Container Databases

MS365 Desired State Configuration

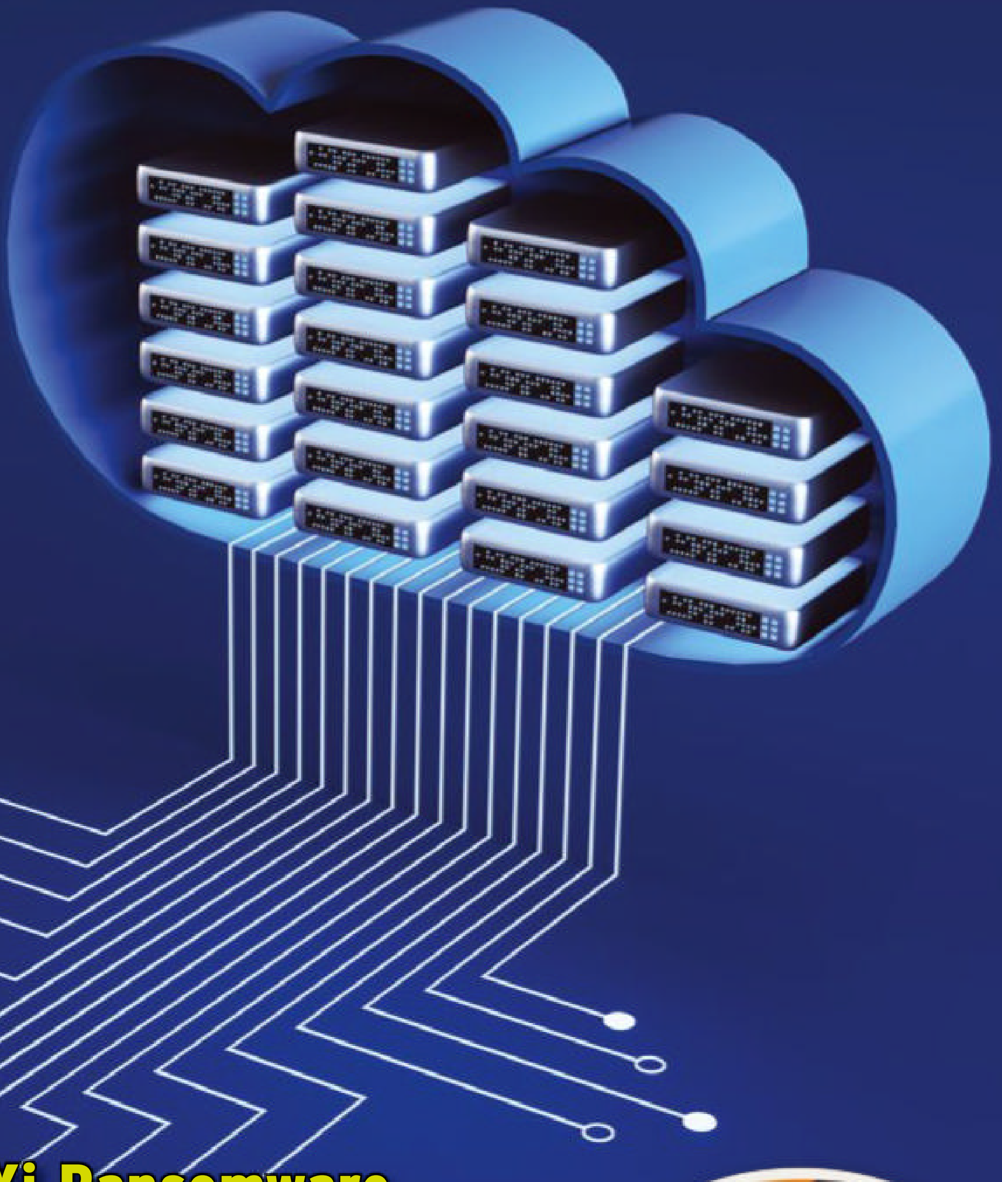
Binary Cyberattack Analysis

ADMIN Network & Security

ISSUE 73

Databases

In clouds, in containers, and scaled out



Kubernetes StatefulSet
Containerize and scale DBs

Purview
Data loss prevention

MS365 DSC
Desired state configuration

Scale Out Cloud DBs
YugabyteDB and Vitess

Win Server 2022 GPOs
Templates and recommendations

ESXi Ransomware Attacks
Now what do you do?

Acceptance Testing
Verify your configuration with goss

Binary Ninja
Analyze cyberattacks and malware

LINUX NEW MEDIA
The Pulse of Open Source

Offshoring and the Great Layoff

No, you're not mistaken and not imagining things. I've written about the Great Resignation, the Great Stay Put, and now the Great Layoff. In case you haven't noticed, the IT industry is in the midst of a massive contraction, in which thousands of workers have lost their jobs. According to one report, 37 percent of tech workers are in a state of worry about the looming possibility of getting laid off **[1]**.

It's a funny thing about the IT industry, and maybe it applies to all industries, but hiring in this field goes in cycles. Boom, bust, boom, bust, and so on. There's never a happy medium where growth, prosperity, and normal attrition exist. What doesn't make sense to me is that technology is still here, no matter what. The work still needs to be done. Services need maintenance, hardware needs to be replaced, and software needs to be compiled and installed, but today, we can do without 20 percent of the workforce we had yesterday?

When the economy slides into a downturn, companies look for ways to cut their budgets. In the high tech field, budgets are often cut through outsourcing to offshore providers. The perception among many executives is that offshoring provides an identical product for less overhead, but customers sometimes see it differently.

Some software companies try to outsource their customer support. Of course, this sometimes works adequately, but sometimes it doesn't work at all. I've been frustrated in the past navigating the language barrier with phone support techs who have only a superficial understanding of the product they are supposed to be supporting.

