk8s.daemon-flanneld is running
  Service snap.microk8s.daemon-containerd is running
  Service snap.microk8s.daemon-apiserver is running
  Service snap.microk8s.daemon-apiserver-kicker is running
  Service snap.microk8s.daemon-proxy is running
  Service snap.microk8s.daemon-kubelet is running
```

Listing 4: Connecting to the Nginx Pod

```
$ curl -k http://10.1.167.10

<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
       body {
              width: 35em;
              margin: 0 auto;
              font-family: Tahoma, Verdana, Arial, sans-serif;
       }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
```

Raspberry Pie

The proof of success is, of course, demonstrating that a workload is running on Kubernetes. As a test, I chose one of the most popular container images on the Internet, the lightweight Nginx webserver. You can create an Nginx deployment with the relatively standard Kubernetes command:

```
$ microk8s kubectl create ➥
  deployment nginx --image=nginx
```

Within just a few seconds the command completes, and an Nginx pod has been created. It will also be re-created if it fails for some reason, because the deployment keeps an eye on it.
Once live, you can run a command that describes all pods in the default namespace (which by default is no pods, other than the newly created web server pod). To see the details of the Nginx pod, enter:

```
$ microk8s kubectl describe po
```

Next, you can take the IP address of the pod and use the `curl` command to connect to it, producing the abbreviated output in Listing 4. In this way, you've proven that you have a workload running courtesy of Kubernetes on your Raspberry Pi.

Ice Cream On Top

When you get a bit more involved with MicroK8s, you can integrate an impressive list of available add-ons, such as those for address resolution and storage services:

```
$ microk8s enable dns storage
```

You can find the list of add-ons online [13]. If you're curious, the `storage` add-on offers "… a default storage class which allocates storage from a host directory," and `dns` installs the excellent CoreDNS. The documentation notes that this may be a mandatory requirement for some applications and that you should always enable it.
If you want to see which commands do what with MicroK8s, visit the Command Reference page [14]. Note you can add multiple nodes to the Kubernetes cluster with the instructions in the `microk8s add-node` link.
Among the basic commands, it is simple to stop MicroK8s and start it up again:

```
$ microk8s stop
$ microk8s start
```

If you want to tidy up your Raspberry Pi after your MicroK8s tests, you can remove the snap package,

```
$ snap remove microk8s
```

and you can tidy up the snap files:

```
$ apt purge snapd squashfs-tools
```

If you encounter any headaches, you should check the Troubleshooting page [12]. You will find a number of hints and tips and discover where to send bug reports.

The End Is Nigh

As you can see, this little Kubernetes distribution is quite something. It operates on almost no compute capacity and only appears to add roughly 700MB of RAM footprint without any workloads, relative to the 2.9GB noted previously, leaving lots of headroom for application workloads.
As the documentation suggests, with a Raspberry Pi, you might embed a Kubernetes distribution on a racing car to push engine metrics up to the cloud for analysis or integrate it with a satellite chassis for telemetry from Earth's orbit. It's intriguing to follow where this space is going, and I hope this look at MicroK8s has given you a welcome insight.
A zero-ops installation of Kubernetes is well worth learning for the future, to keep abreast of associated innovations. ■

Info

[1] K3s: [https://k3s.io]
[2] minikube: [https://minikube.sigs.k8s.io/docs/start]
[3] MicroK8s: [https://microk8s.io]
[4] Amazon Linux 2 live kernel patching: [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/al2-live-patching.html]
[5] Whitepaper: [https://ubuntu.com/engage/microk8s-451research]
[6] Alternative Installs: [https://microk8s.io/docs/install-alternatives]
[7] snap: [https://snapcraft.io]
[8] "Packaging Apps To Run on Any Linux Device" by Chris Binnie, *ADMIN*, issue 41, 2017: [https://www.admin-magazine.com/Articles/Container-Apps]
[9] Snap on Raspberry Pi docs: [https://snapcraft.io/docs/installing-snap-on-raspbian]
[10] Selecting a snap channel: [https://microk8s.io/docs/setting-snap-channel]
[11] Calico: [https://www.projectcalico.org]
[12] MicroK8s troubleshooting: [https://microk8s.io/docs/troubleshooting]
[13] MicroK8s add-ons: [https://microk8s.io/docs/addons#heading--list]
[14] Command Reference: [https://microk8s.io/docs/command-reference]

Author

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

Shop the Shop shop.linuxnewmedia.com

Missed an issue?

You're in luck.

Most back issues are still available. Order now before they're gone!

shop.linuxnewmedia.com

OPA and Gatekeeper enforce policy defaults in Kubernetes

Watchdog

Enforce container compliance in Kubernetes in one of two ways: with Open Policy Agent or Gatekeeper. By Martin Loschwitz

For compliance officers and chief information security officers (CISOs), the motto of the day is clear: Container-based setups need no more and no less compliance and security than their conventional relatives; they need different but equally well-monitored compliance. A container environment is where the Open Policy Agent (OPA) [1] with its Kubernetes sidecar on the one hand and the Gatekeeper policy enforcement service built specifically for Kubernetes (K8s) on the other hand enter the play. Of course, Gatekeeper relies on OPA in the background, as well.

In this article, I introduce OPA and its possible spheres of application and show how integration works with a sidecar or Gatekeeper in K8s.

Flexibility

If you ask developers and admins what they particularly like about containers, you regularly hear the same answers: Containers are flexible, dynamic, easy to manage – at least that's what sworn container fans claim. In fact, containers embody the ideas of agile development particularly well, symbolized by the cloud-ready architecture with its principle of microservices.

What excites developers and admins in terms of flexibility and dynamics, however, regularly puts worry lines on the foreheads of compliance officers and CISOs. All too great is the temptation for many a developer or administrator to use a ready-made image for containers from the Internet, roll it out on their own infrastructure, and just say, "well, it works for me," without considering the security and compliance implications of the operation. This issue has already been addressed in the past, but it doesn't hurt to take at least another quick look at the topic of container compliance.

Compliance

The relevance of security and compliance in the container context can hardly be overestimated. True, containers today no longer regularly run with the rights of the system administrator – although they could if the admin wanted them to, which is also a compliance issue. Nevertheless, a container image from a dubious source with a built-in Bitcoin miner can already endanger the stability of a setup in terms of network and storage.

The dangers of DIY images prove to be at least as great. "Works on my Macbook Air in VMware" has become a catchphrase for naive people who assemble an image locally and then distribute it around the world without providing the sources or a comprehensible list of the steps involved. Any admin who uses such an image will be blown away (at the latest) by the first security audit. If

Photo by Mac Gaither on Unsplash

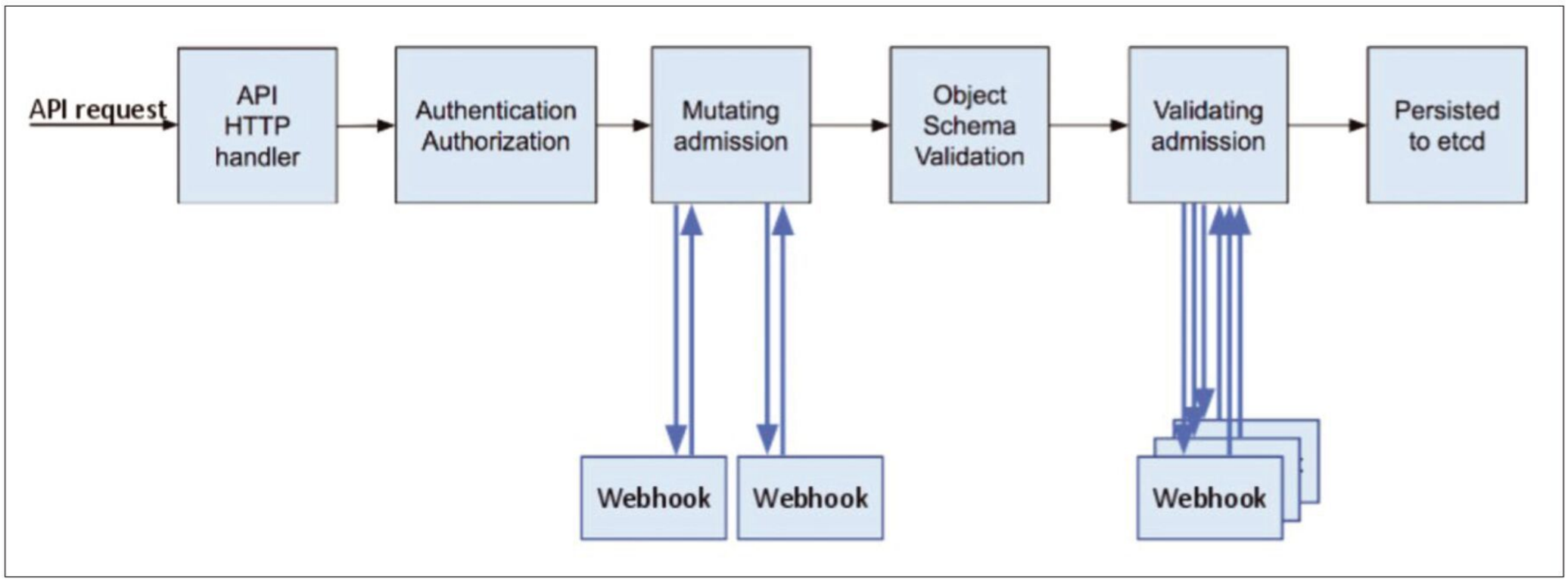


Figure 1: **Admission controllers are a built-in method for providing compliance protection in Kubernetes. © Kubernetes**

a remotely exploitable vulnerability is found in one of the services in the container, finding a way out can be expensive if you don't know how to build the image with a new version of the service or where you can find a legitimate version.

Much Confusion

Users and admins dealing with OPA and Gatekeeper for the first time are regularly confused by their names: Gatekeeper also regularly goes by the name OPA 1.0 on the web, and the terms "Open Policy Agent with Kubernetes sidecar" and "Gatekeeper as Kubernetes policy engine" somehow seem to be related. Anyone dealing with both components would therefore do well in the first step to clarify and understand how OPA and Gatekeeper differ.

However, the question cannot be answered conclusively because OPA and Kubernetes sidecars are two separate components. OPA was originally developed in the context of K8s, but today the product is a standalone component. The purpose of OPA is to provide a completely generic way to define compliance rules and to check for compliance in programs of any kind. A very simple example would be the rule that a web server must not listen on port 80 but use port 443 (i.e., SSL encryption). By means of OPA and an integration of the web server in OPA, you could then check afterwards whether this is actually the case.

Kubernetes comes into play much later. Here, OPA acts as an admission controller (Figure 1). From an admin's point of view, you have two different options. OPA can be connected to K8s by a sidecar. The sidecar then regularly retrieves the defined compliance rules and checks whether the rolled-out application adheres to them. Gatekeeper extends this principle: It belongs as a component to the deployments in K8s and enforces the implementation of compliance rules directly at the container level.

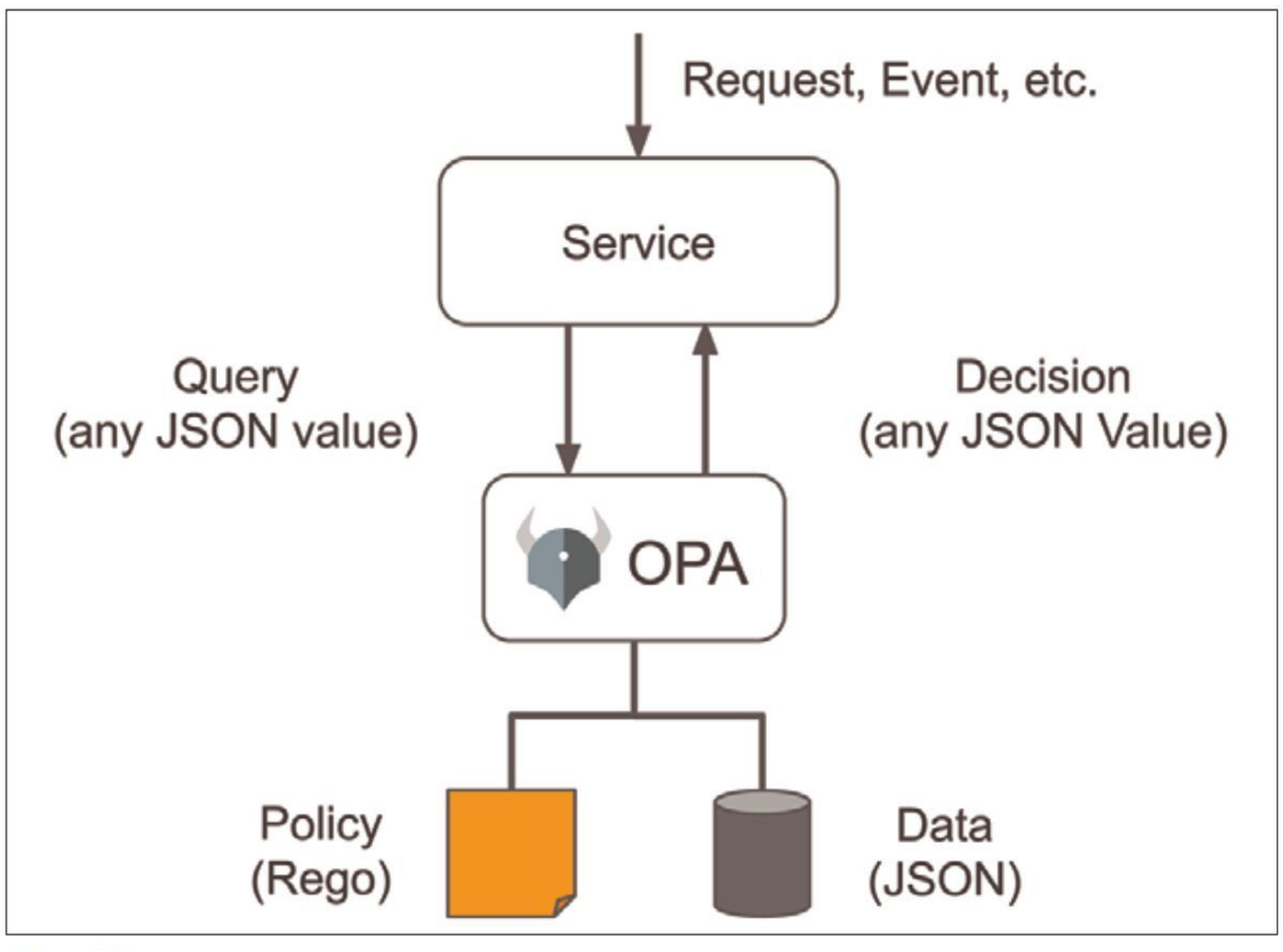


Figure 2: **As the architecture of the Open Policy Agent shows, OPA is a generic tool for policy enforcement. © OPA**

Focus on OPA

Admittedly, the idea behind OPA is not entirely new. Various other implementations of policy frameworks are already on the market, such as Chef InSpec [2], for which you define rules for compliance checks in a separate language and which then enforces the defined parameters. Why is OPA needed at all? The people behind the solution confidently answer this question on their website: Unlike other solutions, OPA is suitable for cloud-native environments. After this statement, however, you are not more informed. Where the strengths of OPA actually lie can only be seen by taking a closer look at the architecture of the solution (Figure 2).

Task Sharing

One core feature immediately stands out in OPA: The tool distinguishes between setting and enforcing its own processes in policy decisions. Other compliance tools follow classic automation, wherein a decision relating to a policy automatically leads to its enforcement. For example, if InSpec detects that a service is listening on a prohibited port, it immediately outputs an error message and stops it. In today's distributed applications, however, this level of automatism might not be desired. This conflict can only be rectified if the policy decision and the consequences to be taken from it come from different instances, which is exactly where OPA sees its use case: It enables external programs to perform compliance tests according to defined rules but then leaves the decision about consequences to the respective applications.

How does this work in practice? Under the hood, OPA distinguishes between its own policy engine and the part of the application that processes data. External applications turn to OPA with requests that contain data in the structured JSON format. For example, a service running on a target system could compile a list of all the host's network interfaces and send it to OPA afterwards. The next step in the process is to use OPA's policy engine.

Rego Declarative Language

The OPA developers have built their own declarative language named Rego for this purpose. However, this development is not completely new: It is strongly reminiscent of the established Datadog, which, however, is simply too old to support the JSON format. Rego essentially consists of Datadog with minimal changes and a JSON extension.

With Rego, you define the set of rules that OPA uses when evaluating incoming compliance requests. The developers provide a complete guide to Rego [3] in their documentation and also address special features, such as the ability to add modules. Undoubtedly, however, the most important factor in Rego is a fixed form for responses to requesting services but no fixed content. Just as the original request must come in JSON format, the response to a compliance request must also reach the counterpart in JSON format. However, you define the content of the response and write the compliance rule, rather than OPA doing so on the basis of any standard assumptions.

In the final step, OPA sends the response to the compliance request to the requesting agency – leaving it entirely up to the agency to decide how to handle any negative outcome.

Complex and Versatile

The described process of policy checking in OPA is admittedly far more complex than this simple example is able to depict, primarily because of the large number of features in OPA. Rego, as a declarative language for setting policy rules, now provides support for more than 150 functions that interpret JSON, iterate over it, and modify it as needed.

Anyone who focuses only on the policy engine in OPA, however, is doing an injustice to the data part of the solution because it is also extremely powerful and, almost more importantly, can put data into context. In concrete terms, this means that OPA's policy engine can not only make decisions on the basis of information from a single program, it can also include data from other services or servers in its calculations. For example, you can configure your systems to offload a variety of different data to OPA on a regular basis. In the next step, checks draw on all the available data.

This versatility makes OPA very different from other compliance tools. Their workflow usually involves relating different data to each other directly (e.g., by defining static profiles). What may have made sense for conventional solutions can cause a problem in dynamic environments such as clouds because the risk is that later admins will find it very difficult to reconstruct checks that were set up ages ago. The implicit changes in the environment in clouds would also mean that profiles and the connections between them would have to be maintained and kept up to date regularly, which becomes a Sisyphean task in highly dynamic environments.

Including Clients

Ultimately, the question remains of how developers integrate OPA into their solutions; you have several possibilities here. The most common variant is to run OPA as a typical Linux daemon and to write requests to it in REST format. This approach is obviously aimed squarely at cloud environments, in which individual components regularly talk to each other over HTTPS anyway. OPA's API is open source and well documented, so developers can integrate it easily.

The second option for OPA integration is to implement the service right at the programming level with a Go library. Because the library is only available for Go, this approach admittedly comes into its own primarily for Go tools. However, the Go programming language is extremely popular, especially in the cloud-ready environment, so this limitation should be fine for most current developers.

Not Only Kubernetes

At this point, at the latest, it also becomes clear that OPA is no longer just an appendage of K8s and never really was. The possibilities of using OPA for compliance decisions go far beyond K8s, as some impressive examples show. The pluggable authentication module (PAM) system in Linux, for example, lets you offload username authentication into separate modules that can be wired up in series. As a kind of proof that OPA can play along here in a meaningful way, its developers have written a PAM module that communicates with OPA in the background (**Figure 3**).

If a user has the admin role in OPA's

Figure 3: With PAM, a use case other than Kubernetes can be found for OPA. However, the tool is probably still most commonly used with Kubernetes.

data, the OPA module allows them to log in – and couple `sudo` with OPA in this way.

The example is not very original because similar effects can be achieved with on-board tools and LDAP. If you take the example a step further, however, things become more coherent: A separate daemon, for example, could compare the users currently logged in on the system with a whitelist of permitted logins. If someone is logged in who is not on the guest list, the tool raises an alarm and an admin takes a look. OPA can also be used well and sensibly in other contexts (e.g., in the Terraform orchestrator or in Envoy, which has already been discussed in detail in *ADMIN* as a mesh for Kubernetes [4]).

This article is ultimately about how to enforce certain compliance issues in K8s. Now that it is clear that OPA provides the appropriate foundations, a concrete question arises: How do you implement a K8s watchdog that, with OPA behind it, takes care of enforcing your defined rules? This is where the two options mentioned at the very beginning of this article come into play again: the `kube-mgmt` sidecar for OPA and Gatekeeper. Basically, both tools aim to enable compliance decisions in a K8s cluster with OPA. However, they do so in very different ways.

Sidecar

As already described, OPA itself does not initiate any actions if a policy check produces a negative result. It also does not initiate the corresponding checks itself but merely provides the engine that external components can use. Of course, you must therefore store suitable compliance rules for K8s in OPA, which is relatively simple. Also, you must provide a component to K8s that reads the workloads there, initiates the compliance check by OPA, and, if necessary, takes action if it fails.

The `kube-mgmt` sidecar, the older of the existing solutions, does several things in parallel for this purpose. First, it rolls out OPA automatically into an existing OPA cluster. The sidecar then takes care of getting OPA up and running, including redundancy, so you do not need to worry about that. The only thing you do have to take care of is providing a suitable SSL certificate. For K8s to communicate with OPA, it insists on an encrypted connection. Once this has been established, OPA can be used in K8s (e.g., to avoid creating resources that fail to meet certain specifications).

Consider the example of an enterprise policy that specifies that all services opened to the Internet must be accessible on port 443 in the *example.net* domain. To do this, you store appropriate policies for K8s in OPA. For this purpose, you would use `kubectl` to create an entry of the `ConfigMap` type; the file contains the plain vanilla instruction for OPA in Rego. Rego supports you with functions such as `fqdn_matches_any` or `fqdn_matches`, which can be used to check hostnames automatically.

The `ConfigMap` object contains only the logic that makes the policy decision according to the parameters of specified hosts. You do not store the domains themselves there but rely on an annotation directive in the pod definition of the namespace where the target services will run. Admins regularly create separate namespaces for different compliance directives in their clusters to keep things clear. Finally, you specify that the existing OPA instance act as the admission controller for K8s, meaning that Kubernetes outsources compliance decisions to OPA.

The rest is then simple. When you launch instances such as ingress controllers in your namespace, you define their basic parameters in the usual way with `serviceName` and `servicePort`. If these contradict the specifications previously made in OPA, K8s refuses to create the corresponding resource and outputs an appropriate justification. To put it casually, compliance control via `kube-mgmt` is thus the smaller and somewhat simpler solution.

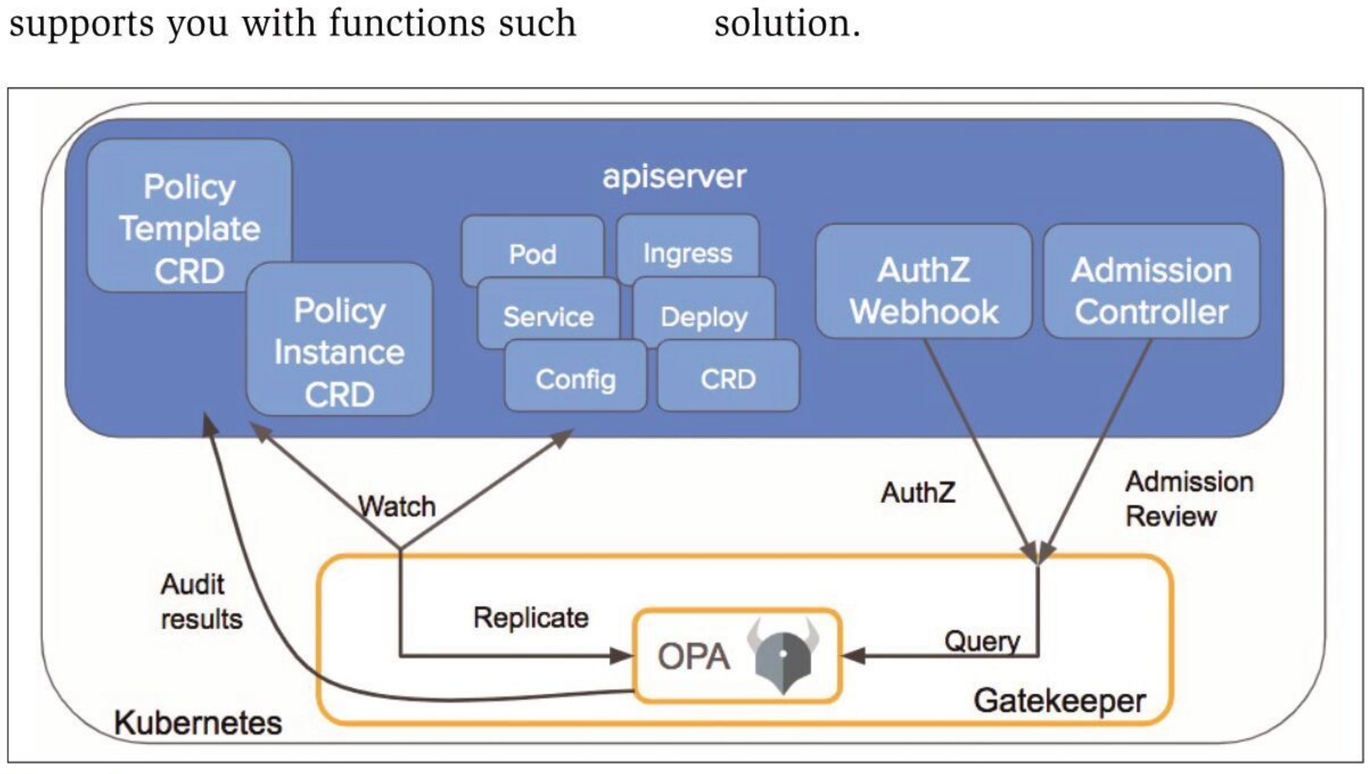


Figure 4: Gatekeeper uses CRDs and its own policy templates in Kubernetes to act as an admission controller and thus determine the fates of resources. © Gatekeeper

Full Solution

Gatekeeper takes a slightly different approach. The differences here are really in the details – and those in the later parts of the setup. First, you need a running OPA instance, even in a setup with Gatekeeper. However, once Gatekeeper is active in K8s, it extends it to include custom resource definitions (CRDs) that allow OPA to be started automatically (Figure 4); then, you do not have just one OPA instance but as many instances as you and users initiate.

Another detail is fundamentally different. When using Gatekeeper, you do not define the compliance rules yourself in OPA. Instead, Gatekeeper comes with a wrapper that you use to import the policy rules as constraint templates, which the user then applies to create constraints. This method proves to be extremely practical in everyday use because it not only lets admins with access to OPA define compliance rules but also normal users with access to K8s. The practical thing is that users are allowed to create as many templates with constraints for OPA as they want and can dynamically apply them to different resources, which is far more flexible than the `kube-mgmt` solution.

```
untitled 2
1  apiVersion: constraints.gatekeeper.sh/v1beta1
2  kind: K8sRequiredLabels
3  metadata:
4    name: ns-must-have-hr
5  spec:
6    match:
7      kinds:
8        - apiGroups: [""]
9          kinds: ["Namespace"]
10   parameters:
11     labels: ["hr"]
12 status:
13   auditTimestamp: "2019-08-06T01:46:13Z"
14   byPod:
15   - enforced: true
16     id: gatekeeper-controller-manager-0
17   violations:
18   - enforcementAction: deny
19     kind: Namespace
20     message: 'you must provide labels: {"hr"}'
21     name: default
22   - enforcementAction: deny
23     kind: Namespace
24     message: 'you must provide labels: {"hr"}'
25     name: gatekeeper-system
26   - enforcementAction: deny
27     kind: Namespace
28     message: 'you must provide labels: {"hr"}'
29     name: kube-public
Line: 33:22 Plain Text  Tab Size: 4
```

Figure 5: Gatekeeper is somewhat better prepared out of the box to be audited than is `kube-mgmt`.

Better Focused on Compliance

Gatekeeper and `kube-mgmt` have many similarities and few – but tangible – differences. Both approaches produce logfiles that can be collected with centralized logging tools such as the Elasticsearch, Logstash, and Kibana stack, or Loki. Moreover, because OPA itself is at the heart of both approaches, it can be configured in

Listing 1: Constraint Template

```
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8srequiredlabels
spec:
  crd:
    spec:
      names:
        kind: K8sRequiredLabels
      validation:
        # Schema for the `parameters` field
        openAPIV3Schema:
          properties:
            labels:
              type: array
              items: string
targets:
  - target: admission.k8s.gatekeeper.sh
    rego: |
      package k8srequiredlabels

      violation[{"msg": msg, "details": {"missing_labels": missing}}] {
        provided := {label | input.review.object.metadata.labels[label]}
        required := {label | label := input.parameters.labels[_]}
        missing := required - provided
        count(missing) > 0
        msg := sprintf("you must provide labels: %v", [missing])
      }
```

Whence Come the Rules

Gatekeeper and `kube-mgmt` have one feature in common: You have to create the compliance rules required for the specific application themselves. Ready-made rulesets for different areas of application cannot be found on the web, at least not yet. It is far beyond the scope of this article to go into detail about Rego and how to use Rego in OPA for Kubernetes. The examples in Listings 1 and 2 are taken directly from the docs for Gatekeeper and show what rule enforcement can look like in principle. The first listing defines the constraint template, and the second implements the concrete constraint with payload data. To load both constraints into a K8s cluster, use:

```
kubectl apply -f
```

The example enforces that the `gatekeeper` label is set for each namespace within the K8s installation. If an admin creates a namespace without this label, K8s returns the request with an error message and refuses to create it.

both cases to send logs of its compliance decisions directly to an external server over HTTP. From that server, for example, you could grab the logs and make them part of an audit log that any certification authorities are likely to want to see (Figure 5). Alternatively, you can use Gatekeeper's audit-safe trail feature that also logs compliance decisions and makes them immediately visible in the constraint entries in K8s.

Gatekeeper or kube-mgmt?

No matter how you spin it, the details that differentiate OPA with `kube-mgmt` on one side from Gatekeeper on the other do not turn out to be earth-shattering. In both cases, the same up-to-date OPA versions are used; in both cases, the administrator has to change the configuration of a Kubernetes cluster to use OPA as an admission controller.

Gatekeeper may have an advantage here because it allows policy configuration from within K8s, whereas with the `kube-mgmt` variant, the admin is up against the OPA instance itself. (See the "Whence Come the Rules" box.) However, this only makes a relevant difference where the admin for the workload in K8s is not the admin running Kubernetes itself.

For those new to permission control and the various factors that can be used to decide whether to allow or disallow resource creation in Kubernetes, `kube-mgmt` is probably a little better than the larger Gatekeeper. ■

Listing 2: Constraint with Payload

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-gk
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Namespace"]
  parameters:
    labels: ["gatekeeper"]
```

ADMIN, issue 59, 2020, pg. 70
[https://www.admin-magazine.com/Archive/2020/59/A-versatile-proxy-for-microservice-architectures/]

Info

[1] OPA: [https://www.openpolicyagent.org/]
[2] InSpec: [https://docs.chef.io/inspec/]
[3] Rego: [https://www.openpolicyagent.org/docs/latest/policy-language/]
[4] "A versatile proxy for microservice architectures" by Martin Loschwitz,

The Author

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

Too Swamped to Surf?

Our ADMIN Online website offers additional news and technical articles you won't see in our print magazines.

Subscribe today to our free ADMIN Update newsletter and receive:

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

bit.ly/HPC-ADMIN-Update

Persistent storage management for Kubernetes

Data Logistics

The container storage interface (CSI) allows CSI-compliant plugins to connect their systems to Kubernetes and other orchestrated container environments for persistent data storage. By Ariane Rüdiger

With ongoing developments in container technology in the area of data management and persistent storage, business-critical applications have been running in cloud-native environments with containers orchestrated by Kubernetes. A container is just a software box without its own operating system. Originally, a container would not only contain the app or microservice, but also all the necessary drivers, dependencies, and data the respective application needed to run. If the container was deleted, all that was gone, which meant that a data store was needed that would stay alive regardless of the existence of a container or its pod.

PVs, PVCs, and Storage Classes

Two variants are available: persistent volumes (PVs) and persistent volume calls (PVCs). PVs are static and defined by the admin in advance, belong to a Kubernetes cluster, and perish with it, but survive the deletion of individual containers. The admin assigns all their characteristic properties: size, storage class, paths, IP addresses, users, identifiers, and the plugin to be used.

Storage classes have certain characteristics such as quality of service (QoS), replication, compression, backup, and more, which the container storage interface (CSI), but not Kubernetes itself, supports. PVs now have many different storage classes in Kubernetes clusters, from local storage to storage attached with Network File System (NFS), iSCSI, or fibre channel to block storage from Azure, AWS, or Google. All PVs of a storage class are accessible through the same API or are coupled to the pod.

PVCs, on the other hand, are requested by the application manager as a storage class according to the application requirements. Depending on the request, a PVC is created from the template of the corresponding storage class and is attached to the pod from that point on (i.e., it also goes down with the pod). A stateful set means that PVCs can be copied across multiple pods. Their properties are defined in the YAML declaration of a pod. Ultimately, the PVC is allocated as much storage as users guess they need for a very specific application.

If a pod is running on a host, you can also define a path to a host directory where the container data will end up, but only as long as the container is actually running on the host. Data blocks or objects related to containerized apps can also be stored locally, but only as long as the container actually remains on this hardware.

Container Storage Interface

Today, the CSI [1] referred to earlier has become very important. It is an interface developed by the Cloud Native Computing Foundation (CNCF) so that storage system providers can connect their systems to Kubernetes and other orchestrated container environments without their own drivers. CSI has gained market acceptance and is supported by many storage vendors.

Before CSI, plugins for volumes had to be written, linked, compiled, and shipped with the Kubernetes code – an expensive and inflexible process – because every time new storage options made their way into the systems, the Kubernetes code itself had to be changed. Thanks to CSI,

Photo by CHUTTERSNAP on Unsplash

this is no longer the case; the Kubernetes code base is now unaffected by changes to the supported storage systems.

A CSI-compliant plugin comprises three components: a controller service, a node service, and an identity service. The controller service controls the storage and includes functions such as Create, Delete, List, Publish, Snapshots, and so on. The container node accesses the storage through the node service. Important functions include volume staging and publishing, volume statistics, and properties. The identity service provides information about the attached storage.

In total, a standards-compliant CSI comprises around 20 functions. If these comply with the CSI specification, administrators have access to a functioning plugin for connecting storage to any container system. The CSI controller service runs on the controller node there, but any number of nodes can be connected by the node service. If a node that used a specific volume dies, the volume is simply published to another node where it is then available. The major container orchestration systems (Kubernetes, OpenShift, PKS, Mesos, Cloud Foundry) now support CSI – Kubernetes as of version 1.13.

Kubernetes complements CSI with its own functions, including forwarding storage class parameters to the CSI drivers. Another option is encrypting identification data (secrets), automatically decrypted by the driver, and the automatic and dynamic start of the node service on newly created nodes. In a Kubernetes environment, multiple CSI drivers can work in parallel. This capability is important when applications in the cluster have different storage requirements. They can then choose the appropriate storage resource, because they are all equally connected to the cluster by CSI. Kubernetes uses redundancy mechanisms to ensure that at least one controller service is always running. Kubernetes thus currently offers the most comprehensive support of all orchestrators for CSI.

However, CSI requires additional middleware software components outside of the Kubernetes core when working in Kubernetes environments. These components ensure the fit between the particular CSI and Kubernetes version in use. The middleware registers, binds, detaches, starts, and stops the CSI drivers. In this way, the external middleware, which was programmed by the Kubernetes team, provides the existing nodes with the required storage access.

Most CSI drivers are written in Go, with the support of the GoCSI framework, which provides about a quarter of the necessary code, including predefined remote procedure code (GoRPC). A special test tool is also available. Dell EMC, for example, uses this framework for some of its storage products.

The numerous open source projects centered around container storage fill functional gaps and deficiencies in Kubernetes, mostly in terms of managing persistent storage and data. Kubernetes needs these add-ons to provide a secure environment for business applications. Currently, about 30 storage projects are on the CNCF's project map, many of which have already been commercialized. I will be looking at Ceph, Rook, Gluster, and Swift in more detail here, in addition to short descriptions of other projects with a focus on container storage.

Ceph Management Tool

The open source storage platform Ceph [2] was developed in its basic form as early as 2004. Today, it is often used with the Rook container storage software. Currently, Red Hat, SUSE, and SanDisk are the main contributors to its development. Ceph is implemented on a distributed computing cluster and is suitable for object, block, and file storage. The system features automatic replication, self-healing, and self-management, while avoiding a single point of failure. Commodity hardware is sufficient for a Ceph environment. The object store is based on Reliable Autonomic Distributed Object Store (RADOS).

The `ceph-mon` cluster monitor monitors the function and configuration of the cluster nodes. It stores information about the placement of data and the general state of the cluster. The `ceph-osd` object storage daemon manages directly attached disk storage (BlueStore), whose entries are recorded in a journal, and the metadata server daemon `ceph-mds` keeps the metadata of all data objects stored in a Ceph system. Managers (`ceph-mgr`) monitor and maintain the cluster and interface (e.g., with external load balancers or other tools). HTTP gateways (`ceph-gwy`) provide an Amazon Simple Storage Service (S3)/Swift interface to the object storage.

Because of its tie-ins to the further history of Ceph, I also need to mention GlusterFS, developed in 2005. The plan was an open source platform for scale-out cloud storage for private and public clouds. In 2011, Red Hat bought Gluster, which has since been acquired by IBM. Red Hat initially marketed GlusterFS as Red Hat Storage Server, then bought Ceph, combined the two technologies, and now markets the solution as Red Hat Gluster Storage.

Rook Storage Orchestration

Rook [3] is a cloud-native storage orchestrator for Kubernetes. It is based on a Ceph cluster (Luminous or higher, Kubernetes 1.6 or higher) and runs a distributed filesystem directly on the storage cluster. Rook provides interfaces for scheduling, life cycle and resource management, security, monitoring, and the cloud user experience. The software is built on top of Ceph's node structure and has additional components that monitor and control the installation and operation of Ceph pods. For example, a Rook agent is installed on each node. It provides part of the storage driver for Kubernetes. The Rook operator, mainly developed by CoreOS, monitors and controls the individual agents and other parts of the Ceph cluster within a Kubernetes environment.

Rook offers the following functions and others: storage management,

even in hyperscaled or hyperconverged storage clusters; effective data distribution and replication; and provisioning of file, block, or object storage for various providers. Rook can also be used to optimize workloads on commodity hardware.

Swift Object Storage in OpenStack

Swift [4] is another important project used in OpenStack to implement object storage with ring structures. The rings are used for mapping between the names of entities stored in the cluster and their physical equivalents on disk. Within the ring are zones, devices, partitions, and replicas. Each partition is replicated at least three times in the cluster. The locations of the copies are stored in the ring mappings.

In case of failure, the ring takes over the switching to intact resources. Data can be isolated within one zone of the ring; replicas are kept in different zones (data center, cabinets, servers, or even switches). Partitions are distributed across all the devices in a ring. The ring does not manage itself, but is managed externally.

The replication mechanism continuously checks for the desired three copies by reading the hash files of the partitions. Zones (racks, servers, one or more drives) are designed to isolate errors. Partitions, on the other hand, are collections of stored data (e.g., account or container databases). Partitions form the core of the replication system.

Each ring is accessed by proxies and their APIs. Proxies are also responsible for coordinating responses and timestamps and handling failures. They have a share-nothing architecture and can scale as needed. At least two need to be present for redundancy. Containers are represented as individual SQLite databases distributed across the cluster. The same is true for accounts. Here, an account database contains all containers that belong to the account. A container database stores all objects in the container.

Other Open Source Projects

The Soda Foundation's Soda Open Data Autonomy (SODA) project [5] is interesting. A uniform API layer is planned, through which applications can access data independent of the underlying storage or logical structures. However, the respective platforms need a SODA plugin.

SODA consists of an infrastructure manager for the entire storage infrastructure. The SODA API acts as a central external interface that seamlessly connects to heterogeneous storage back ends, unifying the usual heterogeneous data and storage management APIs. One controller handles all metadata and state management. The drivers of the different storage back ends are connected to what is known as a SODA dock. There is also a component for multicloud management.

The following is a quick look at some of CNCF's other container-related open source projects:

- Linstor is a Kubernetes-integrated block storage management tool for large Linux clusters and implements persistent block storage for OpenStack, OpenNebula, and OpenShift.
- Longhorn is useful for building distributed block storage in Kubernetes environments.
- OpenEBS implements open container-attached storage in Kubernetes environments, which enables stateful applications to access dynamic local or replicated PVs more easily. Users include Arista, Orange, Comcast, and CNCF.
- Stash backs up stateful applications in Kubernetes environments. The project is based on the Restic open backup application. Stash uses a declarative interface and custom resource definition (CRD) to control backup behavior.
- Velero also backs up Kubernetes resources, but it is also useful for migrations and disaster recovery of persistent volumes between Kubernetes cluster resources.
- MinIO implements an S3 object store for Kubernetes environments without interacting directly with Kubernetes. The solution gets by with a single software layer. Features include erasure coding, encryption, immutable storage, identity management, continuous backup, global data aggregation, and a universal cloud interface. MinIO runs on bare metal and all private clouds but can connect to NAS storage.

Commercial Open Projects

Of course, commercial offerings are also at hand. Portworx [6], recently acquired by Pure Storage, is very successful. The software equips any Kubernetes environment with professional functions for data backup: For example, Portworx backs up volumes from container environments to the cloud and makes application-consistent snapshots. Pure Storage has assured that the Portworx business model will be continued as before, at least for the time being.

Kasten [7], specializing in Kubernetes backup, was recently acquired by backup vendor Veeam, completing Veeam's portfolio of securable infrastructure environments through container landscapes. Other examples of commercialized CNCF projects include:

- Trilio, a solution for protecting Kubernetes, OpenStack, and Red Hat virtualization environments, can create point-in-time snapshots of the corresponding environments.
- Ionir, with its Data Teleport technology, ports data and persistent volumes between different cloud platforms reportedly in less than 40 seconds without manual intervention. Other features include global deduplication, compression, and data recovery. Different infrastructure pools can be merged into a unified, cross-managed data environment. The prerequisite, however, is that the data resides in Kubernetes environments.
- Robin Cloud Native Storage (CNS) from Robin.io makes it possible to deploy applications such as big data, databases, or

artificial intelligence (AI)/machine learning (ML) on cloud platforms very quickly with just a few clicks with its hyperconvergent Kubernetes platform.
The time-consuming setup of the entire container environment for the respective application is completely eliminated. Customers log on to the Robin platform, click on the desired application, and define its parameters with an interactive configuration interface. The Robin environment does the rest. Included are functions such as cloning of data, metadata, and configuration; replication; migration; and snapshots. Robin CNS is CSI compliant and can therefore communicate directly with native Kubernetes tools.

VMware: Farewell to the VM-Only World

Here, I present some examples of how large infrastructure providers implement and manage container storage. VMware manages both containers and virtual machines (VMs) and corresponding stateful services with a virtual storage area network (vSAN) and vSphere. VMware Cloud Foundation [8], meanwhile, is equipped with a CSI interface. VMware drops a policy layer on top of vSAN, virtual volumes (vVols), and Virtual Machine File System (VMFS)/NFS, followed by separate file and block interfaces. The next layer in the stack is the central cloud-native storage control plane. Kubernetes and the persistent volumes defined there can access the control plane through the CSI interface.
The vSAN Data Persistence Platform is essential for stateful services on both a container and VM basis. It provides a docking point for application partners, which include Cloudian, DataStax, Dell, and MinIO. The VMware Cloud Foundation services also let users access container infrastructure in the cloud and VMs with Kubernetes and RESTful APIs. Integration with vSphere is planned. Stateful services from partners can now be managed from partner-built dashboards in vCenter.

Red Hat and NetApp

OpenShift Container Storage [9] is purely software-defined and optimized for the Red Hat OpenShift Container Platform. Files, blocks, and objects are supported. The platform is a part of Red Hat Data Services. The goal is to provide a consistent user experience regardless of infrastructure. Persistent and highly available container storage can be dynamically provisioned and released on demand.
The software is suitable for databases, data warehouses, automating data pipelines, and highly active data in continuous deployment development models. Other application areas include AI, ML, and analytics, which particularly benefit from Kubernetes and microservices-based data services.
Red Hat claims accelerated application development and deterministic database performance as a foundation for data services are benefits of Red Hat Container Storage. Other benefits include simplified storage handling for analytical applications and protection and resilience for persistent volumes and namespaces.
Finally, NetApp deserves a mention. The company seeks to provide users with platform-as-a-service environments with complete management for data tasks within provider infrastructure clouds. Recently, the manufacturer presented two new services that complement Ocean [10] intended for stateless apps in the Kubernetes environment: Astra and Wave.
Astra for Kubernetes protects applications and includes migration and recovery. No software needs to be downloaded, installed, managed, or updated. Features include snapshots for local backup and recovery on the same Kubernetes cluster; application-based disaster recovery, even in another region and on another cluster; and active cloning of applications, along with their data, for migration purposes to another Kubernetes cluster, regardless of its location.
Wave, for more analytical environments, implements a managed Spark environment on AWS and will do so on Azure, Google, and other cloud platforms in the near future. It uses the version of Spark the customer wants to use, along with key tools for streaming data into Spark and for managing queries against Spark data. NetApp has made other announcements in the area of persistent data management in Kubernetes.

Conclusions

Although Kubernetes initially left much to be desired in the way it handled persistent data storage and corresponding enterprise functions, the landscape of potential solutions for theses tasks is now very broad. Because the further proliferation of containers as the future standard infrastructure for cloud environments is very likely, this diversity is likely to increase, so it seems realistic to assume that virtually no current or future storage management challenge will remain unanswered by an efficient software solution in the long run. ■

Info

[1] Container Storage Interface on GitHub: [https://github.com/container-storage-interface/spec/blob/master/spec.md]
[2] Ceph: [https://ceph.io/en/]
[3] Rook: [https://rook.io]
[4] Swift: [https://www.swiftstack.com/product/open-source/openstack-swift]
[5] SODA: [https://sodafoundation.io]
[6] Portworx: [https://portworx.com]
[7] Kasten: [https://www.kasten.io]
[8] VMware Cloud Foundation: [https://www.vmware.com/products/cloud-foundation.html]
[9] Red Hat OpenShift Container Storage: [https://www.redhat.com/en/technologies/cloud-computing/openshift-data-foundation]
[10] NetApp Ocean: [https://spot.io/products/ocean/]

Hone your skills with special editions!

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

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

Available in print or digital format

Check out the full library!

shop.linuxnewmedia.com

FREE DVD!
openSUSE Leap 15.2

JOIN THE LINUX REVOLUTION!
ALL THE SOFTWARE YOU NEED!

GETTING STARTED WITH
LINUX

• MORE POWERFUL • MORE SECURE • MORE FU

LEARN HOW TO SET UP A LINUX SYSTEM TO:
• Listen to Music • Play Games • Process Photos
• Surf the Web • and Much More!

START

2020 Edition
LINUX Special
WWW.LINUX-MAGAZINE.COM

LINUX Special
301 BEST BASH COMMANDS
LINUX SHELL
HANDBOOK 2021 Edition
LINUX Special
SUPERCHARGE
YOUR LINUX SKILLS

Power at Your Fingertips
- Pipe and redirect output
- Monitor processes
- Create custom scripts

Keep this guide as a permanent reference!

LINUX NEW MEDIA
WWW.LINUX-MAGAZINE.COM

FREE DVD!
LibreOffice Full Version

Become a LibreOffice Expert!
2020 Edition
LibreOffice
Dive deep into the world's greatest free office suite

Write Your Own LO Macros
Save time and automate common tasks

Digital Signatures
Lock down your private documents

Edit and Save MS Office Files

Create Professional:
- Text Documents
- Spreadsheets
- Presentations
- Databases

Replace MS Office an Google Docs!

LibreOffice
Includes full versions fo Windows, macOS, and Li
WWW.LINUX-MAGAZINE

LINUX Special
TUNE YOUR LINUX SYSTEM
2021 EDITION
101 COOL LINUX HACKS
Tricks and shortcuts for Linux geeks

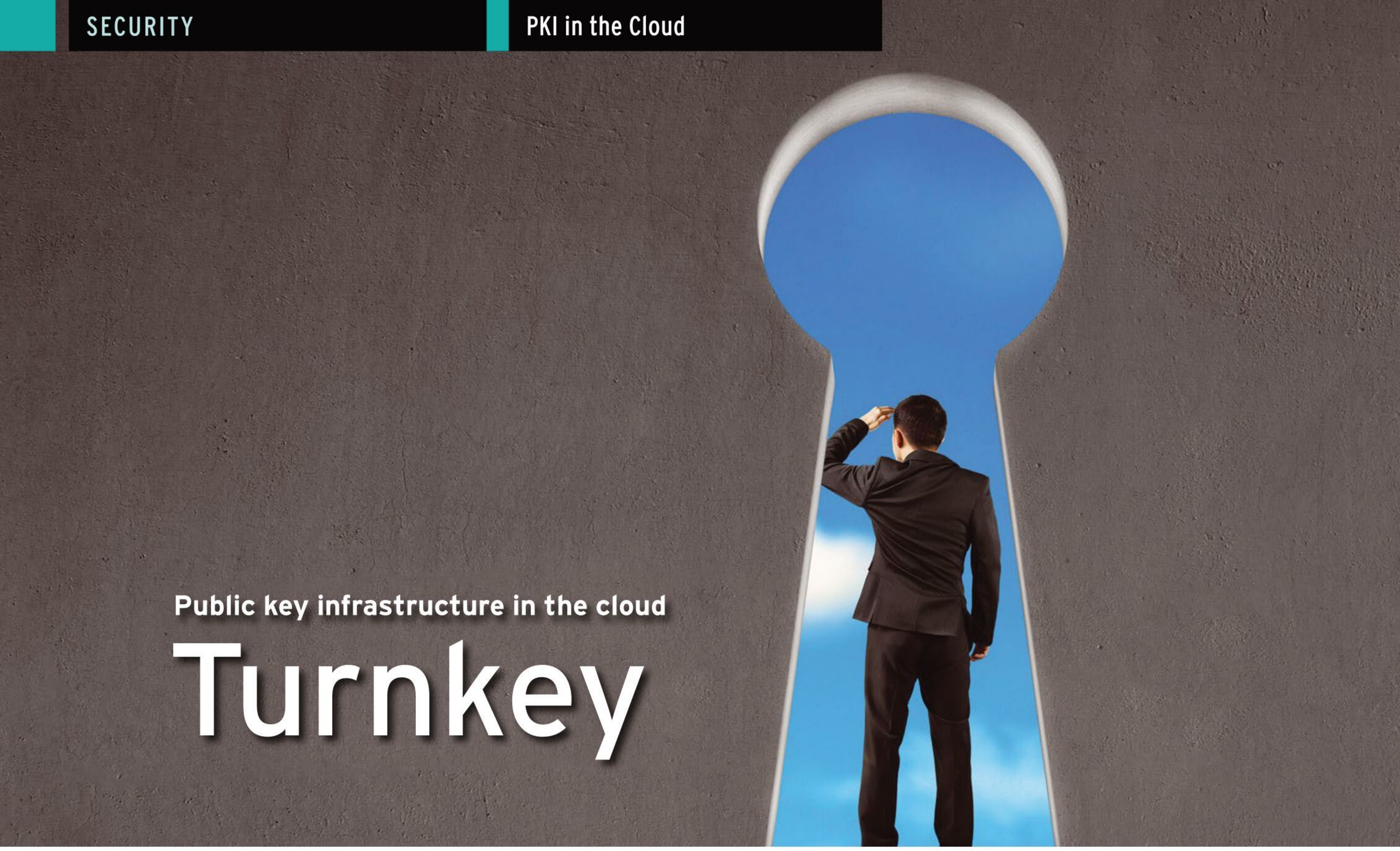
- Recover deleted docs
- Send files without a target IP
- View a handy cheat sheet for your favorite commands

STREAMLINE:
Clean up hidden files

EXTREME CHROOT:
Change to a second distro

Discover the secrets of the experts

WWW.LINUX-MAGAZINE.COM



Public key infrastructure in the cloud

Turnkey

A public key infrastructure in the cloud for secure digital communication maintains the security of an on-premises solution and reduces complexity. By Andreas Philipp

Every industry has a need to authenticate and secure digital communications. The topic of how to communicate securely, whether by a virtual private network (VPN) or over Transport Layer Security (TLS), immediately brings public key infrastructure (PKI) into play. This security infrastructure has spread globally as the most trusted technology to identify people and devices, as well as secure digital communications between participants.

PKI is rightly seen as the entity that provides a trust anchor, which conversely means that a compromised PKI could render an entire digital communication system insecure. Therefore, up to now, organizations have implemented their PKI locally for security reasons.

However, the need for scalability and lower investment or operating costs suggests outsourcing PKI to the cloud. IT security administrators do not have to make any security compromises, and they are spared the need to set up everything from scratch, which they would have to do in an on-premises environment. Whether PKI is better suited as a cloud platform or software as a service (SaaS) essentially depends on the use cases. Adaptability to new regulations and new cloud-native features can also influence the choice.

Classic PKI is Expensive

Setting up the PKI security infrastructure from the hardware security module (HSM) to the database and integrating the detailed processes requires technical expertise to regulate the processes of creating, issuing, and exchanging digital identities in the form of certificates. A new implementation of a further use case pending in a local environment requires extensions to the existing infrastructure and even building new hardware systems.

The security admin also faces some challenges in operations, which is easier for admins with skills that go beyond network administration. Potential hurdles in everyday life, such as managing operating system patches and administering hardware security modules and their backup and restore functions can be overcome more quickly. But what about the increasing global accessibility of corporate services, whether for internal services or in operations, which determine the special requirements for PKI?

One example is the Online Certificate Status Protocol (OCSP) responder information service used as a fundamental component of PKI. For this service to query worldwide whether a certificate has been revoked or blocked, it is necessary to take the transaction load into account. Checking the code signing certificate when a software package is installed while overlooking that the OCSP responder is overloaded and cannot respond is useless.