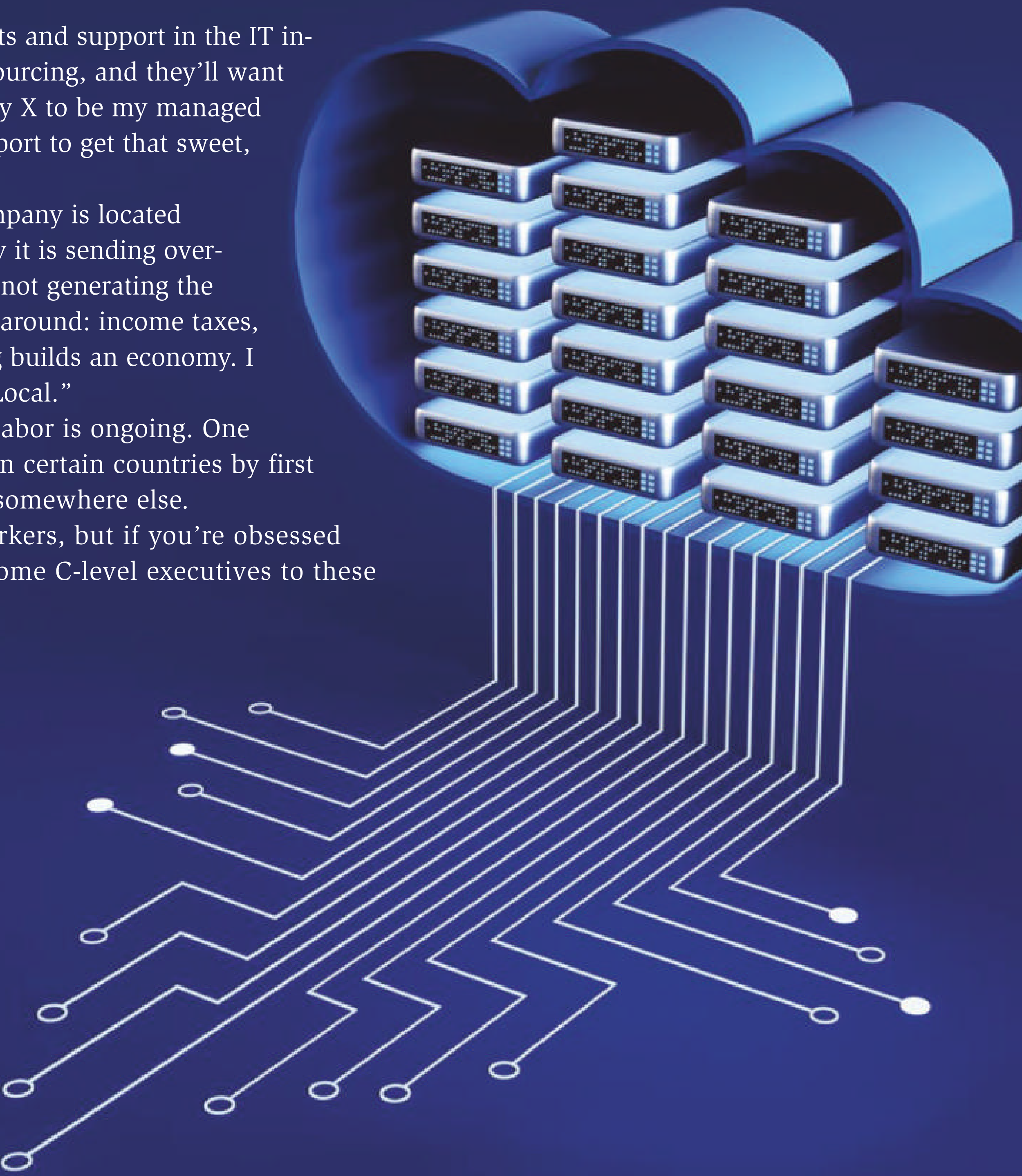
Another problem for companies that provide products and support in the IT industry is that your customers will know you're outsourcing, and they'll want a lower price. I don't blame them. If I hired Company X to be my managed service provider and knew they were offshoring support to get that sweet, cheap labor, I'd want to share in the lower cost too.

Another problem is that the country where your company is located won't receive any economic benefits from the money it is sending overseas. You might be getting cheaper labor, but you're not generating the economic activity needed to turn the local economy around: income taxes, product purchases, and local buying. Local spending builds an economy. I agree with the signs that say "Shop Local" or "Buy Local."

The sadder part of the story is the quest for cheap labor is ongoing. One company I worked for devastated local economies in certain countries by first offshoring to them and then finding cheaper labor somewhere else.

Offshoring is bad for the economy and bad for workers, but if you're obsessed with saving money overseas, how about moving some C-level executives to these cheap labor locations – mic drop.

Ken Hess • ADMIN Senior Editor



Info

[1] Roughly 1 in 3 Workers is Worried about a Layoff: [https://www.cnbc.com/2022/12/19/roughly-1-in-3-workers-is-worried-about-a-layoff.html]

ADMIN
Network & Security

@adminmagazine
@adminmag
ADMIN magazine
@adminmagazine

On the DVD

Manjaro 22.0 Gnome

This Linux-based operating system is fast, powerful, and user-friendly, has no licensing fees, respects your privacy, and is appropriate for beginners and advanced users alike. Manjaro is based on the independently developed Arch operating system. Use it for development, gaming, and 3D at the office or at home, on tablets, mobile devices, desktops, laptops, and boards.

News for Admins

Tech News

JUPITER Exascale Supercomputer Planned in Germany

The JUPITER supercomputer is set to become the first European exascale computing system, according to a recent agreement (**https://eurohpc-ju.europa.eu/one-step-closer-exascale-eurohpc-ju-and-forschungszentrum-julich-sign-hosting-agreement-exascale-2022-12-14_en**) between the European High Performance Computing Joint Undertaking (EuroHPC JU) and the Jülich Supercomputing Centre (JSC) in Germany.

The supercomputer will be the first European system "to surpass the threshold of one billion billion calculations per second" and will support "high-precision models of complex systems and help to solve key societal questions," the announcement states.

JUPITER (which stands for Joint Undertaking Pioneer for Innovative and Transformative Exascale Research) will also "be designed with a strong consideration for sustainability and eco-conscious supercomputing." The water-cooled system will be installed at the Forschungszentrum Jülich research institute in 2023 and will be operated by the JSC.

PineTab2 Linux Tablet Announced

PINE64 has announced details of the forthcoming PineTab2 Linux tablet. In a lengthy blog post (**https://www.pine64.org/2022/12/15/december-update-merry-christmas-and-happy-new-pinetab/**) recapping the past year's activity, Lukasz Erecinski explains the demise of the original PineTab and shares details of the new model.

The "PineTab2 is much more than a spec-bumped version of the original," he says, "it is a complete physical redesign: you're getting a metal chassis that is very sturdy while also being easy to disassemble for upgrades, maintenance, and repair."

To facilitate end-user serviceability, the PineTab2's parts are modular, easy to reach, and easy to replace, Erecinski says, noting that "the camera modules, the daughter-board, the battery, and USB keyboard connector can all be replaced in under 5 minutes."