Local PKI for Complex Customizations

On the other hand, the universal character of PKI also offers advantages in the application because digital identities for a use case, once provided by the established corporate PKI, allow additional use cases to be safeguarded. For example, a company would first establish a PKI that issues digital identities for access to offices and business premises. Smart card or other token technologies, among others, could be used, as well. The next step would be to use these certificates for secure VPN access for employees,

Lead Image © Tsung Lin Wu, 123RF.com

followed by the integration of support staff who need a secure remote maintenance solution.
Server certificates for the entire e-commerce infrastructure, including web servers, load balancers, and server farms, are also conceivable as an extension of PKI. The prerequisite for this approach is a scalable enterprise PKI that can be expanded according to the use cases.

IoT Scenarios Predestined for Cloud PKI

As IoT scenarios continue to grow, so do the requirements for scalability and flexibility, as well as predictable cost models, which are where cloud-based PKI comes into its own and forms the central instance when it comes to applications in the area of machine-to-machine (M2M) communication, device certificates, or TLS encryption in the IoT area.
One example is the healthcare industry where countless IoT use cases illustrate the need for PKI as a Service (PKIaaS) or PKI from the cloud. For example, patient records increasingly need to be available digitally, requiring secure authentication and access in the hospital. Wards also use items such as infusion pumps, in which the software controls medication intake by drip infusion. The only way the software can securely identify any intravenous therapy is by authenticating with a digital certificate. In turn, the machine running the software must ensure that no one tampers with this application. Just to ensure that a patient is administered the correct dose of their medication, multiple digital certificates and PKI-based processes need to interlink successfully, which is the only way to rule out any manipulation of the data, devices, and communication channels.
In a modern hospital, comparable requirements also apply to surgical robots, cooling units, and key cards for security areas such as medical cabinets. In such an IT environment, one advantage of a PKI from the cloud pays off particularly well: Its centralized deployment can be shared among multiple facilities within the hospital operator's setup. Local IT teams do not additionally have to set up and manage local server hardware and applications. Basically, they are faced with the decision of either operating their security architecture as SaaS or as a full PKI platform. The full PKI platform variant is provided within a cloud instance.

PKIaaS or as a Cloud Platform

Cloud is not just cloud these days. As in many other cloud arenas, for PKI, the question arises: PKIaaS or as a cloud platform? PKIaaS offers a fixed set of functions. Billing is per certificate or per device. The approach is an obvious choice if the environment is dominated by standard scenarios that hardly need to be adapted and only a few special cases. Complete individualization is impossible, and deep PKI integration is difficult. The SaaS approach shows its strengths in the provision of standard certificates for servers, TLS, or VPN and pays off immediately because of the inexpensive implementation.
For an extensive PKI implementation or for a very specific use case, relying on a full cloud platform is recommended. This should have deep API support. It is equally important to ensure that billing is based on a single license for an unlimited number of certificates. This means that the system costs less and scales better (e.g., to cover the rapidly increasing IoT use cases). An administrator has full control over a PKI cloud platform and can cover every PKI functionality and component in the cloud.
Digital communication is also influenced by national and international regulations. Adapting to these regulations and integrating corresponding security aspects is one of the strengths of PKI from the cloud, particularly with regard to requirements for the operating environment and the use of approved system components. Some cloud providers cover precisely these aspects. The company uses its PKI from the cloud in the usual way and saves itself costly and time-consuming auditing and certification processes.

New Functions for the Cloud Future

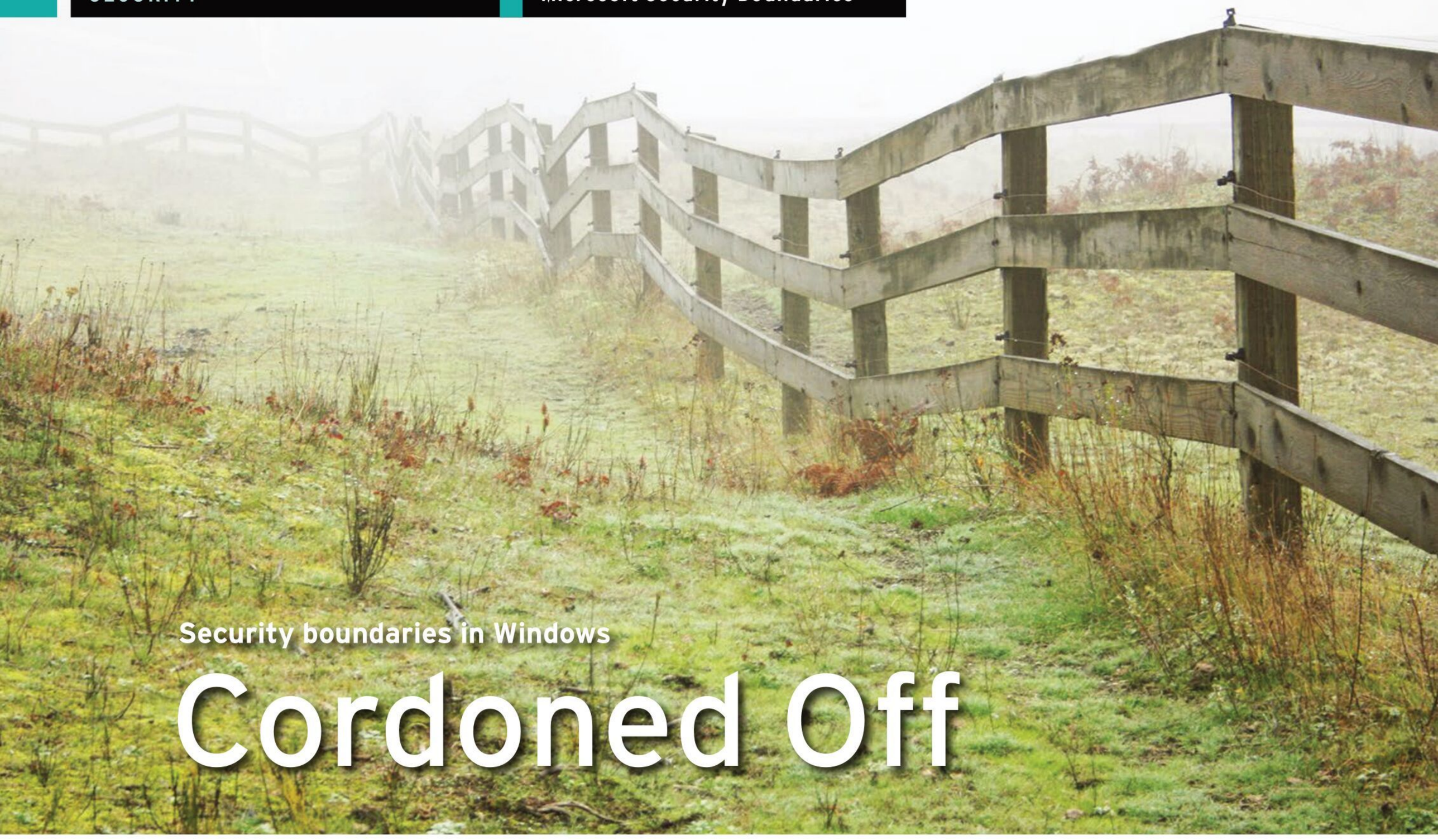
Technological advancements continue, of course. Recent cloud-native features include a dedicated external Validation Authority (VA) that efficiently scales the OCSP. Cost reductions are promised by a feature that supports the AWS Key Management Service. Administrators will be delighted with simplified configuration for clustering, cloud databases, and the integration of a cloud HSM.
The level of integration already taking place in the cloud is illustrated by scaling capacity and throughput, as needed. This capability pays dividends when certificate validation requirements suddenly skyrocket because the PKI user introduces new services or products. Another important advance involves the ability to run a PKI environment with multiple cloud providers. The need may arise from legal requirements. The improvement now is to manage the PKI through one management interface, even though it is used across different clouds.

Conclusions

A PKI is and always has been capable of covering the most demanding use cases for secure digital communication, and this is even more true for the future when considering IoT and M2M environments or new scenarios, such as in connected cars or healthcare. These examples also show that a PKI in cloud operation reduces complexity. Thus far, the opposite has been the case from the critics' point of view. A cloud-based implementation now offers the refreshing approach of beaming the qualities of a proven security architecture into the next decade. ■

Author

Andreas Philipp is Business Development Manager at PrimeKey.



Security boundaries in Windows

Cordoned Off

We look at Microsoft security boundaries and protection goals and their interpretation of the different security areas of Windows operating systems and components. By Matthias Wübbeling

Security architects are familiar with breaking down infrastructure into different levels. The unauthorized transition of a user from one area to another is considered a security breach. Windows offers various protection mechanisms against remote and local attackers. Microsoft first differentiates between different border areas, or security boundaries, and defines separate protection goals for each of these areas. These protection goals not only determine the security of the rolled-out Windows instances but also how the criticality of security vulnerabilities is assessed.

Consider a simple real-world data center example in which a physical area outside the data center is an area for visitors, another area is for customers, and two different areas are for employees, depending on their tasks. If a person enters the visitor area, for example, a transition to the first security zone takes place. As long as this access takes place during normal visiting hours, this action is not problematic. Outside visiting hours (i.e., when the doors are locked), this is obviously a security incident.

If the person is a customer or employee, they can enter the customer area after successful authentication by the gatekeeper. Once in the customer area, access to other areas usually relies on technical security systems.

Customers and employees use a chip card with a PIN for authentication that allows them to enter individual rooms. Support staff can also enter the general staff area in addition to the customer areas. Network administrators are also allowed to enter the rooms with the switches and routers in the data center.

What works in the real world can also be applied to securing operating systems. Microsoft defines nine different security boundaries for its own operating systems, active services, and devices in use, although they are not all hierarchically structured like the security areas in the example above. An associated document in the Microsoft Security Response Center (MSRC) [1] is updated continuously.

Network

The transition from the network to a computer is the outermost boundary of that computer. Non-authorized network users cannot access or manipulate the code and data of users on the computer. A malfunction in the corresponding protection mechanisms, such that unauthorized access is possible, is considered a security breach. Of course, retrieving web pages from the Internet Information Services (IIS) or shared files from file and printer sharing is not a vulnerability as long as this unauthorized access is intentional. The component that separates the network from the computer is the firewall.

Kernel and Processes

Computers also have security zones. Programs and services that do not run under an administrator account cannot access data or code in the operating system kernel area. Even in this case, of course, explicitly intended paths, such as using the operating system functions to request memory and file reads and writes or to open network connections are not a security vulnerability. Microsoft considers any access by programs that were started with

Photo by Jan Canty on Unsplash

administrator rights to be basically unproblematic, even if they execute malware. The administrator kernel has no separate limit.

Processes can run in user mode or kernel mode (i.e., with administrator privileges). The processes started in user mode basically do not get access to the code or data of other running processes, even if the same user started them. Even this case has intended exceptions (e.g., shared memory allocated by the operating system) that are not considered to be security problems. Processes in kernel mode are not affected by this restriction.

AppContainer Sandbox

In Windows 8, Microsoft introduced a sandbox mechanism known as the AppContainer for applications from the Microsoft Store. Different types of isolation can be defined for each sandbox, such as device, filesystem, or network isolation. A sandbox implementation that has an error allowing access to the local network despite network isolation is a security vulnerability. However, if this network access is not restricted for a sandbox – and you can distinguish between intranet, Internet, and server functionality – the application can access the network as desired.

User Separation

A user cannot access or manipulate the code or data of other users. Both files on the filesystem and processes at runtime are included in this security limit. For access by administrators, this limit can also be formally implemented in the filesystem. However, this restriction is not effective. Of course, administrators can, in the absence of an administrator kernel limit, change the access rights of other users' files at any time.

If a Windows user session is running for an authorized user, this account and the processes started in this session cannot access or manipulate other user sessions, particularly to remote desktop sessions, so that, for example, mounted network drives or forwarded printers are not accessible in these sessions.

The browser environment also has restrictions. A website not authorized by the user is bound by the same-origin policy and is not allowed to access or manipulate the code or data of the browser sandbox. However, from Microsoft's point of view, this security limit is only defined for Microsoft Edge and does not include the outdated Internet Explorer or web browsers by other manufacturers.

Virtual Machines

A Hyper-V server guest system, as well as the lightweight Hyper-V containers introduced in Windows Server 2016 (which can be managed with Docker), cannot access the code, data, or settings of another Hyper-V virtual guest without authorization. The Virtual Secure Mode (VSM) introduced in Windows 10 is also based on Hyper-V technology. A microkernel is started and isolates the Local Security Authority Subsystem Service (LSASS) in particular but also hardware such as the Trusted Platform Module (TPM) for apps started in the VSM. This security boundary specifies that code or data within the enclave cannot be accessed from outside the isolation (a so-called enclave).

Components Without Limits

For some Windows components, Microsoft explicitly clarifies that they are not to be considered a security boundary, even if the function suggests other properties. The list only includes those components that are often misinterpreted as a boundary, so it is not complete; it includes, for example, the administrator kernel boundary.

As mentioned before, the administrator or a process started with administrator rights has no restrictions in accessing data structures or kernel code. Microsoft also lists Windows Server containers, which, unlike the "secure" Hyper-V containers, do not isolate with sufficient reliability.

Conclusions

In this article, I provided a brief overview of Microsoft's interpretation of the different security areas of Windows operating systems and components. Understanding these areas will help you plan security mechanisms and identify security issues. In particular, Microsoft also highlights the criticality of security vulnerabilities and paying bug bounties for reports on security vulnerabilities that affect these boundaries. ■

Info

[1] Microsoft Security Servicing Criteria for Windows: [https://www.microsoft.com/en-us/msrc/windows-security-servicing-criteria]

Updating Docker containers with Watchtower

Elevated View

Automatically update software in the Docker universe with Watchtower.

By Matthias Wübbeling

Deploying microservices with Docker is relatively easy. With very little overhead, you can complete complex software installations and offer services in your company. However, controlling an infrastructure that has grown very quickly can often be far more time-consuming than the actual deployment. The progressive use of microservices, and with it the need to provision individually customized ecosystems within containers, leads to a confusing update jungle. If something goes wrong in the process, an unreachable service is in many ways less critical than a service that is vulnerable to attackers. Possible downtime is then supplemented by close monitoring and fallback strategies. Regardless of how you manage your Docker containers – whether directly with Docker itself, with Docker Compose, or with one of the many other tools – you will want a working backup solution, and you will want to ensure that software updates are automated to the greatest extent possible. In this article, I look at Watchtower as an option for automatically updating your Docker containers.

Launching Watchtower

The initial setup for Watchtower is roughly equivalent in scope and complexity to changing a light bulb. If all tools are ready and the Docker daemon is running, you can launch Watchtower with:

```
docker run ➥
  -d --name watchtower ➥
  -v /var/run/docker.sock:➥
     /var/run/docker.sock ➥
  containrrr/watchtower
```

This command sends the launched container directly into the background with `-d` (detach). You can freely select the name by specifying `--name` and use the `-v` option to include areas of the filesystem as a volume in the container's process group. The command includes the Docker daemon's communication socket because that is how Watchtower communicates and transmits its commands. The last argument of the command line passes in the Watchtower path in the Docker Hub registry [1].

Once started, Watchtower takes direct control and monitors the availability of new image versions for all running Docker containers. If a new version is available, Watchtower downloads it and restarts the affected containers. All the parameters of the containers (i.e., the parameters you specified with `docker run`) are taken into account and passed in accordingly on restart. In this way, mounted volumes and shared ports persist despite updates.

Please note that Watchtower also takes into account tags that you attach to the images. The best known tag is probably *latest*, which lets the creator of an image publish the fact that the image is the latest available version. However, when updating from *latest*, Watchtower also jumps to new major release versions. If you have a piece of software that is version 3.8.12, you might see version 4.0.1 start up as an updated image, which might not always be what you want.

Photo by Alan on Unsplash

If the software you use supports extensions through a plugin system, for example, these extensions often are not immediately compatible with the next major release. Therefore, a good practice among creators of Docker images is to assign tags that are as complete as possible. Version 3.8.12 is then assigned the tags *latest*, *3*, *3.8*, and *3.8.12* at the time of release. Therefore, as an administrator, you can define your desired version and the update strategy for Watchtower. If you start an image with *:3*, you will get version 3.9, 3.10, and so forth in the future. An image that starts *:3.8* will give you 3.8.13 and so on, whereas *:3.8.12* will mean you do not receive any updates.

Selectively Distributing Updates

Probably the most important feature that developers can include with a Docker image is the update and migration capability of the application it contains. Importing new container images, then, also results in automatically adjusted startup times in all necessary configuration and database settings of the provided software. Therefore, it depends on the container's entry point, configured as a `RUN` command within the Dockerfile, whether a container image can be updated easily. Most official images have this capability.

You might want to have Watchtower automatically update only selected containers. Docker lets you pass in additional arguments to a container's entry point with the `docker run` command. Watchtower uses this ability to pass in container names and thus define which containers should be updated. To include only the containers for Nextcloud, Elasticsearch, and Mailman, extend the command as follows:

```
docker run ➥
  -d --name watchtower ➥
  -v /var/run/docker.sock:➥
     /var/run/docker.sock ➥
  containrrr/watchtower Nextcloud ➥
    Elasticsearch Mailman
```

Keep in mind that this selection also removes Watchtower itself from the list of automatically updated containers. If you want the tool to update itself, you need to specify it.

Another way to select containers is with container labels, which you assign to a container at startup time with the `-l` option. One advantage of this method is that you can also select containers that you create after Watchtower has been launched; then you tell Watchtower only to update containers with the corresponding label by the environment variable

```
-e WATCHTOWER_LABEL_ENABLE=1
```

during the `docker run` command. To add the appropriate label to all other containers that you start before or after Watchtower, use:

```
-l com.centurylinklabs.watchtower.➥
   enable="true"
```

Again, if you want Watchtower to update itself, you need to start the container with the appropriate label.

Creating Schedules and Trial Runs

If you want to manage the times at which Watchtower checks the containers, you can use the `WATCHTOWER_SCHEDULE` environment variable by filling the variable with a string in the extended Cron syntax of the Go programming language, on which Watchtower is essentially based [2], corresponding to the known Cron syntax of your Unix system, with a field for seconds at the front. If you want Watchtower to start working every six hours, add the argument

```
-e WATCHTOWER_SCHEDULE="0 0 */6 * * *"
```

to the `docker run` command.

As discussed earlier, if you use containers for which automatic updates are difficult or unreliable, you can also configure Watchtower simply to check the container images for updates and notify you when they are available. Watchtower supports various approaches, such as email or instant messaging with Slack, Microsoft Teams, and other services.

To enable email notification, you first need to protect the corresponding container against automatic updates by assigning a label specifying

```
-l com.centurylinklabs.watchtower.➥
   monitor-only="true"
```

when creating the container. Extend the Watchtower environment as shown in Listing 1 to include the environment variables for configuring an email gateway. In this way, you can permanently keep track, and you can manually trigger any updates that become available.

Listing 1: Email Gateway

```
-e WATCHTOWER_NOTIFICATIONS=email
-e WATCHTOWER_NOTIFICATION_EMAIL_FROM=wt@<your-co>.com
-e WATCHTOWER_NOTIFICATION_EMAIL_TO=notify@<your-co>.com
-e WATCHTOWER_NOTIFICATION_EMAIL_SERVER=smtp.<your-co>.com
-e WATCHTOWER_NOTIFICATION_EMAIL_SERVER_PORT=587
-e WATCHTOWER_NOTIFICATION_EMAIL_SERVER_USER=from
-e WATCHTOWER_NOTIFICATION_EMAIL_SERVER_PASSWORD=<password>
-e WATCHTOWER_NOTIFICATION_EMAIL_DELAY=2
```

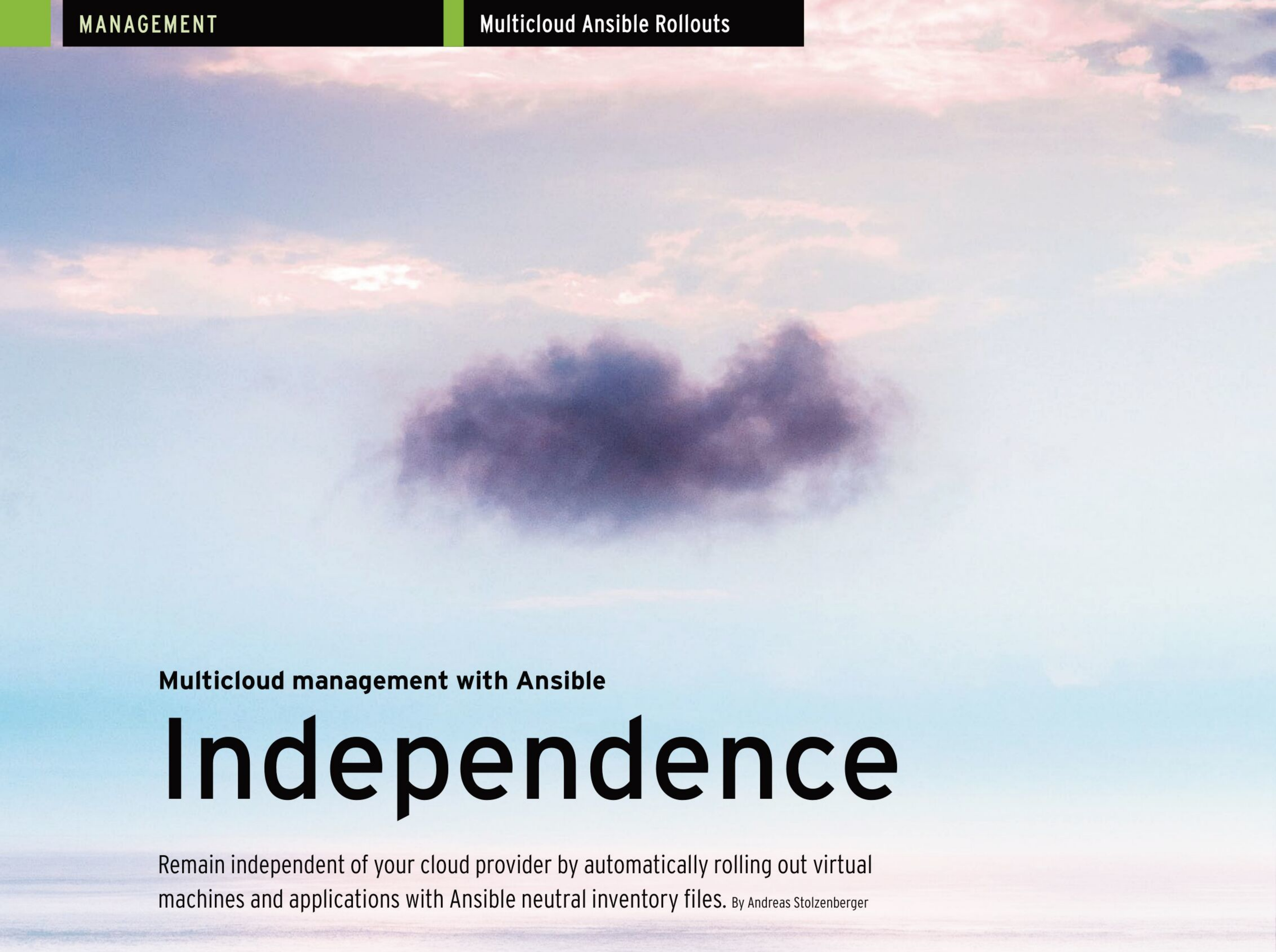
Conclusions

To keep your running Docker containers up to date, add Watchtower as an additional container to your environment. The initial configuration is child's play, with Watchtower's comprehensive configuration options fulfilling almost every need. In this article, I looked into the first steps for using and automatically updating software in the Docker universe with Watchtower. ■

Info

[1] Watchtower on GitHub: [https://hub.docker.com/r/containrrr/watchtower]

[2] CRON format: [https://pkg.go.dev/github.com/robfig/cron@v1.2.0#hdr-CRON_Expression_Format]



Multicloud management with Ansible

Independence

Remain independent of your cloud provider by automatically rolling out virtual machines and applications with Ansible neutral inventory files. By Andreas Stolzenberger

Many cloud providers vie for the user's favor. Besides the top dogs, Amazon Web Services (AWS), Azure, and Google Cloud Platform (GCP), smaller regional or specialized providers are increasingly offering cloud computing resources, which is good for the user because competition is known to stimulate business and prompt price drops. Precisely because companies have a choice when it comes to cloud services, they will not want to bind themselves to a single vendor.

However, very few professional cloud users click through the providers' pretty web GUIs to roll out dynamic resources. The whole thing has to work quickly and automatically. All cloud providers, therefore, offer powerful tools for the command line, which can also be used to generate scripts that automate the rollout.

Of course, this is exactly what could shoot down your desired independence from the provider. After all, anyone who has invested a large amount of time developing fancy scripts for AWS cannot easily switch to GCP or Azure without first switching their automation to a different toolset. Two things can help in such a case: Ansible as an independent automation tool and a modular abstraction strategy.

Independence

Nobody rolls out empty virtual machines (VMs) for their own sake on platforms such as AWS or GCP: The decision is determined by the application. Providers are offering more and more convenient and preconfigured services for this purpose. Want MariaDB? No problem: Here's a pre-built image to roll out directly to AWS. The user saves themselves the trouble of separate operating system (OS) and database installations. This scenario sounds tempting, especially for the cloud provider, because it ties the user firmly to the platform and precisely to this one template, which is not available for another platform in this form.

To remain independent, administrators need to separate the VM rollout from the distribution of the applications. Minimal OS templates that are more or less identical on all cloud platforms can help. Alternatively, you can use a tool (e.g., `lorax-composer` [1]) to build your own OS templates, which can then be uploaded to the respective cloud environment. With a modular rollout like this, admins write their installation and configuration playbooks independently of the cloud they use. The rollout process simply needs to communicate a set of parameters to the application automation component in a vendor-independent format. Whatever that component may be, the parameters include:

- The external IP address through which the VM is accessible to Ansible and later the application.
- The internal IP address that the application uses to communicate with other VMs in the same rollout.
- The purpose or type of machine.

Photo by Pawel Nolbert on Unsplash

For Ansible to address the target systems for the application rollout, it needs a vendor-independent inventory. The Ansible modules for GCP, AWS, and Azure, of course, provide scripts for dynamic inventories, but there is a catch: For example, if you want to query the inventory of a GCP project dynamically and thus discover the internal IP address of a VM, you have to use the `gce_private_ip` variable. On AWS this would be `ec2_private_ip`.

Even in Ansible, the cloud modules come from the respective providers, and the designation of Ansible "facts" and variables is based on provider standards across the board. It is up to you as the administrator to ensure a neutral inventory.

Dynamic Static Inventory

A simple trick to stay as neutral as possible in cloud usage works like this: The vendor-specific playbook that rolls out the required VMs on the target platform also creates a static inventory file and there enters the variables required by the application rollout in an independent form. The application rollout playbook, in turn, takes all the necessary variables from the inventory file and can then work with any target platform, whether AWS, GCP, or even on-premises environments such as vSphere.

Here, I look at a vendor-neutral cloud rollout with two playbooks: one for AWS and the other for GCP. At the end, you will have a set of VMs and a matching neutral inventory. The examples shown here assume that the clouds you are using have been prepared, and that you have created the required users and SSH keys, as well as the appropriate subnets, security groups, and firewall rules.

By the way, if you work with variable data in an Ansible playbook, you should always keep the data separate from the playbook logic (i.e., never hard code the data directly into the playbook). Moreover, you always need to think first about the kind of data you need to capture and the kind of data you need to register before moving on to the logic.

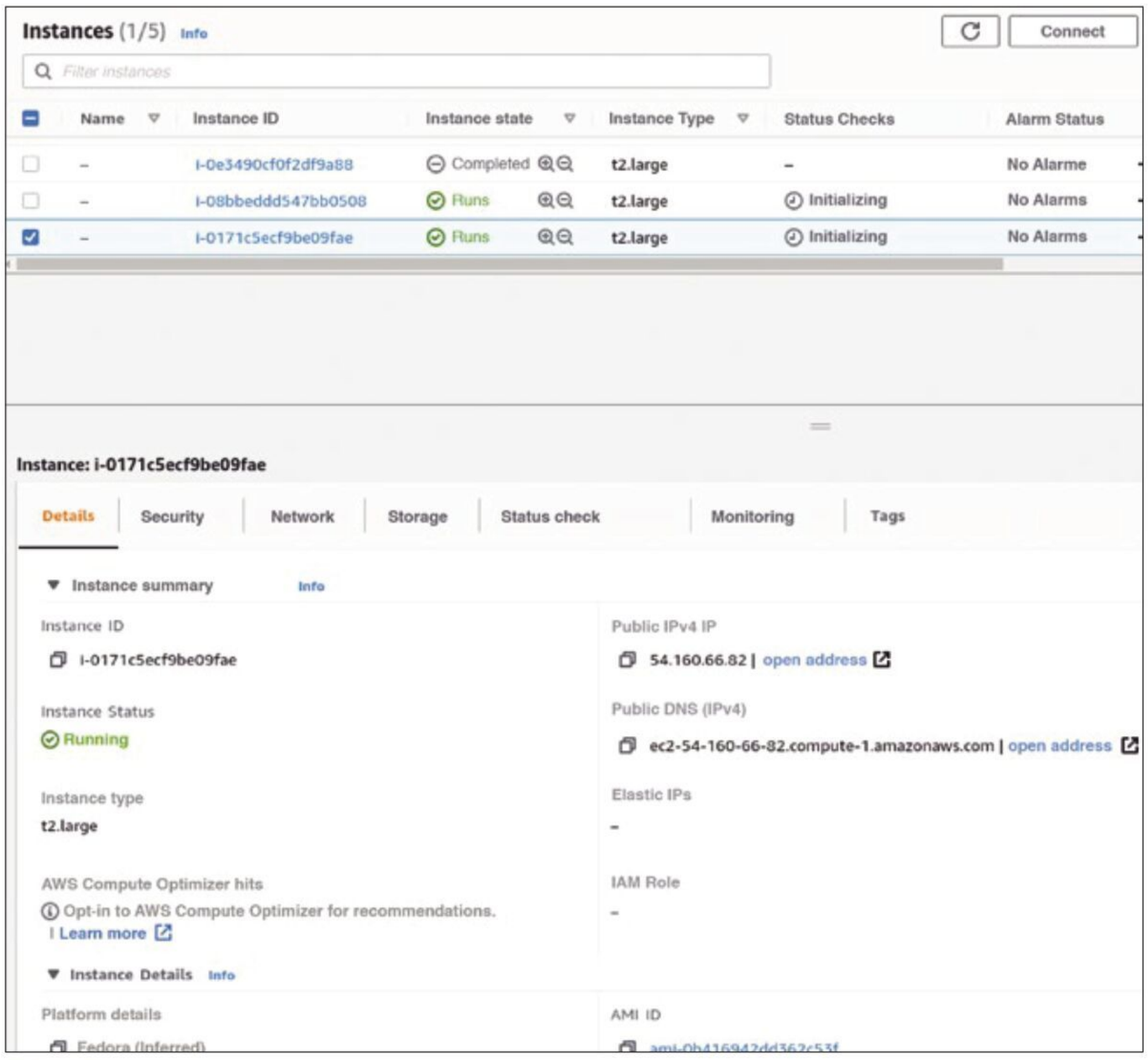


Figure 1: GUIs and functions of cloud providers differ greatly. AWS, for example, registers a domain name system (DNS) name for new VMs, but a customized, mnemonic name cannot be assigned.

To begin, I'll start with the variable declaration.

Providing Basic Data: AWS

To talk to a cloud, you need to authenticate and specify the region or project in which you are working. The variable declaration `ec2_vars.yml` for AWS therefore starts:

```
---
ec2_access_key: (key)
ec2_secret_key: (secret)
ec2_region: us-east-1
ec2_key_name: (keyname)
ec2_security_group_id: sg-<ID>
ec2_vpc_subnet_id: subnet-<ID>
ec2_hostfile: ec2_hosts
```

The parameters are largely self-explanatory, with the exception of `ec2_hostfile`, which refers to the inventory file (yet to be created).

Defining VM Specifications

The EC2 module for Ansible can roll out multiple VMs at once with a single command. However, they are then all the same size and are based on the same template (Figure 1). However, you want to remain flexible and be able to assign individual sizes and templates to the VMs. You also want to provide them with additional information for the application rollout that follows. In the example, I will be assigning the machines a `purpose` (e.g., `database` or `webserver`). Listing 1 shows the continuation of `ec2_vars.yml`. Ansible refers to the hierarchical variable structure of `vms` as a dictionary, which can later be used to build

Listing 1: AWS VM Specs `ec2_vars.yml`

```
vms:
   one:
     type: t2.large
     image: ami-0b416942dd362c53f
     disksize: 40
     purpose: database
   two:
     type: t2.small
     image: ami-0b416942dd362c53f
     disksize: 20
     purpose: webserver
```

a loop. The example creates two Fedora minimal VMs with different disk and VM sizes. Of course, the `vms` dictionary can list many more machines, and you can add more variables that will end up in the inventory later. In a loop over `vms`, the items end up in the following Ansible variables,

```
vms:
   one: -> {{ item.key }}
      type: t2.large -> ➥
            {{ item.value.type }}
...
```

that tasks can access within the loop playbook.

Listing 2: Playbook `rollout_ec2.yml`

```
---
- hosts: localhost
   gather_facts: False
   vars_files:
   ec2_vars.yml
   tasks:
   - name: Create Hosts file
     file:
        path: "{{ ec2_hostfile }}"
        state: touch
   - name: hosts file Header
     lineinfile:
        path: "{{ ec2_hostfile }}"
        line: '[ec2]'
   - name: Loop VM Creation
     include_tasks: ec2_loop.yml with_dict: "{{ vms }}"
```

Listing 3: Playbook `ec2_loop.yml`

```
- name: Launch EC2 Instances
   ec2:
     access_key: "{{ ec2_access_key }}"
     secret_key: "{{ ec2_secret_key }}"
     region: "{{ ec2_region }}"
     key_name: "{{ ec2_key_name }}"
     instance_type: "{{ item.value.type }}"
     image: "{{ item.value.image }}"
     vpc_subnet_id: "{{ ec2_vpc_subnet_id }}"
     group_id: "{{ ec2_security_group_id }}"
     count: 1
     assign_public_ip: yes
     wait: true
     volumes:
       - device_name: /dev/sda1
         volume_size: "{{ item.value.disk size }}"
         delete_on_termination: true
   register: ec2_return
...
```

Completing the Inventory

The logic of the `rollout_ec2. yml` playbook (Listing 2) first creates the machines and adds all the necessary information to the static inventory. The playbook runs on the local host and first creates the static inventory file with its header; then, it loops over the dictionary for each VM to be created, calling the `ec2_loop.yml` playbook (Listing 3) each time.

The first task uses Amazon's EC2 module and creates the VM. Because you start the module once per loop, you set the VM count statically to 1. The `wait: true` switch is, unfortunately, necessary. From time to time, the automation runs faster than AWS itself, then the task without a wait completes before Amazon has even assigned a public IP address to the machine. Of course, the automation needs this missing IP address. At the end, `register` saves the return values of the EC2 module in the variable `ec2_return`:

```
- name: Debug ec2_return
   debug:
     msg: "{{ ec2_return }}"
     verbosity: 2
```

The variable `ec2_return` is a JSON construct. It outputs the debug command fully on the command line, which helps Ansible developers determine the information they need in JSON during an initial trial run and then be able to use the correct variable structure for the rest of the process. You can remove the task in the finished playbook, of course. Here, it is simply included with the `verbosity: 2` flag. In this case, this step will only be executed if you start the playbook with `-vv` (i.e., debug level 2). Now complete the `ec2_loop. yml` playbook (Listing 3) with the following lines:

```
- name: Loop Output to file
   lineinfile:
     path: "{{ ec2_hostfile }}"
     line: ➥
       '{{ ec2_return.instances[0].➥
          public_ip }} ➥
       vmname={{ item.key }} ➥
       privateip=➥
         {{ ec2_return.instances[0].➥
            private_ip }} ➥
       purpose={{ item.value.purpose }}'
```

After debugging the return array (a variable that stores multiple values with an index), you now know where information like public and private IP addresses are in the array; Ansible appends them to the static inventory with `lineinfile`. The `ec2_return_instance[]` variable is an array because the `ec2` module can create multiple VMs in a single pass. For example, after a successful

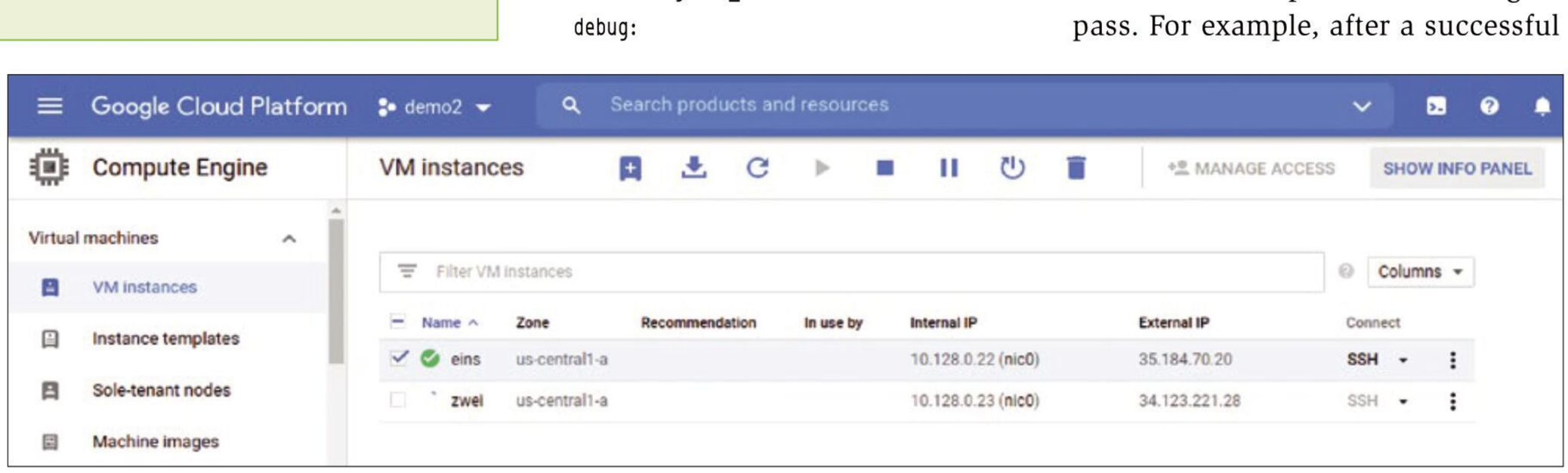


Figure 2: Google not only sorts VMs by region but also manages machines in projects. The admin gives each machine a name when it is created, but GCP does not automatically register a DNS resolution.

rollout of the two test VMs on AWS, the inventory created in this way will have the following information:

```
[ec2]
100.25.193.49 ➥
  vmname=one privateip=172.31.63.60 ➥
  purpose=database
54.173.55.227 ➥
  vmname=two privateip=172.31.63.241 ➥
  purpose=webserver
```

What ends up in the inventory is up to the user. You can, of course, add more information from the loop or variable declaration to the inventory that the later application rollout will be able to use. In the case of AWS, for example, it makes sense to write the ID of the machine from the array `ec2_return_instance-` to the inventory, too. A rollout playbook doesn't need this parameter, but a playbook that deletes the VMs in the AWS cloud after using the service needs to know these IDs.

And with Google

Google Cloud Platform organizes resources differently from AWS. It sorts VMs into projects and regulates networks and firewalls somewhat differently (Figure 2). The associated Ansible modules create the VMs in several steps. However, the end result is an identical inventory to that after the AWS rollout. The `gcp_vars.yml` variable declaration is shown in Listing 4. Parameters such as networks and firewall rules are optional, and GCP sets the default values of the project. The `vms` dictionary looks almost identical to the AWS declaration. Only the names and paths for `image-type` and `template-source` use different values. The GCP rollout script, `roll-out_gcp.yml` (Listing 5) contains basically the same statements as for the AWS rollout, except the loop playbook for GCP differs

Listing 4: GCP VM Spec `gcp_vars.yml`

```
---
gcp_credentials_file: credential-file.json
gcp_project_id: myproject-123456
gcp_zone: us-central1-a
gcp_hostfile: gcp_hosts
vms:
   one:
     type: e2-standard-4
     image: projects/centos-cloud/global/images/family/centos-8
     disksize: 40
     purpose: database
   two:
     type: e2-standard-2
     image: projects/centos-cloud/global/images/family/centos-8
     disksize: 20
     purpose: webserver
```

Shop the Shop ⇒ shop.linuxnewmedia.com

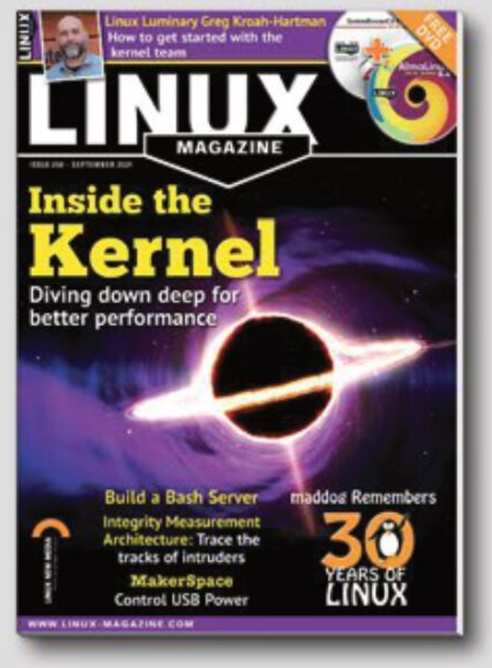
Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

Want to subscribe?

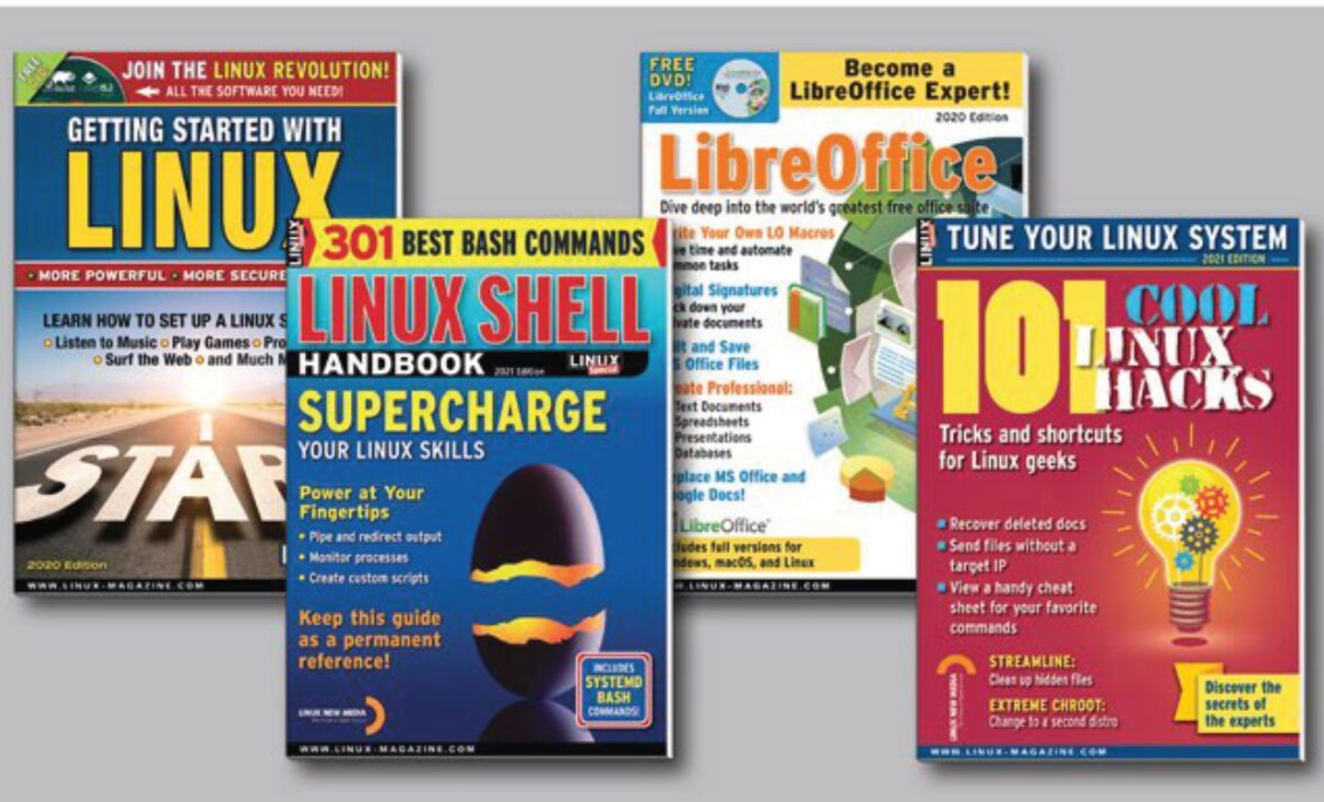
Searching for that back issue you really wish you'd picked up at the newsstand?

shop.linuxnewmedia.com

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



Listing 5: Playbook `rollout_gcp.yml`

```
---
- hosts: localhost
   gather_facts: False
   vars_files:
   gcp_vars.yml
   tasks:
   - name: Create Hosts file
     file:
     path: "{{ gcp_hostfile }}"
   state: touch
   - name: hosts file Header
     lineinfile:
        path: "{{ gcp_hostfile }}"
        line: '[gcp]'
   - name: Create GCE VMs
     include_tasks: gcp_loop.yml with_dict: "{{ vms }}"
```

Listing 6: Loop Playbook for GCP

```
- name: create a disk
   gcp_compute_disk:
     name: "{{ item.key }}-disk"
     size_gb: "{{ item.value.disksize }}"
     source_image: "{{ item.value.image }}"
     zone: "{{ gcp_zone }}"
     project: "{{ gcp_project_id }}"
     auth_kind: serviceaccount
     service_account_file: "{{ gcp_credentials_file }}"
     state: present
   register: disk
- name: create a instance
   gcp_compute_instance:
     name: "{{ item.key }}"
     machine_type: "{{ item.value.type }}"
     disks:
     - auto_delete: 'true'
        boot: 'true'
        source: "{{ disk }}"
     network_interfaces:
     - access_configs:
        - name: External NAT
           type: ONE_TO_ONE_NAT
     zone: "{{ gcp_zone }}"
     project: "{{ gcp_project_id }}"
     auth_kind: serviceaccount
     service_account_file: "{{ gcp_credentials_file }}"
     state: present
   register: gcp_return
```

Listing 7: Playbook for Application Rollout

```
---
- hosts: all
   remote_user: ssh-user
   become: yes
   tasks:
   - name: DDNS
     include_role: name: common
   - name: Common Config
     include_role: name: common
   - name: Config Database
     include_role: name: database
     when: purpose=="database"
   - name: Config Webserver
     include_role: name: nginx
     when: purpose=="webserver"
```

significantly from the AWS rollout, as Listing 6 shows.

If you want to determine the disk size of the VM yourself, you first need to create and adapt a disk from the template in a separate `gcp_compute_disk` task. Without this separate task, the following module would create a disk of the size stated in the OS template. In the second task, GCP then builds the VM with the previously created disk. In GCP, the administrator can specify the name of the machine. Again, this does not matter for the upcoming application rollout, but the VM name in a later playbook identifies the VMs to be deleted:

```
- name: debug output
   debug:
     msg: "{{ gcp_return }}"
     verbosity: 2
```

As previously with AWS, you need to analyze the JSON structure of the return code in a first test run to extract the correct parameters in the next step and add them to the playbook (Listing 6):

```
- name: Add Instance Data to Host File
  lineinfile:
    path: "{{ gcp_hostfile }}"
    line: '{{ ➥
      gcp_return.networkInterfaces[0].➥
      accessConfigs[0].natIP }} ➥
      vmname={{ item.key }} ➥
      privateip={{ gcp_return.➥
      networkInterfaces[0].networkIP }} ➥
      purpose={{ item.value.purpose }}'
```

The GCP return variable also contains arrays. In GCP, VMs with multiple network adapters can be built in an automated process; the neutral inventory file `gcp_hosts`,

```
[gcp]
34.71.6.79 ➥
  vmname=one privateip=10.128.0.20 ➥
  purpose=database
34.66.26.41 ➥
  vmname=two privateip=10.128.0.21 ➥
  purpose=webserver
```

is also available in GCP at the end.

Preparing the Application Rollout

Before the application rollout on the newly created machines, it's time for some housekeeping. In addition to the usual steps, such as configuring repositories; installing runtimes and dependencies; and creating users, groups, and rights, you need to register the systems with a dynamic DNS (DDNS) server that is independent of the cloud provider. In this way, you can always reach the application with its own DNS name. A suitable playbook refers to previously declared roles for the application rollout (Listing 7).

The `ssh-user` matches the key pair of the target servers. If your company has different users for GCP and AWS, you can also declare `ssh-user` as a variable in the inventory. The DDNS and `Common Config` tasks run on all rolled-out machines. Then, you can use the `when` filter to assign the roles only to the machines with the appropriate `purpose`. If you have not yet written your own roles for application rollouts in Ansible, the community repository, Galaxy [2], provides many prebuilt roles for the usual suspects, such as PostgreSQL, MariaDB, Apache, and Nginx.

Conclusions

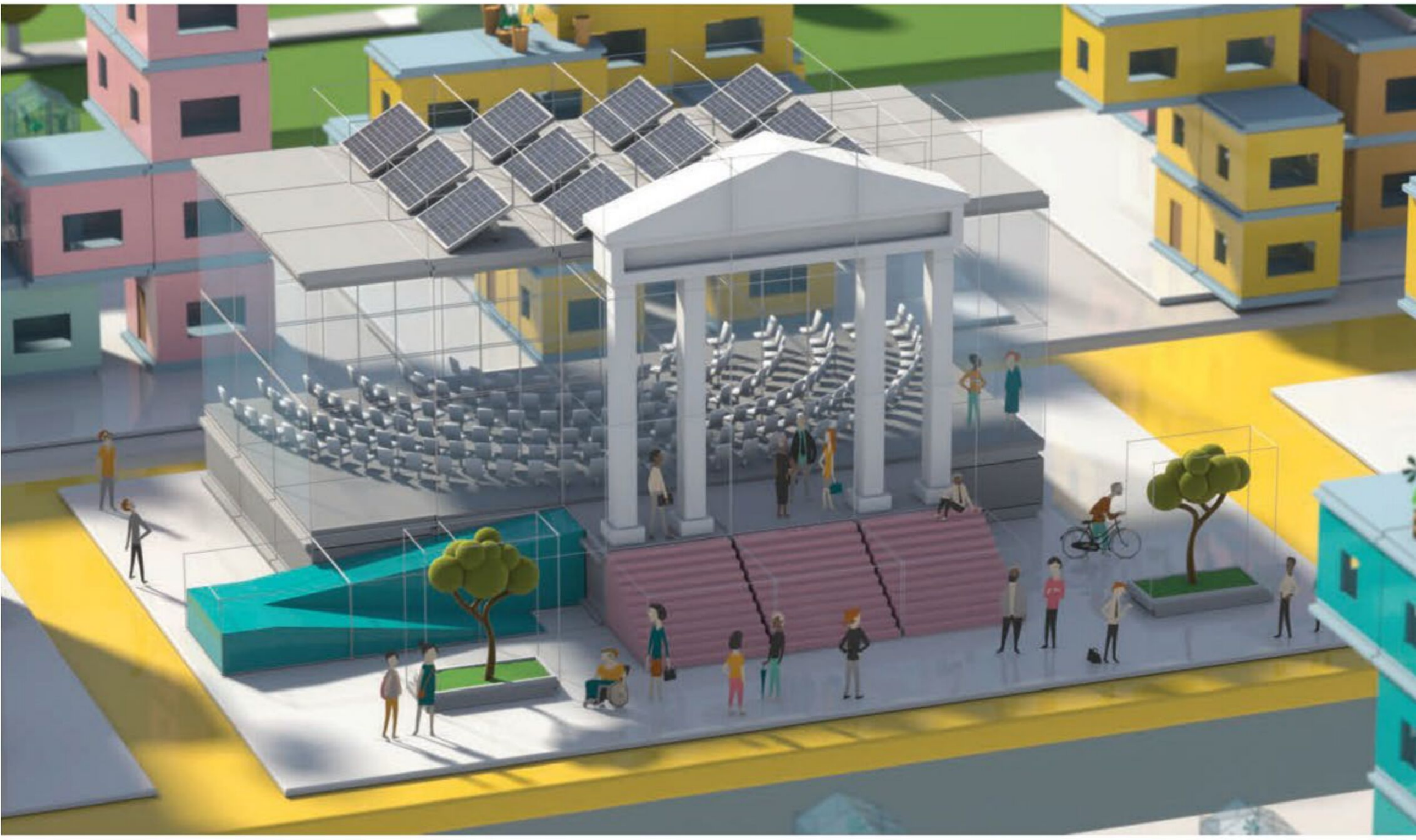
With Ansible as an automation tool and an abstraction layer – in this case, a neutral inventory file – administrators can set up their cloud rollouts independent of the vendor. Only the automation part that works directly with the VMs in each cloud remains vendor dependent. However, managing the applications running on the cloud resources has nothing to do with the underlying virtualization layer. The strategy presented here can therefore also be extended to include local virtualizations (e.g., with vSphere or Red Hat Virtualization). ■

Info

[1] lorax-composer: [https://weldr.io/lorax/lorax-composer/lorax-composer.html]

[2] Ansible Galaxy: [https://galaxy.ansible.com]

Public Money
Public Code



Modernising Public Infrastructure with Free Software

fsfe Free Software Foundation Europe

Learn More: https://publiccode.eu/

U-Move backs up, restores, and migrates Active Directory environments

Move It!

U-Tools Software's U-Move promises significantly simplified backups and restores of Microsoft's directory service in the event of a disaster, during migrations, and when setting up test environments. By Christian Knermann

Backing up and restoring Windows servers is considered a difficult undertaking when it comes to forests and domains in Microsoft's Active Directory (AD) service. Software vendor U-Tools promises to make backing up and restoring AD environments easier with its U-Move tool. Backups are said to take significantly less space than with the native tools, and U-Move promises simpler migrations when switching to a newer version of the Windows Server operating system, cloning or copying an AD environment, or testing an isolated lab environment, as well as for production operations in the scope of a cloud migration.