Two PineTab2 versions — featuring 8GB RAM with 128GB flash and 4GB RAM with 64GB flash storage — will be available upon launch, which is expected "sometime after the Chinese New Year," Erecinski says. Pricing of the tablet has yet to be determined.

NIST Says Use of SHA-1 Algorithm No Longer Advisable

The National Institute of Standards and Technology (NIST) has deprecated use of the SHA-1 cryptographic algorithm.

According to the announcement (**https://www.nist.gov/news-events/news/2022/12/nist-retires-sha-1-cryptographic-algorithm**), the algorithm "has reached the end of its useful life," and NIST

DrupalCon
PITTSBURGH**2023**
5-8 JUNE

David L. Lawrence Convention Center
events.drupal.org/pittsburgh2023

Join us — in person! — for DrupalCon Pittsburgh, where the people who make amazing digital experiences possible come together to make them even better.

Register now to save **$100 USD**
Early bird rate ends on **2 April 2023**

recommends that IT professionals replace it with newer, more secure options. “We recommend that anyone relying on SHA-1 for security migrate to SHA-2 or SHA-3 as soon as possible,” said NIST computer scientist Chris Celi.

SHA-1, which stands for “secure hash algorithm,” has been in use since 1995 and was “one of the first widely used methods of protecting electronic information.” However, the announcement notes, “as attacks on SHA-1 in other applications have become increasingly severe, NIST will stop using SHA-1 in its last remaining specified protocols by Dec. 31, 2030.”

Attackers Use PRoot to Expand Scope of Linux Attacks

The Sysdig Threat Research Team (**https://sysdig.com/blog/proot-post-explotation-cryptomining/**) recently reported attackers “leveraging an open source tool called PRoot (**https://proot-me.github.io/)** to expand the scope of their operations to multiple Linux distributions.”

Typically, the researchers note, attacks are “limited by the varying configurations of each Linux distribution.” Using PRoot, however, “there is little regard or concern for the target’s architecture or distribution since the tool smooths out the attack struggles often associated with executable compatibility, environment setup, and malware and/or miner execution," Sysdig says.

Bill Toulas at Bleeping Computer (**https://www.bleepingcomputer.com/news/security/hackers-hijack-linux-devices-using-proot-isolated-filesystems/**) explains it this way: “Hackers are abusing the open source Linux PRoot utility in Bring Your Own Filesystem (BYOF) attacks to provide a consistent repository of malicious tools that work on many Linux distributions. A BYOF attack is when threat actors create a malicious filesystem on their own devices that contain a standard set of tools used to conduct attacks.”

A runtime detection layer, such as Falco (**https://www.cncf.io/projects/falco/**), can help observe this type of threat and reduce your risk of exploitation, Sysdig says.

WSL Version 1.0.0 Now Available

The Windows Subsystem for Linux (WSL) has dropped its “Preview” label and is now generally available as version 1.0.0 (**https://github.com/microsoft/WSL/releases/tag/1.0.0**) in the Microsoft Store.

According to the announcement (**https://devblogs.microsoft.com/commandline/the-windows-subsystem-for-linux-in-the-microsoft-store-is-now-generally-available-on-windows-10-and-11/**), the Store version will now be the default for new users who run `wsl --install`.

Existing WSL users can easily upgrade by running `wsl --update`. “Using the Store version of WSL allows you to get updates to WSL much faster compared to when it was a Windows component,” says Craig Loewen (**https://devblogs.microsoft.com/commandline/a-preview-of-wsl-in-the-microsoft-store-is-now-available/**). As of this release, WSL in the Store will be available on Windows 10 in addition to Windows 11.

Additionally, Loewen notes, “with the Store version of WSL, there are a lot of names to keep track of.” There are two types of WSL distros, Loewen explains: WSL 1 and WSL 2. “These matter for how your distro runs and behaves, as they have different architectures. WSL 2 distros have faster file system performance and use a real Linux kernel, but require virtualization.” Refer to the WSL version comparison (**https://learn.microsoft.com/windows/wsl/compare-versions**) to learn more.

Demand for Tech Talent Remains High

Despite industry layoffs, economic challenges, and reported hiring freezes, demand for tech talent remains high (**https://www.fosslife.org/open-source-jobs-report-talent-high-demand**), according to a recent survey from Dice, “with more than 3.8 million tech jobs posted in 2022 so far.” And, many technology professionals are still looking to change jobs.

More than half of respondents (52 percent) to Dice’s 2022 Tech Sentiment Report (**https://www.dice.com/recruiting/ebooks/dice-tech-sentiment-report/**) said they were likely to switch jobs in the next year (up from 44 percent last year).

Employer reputation and culture are among key considerations for job seekers, as “nearly 90 percent of tech professionals feel an employer’s brand is important when considering a new employer, and nearly 8 in 10 said they would not apply for a higher paying job at a company with a bad reputation," the Dice report says.

Respondents ranked the following factors as important in terms of employer brand:

- Culture (85%)
- Corporate reputation (66%)
- Product and purpose (62%)

The ability to work remotely is also important, Dice says, as 60 percent of respondents said "100% remote" was their most desired workplace setting compared to 53 percent in 2021.

Frontier Supercomputer Retains Lead on TOP500 List

The Frontier system (**https://www.olcf.ornl.gov/frontier/**) at Oak Ridge National Laboratory (ORNL) is the top-performing supercomputer in the world, according to the recently released 60th edition of the TOP500 list (**https://www.top500.org/news/ornls-exaflop-machine-frontier-keeps-top-spot-new-competitor-leonardo-breaks-the-top10/**).

Frontier is the first U.S. supercomputing system with a performance exceeding one EFlop/s, the report says. "Frontier is the clear winner of the race to exascale (**https://www.fosslife.org/power-exascale-computing**), and it will require a lot of work and innovation to knock it from the top spot."

The Top 5

- Frontier — The HPE Cray EX system is installed at ORNL in Tennessee, "where it is operated for the Department of Energy (DOE). It currently has achieved 1.102 EFlop/s using 8,730,112 cores," the report says.
- Fugaku — The Fugaku system installed at the RIKEN Center for Computational Science in Kobe, Japan remains at #2 with an HPL score of 0.442 EFlop/s.
- LUMI — This HPE Cray EX system at the EuroHPC center at CSC in Finland underwent a major upgrade to hold on to the #3 spot, with a score of 0.309 EFlop/s.
- Leonardo — At #4, the new Leonardo system installed at EuroHPC/CINECA in Bologna, Italy scored 0.174 EFlop/s with 1,463,616 cores.
- Summit — This IBM-built system, also at ORNL, rounds out the top five with a score of 148.8 PFlop/s on the HPL benchmark.