Before I proceed to investigate whether the manufacturer actually keeps this promise, just one more note: The primary purpose of Windows Server Backup and of U-Move is restoring domain controllers (DCs) or a complete AD environment. Both tools can, in principle, also restore individual elements from an AD but only with disproportionately high overhead.

Licenses from Small to Large

U-Move (see the "U-Move for AD" box) is available as a Small Business license for a forest with a single domain and up to 50 user objects. The Domain type license also backs up a forest with one domain but with no limit on the number of users; finally, the Enterprise license covers any number of domains with no limit on the number of users.

All licenses entitle the user to use the product permanently and also include basic technical support and updates to new versions for one year. The Enterprise variant includes extended technical support in the first year, even outside the manufacturer's business hours and on weekends. On request, the vendor offers support extensions for all license types for a period of one to five years, provided the order for the extension is received up to 60 days before the end of the first year.

Although Microsoft has long since discontinued support for the forefathers of Windows Server, U-Move retroactively supports all editions down to and including Windows Server 2003, including its smaller siblings Small Business Server and Essentials.

Easy Local or Remote Installation

For this article, I used the software in domains with DCs running Windows Server 2016 and 2019. The setup routine presents the license terms and then asks for the target path and

U-Move for AD

Product

Software for backup, migration, and recovery of Microsoft Active Directory.

Manufacturer

U-Tools Software LLC [1]

Price

One-time purchase including support for one year:

- Small Business around $200
- Domain $489
- Enterprise $2,200

Support extension for one to five years costs 20 percent of the purchase price per year.

System requirements

Backs up Microsoft Windows Server from version 2003 and Small Business Server in versions 2003 to 2011 [2].

Photo by Erda Estremera on Unsplash

license key. I had received a 30-day trial license from the vendor. That was the complete install process – done. The tool took up less than 30MB of hard disk space.

When first launched, U-Move says hello and gives a short introduction to the various possible uses, each with direct links to the appropriate chapters of the very extensive and comprehensibly formulated online help. I was also able to access the Help top left in the window. The *Connect* link next to it starts U-Move on a remote computer if desired, without the software already having to be installed there.

If U-Move does not yet exist on the target, the U-Move remote agent is installed. For both installation and remote control, the Windows Defender firewall on the target system must allow the rules in the File and Printer Sharing, Remote Desktop, and Remote Event Log Management groups. If so desired, U-Move will use a different user account to log on to the remote system and transfer the license code of the local installation.

Fast and Lean Backups

To begin, I wanted to use U-Move on the local server. The functions are clearly arranged on the six tabs *Back up*, *Restore*, *Schedule*, *Clone*, *Upgrade*, and *Undo*. Each of these actions has an introductory text about its purpose, again accompanied by links to the appropriate chapters of the comprehensive help. The help not only explains how to use the tool but also the technical interrelationships and processes in AD, so less experienced AD admins, in particular, will find great support here.

From the *Back up* tab, you can manually create an initial backup of the AD. The wizard guides you through the easy-to-follow required steps. First, U-Move asks for a local path for the *staging folder*, which is the folder in which the software collects all data belonging to the backup. If you already have data from a previous backup in this path, U-Move warns you and asks for permission to delete. After that, you are able to choose whether U-Move is subsequently to pack the data into a backup archive of the BKF file type or leave it in the staging folder. The BKF archive is the recommended variant. In the next steps, you are prompted for the target path and name for the file and optionally for a password for protection. In the penultimate step, after right-clicking on the application window, you are taken to the *Advanced* settings (Figure 1), where you can optionally include Exchange, SharePoint, or Windows Server Update Services (WSUS) databases in the backup. However, these options will mainly be relevant for very small environments and installations with Small Business Server because Microsoft otherwise recommends installing these services from the AD roles on separate servers. I opted for the default settings. The wizard then presents a summary of your selected options, and you press *Finish* to start the backup process – which took only a few minutes and created a file for my backup weighing in at less than 180MB in the target directory.

Restore Without Complications

Next, I deleted various organizational units and user, computer, and group policy objects from my environment and proceeded with the *Restore*, which lets you reinstate an earlier state of the AD on a DC that is still functional. If you want to restore the AD on a newly installed replacement machine, the wizard offers the *Clone* tab.

In the first step, you are allowed to choose between a *Simple restore* or *Comprehensive restore*, which also restores additional information, such as the databases of application servers, if desired. In my case, the *Simple restore* was fine because I only needed to restore the AD, including the SYSVOL share.

In terms of possible sources for a restore, U-Move proved to be extremely flexible (Figure 2). As an alternative to a BKF file or a previously unpacked staging folder, a volume shadow copy of the local hard disk, a backup data set of the Windows server backup at a local or remote location, or the hard disk of a system that is no longer bootable were available for selection. In the case of a no longer bootable system, a physical or virtual hard disk, an image file obtained with third-party tools, or otherwise extracted data would be eligible. U-Move recovers data in almost any scenario as long as the original DC's data is still readable under the `C:\Windows`, `C:\Users`, and `C:\ProgramData` paths. For installations in deviating paths, the online help describes in detail what information the restore requires.

In the next step, I decided to restore a BKF file and specified the path to my archive, followed by an empty staging folder. The wizard unpacked all the data contained in the backup into this

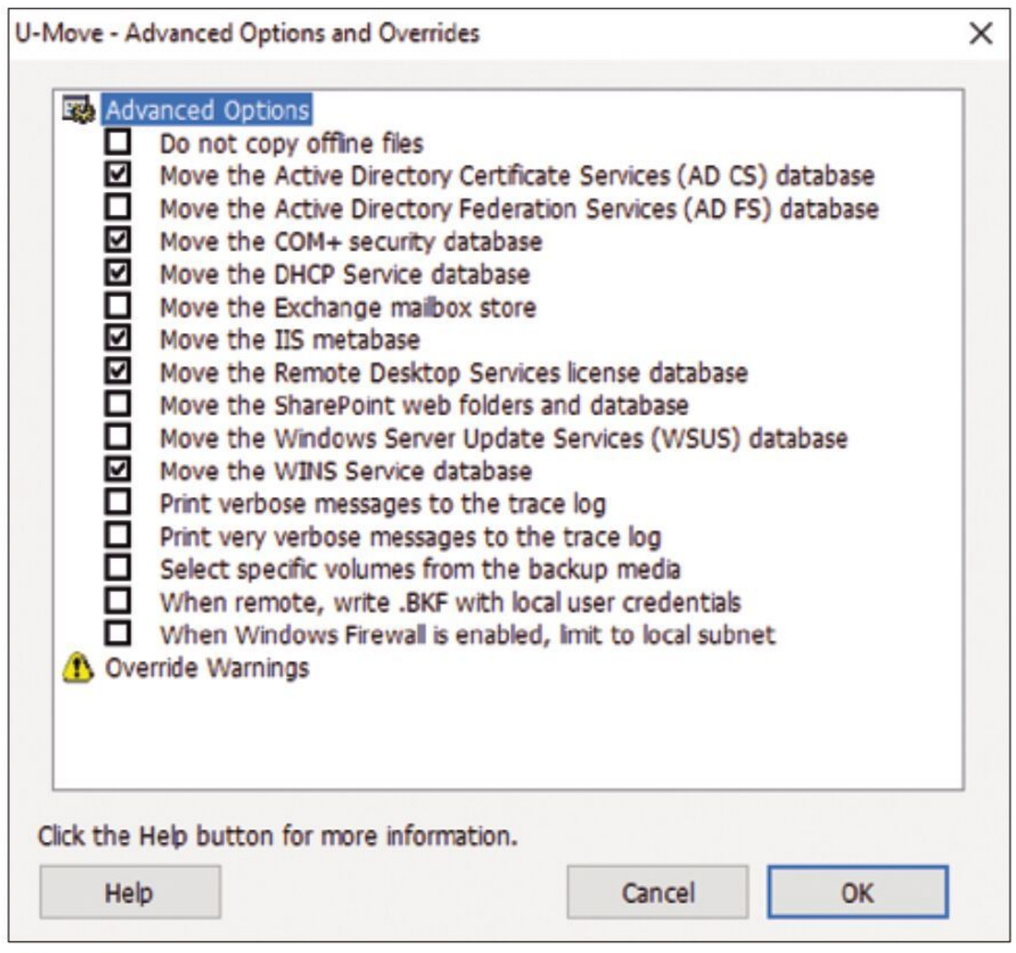


Figure 1: U-Move optionally backs up Exchange, SharePoint, and WSUS databases.

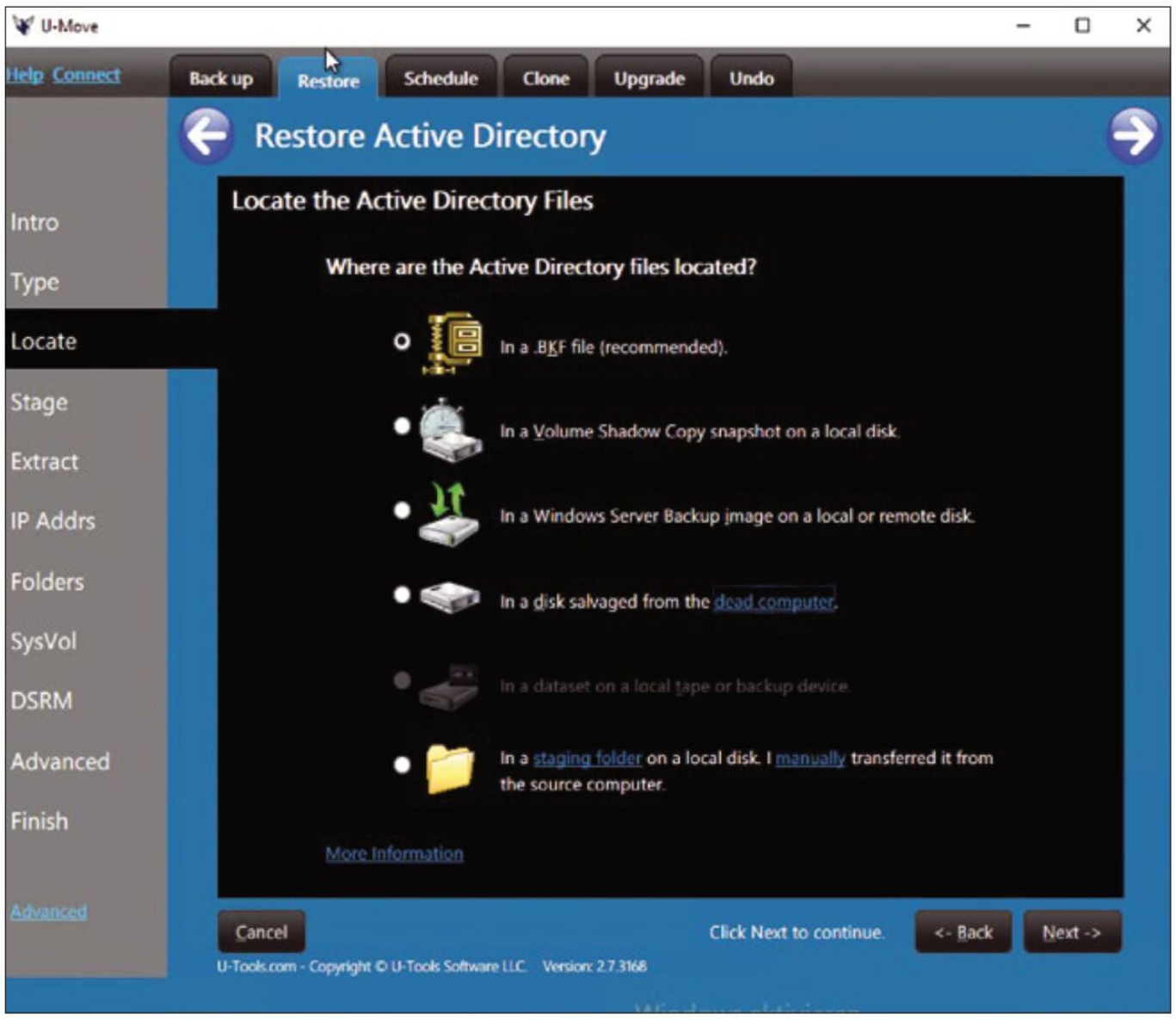


Figure 2: U-Move uses various sources for the restore.

folder, showing an overview of the backup contents on the filesystem. The wizard skipped the configuration of IP addresses and target directories as part of the simple restore.

In the dialog step that follows, you have to decide between an *Authoritative Restore* or *Non-authoritative Restore*. In the case of the third option, *Normal Restore*, U-Move would merge the contents of the SYSVOL shares of several DCs in the case of inconsistent DFS replication – an undertaking that the manufacturer explicitly warns against in the GUI and in the online help under the *Normal Restore is Abnormal* head. Because I wanted to reinstate an earlier state of my AD database, I opted for the first option, the *Authoritative Restore*. Again, once I pressed *Finish* to start the operation, U-Move restored the backup in less than a minute and triggered a reboot. Afterward, I could see for myself that U-Move had restored the AD to its previous state without any complications.

Flexible Backup According to Schedule

The third tab, *Schedule*, lets you configure a regular backup on a schedule, either daily at a specific time, weekly on a specific day, or on a user-defined schedule. You are also allowed to choose whether the backup is to run in the context of the local system or under a different user account, which has the advantage that this user can interactively follow the progress of the backup if they are logged in at the appropriate time. Furthermore, a user with appropriate permissions can also write the backup directly to a network share, which the local system itself is not able to do.

In this case, I specified the destination path for the backup and how many version levels I wanted U-Move to keep. By default, the tool keeps the last 14 backups. As with a manual backup, you have options for password-protecting the backup and for configuring notifications when the task is complete by local or remote system messages and email.

These scheduled backup options are also provided in the *Advanced* section in the sidebar. Once the configuration is completed, U-Move automatically creates a scheduled task. Afterward, the new *Cancel the scheduled backup* option appears in U-Move's Schedule Wizard for removing the task, if so desired. Alternatively, the scheduled task could simply be deleted manually with onboard Windows tools.

Move DCs by Cloning

All the scenarios on the *Clone* tab assume that the original DC is no longer accessible, and the recovery targets another machine on the same or a different network. U-Move differentiates between the fastest possible recovery on the same network in the context of disaster recovery, a planned migration to a replacement machine, copying to an isolated test environment, and migrating to a cloud or another network segment. The scenarios differ primarily in the recovery steps that U-Move recommends but, more importantly, in whether or not the recovery adopts the IP address and other network settings from the original system.

In my tests, I installed a machine with a different name and IP address on the local network parallel to the DC, without joining it to the domain. After shutting down the DC, I installed U-Move on the new system and set about restoring from the BKF file, which was similar to the restore process before.

However, the wizard now asked additional questions about handling the network settings. I decided to use the option *I am replacing the old domain controller on the same network. Copy the IP addresses,* and I was then prompted to review and confirm the IP address and DNS settings of the original system stored in the backup. U-Move then took care of everything else automatically, installing all the Windows components necessary for operation in DC mode and restoring the AD database, SYSVOL share, and DNS server. After the obligatory reboot, the server booted with the identity of the original DC, and the AD environment was back on track in no time.

A second restore of the domain in a completely separate test envi-

ronment also went ahead without complications. In this case, I wanted to duplicate the AD on a virtual machine (VM) in the Microsoft Azure cloud away from the local network. There, too, I had installed U-Move and started the clone wizard. This time I selected the option *I am cloning the domain controller to an isolated test lab. Do not copy the IP addresses*; then, I only had to intervene with the VM's network settings manually and configure local loopback address 127.0.0.1 as the primary DNS server and the Azure cloud's external DNS server as the secondary because U-Move did not change the network settings as instructed. Apart from that, this process was also fully automated, and after rebooting the machine, I had an identical clone of the production AD.

Well-Managed Migration

U-Move supports migrations to newer versions of the Windows Server operating system. As part of the test, I wanted to migrate a domain from Windows Server 2016 to 2019. The wizard on the *Upgrade* tab, although not fully automated – unlike the previous procedures – proves to be just as useful. The tool guides you through the necessary steps, removing the need for all that test overhead that you would otherwise have to go through manually as part of the migration. It checked up front that the new server and domain met all the requirements and then took care of replication tests and Flexible Single Master Operation (FSMO) role relocation (Figure 3), and it helped decommission the old DC.

To convince myself that this operation worked, I first added an instance of a Windows Server 2019 member server to my DC that was running Windows Server 2016. After launching the upgrade wizard, it prompts you to create a new project, which means U-Move can pause a migration and continue it at a later time. In the next step, U-Move connects to the old DC (in this case, the local server) and to the designated target server. On the target, it installs the U-Move remote agent and checks both servers and the AD for suitability for the upgrade, presenting a detailed report after doing so.

In the following dialog step, however, you have to get involved: U-Move explains the steps for manually installing the AD services on the target server and then upgrading it to the DC in the domain. After the obligatory reboot, U-Move takes over again, checking the new DC, as well as the replication between the two, and then recommends that you create a backup of both DCs. This was followed by a check of DNS replication and NTP settings, as well as moving the FSMO roles – a significant reduction in workload compared with doing all these steps manually. Last but not least, U-Move walks you through the optional postmigration steps and helps you remove the old DC on Windows Server 2016.

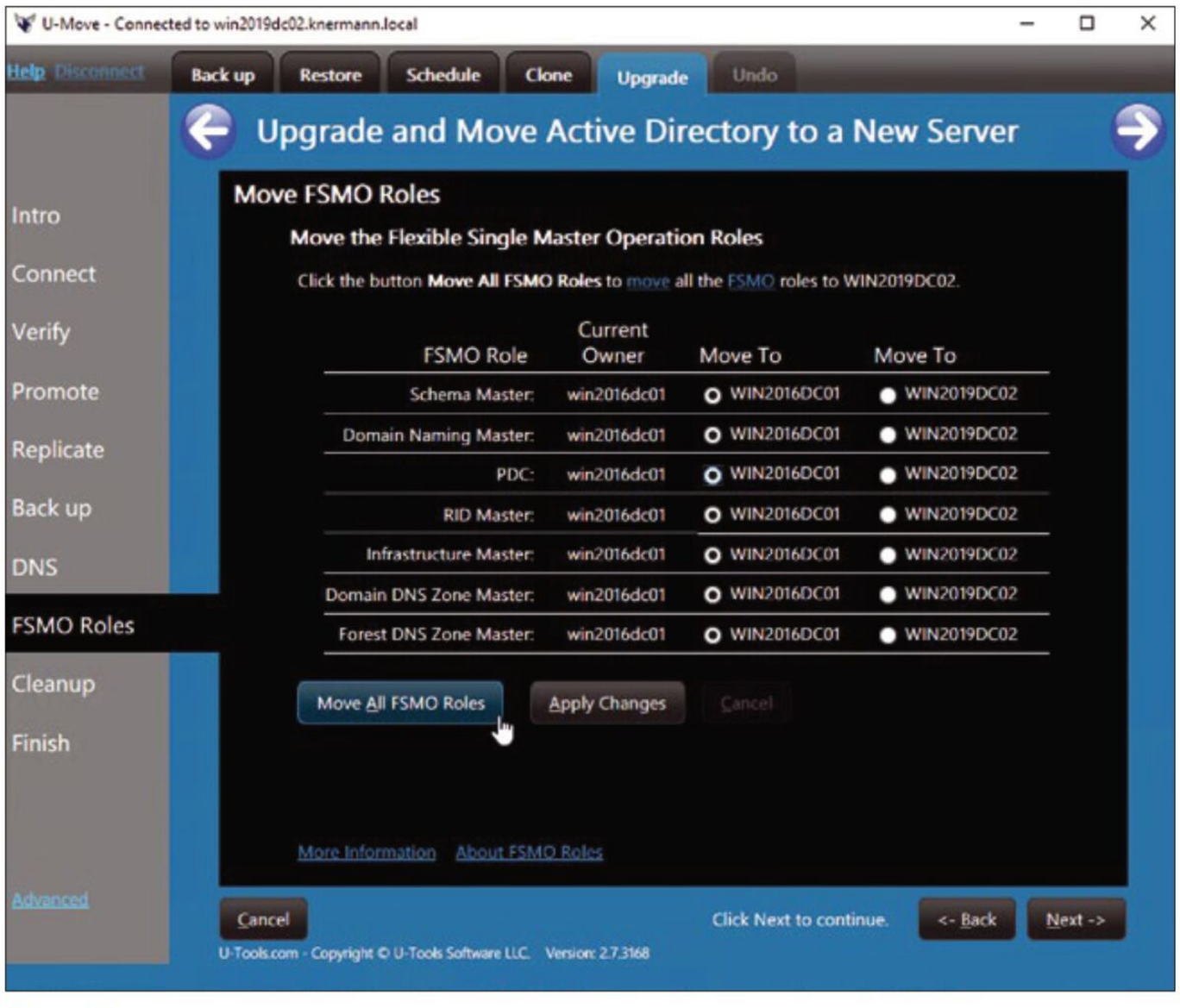


Figure 3: U-Move takes care of FSMO role relocation.

The Verdict

Rating

- Backup speed and size: 8
- Simple restore: 7
- Clone: 8
- Migration support: 7
- Online help: 8

Suitability

- Perfect as a supplement or alternative to Windows Server Backup.
- Restrictions apply for companies running additional services not supported by U-Move on DCs.
- Not useful for companies without Active Directory.

Conclusions

U-Move for Active Directory backs up and restores individual DCs and Active Directory environments faster and more easily than would be possible with the on-board tools that come with Windows Server. (See "The Verdict" box.) With the wizards and the extensive and comprehensible online help, the tool offers massive relief for migration projects or for setting up a test environment identical to your production AD. ■

Info

[1] U-Move: [https://u-tools.com/u-move]
[2] Requirements: [https://u-tools.com/help/Requirements.asp]

App Proxy support for Remote Desktop Services

Full Supply

Support flexible working environments with Remote Desktop Services and Azure AD Application Proxy. By Florian Frommherz

Azure Active Directory Application Proxy (AAP) has found its way into many organizations during the pandemic as an approach to delivering internal applications quickly and securely to stay-at-home employees. Security comes from Application Proxy (App Proxy) integration with Conditional Access, which can enforce multifactor authentication (MFA) and ensure access from trusted, managed devices tagged as "healthy." The architecture makes deployments simple. The proxy does its work with outbound network connections to the cloud only – central IT does not need to drill down into firewalls [1].

Many applications continue to make use of a full-fledged client architecture, according to which the client talks to the back end with special or proprietary protocols – or the back end cannot be easily published. In other use cases, especially when the client does not remain on the user's device but is also to be made available, the standard scenarios of a classic HTTP proxy end. A trick provides a way out: By publishing a session on a VM or session host, entire applications can be published, provided the solution makes the session accessible by a gateway or proxy over HTTPS. If you can publish the Citrix or remote desktop environment with App Proxy, you can also handle these scenarios.

In other words, you are changing the task to one of providing clients with access to a session server, which you do in as simple a way as possible, with single sign-on (SSO) – but with protection, of course. The session server then gives access to clients that do not need to talk to the back end over HTTP protocols.

Get Prepared

The implementation with Microsoft technology envisages Remote Desktop Services (RDS) for this task in combination with AAP, which publishes the RDS, supports SSO, and adds safeguards with the help of Conditional Access. This scenario allows the IT team to offer both convenient access to services with SSO and to use the set of rules from Conditional Access to define policies that only allow access by known devices perceived to be healthy or MFA for the connection.

In the simplest case, for a proof of concept, an Azure Active Directory (AD) tenant and a server running both RDS and the App Proxy agent will do the trick. For a production environment, you will want to follow Microsoft's recommendations for the number of sessions and users you are targeting. RDS comprises multiple roles; for publishing with App Proxy, the focus will be on the RD Web Access role, which handles session-to-HTTP translation. In the example here, AAP is the bridge between clients connecting from the home office and the internal network, where the session hosts and RD Web Access reside. The servers, to enable SSO, also all need to be members of a Windows AD.

Also on the procurement list is a TLS/SSL certificate for the RD web app, which is used with a private key in Azure AD (AAD) and for RDS. If you want to reach the RD

Photo by Iwona Castiello d'Antonio on Unsplash

services (in this example, by *https://websession.contoso.com*), you need the appropriate certificate for this address. You will then use it to install the RD roles on Windows Server. You should also be able to create a new canonical name (CNAME) for the target domain so that you can redirect the URL of the RDS to the unique publishing URL you get from Azure AD.

Installing Roles and Certificates

For the setup, it is best to start small: In a test environment, you can store all the required server roles for Remote Desktop and the application proxy on one server. If you plan to expand the setup slowly afterwards, you can also start with multiple servers. The RD Web Access and RD Gateway roles can be concentrated on one server and the remaining Remote Desktop roles distributed to another or multiple servers.

To start the RD setup, go to the Server Manager and, depending on the target architecture, select *Standard deployment* for multiple servers sharing different roles or *Quick Start*, then *Add roles and features*. The first RDS roles end up on a server, but the RD Gateway role is not yet included. Continue with *Quick Start* for easy deployment, then install the Gateway role. For publishing applications shared in virtual sessions, select *Session-based desktop deployment*.

Once the basic installation is complete, you will see a graphic on the target servers on the left *Remote Desktop Services* menu under *Overview* in the Server Manager; this shows you the installed roles. You can add new servers for *RD Gateway* and *RD Licensing*; the other roles are already installed. Check here that all roles are present.

If the gateway is already installed, select *Tasks* | *Edit Deployment Properties* and check that the name of the server is correctly in fully qualified domain name (FQDN) form for the self-signed SSL certificate. Under *Logon method*, make sure that *Password Authentication* is chosen in the drop-down menu and *Use RD Gateway credentials for remote computers* is checked (Figure 1). In case of a new RD Gateway installation, click on *Configure certificates* before closing the dialog. If the RD Gateway is already available, check in the *Certificates* section whether all SSL certificates are stored for all roles.

To configure SSL certificates for internal and external trusts, you can use the *Create new certificate* and *Select existing certificate* buttons for each role. The *Select existing certificate* option lets you select the certificate in PFX format (i.e., with a private key) by specifying the password and allowing its use for the respective role. However, you have to repeat this process of adding the certificate for all roles so that *Success* appears in the *State* column at the end of each role line.

If no app is configured for RD access yet, you need to create a collection where you publish the apps. If you have chosen quick deployment, this is already done, including publishing Paint and Calc as sample apps. Otherwise, you need to create a new collection in the Server Manager in RDS under *Collections*.

If you have chosen the same DNS name internally and externally, as in the example, you need to register the name internally in the DNS: The CNAME for *websession.contoso.com* should point to the name of the machine with RD Web Access and the RD Gateway.

Finally, to prepare pre-authentication for configuration with App Proxy, you need to customize the collection in PowerShell. The following PowerShell command sets the collection to pre-authentication, listening for the public DNS name that is also used from the Internet:

```
Import modules RemoteDesktop
Set-RDSessionCollectionConfiguration ➥
 -CollectionName "QuickSessionCollection" ➥
 -CustomRdpProperty "pre-authentication ➥
  server address:s:websession.➥
  contoso.com/ `n require ➥
  pre-authentication:i:1"
```

Note the `` `n `` delimiter between the URL and the `require pre-authentication` properties.

Configure the Proxy

Once the service is available locally, you can start using App Proxy to

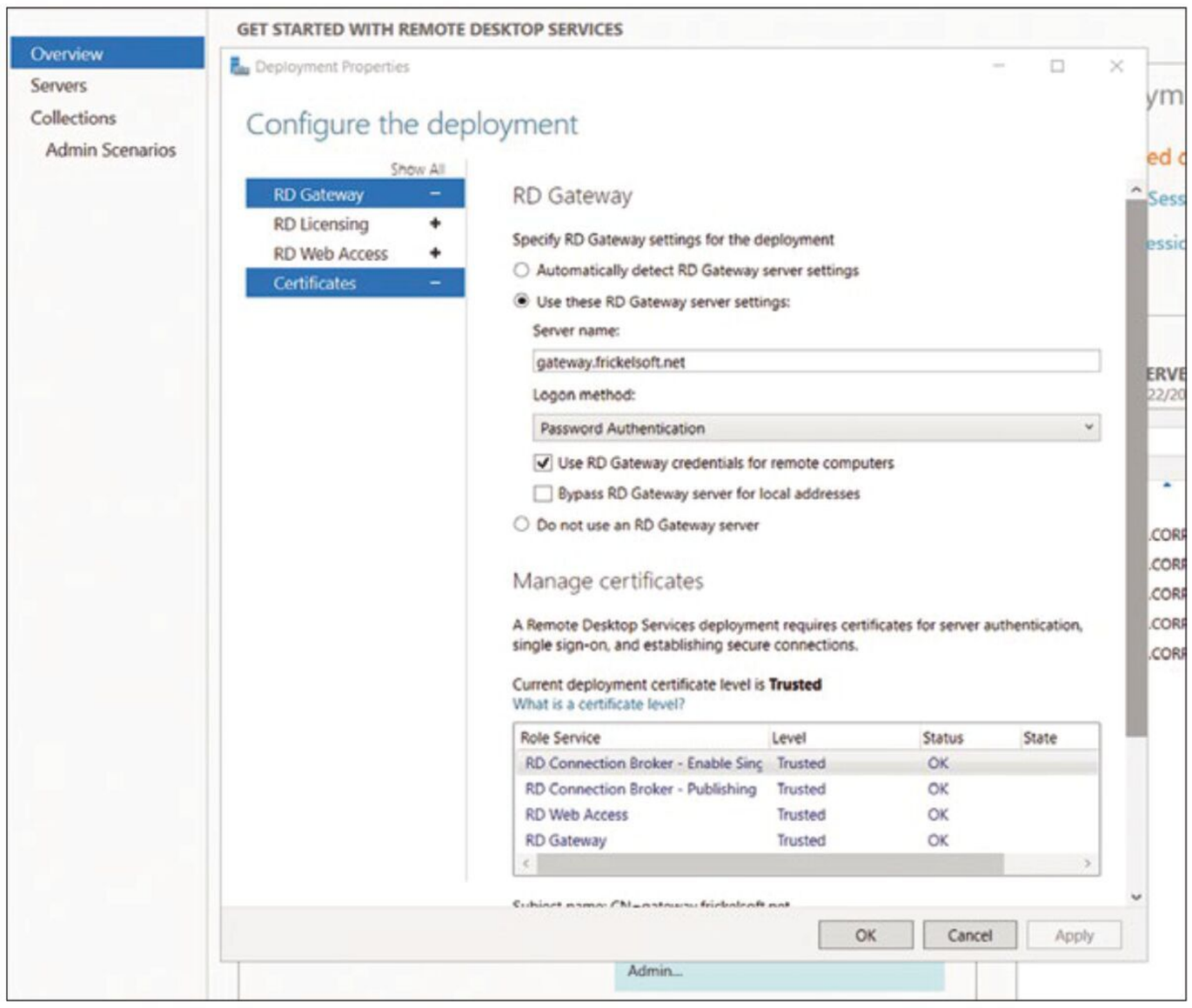


Figure 1: Remote desktop components must be able to use the correct certificates for secure publishing.

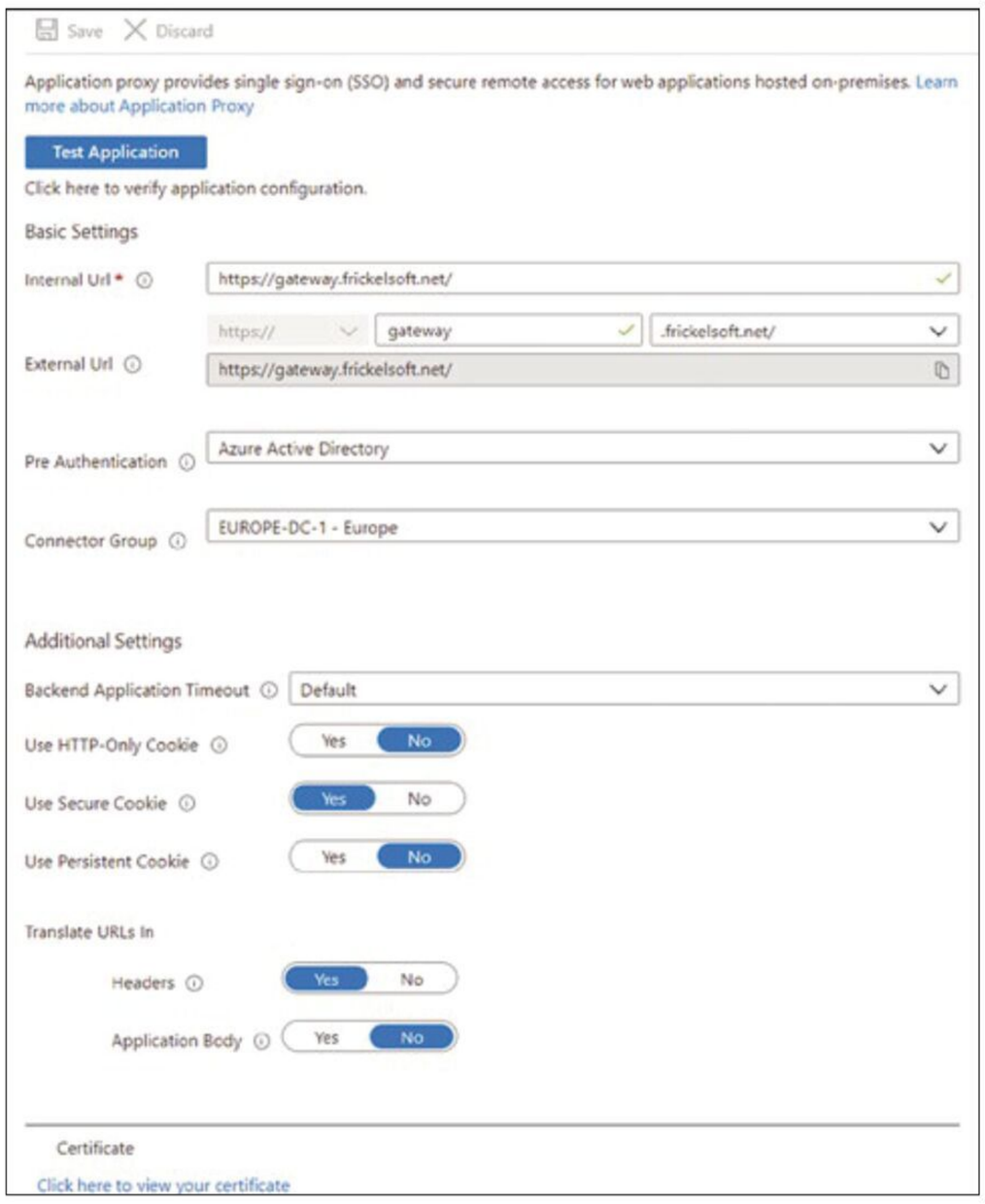


Figure 2: For secure RD application sharing, configure Azure AD App Proxy with AAD pre-authentication.

configure publishing to the outside world. You have two options: either you set up simple publishing or you let AAD perform pre-authentication, after which Conditional Access enforces further rules.

If you do not already have an App Proxy agent installed on your local network that you can use for publishing, you will want to go to the AAD portal (logged in as administrator) and download the agent from *Application Proxy* | *Download connector service*. Transfer the installation file to either the RD host or another Windows server that has a network connection both outbound to the Internet and to the Remote Desktop Server. After you start the installer, it will prompt you for permissions to register the connector in the cloud with a logon window for AAD. You can specify a user who occupies the *Application administrator* role. It does not have to be the *Global administrator*.

Shortly after installation, when you reload the Azure AD portal, take a look at *Application proxy* and find the agent marked with a green arrow, which tells you that the connection is up. The next step is to create the publication in AAD. First, click on *Enterprise applications* | *+New application*. Under the *On-premises applications* heading, you will see a suggestion for *Add an on-premises application*, which you can accept. Now the App Proxy publication wizard starts up.

Enter the application name and the internal and external URLs, which should be the same (Figure 2). In the selection field for *External Url*, the tenant-specific *<Tenant-Name>.msappproxy.net* domain is preselected, but you can select any domain for which you have completed domain registration, including validation in AAD. Therefore, you also can use internal domain addresses, as long as the internal domains are resolvable on the Internet. Finally, select *Azure Active Directory* as the option for *Pre Authentication*. The rest of the default settings are correct.

At the end of the dialog, you will see a message stating that you need to create a CNAME record for the target domain, which is required for redirection from *websession.contoso.com* to *websession-contosotenant.msappproxy.net* and avoids certificate errors. Once you have saved the App Proxy application with *+Add*, create this CNAME with a time to live (TTL) of 3600 seconds in the public DNS that clients go to outside the network. Once the application is registered, note the *Application ID* and *Object ID* in the app settings in the *Overview* – you'll need these again right away if you want to publish the HTML5 web client.

In *Properties* check whether *User assignment required?* is set to *Yes*, then the AAD will check before pre-authentication whether an administrator has assigned the RDSs to the user and whether access is desired. You now have several levers for restricting access to the services. You can define at both the gateway level and the AAD level which employees are allowed to connect to RD Web Access. Then, you can add definitions in *Users and Groups*.

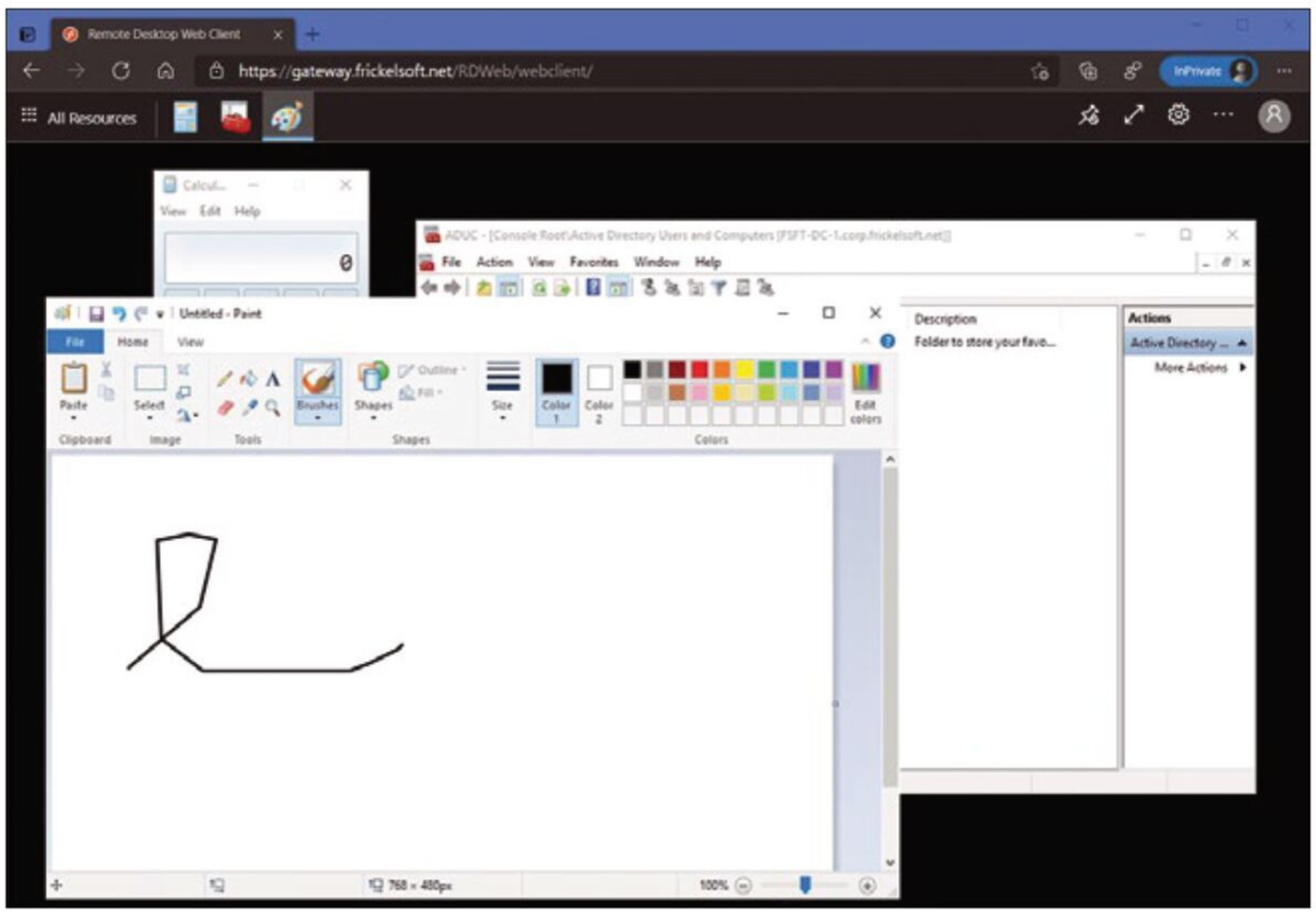


Figure 3: With the HTML5 client, application sharing looks more modern.

Enable the Web Client for HTML

To use the HTML5 web client for RDS, which offers a modern user interface (Figure 3) and is no longer based on an ActiveX add-in, install the client with the following PowerShell commands on the RD Web Access server:

```
Install-Module -Name RDWebClientManagement
Install-RDWebClientPackage
Import-RDWebClientBrokerCert ➥
  <path to CER file>
Publish-RDWebClientPackage ➥
  -Type Production -Latest
```

It is a good idea here also to customize the URL that you have shared in App Proxy so that employees are then automatically redirected to the HTML5 variant.

When you import the broker certificate in the third step, you need to specify the certificate for your publication without the private key in CER format. If you have not yet installed `PowershellGet` on the server, do so first:

```
Install-Module -Name PowershellGet -Force
```

When you set up the initial share in App Proxy, a share is created in the path *https://websession.contoso.com/RDWeb*, which automatically starts the traditional web interface. If you want to switch to the HTML5 web client, you can change the URL in App Proxy with PowerShell. To do this, you need the AAD PowerShell module and Application Administrator permissions:

```
Import modules AzureAD
Connect-AzureAD
Get-AzureADApplication | ➥
  ? {$_.AppID -eq "033deed3-eddf-459a-➥
     a8c4-99b067f6186b" } | ➥
  Set-AzureADApplication ➥
  -Homepage https://websession.➥
   contoso.com/RDWeb/webclient
```

The `AppID` you are looking for is the Application ID created during App Proxy publishing and stored with the Enterprise application object in *Properties*. When the application object has accepted the new home page, adjust the associated Enterprise application object. This time, take the object ID of the Enterprise application, for example:

```
Set-AzureADServicePrincipal ➥
  -ObjectId 4c2e134a-9884-4716-81e8-➥
   36a1eaea1b2b ➥
  -Homepage https://websession.➥
   frickelsoft.net/RDWeb/webclient
```

Give the AAD a few minutes to apply the changes and test the connection again. To do this, open an Incognito window and go to *https://myapplications.microsoft.com* for a user who should be able to access the RD services. Log in with valid credentials and then select the RD services from the list of published applications. You should be taken immediately (with SSO) to the HTML5 web client.

Conclusions

Existing Remote Desktop implementations can be published with relative ease thanks to App Proxy. Having the right certificates and adjusting the internal and external names for the web components is important. With Azure Active Directory publishing mode as pre-authentication, you can now protect the entire RD web app as an application with Conditional Access. At the same time, you can force all employees either to use multifactor authentication or, alternatively, to work from a known, healthy device when connecting to Remote Desktop by publication. ■

Info

[1] **Azure AD App Proxy:** [https://docs.microsoft.com/en-us/azure/active-directory/app-proxy/application-proxy]

PowerShell scripts for managing Microsoft 365 components

Master Key for the Cloud

Manage the various components of Microsoft 365 with PowerShell scripts that use modules culled from various Microsoft products. By Florian Frommherz

Different components of Microsoft 365 use different portals for managing services such as Teams, SharePoint, and Exchange, making administration difficult. With an arsenal of scripts and the appropriate PowerShell modules, however, many recurring activities can be conveniently controlled from the command line.

Many companies use Microsoft Teams when it comes to enterprise collaboration. The system relies on Microsoft 365 Groups to assign permissions in Exchange, SharePoint, and in itself and to control its functions. Microsoft 365 Groups are stored in Azure AD and are managed there – including the memberships for internal and external users. It's a good idea to start with Groups.

Controlling Microsoft 365 Groups

A new Microsoft 365 group can act as a team in different ways. You can either take the Exchange PowerShell approach with

```
Connect-ExchangeOnline ➥
  -userPrincipalName <user@example.com>
New-UnifiedGroup ➥
  -DisplayName "<groupname>" ➥
  -Alias "<groupalias>" ➥
  -Owner <user@example.com>
```

or use the Azure AD PowerShell modules:

```
Connect-AzureAD
New-AzureADMSGroup ➥
  -DisplayName "<groupname>" ➥
  -MailNickname "<groupalias>" ➥
  -GroupTypes "Unified" ➥
  -MailEnabled $true ➥
  -SecurityEnabled $true
```

The `Unified` group type identifies the Microsoft 365 groups that are used for Teams and Yammer, as well as permissions and mailing. Azure AD PowerShell distinguishes between the `New-AzureADGroup` and `New-AzureADMSGroup` cmdlets for traditional and Microsoft 365 groups, respectively.

If you have not installed the Exchange Online PowerShell cmdlets, do so in a PowerShell session as administrator and import the module as a normal user:

```
Install-Module ExchangeOnlineManagement
Import-Module Exchange-OnlineManagement
```

You need to be aware of one difference between the Exchange and Azure AD ways of creating groups. If you take the Exchange route, you create an associated mailbox for the group directly, whereas in Azure AD (AAD) you first initiate the creation in the directory and then create the mailbox after AAD and Exchange are synchronized.

For example, for a new sales campaign, you can easily add staff from one campaign who are already members of a team as members of the new team:

```
Get-AzureADGroupMember ➥
  -ObjectId e45712da-4a52-422c-➥
   94c3-b158d366945a ➥
```

Lead Image © vska, 123RF.com

```
| % { Add-AzureADGroupMember ➥
     -ObjectID 378f9975-143d-418d-b735-➥
      96ab403e75f9 ➥
     -RefObjectId $_.ObjectId }
```

This command first reads the members of the old campaign and then writes them to the new team (identified by `ObjectID`). In the `foreach` loop (starts with `%`), each member is considered and passed as `RefObjectID`. Group owners who do not play a central role in the life cycle of traditional groups (e.g., from Windows AD) are particularly important in Teams. The owners can configure the team in detail and are the contact persons for reviews of members:

```
Add-AzureADGroupOwner ➥
  -ObjectId 7615d111-e04b-493a-9992-➥
   dca9493828fd ➥
  -RefObjectId (➥
   Get-AzureADUser -SearchString ➥
   <User@example.com>).ObjectId
Get-AzureADGroupOwner ➥
  -ObjectId 7615d111-e04b-493a-➥
   9992-dca9493828fd
```

Groups that have fewer than one owner need closer attention. The command

```
Get-AzureADMSGroup ➥
  -Filter "groupTypes/➥
   any(c:c eq 'Unified')" ➥
  -All:$true ➥
  | ? { (Get-AzureADGroupOwner ➥
       -ObjectId $_.Id).Count -lt 1 } ➥
  | Export-CSV C:\temp\missing-owners.csv
```

finds more owners and defines them.

Managing Guest Access to the Tenant

Before you create many teams and groups, you need to familiarize yourself with the tenant settings. Guest access for external users is now allowed as a basic configuration in Microsoft Teams. If you want to make Microsoft 365 groups or teams inaccessible to external users, you can use an AAD setting that you copy as a template and then apply to the groups. This also works the other way around if you want to prohibit guest access with the tenant settings but allow external members of individual teams:

```
$template = ➥
  Get-AzureADDirectorySettingTemplate ➥
  | ? {$_.displayname ➥
       -eq "group.unified.guest"}
$preventGuests = ➥
  $template.CreateDirectorySetting()
$preventGuests["AllowToAddGuests"]=$false
```

Then, apply the setting to the groups that will no longer be able to include external members:

```
Get-AzureADMSGroup ➥
  -Filter "groupTypes/➥
          any(c:c eq 'Unified')" ➥
  -All:$true ➥
  | ? {$_.displayName -like "Finance*" } ➥
  | % { New-AzureADObjectSetting ➥
       -TargetType Groups ➥
       -TargetObjectId $_.Id ➥
       -DirectorySetting $preventGuests }
```

The command first searches for all Microsoft 365 groups with the prefix *Finance* and then applies the settings.

Controlling Groups with Labels

Labels from the Security and Compliance Center are more elegant and better automated (Figure 1). These labels can be used in many different places in the Microsoft Cloud, are not only used to encrypt email, and can classify and restrict memberships of teams. To use the labels in Azure AD for groups, you first need to enable the labels:

```
$template = ➥
  Get-AzureADDirectorySettingTemplate ➥
  | ? {$_.displayname -eq "group.unified"}
$copy = $template.CreateDirectorySetting()
$copy["EnableMIPLabels"] = $true
New-AzureADDirectorySetting ➥
  -DirectorySetting $copy
```

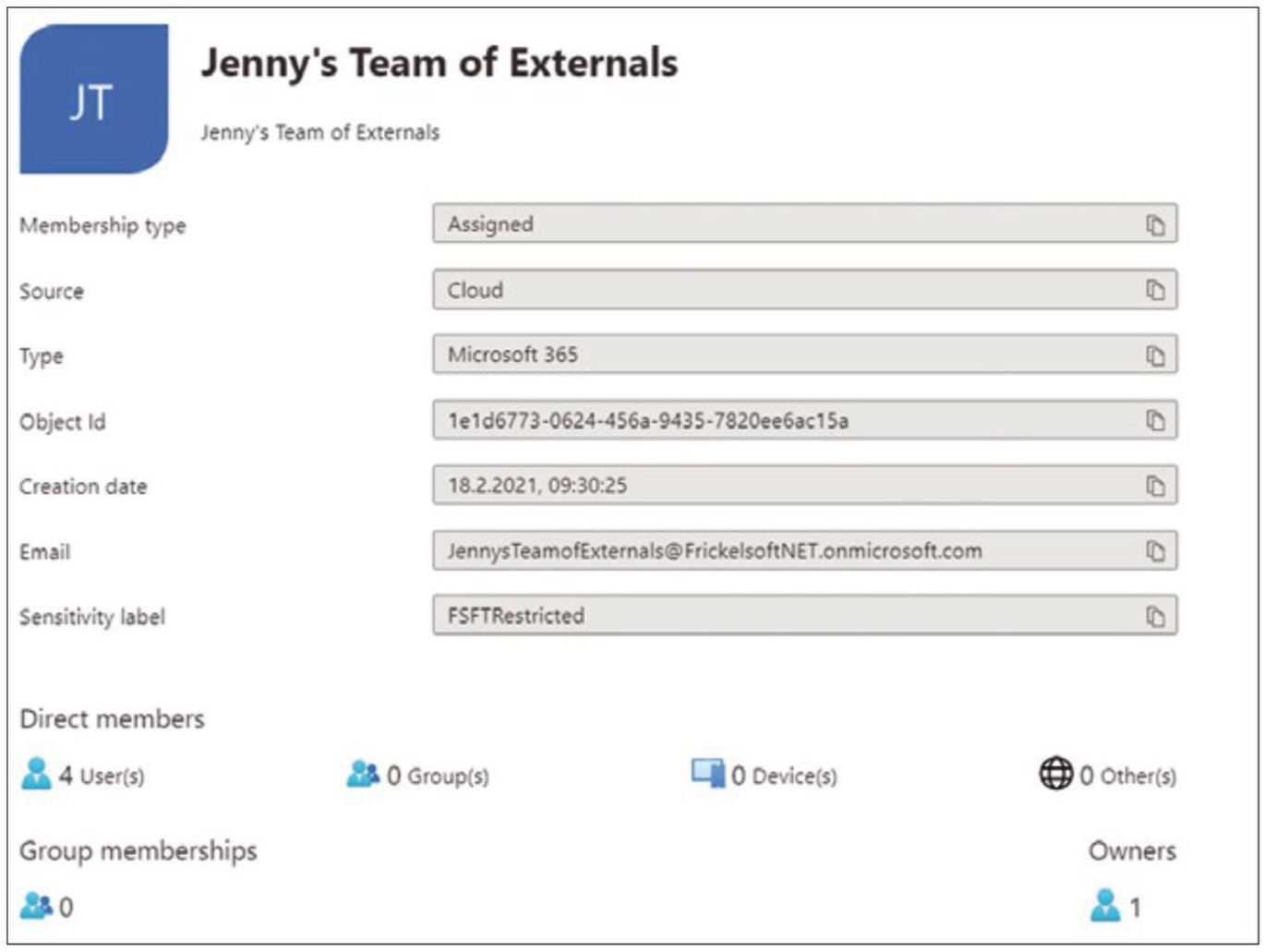


Figure 1: The successfully attached label can be seen in the properties of the group in the Azure portal.

Listing 1: New Label

```
Connect-IPPSSession -UserPrincipalName <compliance-admin@frickelsoftnet.onmicrosoft.com>
New-Label -DisplayName "FSFTTopSecret" -Name "<Frickelsoft top secret>" -Tooltip "<This is a confidential
  file>" -LabelActions '{"Type":"protectgroup","SubType":null,"Settings":[{"Key":"privacy","Value":
  "private"},{"Key":"allowemailfromguestusers","Value":"false"},{"Key":"allowaccesstoguestusers","Value":
  "false"},{"Key":"disabled","Value":"false"}]}'
```

Next, create a new label with the appropriate cmdlet from the Exchange Online PowerShell modules. However, the commands first need to connect to the Information Protection (IP) endpoint, assume the role of a compliance admin, and define a new label (Listing 1). The last command comprises two parts: creating the label and the additional information in `LabelActions` that defines the label's rules about group memberships and creating permissions for external guests.