NetApp BlueXP Released

NetApp has announced general availability of BlueXP, "a unified control plane delivering a simple hybrid multi-cloud experience for storage and data services across on-premises and cloud environments."

BlueXP (**https://www.netapp.com/bluexp/**), which integrates all of NetApp's on-premises storage offerings into a single console, is part of the company's evolved cloud (**https://www.netapp.com/blog/cloud-evolution-better-hybrid-multicloud-experience/**) approach in which "cloud is fully integrated into your architecture and operations." NetApp BlueXP capabilities include:

- Unified storage management
- AIOps-driven health
- Cyber resilience
- Governance at a glance
- Seamless mobility

According to the announcement (**https://www.netapp.com/newsroom/press-releases/news-rel-20221101-530154/**), BlueXP is also the preferred method to manage NetApp's ONTAP data management software. And, the most recent release of ONTAP (**https://www.netapp.com/blog/new-ontap-innovations-november2022/**) features major changes, such as tamper-proof snapshots, reduced ransomware exposure, and hardened security incorporating Zero Trust principles.

Why databases are moving to the cloud

Flexibility

Demand for cloud databases continues to increase, not only because of better scalability and availability, but because of lower investment and operating costs. We'll look at some of the limitations. By Gregor Bauer

Opinions are divided into two major camps as to what constitutes a cloud database. The first camp claims they are databases provided by hyperscalers such as Amazon, Azure, and Google under the database as a service (DBaaS) model. The other, more common, view is that they can be any database that someone deploys within any cloud infrastructure, whether it be a private cloud, public cloud, hybrid cloud, multicloud, or hosted DBaaS. It's just a matter of where the data is stored and how it is accessed. In the case of on-premises databases, access is over the corporate network (and VPN, if necessary) to internal database servers, whereas cloud databases will rely on Internet connections to external (in some cases internal) cloud servers.

Deployment Models

Generally speaking, cloud databases have three popular deployment models. In the first case, the cloud database is operated in-house, either in the company's own data cloud or a public cloud. In the second case, a hoster provides DBaaS. The third variant involves consuming the database as a managed service.

In the first scenario, the cloud database runs on an internal or external virtual machine (VM), but operations are handled by the company's own database administrators. In contrast, DBaaS involves the hoster or provider provisioning the database and hardware infrastructure as part of a subscription agreement, depending on the service level agreement (SLA). You are still responsible for ongoing operation of the database, though. If you want the provider to handle database operations, taking the load off the internal IT administrator's shoulders, the third option is a database as a managed service, or managed hosting.

The deployment model has no influence on the database type. Both relational SQL and document-oriented NoSQL databases can be operated as cloud databases. The different levels of flexibility make an important difference, though. SQL databases rely on fixed table structures. Changes to this row-column scheme cause major overhead, if they are possible at all. The younger, cloud-native NoSQL databases, on the other hand, have a far more flexible data model based on JSON documents instead of tables that can be changed faster and with less overhead.

When people talk about cloud databases, they often overlook the fact that using them only really makes sense if both the applications and the data are migrated to the cloud.

Photo by Wesley Tingey on Unsplash

Put simply, a cloud database only makes sense if the cloud applications also use the data stored there. NoSQL databases cope better with the greater volatility of cloud apps than traditional SQL databases. Their operating principle dates back to a time when clouds were still the exclusive domain of meteorologists.

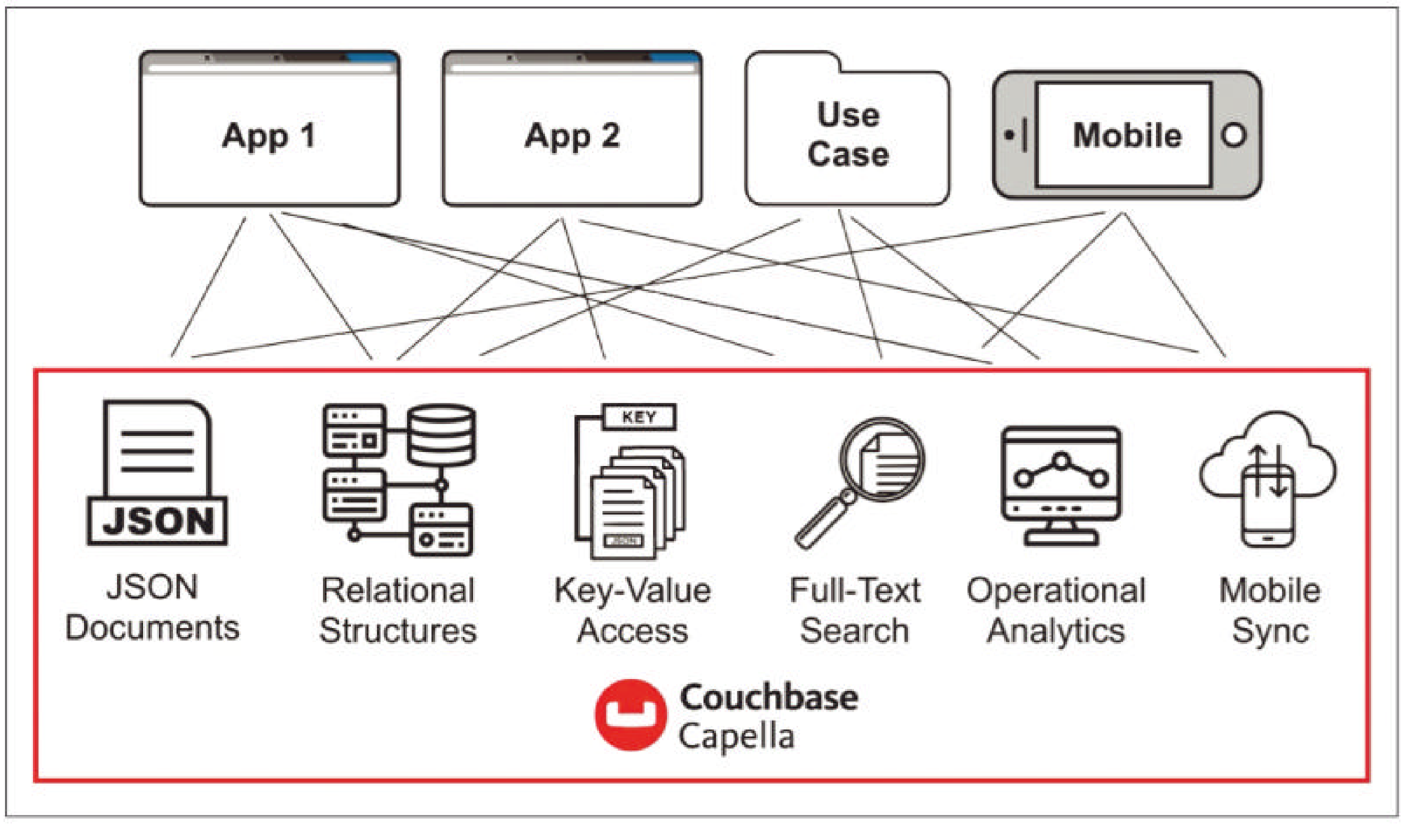


Figure 1: Cloud databases are (almost) unbeatable for scalability and availability (courtesy of Couchbase [1]).

Advantages

Three major features argue for the use of cloud databases: scalability (Figure 1), fast and secure availability, and (purportedly) lower costs.

Extending databases that you run in your own IT infrastructure requires investing a large amount of cash in additional hardware and having at least the same amount of patience until everything is installed and running. Cloud databases, on the other hand, scale more-or-less in real time and in both directions. They are highly flexible and can adapt quickly to changing requirements, which means a sales slump can be cushioned just as quickly as a sudden boom in demand, without expensive hardware lying idle or having to be ordered hastily. Moreover, you can do without tricks like thin provisioning, which translates to rapid availability of database resources that depend on current demand. Long-term resource planning and preemptive, and costly, hardware (over)dimensioning are a thing of the past.

The demands on the IT infrastructure and administration drop at the same time. Depending on the design, expensive hardware installations can be almost completely eliminated, and the space required for them is reduced accordingly. On top of this, managed hosting of cloud databases means you no longer need a dedicated administrator to operate, support, and maintain the database, because it is handled by the database operator. Given the tight staffing situation in this area, this consideration is quite significant.

One important advantage is simple handling of cloud databases, which can be controlled and used over a web interface.

On the other hand, you do need to invest in fast and stable broadband connections when operating cloud databases. The costs for this are usually worthwhile compared with the potential savings as just described. However, some caution is advisable when it comes to the cost of database usage. In many cases, only the standard hyperscaler functions are genuinely inexpensive. High surcharges will often apply for special cases or individual requirements.

The pay-per-use or pay-as-you-go options, wherein only the database resources you actually use and consume are billed, is attractive. Of note is that investments in hardware and software in many countries is depreciated over several months or years for tax purposes, whereas subscription costs for cloud databases generally can be recovered directly as tax-reducing operating costs.

Migration and Security

Anyone who wants to transfer their data – or even just part of it – to the cloud faces the often underestimated problem of cloud migration. The greater the volume of data to be migrated to the cloud, the higher the costs. It makes sense to clarify in advance what data you still need and what is dispensable or even superfluous and, in turn, a potential security risk. Other security and compliance issues are related to confidential information, such as the personal data of your customers, contractors, and employees.

Data cleansing and analysis is a good idea before migrating to the cloud. Data cleansing is primarily about removing redundant, obsolete, and trivial (ROT) data. The aim of data analysis is to clarify which of the remaining data should be allowed to move to the cloud and which should remain in-house for security reasons or issues with particularly sensitive intellectual property. In the course of cloud preparation, you might want to put an end to an undesirable spread of databases in the company. Before moving to the cloud, weigh up the extent to which it makes sense to consolidate databases into a unified database management system (DBMS).

When it comes to security, cloud providers go the extra mile and, in their own interest, ensure a general level of security that is often higher than the individually installed protection mechanisms in private clouds or in-house data centers. Even database-critical issues such as failover and backup are handled by the provider in the case of DBaaS and managed services. Fail-safes and high availability of the cloud databases are implemented in the availability zones in which database clusters are formed. To do this, the cloud database needs appropriate replication mechanisms (e.g., cross-data center replication (XDCR)) that constantly replicate between clusters. Alternatively, you can fall back on cloud resources with a redundant design in the background. One commonly reported problem with cloud databases is speed, which does not primarily depend on the database itself; instead, it is a function of the general performance of the virtualized environment. The bandwidth is high, but generally does not achieve the performance of a bare-metal installation, even in the best of clouds. In contrast, latency times in cloud databases, attributable to the centralized cloud architecture, can be critical. All data first needs be transferred to one or more data centers – usually far away – and then returned from there. This process takes more time than accessing or transferring data in internal database servers. This limitation above all affects real-time applications, such as those typical for scenarios in the Internet of Things (IoT) environment or in the Industrial Internet of Things (IIoT).

Edge Computing

Cloud databases therefore only play a subordinate role in edge computing. An additional limiting dependency is the constant need for online connections. Even a brief loss of connection could bring entire production lines to a standstill. Resilience looks different. Therefore, sensor data is processed in real time directly on-site with edge databases, which have reduced functionality tailored to edge needs (e.g., analytical capabilities to evaluate peripheral data). On the other hand, edge databases do not need to be particularly powerful or highly scalable, nor do they need full-text search capability.
Cloud databases enter the scene again for consolidation downstream and to store data no longer needed at the edge. For this to happen, you need a database that maps the decentralized architecture of edge computing and supports its special features. The pared-down edge database and the full-format database in the cloud need to understand each other and complement each other in terms of functionality.