In this example, groups classified with the label can only be joined with the owner's permission (`privacy: private`), and external members are not allowed (`allowaccesstoguestusers: false`). For deployment, you assign the label (often together with other labels in a production environment) to a label policy and trigger the synch between Exchange and the Compliance Center for Azure AD:

```
New-LabelPolicy -Name "<policyname>" ➥
  -Labels "<secretfiles>"
Execute-AzureADLabelSync
```

The label should reach Azure AD after a few minutes.

Finally, it's time to pin one of the labels on an existing or new Microsoft 365 group. To connect the label and the team, you need the unique ID of the label; you can display an overview of all labels and each `immutableID` with PowerShell:

```
Get-Label | ft ImmutableID, Name
```

The table output from the command provides the assignment of the IDs to the label names; you then use the ID of the correct label with the `LabelID` property when you create or modify the team:

```
New-AzureADMSGroup ➥
  -DisplayName "<groupname>" ➥
  -MailNickname "<groupalias>" ➥
  -GroupTypes "Unified" ➥
  -MailEnabled $true ➥
  -SecurityEnabled $true ➥
  -LabelId f460a5b0-8d8e-4ac1-bb92-➥
   afb9ea22f9da
```

If you followed the steps and created the `LabelActions` as shown in the example, the labeled team will no longer accept new members from other tenants.

Long-Term Group Management

To manage AAD groups in a valid way in the long term, it makes sense to have a lifecycle policy that supports all or selected groups. In the following example, the selected groups need to be renewed after 180 days. Groups that have no owner are reported to the notification email:

```
New-AzureADMSGroupLifecyclePolicy ➥
  -GroupLifetimeInDays 180 ➥
  -ManagedGroupTypes "Selected" ➥
  -AlternateNotificationEmails ➥
   "<user@example.com>"
```

Alternatively, you can specify `None` or `All` for `ManagedGroupTypes`; these Microsoft 365 groups are subject to the lifecycle policy. The policy ID is returned, which you can then pin on group objects:

```
Get-AzureADMSGroup ➥
  -SearchString "Project" ➥
  | % { Add-AzureADMSLifecyclePolicy➥
      Group -Id 5d69168d-b3e4-410a-b0a5-➥
       c729703ebc86 -GroupID $_.Id }
```

SharePoint Inspection

Microsoft Teams uses SharePoint to store the data in the background, securing access to all documents and resources for collaboration. Of course, SharePoint Online is also used without Teams in many places, either as a team and collaboration platform or for personal data storage on OneDrive. With the SharePoint PowerShell modules [1], you start an inventory and connect with:

```
Connect-SPOService ➥
  -Url <https://frickelsoftnet-admin.➥
       SharePoint.com>
```

You can display an overview of the sites with

```
Get-SPOSite -Limit all
```

Then, you will see not only the URLs but also the storage used and – where available – the owners of the sites. You will notice that owners cannot be found for all sites. However, if you look more closely, the `Owner` field is listed in the details:

```
Get-SPOSite ➥
  -Identity <https://frickelsoftnet.➥
   SharePoint.com/sites/Project> | fl
```

The `Owner` field contains a GUID that indicates a user or group but has a special suffix: `_o`. On closer inspection, this phenomenon occurs for all sites that originated from Microsoft 365 Groups or Teams. If you cut off the suffix and ask the AAD for a group with the resulting correct GUID, the group name is found:

```
Get-SPOSite ➥
  -Identity <https://frickelsoftnet.➥
   SharePoint.com/sites/Project> ➥
  | SELECT Owner ➥
  | % { Get-AzureADGroup ➥
      -ObjectID ($_.Owner).TrimEnd("_o") }
```

In this way, you can quickly build a command for all sites. Microsoft 365 and Teams sites are recognized because they result from a SharePoint template that has `Group*` in its name:

```
Get-SPOSite -Limit all ➥
  | % { if($_.Template -like '<GROUP*>') ➥
      { $owner = Get-SPOSite ➥
       -Identity $_.URL ➥
       | SELECT -ExpandProperty Owner; ➥
       $owner = (Get-AzureADGroup ➥
        -objectID $owner.TrimEnd("_o")).➥
        displayName } else ➥
      { $owner = $_.Owner} ➥
      Write-Host $_.URL $owner }
```

Alternatively, the group on which the team, and thus Microsoft SharePoint, is based can also be read out in the `GroupID` attribute in the SPO site details – without the suffix.

Whom collaboration is allowed with is determined either at the tenant or site level. The `SharingCapability` attribute describes the tenant setting:

```
Get-SPOTenant | SELECT SharingCapability
```

The supported values are `Disabled` for disabled external sharing and internal-only use, `ExistingExternalUserSharingOnly` for collaboration with existing guests, and `External-UserAndGuestSharing` for existing as well as new guests included by SharePoint email one-time password (OTP). The sharing settings for sites can differ from the tenant setting, so it is worth taking a look at the site:

```
Get-SPOSite ➥
  -Identity <https://frickelsoftnet.➥
   SharePoint.com/sites/Project> ➥
  -Detailed | SELECT SharingCapability
```

The `ExternalUserAndGuestSharing` setting in particular integrates partners and suppliers outside of your organization with an OTP that reaches the user by email. You then create a SharePoint account for the external party. If you use a managed Azure AD and want to prevent SharePoint accounts being created, you can use PowerShell to force the SharePoint tenant to send Azure AD business-to-business (B2B) invitations – even by email OTP if necessary:

```
Set-SPOTenant ➥
  -EnableAzureADB2BIntegration $true
Set-SPOTenant ➥
  -SyncAadB2BManagementPolicy $true
Set-SPOTenant ➥
  -CustomizedExternalSharingServiceUrl ➥
   <https://sharing.frickelsoft.net/➥
    external-access>
```

The function is still in the final stages of development at Microsoft but is already available in the Public Preview. Once enabled, the external accounts between Azure AD and SharePoint are no longer any different, and SharePoint's own OTP solution is no longer necessary.

Logging Activities

Provided you have the appropriate licenses, you can use the Compliance Center in Microsoft 365 to generate alerts when something happens on your tenant that you either don't want to happen or at least want to keep a closer eye on. These alerts are structured such that you can define what action triggers the alert and who is notified. Also, you can stipulate that the alert is only triggered if someone specific takes the action. An overview of existing alerts is displayed with:

```
Get-ActivityAlert | ft Description,Name
```

A simple example would be when a user shares a SharePoint page or file with an external party. If you want to watch more closely when user Jenny generates a new invitation, create the alert with:

```
New-ActivityAlert ➥
  -Name "<External Sharing Alert>" ➥
  -Operation sharinginvitationcreated ➥
  -NotifyUser <alarms_m365@frickelsoft.➥
   net>, <florian@frickelsoft.net> ➥
  -UserId <jenny@frickelsoft.net> ➥
  -Description "<Triggers an alert if ➥
   Jenny generates a sharing invitation>"
```

If you are interested in knowing when user Sarah deletes a team, use the new alert:

```
New-ActivityAlert ➥
  -Name "Team deletion alert" ➥
  -Operation teamdeleted ➥
  -NotifyUser <alarms_m365@frickelsoft.➥
   net>, <florian@frickelsoft.net> ➥
  -UserId sarah@frickelsoft.net ➥
  -Description "<If Sarah deletes a ➥
   Microsoft 365 group or team, an alert ➥
   is triggered>"
```

The email then mailed to your Inbox refers you to the admin Compliance Center, where you can investigate the details.

The AAD log also contains the group changes, and the AAD Preview module (set up with `Import-Module AzureADPreview`) allows you to perform specific searches there:

```
Get-AzureADAuditDirectoryLogs -Filter ➥
  "initiatedBy/user/userPrincipalName eq ➥
   '<user@example.com>' and ➥
   ActivityDisplayName eq 'Delete group'"
```

If you are interested in the last changes to the group before it was deleted, you can expand the previous command and extract the object ID of the group. Simply add

```
| SELECT ➥
  -ExpandProperty TargetResources ➥
  | SELECT ID
```

The object ID (*1c9c09d7-4b3c-4f37-b2cf-e3b8ad0a2ecf* in this example) can then be used to search for audit entries:

```
Get-AzureADAuditDirectoryLogs ➥
  -Filter "targetResources/any (➥
   tr:tr/id eq '1c9c09d7-4b3c-4f37-b2cf-➥
   e3b8ad0a2ecf')" ➥
  | ft ActivityDisplayName, ➥
    ActivityDateTime
```

As far back as the audit log goes (30 days in Azure AD Premium tenants), you will then see a history of the changes that the group has undergone – from the creation of new owners, as well as new members and updates to attributes, to deletion.

Conclusions

In this article, I looked at Microsoft 365 and worked with groups, teams, external users, and sensitivity labels. To get a handle on Microsoft 365, you can't get around PowerShell. Unfortunately, no all-encompassing PowerShell master module covers the most important functions.

Administrators still need to rely on the modules for Azure AD, Azure AD Preview, Exchange Online Management, SharePoint Online, Microsoft Teams, and so on. Therefore, it is always a good idea to create a script repository in a solid development environment such as Visual Studio Code or a smart text editor such as Notepad++. ■

Info

[1] **SharePoint Online Management Shell:** [https://www.microsoft.com/en-us/download/details.aspx?id=35588]

Flexible software routing with open source FRR

Special Delivery

The FRR open routing stack can be integrated into many networks because it supports a large number of routing protocols, though its strong dependence on the underlying kernel means it requires some manual configuration.

By Benjamin Pfister

In the past, many network administrators used pre-installed and expensive appliances for routing. Although this was used for a long time as a reliable solution, it is no longer suitable for more flexible use. For example, if you have highly virtualized server environments, the norm today, hardware appliances don't really make sense.

New approaches, such as network function virtualization (NFV) or software-defined networking (SDN), decouple networks from the hardware. Routing therefore needs to support integration into a service chain (i.e., the concatenation of the required services, alongside other functions such as firewalling or intrusion detection and prevention).

Moreover, it used to take a large amount of space, power, and money to learn in a test environment how routing protocols work with real hardware routers or to recreate specific behaviors. Network administrators who wanted to test their implementations for vulnerabilities had to build elaborate hardware environments or write their own routing stacks.

In this article, I look at an open routing stack provided by the open source project Free Range Routing [1], usually known as FRRouting or FRR.

Open Source Routing Stack Remedy

An open source routing stack can be an alternative to classic routers. In contrast, the conventional monolithic architectures with specialized application-specific integrated circuits (ASICs) or field-programmable gate arrays (FPGAs) for hardware-optimized packet forwarding obviously cannot directly offer the same performance optimizations; however, in some constellations, it is not so much a question of forwarding performance, but more likely about the functionality of the control plane, as when validating a network design. Control plane refers to the functions that can be used to control networks – Spanning Tree on Layer 2, for example, and the corresponding routing protocols on Layer 3.

However, technologies such as the Data Plane Development Kit (DPDK) [2] for performance optimization, developed by Intel in 2010 and now under the auspices of the Linux Foundation, now also allow direct access to physical resources without placing too much strain on remaining resources such as the CPU.

Single-root I/O virtualization (SR-IOV) can also provide the necessary flexibility in virtualized environments to access a native hardware resource with multiple virtualized network functions. In combination with the currently much-hyped SmartNICs (programmable network adapter cards with programmable accelerators and Ethernet), optimized packet forwarding could usher in a new network architecture with an open routing stack. Here, network functions are to be outsourced from the host to specialized network cards, such as encryption functions for virtual private networks (VPNs), deep packet inspection for next generation firewalls, and offloading of routing tasks,

Photo by Sahand Babali on Unsplash

which makes SmartNICs interesting for software routers.

Free Range Routing

FRR emerged as a fork from the Quagga project [3]. Quagga itself has been known to some administrators for years as a component of other open source projects, such as pfSense [4], into which FRR can now also be integrated. But Quagga is also used as the routing stack of Sophos's unified threat management software as a routing substructure. Quagga itself was created in 2002 as a fork from the Zebra project, which is no longer maintained.

The fork from Quagga to FRR came about because of the large backlogs of patches and the slow evolution of Quagga. FRRouting has a four-month release cycle. Currently, the project is under the care of the Linux Foundation, to which it was handed over in April 2017. However, many organizations also very actively support the development work, including VMware, Orange, Internet Systems Consortium (ISC), Nvidia, and Cumulus Networks. As a licensing model, it is under the GPLv2 + .

FRR's routing stack can adapt very flexibly to different environments, as is shown by the many implementations that include the previously mentioned open source firewall system pfSense but also its fork OPNsense and the complete Network Operating System (NOS) VyOS. Furthermore, routing functionalities of data center switches can also be implemented with FRR, as is demonstrated by the integration of the routing stack in switches from Cumulus Networks.

It should be noted that FRR is only responsible for the control plane. The decision about forwarding IP packets is made by the kernel of the underlying operating system.

Flexible Architecture

Before I go into the individual performance features, I will first take a look at the architecture. C is the programming language with individual additions in Python. Otherwise, FRR differs fundamentally from classic network operating systems. As I already pointed out at the beginning of this article, the nature of the software architecture in classic routers is monolithic and is attributable to scarce resource availability at the time. In these cases, all processes for the dynamic routing protocols are already activated out of the box, which generates unnecessary load, opens up attack vectors, and increases complexity. For example, each routing process communicates directly with the other dynamic routing protocol when redistributing from one to the other. A different interface must be known and documented for each routing protocol, and the programming overhead grows with each additional protocol. FRR solves this problem more elegantly than the software architecture in classic routers. To do so, it introduces a central and protocol-independent mediator process daemon (Figure 1) named Zebra. A dynamic routing protocol, such as the Border Gateway Protocol (BGP), binds to this daemon with the Zebra API (ZAPI). Together with the dynamic routing protocol daemons, the Zebra daemon forms the control plane. However, packet forwarding itself is handled by the kernel of the underlying operating system.

The routes from the Zebra process now have to be transferred from the userspace process to the kernel of the operating system through a socket-based interface known as the Netlink bus. The interface has a function for adding new routes (`RTM_NEWROUTE`), as can be seen in Figure 2, but it can

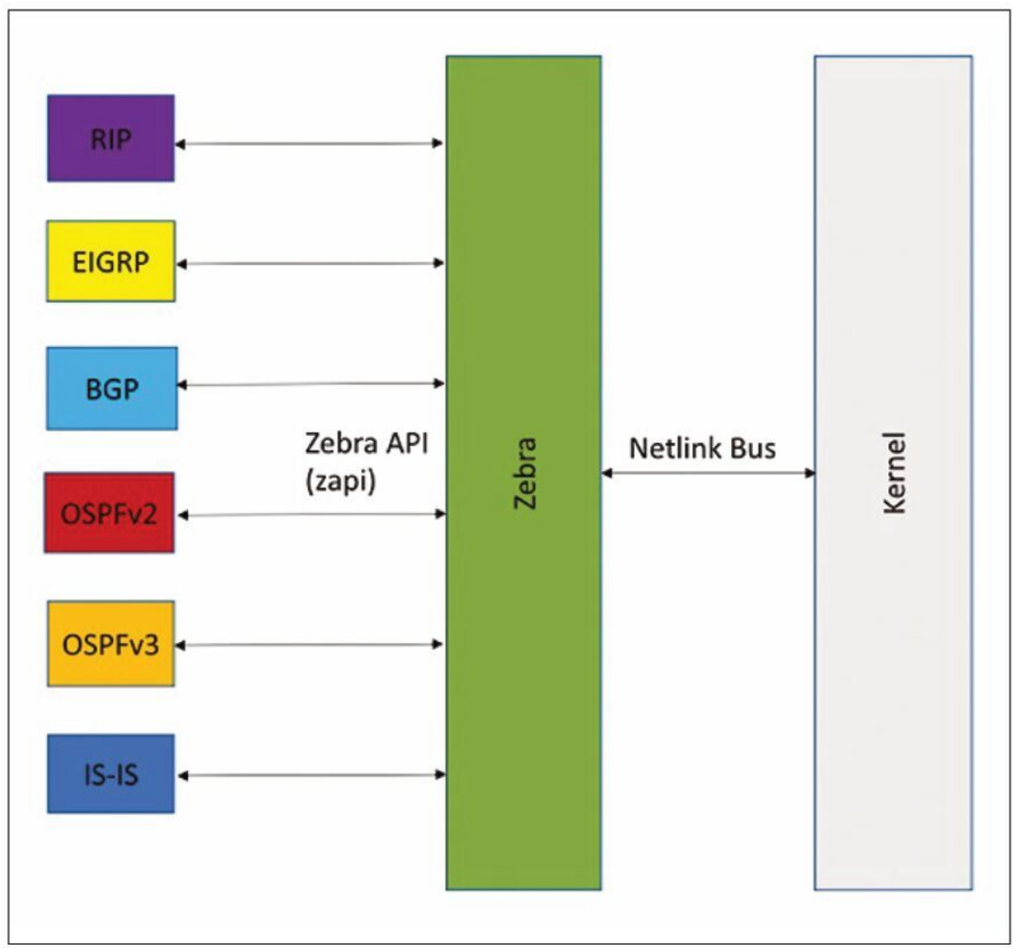


Figure 1: In the FRR architecture, the dynamic routing protocols (left) connect to the Zebra central and protocol-independent mediator process through the Zebra API.

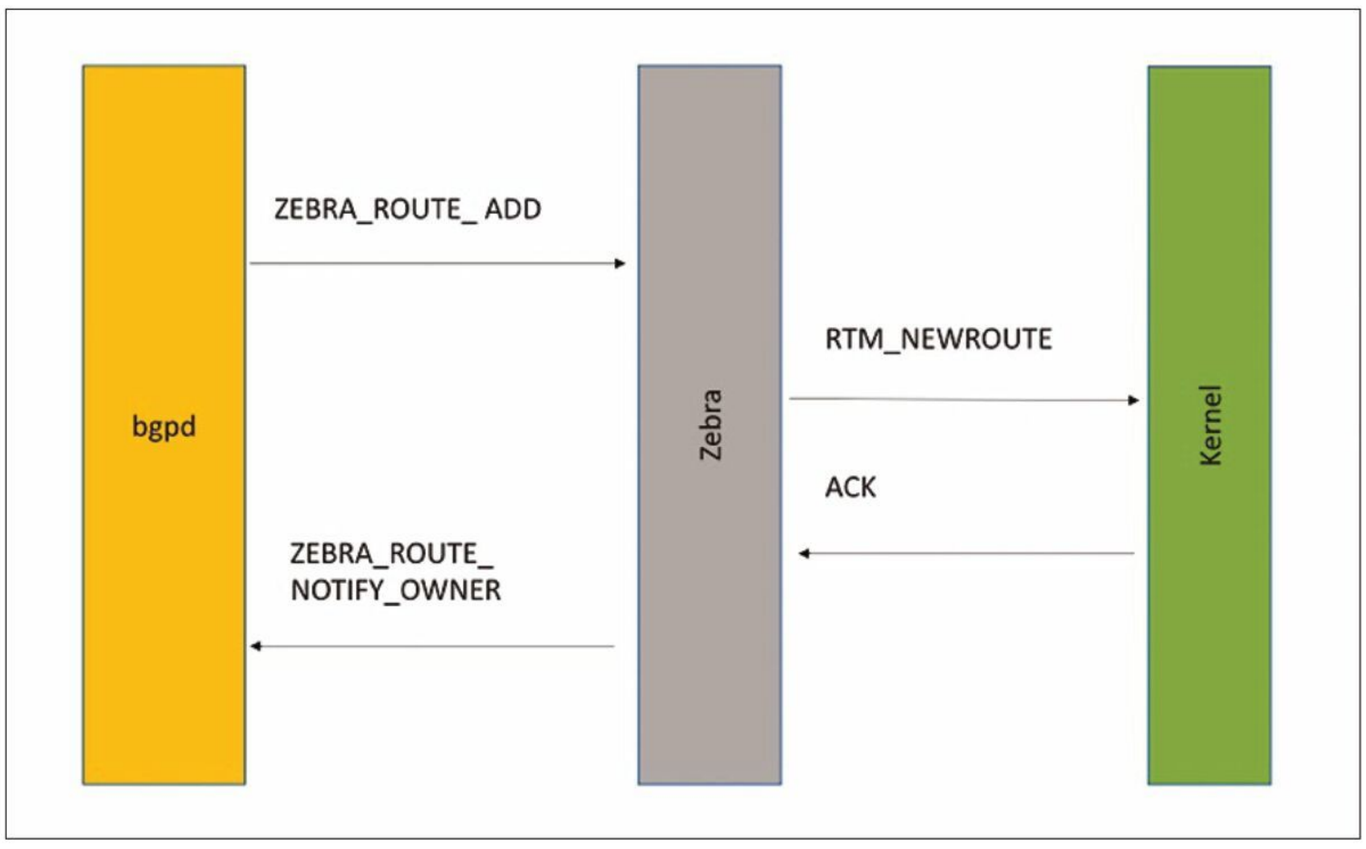


Figure 2: Flow for propagating a new route from FRR's BGP daemon into the kernel.

also signal new routes in the kernel to the Zebra process. The BGP daemon (`bgpd`) passes the route to the Zebra daemon through the Zebra API (`ZEBRA_ROUTE_ADD`). The daemon uses the Netlink bus function `RTM_NEWROUTE` to pass the new route to the kernel. Confirmation takes place afterward.

The architecture, relying on the middleman process, facilitates the integration of new routing protocols because there is a uniform interface (ZAPI). In redistributing from routing process A to B, routing process A gives its routes to the Zebra process by way of the Zebra API, and ZAPI passes them to routing process B. Errors and crashes in one protocol do not necessarily affect other daemons, which basically improves overall availability.

To use a dynamic routing protocol, you need to enable it in `/etc/frr/daemons`. Only the Zebra daemon and `watchfrr` daemon, which detects faulty daemons and restarts them if necessary, are already enabled after installation. For all others, you need to configure `/etc/frr/daemons`.

Classic and Northbound Layer Configuration Model

The management interface is undergoing some upheaval right now between the classic configuration model and the model with a Northbound layer (highest layer in an SDN architecture) designed to improve flexibility and offer support for modern API-based configuration interfaces such as RESTCONF and NETCONF. I'll look at the classic configuration model first: the command-line interface (CLI) for FRR daemons, `vtysh` [5]. The shell is based on the classic NOS. If you use Cisco Internetwork Operating System (IOS), you will be able to find your way around quickly because the syntax is very similar. Cisco IOS supports a user EXEC mode as well as a privileged EXEC mode, in which you can use the `configure` terminal command to switch to the global configuration mode and subsequently to the interface-specific configuration modes and routing protocol configuration modes.

The `vtysh` CLI config tool also supports a privileged EXEC mode and a global configuration mode, as well as some more specific modes. Saving with the `write memory` command should also be familiar to some administrators. However, the way you store the saved startup configuration is different. Although IOS-based systems store the startup configuration in NVRAM, it requires the VTYSH integrated configuration file on FRR, which is enabled by default in the `/etc/frr/vtysh.conf` file. This kind of configuration operation gives you a summarized configuration file. Each daemon parses this file at startup and pulls the relevant components. Assuming a separate configuration file will be used for each daemon, you need to use the

```
no service integrated-vtysh-configuration
```

command. To check whether an integrated configuration file exists, you need to check `/etc/frr/vtysh.conf`. If the contents match

```
service integrated-vtysh-config
```

then an integrated configuration exists. However, as is also currently à la mode with commercial manufacturers, the project seeks to use API-based configuration models to enable better support for network automation. The developers' goal is a completely programmable routing stack through the gradual integration of the YANG data model. For this purpose, a Northbound layer is used, to which the different configuration interfaces can then connect. Figure 3 shows an example of the old and new models; however, the entire process is still a work in progress because far-reaching changes in the source code are necessary.

Variety of Dynamic Routing Protocols

FRR as a routing stack cannot be compared with a complete network operating system because it is specifically used to publish and receive

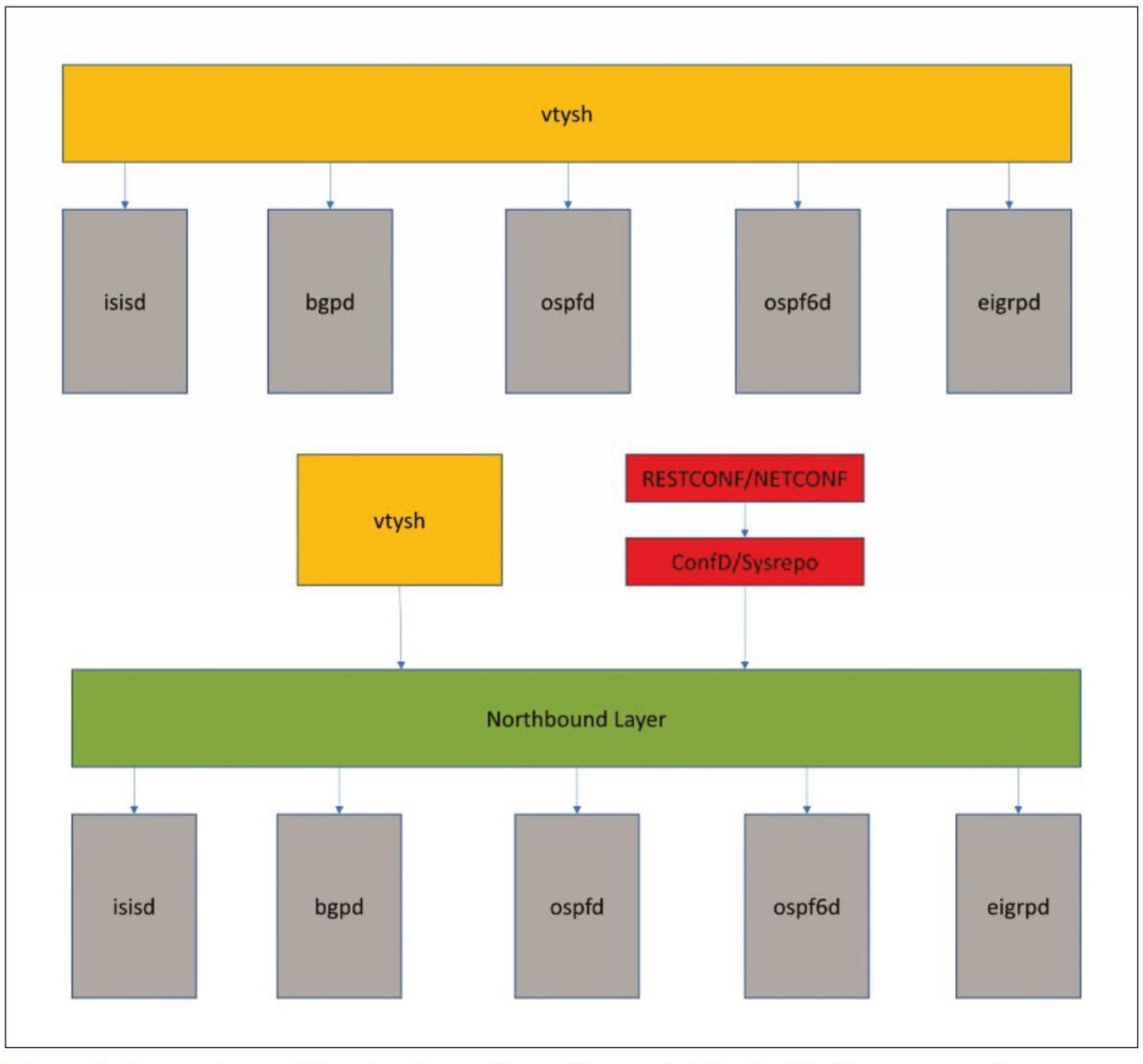


Figure 3: Comparison of the classic configuration model (top) with the more modern configuration model (bottom) that includes optimizations for APIs, such as NETCONF and RESTCONF.