The Cloud Is Everywhere

That said, cloud databases can be useful in virtually any conceivable deployment scenario. In terms of functionality, they are by no means inferior to in-house databases. They come in the SQL and NoSQL database flavors that harmonize well with virtualized and containerized environments (keyword: cloud native). They typically tie up fewer resources and are readily available, highly scalable, and usually less expensive to operate. ■

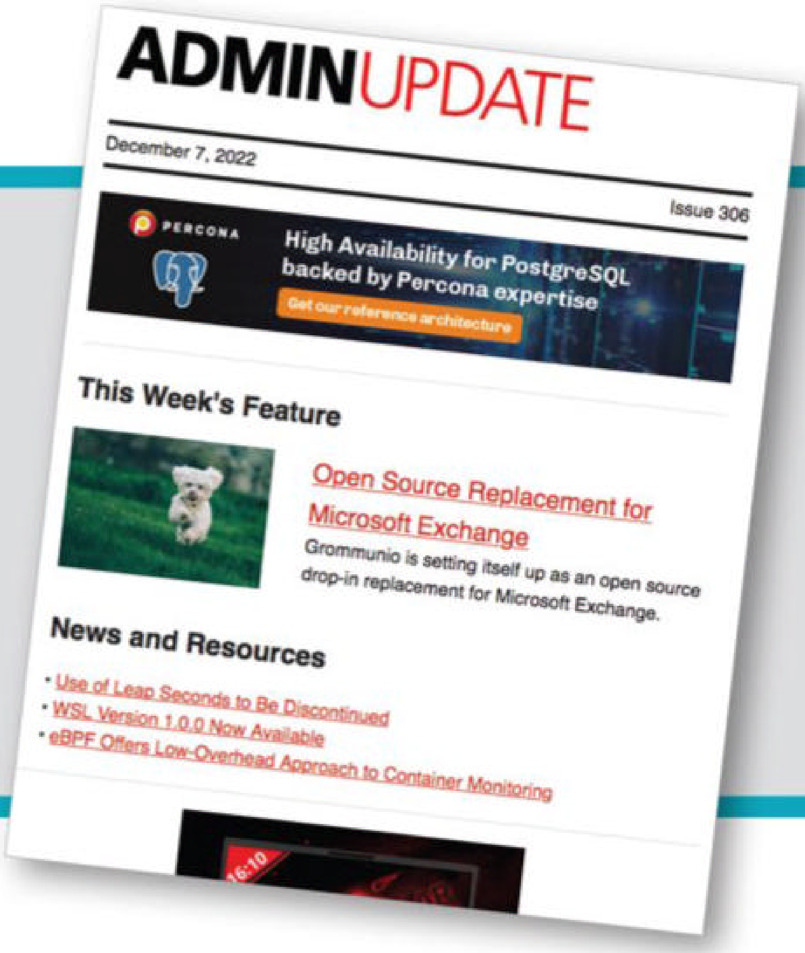
Info

[1] Couchbase: [https://www.couchbase.com/products/capella]

The Author

Gregor Bauer is Senior Solutions Engineer Central Europe and a technical team leader and presales engineer at Couchbase, a US enterprise database provider. He focuses on NoSQL databases, web technologies, device management, edge computing, and mobile applications.

■



ADMINUPDATE

December 7, 2022

Issue 306

High Availability for PostgreSQL backed by Percona expertise

Get our reference architecture

This Week's Feature

Open Source Replacement for Microsoft Exchange

Grommunio is setting itself up as an open source drop-in replacement for Microsoft Exchange.

News and Resources

- Use of Leap Seconds to Be Discontinued
- WSL Version 1.0.0 Now Available
- eBPF Offers Low-Overhead Approach to Container Monitoring

Databases in the Google Cloud

Spoiled for Choice

The Google Cloud Platform offers a wide range of different databases for various purposes. By Guido Söldner

Even in the cloud age, a classic IT task is operating databases. In addition to SQL relational databases, non-relational databases are making inroads into the corporate landscape. Today, companies likely use more than just one kind of database.

In principle, cloud databases can be operated in the same way as databases in an organization's data center. The admin team simply installs the database as a virtual machine. Administration in this case is no different from local administration. Aspects such as high availability, patching, backup and restore, disaster recovery, and scaling remain the administrator's responsibility.

To make it as easy as possible for companies to operate databases, cloud providers also offer databases as managed services, which means the customer is, by and large, only responsible for the application and the database logic but does not need to worry about operations issues.

Choosing a Database

Google is one of the cloud pioneers: Google Cloud Platform (GCP) is one of the three largest cloud providers alongside Amazon Web Services (AWS) and Microsoft Azure. As such, it offers a comprehensive portfolio of database technologies, but choosing the right database is not always easy and needs to be carefully considered. A catalog of criteria can help:

- The access method is important: Legacy applications such as billing software use SQL to access the database. The database needs to have an appropriate relational schema, support highly available operation, and, of course, maintain a consistent dataset at all times.
- Access techniques can differ: Relational databases rely on SQL commands (e.g., `SELECT`, `INSERT`, `UPDATE`, `DELETE`) and usually support transactions. In contrast, NoSQL databases often use a REST interface, and object stores use files in the form of objects. Alternatively, a database can also support streaming data and batch operations.
- The storage volume is relevant: Relational databases are usually in the range of a few terabytes, whereas data warehouse systems and NoSQL databases break through the petabyte barrier, with no conceptual size restrictions for object stores.

Cloud storage is not a database in the purest sense, but because it is used as a storage back end in many NoSQL and big data scenarios, it still makes sense to look at it as such. Cloud storage keeps data in buckets. If you compare this to a filesystem, all the objects are stored in a flat namespace with no folders. Objects are replicated and are therefore always stored with high availability. Thanks to replication, durability is very high: up to 99.9999999 percent. The object store is accessed over REST or API interfaces. As a result, and because of the architecture, latencies when retrieving objects are fairly high – almost always in the double-digit millisecond range. Therefore, object storage is generally used for unstructured data.

Cloud SQL Databases

The Cloud SQL service offers both MySQL, PostgreSQL, and Microsoft SQL Server as managed services. Google operates the database, which means that patches and updates are Google's responsibility. Backup and restore tasks can be easily configured when you create the database – but you still have to set up user access yourself.

Creating a database is relatively easy in the graphical user interface (Figure 1). To do this, click on *Databases* |

Photo by Jon Parry on Unsplash

SQL in the menu at top left. If no database exists as yet, you can continue directly with the wizard and select *Create Instance*. MySQL versions 8.0, 5.7, and 5.8 and PostgreSQL versions 9.6, 10, 11, 12, 13, and 14 are now available for selection. Alternatively, a managed database instance with Microsoft SQL Server versions 2019 and 2017 is available.

A unique name is required to create the database. It's a good idea to use random characters, if possible, because after you delete an instance, you cannot reuse the name for seven days. The region is also important. If you want to provide the database in Frankfurt, Germany, for example, select *europe-west3* as the region. If you want the database to be highly available, use the *Multiple zones (Highly available)* option under *Zonal Availability*. You can specify the *Primary Zone* and the *Secondary Zone*, as well. High availability incurs higher costs because a standby server is required to ensure that failover is possible at all times.