IP-based routing information on the basis of supported dynamic routing protocols and parameterized policies. It also takes care of passing this information to the operating system kernel over the Netlink bus. It does not provide additional functions such as firewalling, network address translation (NAT), or quality of service (QoS). For this, you still need to rely on the functionality of the underlying operating system or third-party systems.

FRR's feature set includes a wide variety of dynamic routing protocols. It should be noted that BSD-based operating systems do not support all functions. A close look at the feature matrix [6] is also recommended for Linux because individual functions require a certain kernel version. However, if you observe the minimum kernel versions, all functions are basically available under Linux.

FRR can be used to set up static, policy-based, and dynamic routes. The dynamic routing protocols include both interior and exterior gateway routing protocols. Path-vector, distance-vector, and link-state protocols are available depending on the desired deployment scenario. BGP is one of the exterior gateway protocols, so it can also be used as an Internet edge router on corporate or provider networks.

However, BGP not only impresses on this interface, it also forms the basis for the Border Gateway Protocol Ethernet virtual private network (BGP EVPN), on which the virtual extensible local area network (VXLAN) runs as a data plane, offering the possibility of Layer 2 links between data centers across Layer 3 borders. Additionally, it can be combined with equal-cost multipathing (i.e., the simultaneous and equivalent use of several available links). Layer 2 linking is particularly interesting if virtual machines are running on the corresponding host. Provided the hypervisor you use supports it, the virtual machines can be migrated between sites without modifying the IP addresses, and it is possible without the restrictions known from the spanning tree method.

In particular, FRR can hold its own in terms of Interior Gateway Protocols (IGPs). If you take a look at the supported IGPs, you will find well-known protocols such as Routing Information Protocol (RIP) version 2 and Open Shortest Path First (OSPF) version 2 for IPv4, as well as their IPv6 derivatives RIPng and OSPFv3. Additionally, Intermediate System-to-Intermediate System (IS-IS) and OpenFabric – which was inspired by IS-IS but optimized for spine-leaf architectures in data centers – have found their way into the feature list.

The developers have also found a solution for integration into networks based on the Enhanced Interior Gateway Routing Protocol (EIGRP), which was initially proprietary to Cisco before being published in RFC7868. Support is still considered alpha according to the note on GitHub; however, I could not find any bugs in my tests. Additionally, an implementation of Babel, a distance-vector routing protocol, is optimized for wireless mesh networks. A list of supported protocols can be found online [6].

If FRR is to be used on large enterprise networks with multiprotocol label switching (MPLS), the Label Distribution Protocol (LDP) daemon can be used to distribute the MPLS labels. However, this requires additional configuration and support in the kernel. The virtual routing and forwarding (VRF) feature is also suitable for larger multitenant networks or routing isolation. Multiple routing instances exist that cannot see each other without deliberate route leaking [7]. Overlapping IP address blocks in different VRFs are possible.

Although all of the above protocols are suitable for unicast traffic, FRR also offers a solution for multicast traffic, for which a protocol-independent multicast (PIM) daemon is available. Also, FRR offers several functions to increase availability. For example, errors between routers can be detected more quickly with bidirectional forward detection (BFD). If several routers are used as the first hop, a First Hop Redundancy Protocol (FHRP) is required. FRR offers the Virtual Router Redundancy Protocol (VRRP) for this purpose. From the daemon for the Next Hop Resolution Protocol (NHRP), it is also possible to set up dynamic multipoint VPN solutions.

Installing FRRouting

You can obtain and install FRR from various sources. The most flexible, but also most complex, option is to clone the code from the official GitHub repository and then compile and install it yourself. This method gives you access to the latest releases. The more convenient, but not always completely up-to-date, option is to install from the operating system sources. I chose the latter method and installed FRR on an Ubuntu 20.04 server:

```
sudo apt update
sudo apt install frr
sudo vi /etc/sysctl.conf
net.ipv4.conf.all.forwarding = 1
net.ipv6.conf. all.forwarding = 1
```

The last two lines enable routing for IPv4 and IPv6 at the operating system level.

Configuring FRR

Before I go into the configuration example in more detail, some preparation is required. You need to enable the desired routing protocol in `/etc/frr/daemons`, which is ready after reloading. Use

```
show watchfrr
```

in the VTYSH shell to check which daemons are active.

Discussing the configurations of all supported dynamic routing protocols is well beyond the scope of this article, so I am limiting myself to a simple example for an eBGP link between two routers. Each of the two routers announces the IP address of the respective loopback interface to

Listing 1: FRR1 Netplan YAML Config

```
network:
  ethernets:
    ens33:
      dhcp4: no
      addresses:
        - 192.0.2.0/31
    lo:
      addresses:
        - 192.168.1.1/32
  version: 2
  renderer: networkd
```

Listing 2: FRR2 Netplan YAML Config

```
network:
  ethernets:
    ens33:
      dhcp4: no
      addresses:
        - 192.0.2.1/31
    lo:
      addresses:
        - 192.168.2.2/32
  version: 2
  renderer: networkd
```

Listing 3: BGP Config Snippets

```
01 #FRR1
02 router bgp 65540
03   neighbor LAB peer-group
04   neighbor LAB remote-as 65541
05   neighbor LAB password t0ps3cr3t
06   neighbor 192.0.2.1 peer-group LAB
07   ...
08   address-family ipv4 unicast network 192.168.1.1/32
09   exit-address-family
10 #FRR2
11 router bgp 65541
12   neighbor LAB peer-group
13   neighbor LAB remote-as 65540
14   neighbor LAB password t0ps3cr3t
15   neighbor 192.0.2.0 peer-group LAB
16   ...
17   address-family ipv4 unicast network 192.168.2.2/32
18   exit-address-family
```

the opposite router over eBGP. To do this, you assign IP addresses to the loopback interfaces in line with the IP addresses shown in Figure 4 on the underlying Ubuntu 20.04 server; then, you give the *ens33* interface an IP address on the transfer network. To do this, you generate the YAML files for Netplan (Listings 1 and 2) to obtain a bootable configuration, which is enabled by

```
sudo netplan apply
```

Now that the hosts can reach each other on transfer network 192.0.2.0/31, the FRR configuration can take place. I used VTYSH for the parameterization work. Listing 3 shows the configuration components for eBGP. This example is only to illustrate functionality; it is not intended for production use.

Now I'll walk through the configuration from Listing 3 with FRR1. In the first step, you need a BGP neighborhood. To do this, line 2 parameterizes autonomous system (AS) 65540; then, line 3 creates the LAB peer group, so you can apply the peer configuration to other BGP peers. The remote AS 65541 is used in line 4. To harden the peering, line 5 configures an MD5 password, then line 6 binds neighbor 192.0.2.1 (FRR2) to the LAB peer group, so it inherits the appropriate parameters. Once peering is in place between the routers, line 8 enters the address family and configures the host route 192.168.1.1/32 for publication.

After the configuration is complete on FRR2, you can check the peering status with

```
show ip bgp summary
```

which is familiar from classic network operating systems. To query the routes learned by BGP, type

```
show ip route bgp
```

The output shows you a host route of 192.168.2.2/32 on FRR1.

Conclusions

FRR offers a flexible routing stack for anything from lab environments to production data center networks. FRR also makes a good impression when facing Pentest-Tools for routing protocols such as Routopsy. The large number of routing protocols supported means that FRR can be integrated into many networks but with a downside: Parameterization – especially in combination with additional functions such as a network address translation – seems fragmented in some cases because of the strong dependence on the underlying kernel. ■

Info

[1] FRRouting on GitHub: [https://github.com/FRRouting/frr]

[2] Developer Quick Start Guide: [https://www.dpdk.org]

[3] Quagga routing suite: [https://www.quagga.net]

[4] pfSense: [https://www.pfsense.org]

[5] VTYSH docs: [http://docs.frrouting.org/projects/dev-guide/en/latest/vtysh.html]

[6] FRR feature matrix: [http://docs.frrouting.org/en/latest/overview.html#feature-matrix]

[7] Route leaking: [https://www.cisco.com/c/en/us/index.html]

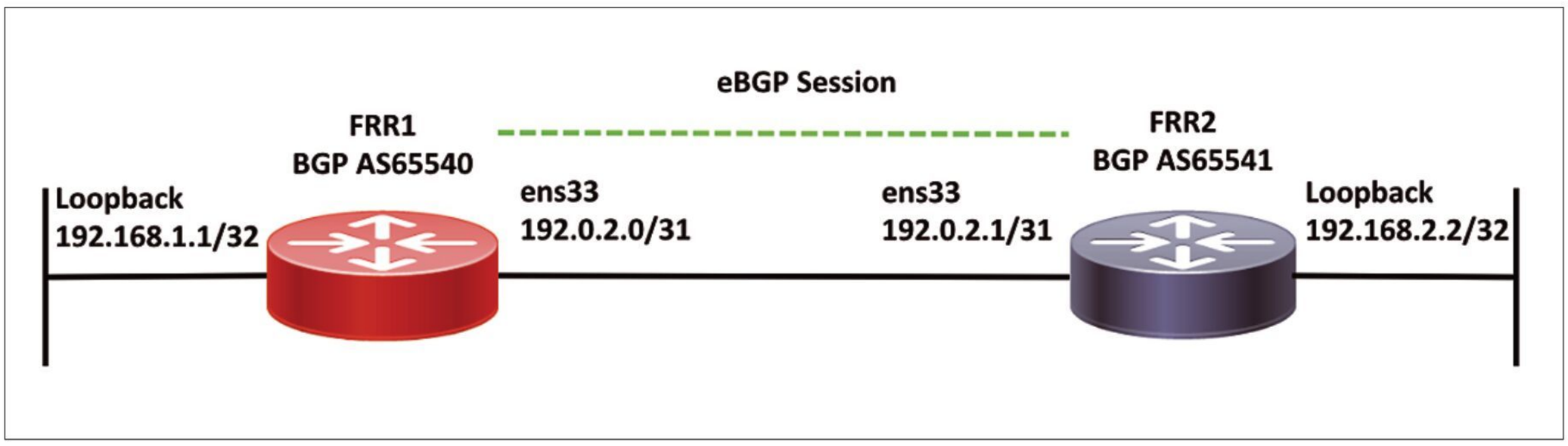


Figure 4: An external BGP (eBGP) session between the hosts FRR1 and FRR2. Each of the two routers announces its loopback IP address over BGP.

REAL SOLUTIONS *for* REAL NETWORKS

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

Improve your admin skills with practical articles on:

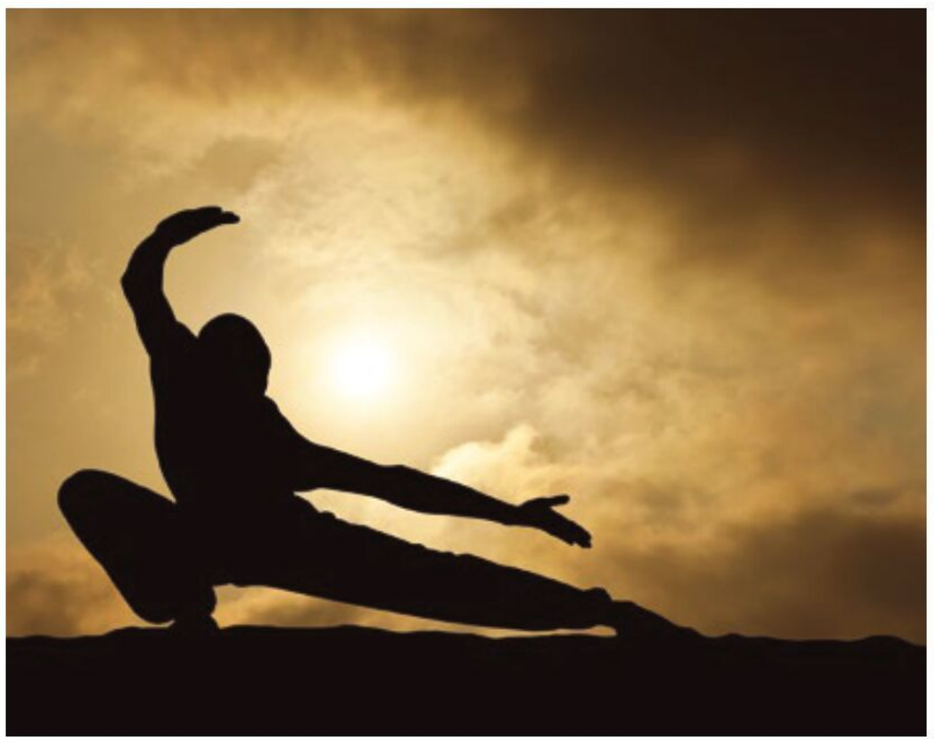
- Security
- Cloud computing
- DevOps
- HPC
- Storage and more!

GET IT FAST with a digital subscription!

6 issues per year!

ORDER NOW

shop.linuxnewmedia.com



Comparing startup times of Linux distros in the cloud

Cloud Startup

A cloud speed test pits Linux distributions against one another. By Federico Lucifredi

Launching instances on Amazon Elastic Compute Cloud (EC2) can be a lengthy affair for those used to instant gratification, taking entire minutes to complete. Historically, Windows instances have brought up the rear by taking considerably longer to initialize than Linux-based instances because of `sysprep` and the reboot that follows. The open source world is somewhat more streamlined, but there is still considerable variance between Linux distributions, with one to two minutes being a reasonable expectation of first availability.

One Ping Only, Please

Trying to connect multiple times as an instance boots is inelegant. Fortunately, you can use shell-fu to script your way out of this. The BSD version of `ping` [1], notably on macOS, includes a convenient "one ping only" option (`-o`) that I would like to think honors Sean Connery's famous quote in *Hunt for Red October*. The option terminates `ping` once the first reply is received. Like Connery's character, Captain Marko Ramius, you can use this option to ask for "one ping only, please" (Listing 1). Whether you launch the instance with the Amazon Web Services (AWS) Console [2] or the AWS command-line interface (CLI) [3], you can easily find the moment it becomes accessible; however, that is only part of the picture. An instance is only useful as the service it exists to provide can be reached by users. Completing service bootstrap is distinct from the point in time when the kernel network stack becomes active. The SSH daemon is present in nearly all official distribution images, and it provides a convenient testing standard for service availability. Perhaps less steeped in movie lore, but nonetheless equally effective, is this GNU-compatible one-liner, waiting in a loop for the SSH service to start up (Listing 2).

Listing 1: One Ping Only

```
$ ping -o 52.90.56.122; sleep 2; ssh ubuntu@52.90.56.122
PING 52.90.56.122 (52.90.56.122): 56 data bytes
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1

Request timeout for icmp_seq 2
64 bytes from 52.90.56.122: icmp_seq=3 ttl=48
time=40.492 ms
[ output truncated ]
Welcome to Ubuntu 20.04.1 LTS
(GNU/Linux 5.4.0-56-aws x86_64)
```

Listing 2: Waiting for SSH

```
$ until ssh ubuntu@52.90.56.122; do sleep 1; done
ssh: connect to host 52.90.56.122 port 22: Connection refused
ssh: connect to host 52.90.56.122 port 22: Connection refused
[ output truncated ]
Welcome to Ubuntu 20.04.1 LTS
(GNU/Linux 5.4.0-56-aws x86_64)
```

Booting Benchmarks

These two tests provide good insight into the initialization speed of stock operating system images, with the notable exception of Microsoft Windows, where SSH is absent (see the "Hibernate Support" box). FreeBSD Security Officer Emeritus and author of Tarsnap [4] Colin Percival has done all the hard work and released a new tool to time these events. The `ec2-boot-bench` tool [5] takes four measurements: (1) the duration of the RunInstances API call, (2) the time to transition from a pending to a running state, (3) how much longer the network stack took to initialize (RST TCP response), and (4) actual service availability (SYN + ACK TCP response). The `ec2-boot-bench` tool uses the SSH service's availability for the fourth metric, just as the shell script in Listing 2, but you can easily change the port with a recompile and use Colin's code to benchmark the specific service of your own Amazon Machine Image (AMI).

The first two metrics do not vary with distribution, and Colin himself has published [6] data indicating an average of 1.5 and 6.9 seconds, respectively; however, these numbers may vary with AWS region, availability zone, or instance type and are part of the delay experienced by the user. My own measurements in the `us-east-1` region are consistent with Colin's findings and average around 1.7 and 8.5 seconds, respectively (Figure 1). I used `T3.micro` instances for my tests instead of `C5.xlarge`, accounting for some of the variance between our results.

The tool itself is is still a rough instrument, with no distribution packages, man pages, or GitHub docs – and no way to check API errors. (I will be submitting a couple of patches for these before this article goes to press.) However, it does its job remarkably well already. Instances are benchmark-terminated, but if you Ctrl + C the tool, you might have stragglers left running: Remember to clean up after tests or your AWS bill will suffer.

Ubuntu's Lot

Ubuntu on 64-bit instances appears to have made consistent progress over the years, cutting startup time by nearly a third, from 14.6 to 9.9 seconds (32%), as shown in Table 1 for 10 sample averages after a warm-up run. The latest Ubuntu Server LTS release, Focal Fossa (20.04), can be expected to be ready for use in 15 seconds once the EC2 virtual machine has been initialized, compared with 18 seconds with Trusty

Lead Image © Lucy Baldwin, 123RF.com

```
root@focal:~# ./ec2-boot-bench --keys credentials --region us-east-1 --ami ami-05dc324761386f3a9 --itype t3.micro
RunInstances API call took: 1.802059 s
Moving from pending to running took: 10.335988 s
Moving from running to port closed took: 9.912614 s
Moving from port closed to port open took: 4.727956 s
root@focal:~# ./ec2-boot-bench --keys credentials --region us-east-1 --ami ami-05dc324761386f3a9 --itype t3.micro
RunInstances API call took: 1.487366 s
Moving from pending to running took: 7.287881 s
Moving from running to port closed took: 10.558349 s
Moving from port closed to port open took: 5.620171 s
root@focal:~# ./ec2-boot-bench --keys credentials --region us-east-1 --ami ami-05dc324761386f3a9 --itype t3.micro
RunInstances API call took: 1.652186 s
Moving from pending to running took: 9.647496 s
Moving from running to port closed took: 8.556996 s
Moving from port closed to port open took: 4.557058 s
root@focal:~# ./ec2-boot-bench --keys credentials --region us-east-1 --ami ami-05dc324761386f3a9 --itype t3.micro
RunInstances API call took: 1.660319 s
Moving from pending to running took: 8.610069 s
Moving from running to port closed took: 9.640627 s
Moving from port closed to port open took: 5.319883 s
root@focal:~#
```

Figure 1: A few runs with `ec2-boot-bench`. A shell loop is recommended.

Table 1: Ubuntu Startup Improvement

Distribution	AMI Tested	OS Boot (s)	Service Start (s)	Total (s)
Trusty (14.04)	ami-05dc324761386f3a9	14.581	3.170	17.75
Xenial (16.04)	ami-0133407e358cc1af0	9.300	7.001	16.301
Bionic (18.04)	ami-0186d369d234b536f	12.946	5.608	18.554
Focal (20.04)	ami-000b3a073fc20e415	9.934	4.938	14.872

Table 2: RHEL 8, Alpine, and Clear Linux

Distribution	AMI Tested	OS Boot (s)	Service Start (s)	Total (s)
Amazon Linux 2	ami-087c17d1fe0178315	10.480	2.307	12.787
Debian 10	ami-07d02ee1eeb0c996c	7.438	4.202	11.640
RHEL 8	ami-0b0af3577fe5e3532	15.497	53.561	69.058
Clear Linux 35000	ami-078bad975fd5aa9f3	2.811	0.025	2.836
Alpine Linux 3.14.2	ami-026896c9df188bad2	2.562	8.213	10.775

Tahr (14.04). The more than doubling of the SSH startup time between Trusty and Xenial may show how the transition to systemd [7] affected Ubuntu's overall optimization, or it may just be an artifact of startup ordering because the resulting total is not significantly affected.

Hibernate Support

Newly started instances may take longer to be ready for reasons extending beyond boot time. On the first run, many distributions run one-time instance setup code (e.g., AWS Linux and FreeBSD will install package updates if any are available). Moreover, services can have a much longer startup time than SSH. Fortunately, EC2 now supports Hibernate [8]. The ability to save the RAM state of a running system was developed for mobile computing, but it is increasingly becoming useful for quickly bringing cloud instances online with significant state. Hibernate works by dumping to Elastic Block Storage (EBS) the whole content of RAM and restoring it as needed, which is a much faster process than recreating all that state – and much cheaper to operate than keeping the instance running.

Fastest and Slowest

The slowest numbers were posted on RHEL 8 with nearly 70 seconds, to the consternation of some on the RHEL team. A network misconfiguration in the RHEL AMI used for testing caused a failed DNS lookup to hold up the `cloud-init` [9] configuration of userspace. The issue has already been remedied as I write and should no longer be there by the time you read this column [10] [11], as the RHEL AMI is updated. On the winning side, Intel's speedy Clear Linux spins up in under three seconds most of the time, although some unexplained outliers in the 10-second range occur with fast-starting distributions (Table 2). The service startup time is essentially zero, and not even the ever popular Alpine Linux can keep up, on account of its longer service initialization times. Debian and AWS Linux 2 have times in line with Ubuntu, so you can pretty much expect to have your newfangled `T3.micro` instance in about 20 seconds.

If you can spend a few minutes re-building the code to change some configuration, you now have a way to measure your AMI's startup times consistently that can be integrated into your continuous integration/continuous deployment (CI/CD) pipeline. As RHEL's example shows, it is a good idea. ■

Info

[1] ping(1) man page: [https://manpages.ubuntu.com/manpages/focal/en/man1/ping.1.html]

[2] AWS Management Console: [https://console.aws.amazon.com/]

[3] AWS CLI: [https://aws.amazon.com/cli/]

[4] "From bsdtar to tarsnap: Building an online backup service" by Colin Percival; September 28, 2013: [https://www.tarsnap.com/download/EuroBSDCon13.pdf]

[5] ec2-boot-bench: [https://github.com/cperciva/ec2-boot-bench]

[6] "EC2 boot time benchmarking" by Colin Percival; August 12, 2021: [https://www.daemonology.net/blog/2021-08-12-EC2-boot-time-benchmarking.html]

[7] systemd: [https://www.freedesktop.org/wiki/Software/systemd/]

[8] "New – Hybernate your EC2 instances" by Jeff Barr, *AWS News Blog*; November 28, 2018: [https://aws.amazon.com/blogs/aws/new-hibernate-your-ec2-instances/]

[9] cloud-init: [https://cloudinit.readthedocs.io/en/latest/]

[10] Bugzilla issue 1862930: [https://bugzilla.redhat.com/show_bug.cgi?id=1862930]

[11] Bugzilla issue 1994804: [https://bugzilla.redhat.com/show_bug.cgi?id=1994804]

The Author

Federico Lucifredi (@0xf2) is the Product Management Director for Ceph Storage at Red Hat and was formerly the Ubuntu Server Product Manager at Canonical and the Linux "Systems Management Czar" at SUSE. You can read more from him in the new O'Reilly title *AWS System Administration*.

Linux Magazine is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!

Expand your Linux skills with:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

Subscribe now!

shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY WITH OUR PDF EDITION

ADMIN Network & Security

NEWSSTAND

Order online:
bit.ly/ADMIN-Newsstand

ADMIN is your source for technical solutions to real-world problems. Every issue is packed with practical articles on the topics you need, such as: security, cloud computing, DevOps, HPC, storage, and more! Explore our full catalog of back issues for specific topics or to complete your collection.

#64/July/August 2021

Bare Metal Deployment

Setting up, automating, and managing bare metal deployments gets easier with the tools presented in this issue.

On the DVD: Rocky Linux 8.4 (Minimal Install)

#63/May/June 2021

Automation

This issue we are all about automation and configuration with some tools to lighten your load.

On the DVD: Ubuntu 21.04 Server

#62/March/April 2021

Lean Web Servers

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

On the DVD: Fedora 33

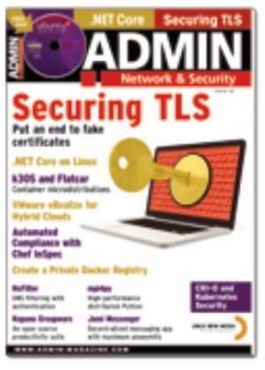
#61/January/February 2021

Secure Containers

Security is the watchword this issue, and we begin with eliminating container security concerns.

On the DVD: Clonezilla Live 2.7.0

#60/November/December 2020



Securing TLS

In this issue, we look at ASP.NET Core, a web-development framework that works across OS boundaries.

On the DVD: Ubuntu Server Edition 20.10

#59/September/October 2020

Custom MIBs

In this issue, learn how to create a Management Information Base module for hardware and software.

On the DVD: CentOS 8.2.2004

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

Erik Bärwaldt	12
Chris Binnie	48
Florian Frommherz	80, 84
Ken Hess	3
Jan Kappen	20
Christian Knermann	76
Jeff Layton	30
Martin Loschwitz	24, 52
Sandro Lucifora	36
Federico Lucifredi	94
Benjamin Pfister	88
Andreas Philipp	64
Dr. Holger Reibold	42
Ariane Rüdiger	58
Thorsten Scherf	10
Andreas Stolzenberger	70
Jack Wallen	8
Matthias Wübbeling	66, 68

Contact Info

Editor in Chief
Joe Casad, jcasad@linuxnewmedia.com

Managing Editors
Rita L Sooby, rsooby@linuxnewmedia.com
Lori White, lwhite@linuxnewmedia.com

Senior Editor
Ken Hess

Localization & Translation
Ian Travis

News Editor
Jack Wallen

Copy Editors
Amy Pettle, Aubrey Vaughn

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen, Illustration based on graphics by Somchai Suppalertporn, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Publisher
Brian Osborn

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2021 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by hofmann infocom GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN (ISSN 2045-0702) is published bimonthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. September/October 2021. Periodicals Postage paid at Lawrence, KS. Ride-Along Enclosed. POSTMASTER: Please send address changes to ADMIN, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Published in Europe by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Custom Tailored Suit

Select essential parts for your mobile computing needs

Processor
Intel or AMD CPU

Graphics Card
Integrated or dedicated

System Memory
Upgrade any time

Data Storage
Fast, large and switchable

Network
Wireless, Cellular or Gigabit LAN

100% Linux

5 Year Warranty

Lifetime Support

Built in Germany

German Privacy

Local Support

TUXEDO COMPUTERS

tuxedocomputers.com

A Linux Distribution For Professionals

openSUSE Leap 15.3

Leap is a desirable distribution for IT

professionals, entrepreneurs,

hobbyists, small businesses

and educational practitioners

get.opensuse.org