When it comes to hardware sizing, you have a wide choice; you can run a small test database with PostgreSQL or MySQL on a shared core with a virtual (v)CPU and 614MB of RAM. On the other side of the scale, the maximum limit is 96 vCPUs and 624GB of RAM (at the time of writing). Hard disk storage is available in the form of hard drives (HDDs) and solid-state disks (SSDs).

Size also determines disk performance; for each gigabyte of storage space you have 30 input/output operations per second (IOPS) for reading from and writing to SSDs, but these values are lower for writing on legacy magnetic drives. The data throughput is similar and also linked to size.

By default, databases are given public IP addresses. If this setup is undesirable for security reasons, deploy the database to a virtual private cloud (VPC) with private IP addresses only. Both normal and shared VPCs are supported.

In the backup options, you can set a time window in which an automatic backup will take place and enable point-in-time recovery to restore an arbitrary database status in a granular way in the event of a problem.

By default, Cloud SQL stores the last seven backups. You also configure the backup region at this point. In addition to selecting a dedicated region, multiregional backup is available. For example, you can choose to create backups in various regions within the European Union.

Because Cloud SQL is a managed database service, you do not need to set time windows for patching and updating; instead, this process is handled, like the other options, in the graphical user interface, where you can configure database-specific options, as well. For PostgreSQL, this process includes auditing queries or the number of possible concurrent connections, for example. One item of interest to developers is the *Insights* option, which can be used to identify performance problems at database runtime.

Of course, Cloud SQL databases can be created with tools other than the graphical user interface. The command-line interface or Terraform are especially popular. A simple call to create a MySQL database would be:

```
gcloud sql instances create myinstance ➘
  --database-version=MYSQL_8_0 ➘
  --cpu=2 ➘
  --memory=7680MB ➘
  --region=europe-west3
```

The corresponding Terraform resource definition,

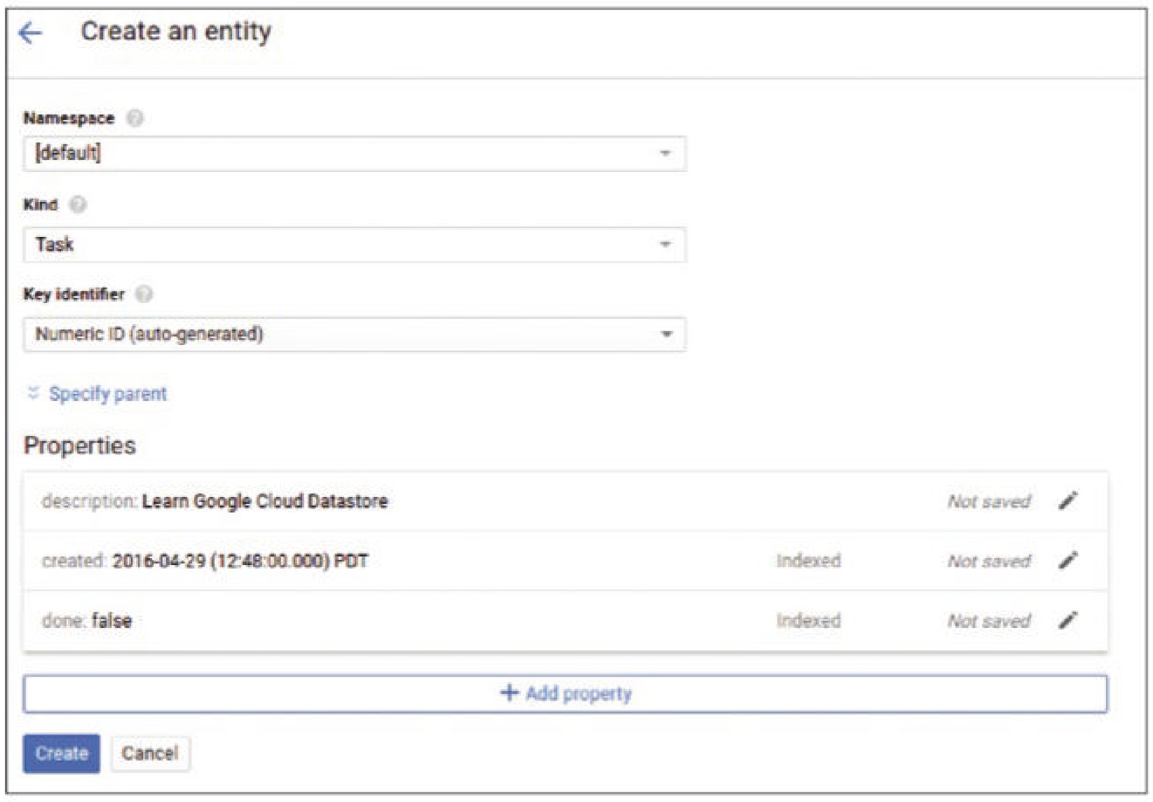
```
resource "google_sql_database_instance" ➘
  "instance" {
    name = "mysql-instance"
    region = "us-central1"
    database_version = "MYSQL_8_0"
    settings {
      tier = "db-n1-standard-2"
    }
  deletion_protection = "true"
}
```

is not difficult either.

Cloud Spanner Worldwide

Cloud Spanner is an interesting choice for companies that want the benefits of a relational database and the scale of a NoSQL database, with capacities in the petabyte range and an operational scope across multiple regions at the same time. This feature is especially interesting for applications that tolerate very little or no downtime. Unlike classic relational databases, Cloud Spanner scales horizontally rather than vertically. Replication takes place automatically, and the database remains in a globally consistent state at all times.

Cloud Spanner supports a wide range of functions of classic databases, including system compatibility with ANSI 2021 standard SQL and support



Namespace: [default]
Kind: Task
Key identifier: Numeric ID (auto-generated)

description: Learn Google Cloud Datastore — Not saved
created: 2016-04-29 (12:48:00.000) PDT — Indexed — Not saved
done: false — Indexed — Not saved

Figure 1: Importing data into Cloud SQL is easy in the GUI.

for transactions. Like other databases in the Google Cloud, Cloud Spanner is integrated into the GCP ecosystem and offers encryption with key management service (KMS), audit logging, and integration with identity and access management. In addition to typical interfaces such as Java Database Connectivity (JDBC), client libraries exist that support access in the Java, Python, and Node.js programming languages.

When migrating an existing application, you need to keep in mind that not all features of classic relational databases are supported. For example, sequences cannot be used for primary keys. Sequences make unique lines with numeric values. However, because Cloud Spanner distributes the data to nodes on the basis of the primary key, and the primary key must be alphanumeric, sequences are ruled out.

Another downer is the lack of support for triggers, stored procedures, and user-defined functions (UDFs). For this reason, Cloud Spanner might not be suitable for your application. However, a number of criteria still make Cloud Spanner interesting, including applications that have become too large for a single database or that only run at all as classic databases after optimization measures such as database sharding. Transactional consistency or global availability requirements also favor Cloud Spanner.

Listing 1: Accessing Cloud Firestore with Node.js

```
async function addTask(description) {
  const taskKey = datastore.key('Task');
  const entity = {
    key: taskKey,
    data: [
      {
        name: 'created',
        value: new Date().toJSON(),
      },
      {
        name: 'description',
        value: description,
        excludeFromIndexes: true,
      },
      {
        name: 'done',
        value: false,
      },
    ],
  };
  try {
    await datastore.save(entity);
    console.log(`Task ${taskKey.id} created
                successfully.`);
  } catch (err) {
  console.error('ERROR:', err);
  }
}
```

Cloud Firestore

In comparison, Cloud Firestore resides in the NoSQL camp and acts as a document database. In contrast to the databases already described, Cloud Firestore is completely serverless and has no management overhead. With this database, you only pay for what you use and don't have to worry about rightsizing or underutilized resources. The pricing model envisages users paying a certain sum for the volume of data stored (in gigabytes) and for the number of read, delete, and write operations.

Google sees Cloud Firestore's main area of application in mobile, web, and Internet of Things (IoT) databases. Offline mode for smartphones or tablets is particularly interesting. As soon as the user is back online, they can synchronize with the Firebase back end in the Google Cloud with an integrated live synchronization tool. If your users work with multiple devices, real-time updates are important because they support synchronization of data across different connected devices. One benefit for developers is that Cloud Firestore, a highly scalable NoSQL database for applications, supports transactions.

Cloud Firestore also has some technical drawbacks: It is not designed for use cases that require high write performance. If you put a great deal of emphasis on analytics, Cloud Firestore will not necessarily be your first choice; in this case, you should take a closer look at Cloud SQL or Cloud Bigtable. Normal read processes, on the other hand, are straightforward and quick and can be further optimized with the Memcached caching service.

Much like Cloud Spanner, Cloud Firestore provides automatic replication for multiple regions while ensuring strict data consistency. Google states the availability of the data as 99.999 percent. The volume of data that can be stored in Cloud Firestore is broadly similar to Cloud SQL and is in the terabyte range. Saving data in Cloud Firestore is a quick process: Unlike a relational database, you do not need to create a schema, so storage can begin immediately.

Applications access the database over the client API. As with Google's other products, client libraries exist in a variety of programming languages. Listing 1 shows how this works with Node.js. The listing creates a key and an entity. The key `Task` references the key and contains a data element with the fields `name`, `description`, and `done`. It then interacts with the Cloud Firestore API and stores the entity.

In the same way –

```
const query = datastore
  .createQuery('Task')
  .filter('done', '=', false)
  .filter('priority', '>=', 4)
  .order('priority', {
    descending: true,
  });
```

– queries can be implemented with the client library.

Low Latency with Cloud Bigtable

Cloud Bigtable is a fully managed NoSQL database that originates from the big data environment where you can store data in the petabyte range for low-latency access. The underlying storage engine takes care of scaling, lifting the need to optimize off the admin's shoulders.

This kind of performance is achieved by Cloud Bigtable's internal separation of processing and storage. Figure 2 shows how clients access nodes. A Cloud Bigtable node is responsible for a subset of data. If more data is available or more performance is desired, you can increase the number of nodes. HDDs or SSDs are available as the storage types. HDDs are less expensive in principle, but they suffer from significantly higher access latency, especially in random read processes. In contrast, they perform well in batch analytics applications such as machine learning or data analytics involving a large amount of sequential reading. In contrast, SSDs are recommended for real-time access, such as ad serving or recommendation apps. Storage is billed by data volume (in gigabytes).

The number of Cloud Bigtable nodes can either be set manually or managed automatically, which allows you to avoid unforeseen costs – although you do need to keep an eye on the utilization status of the Cloud Bigtable cluster – or with autoscaling, which automatically sets the number of nodes and adds or removes them depending on the workload. For a highly available cluster, you can set up a second cluster and replication.

The main use cases for Cloud Bigtable are streaming and batch operations, as well as applications that require low latency when accessing data. Cloud Bigtable is usually accessed over the Apache HBase interface, which avoids vendor lock-in because access is not handled by a proprietary interface.

Data is stored as key-value pairs in tables. Each row describes an entity and each column has an individual value. Columns that belong together can be grouped into families. Indexing goes by the line key.

Data Warehouse with BigQuery

BigQuery is a fully managed, serverless data warehouse. As such, it scales easily into the petabyte range – without operational overhead. Google claims that the total cost of ownership is 26 to 34 percent cheaper compared with competitors. A special feature that BigQuery has had for some time is machine learning functionality. In particular, it provides an integrated platform for classification and regression models that is based on structured data. Machine learning functionalities such as training a model or executing predictions are easy to learn and can be handled with normal SQL syntax.

Typical fields of application are recommendation systems or regression analyses. With BigQuery Omni, Google has opened up to other cloud providers and supports querying data stored in AWS or Azure, for example. Like Microsoft with its Power BI, Google also offers a fast in-memory analytics service that lets users analyze large volumes of data with sub-second response times. Additionally, BigQuery also acts as a system for analyzing spatial-geographical information.

In addition to the option of local data, you can load data into BigQuery from various sources, such as cloud storage buckets or Google Drive data. Special connectors exist for a number of third-party systems that can be addressed with the BigQuery data service. If you need your own special logic when transforming the input data, the Google Cloud Data services Fusion and Dataflow are there to help you. A number of other partner integrations exist, as well. Data is stored in tables that can be combined to create datasets and assigned the appropriate authorizations. The easiest way to work with BigQuery is through the Google Cloud console (Figure 3).

Conclusions

The Google Cloud platform offers a wide range of products for operating databases in the cloud. Depending on the deployment scenario, you can choose whether low latency, storing very large data volumes, or even machine learning is the focus. Moreover, you can specify whether classic SQL is required or whether you are happy with NoSQL services. ■

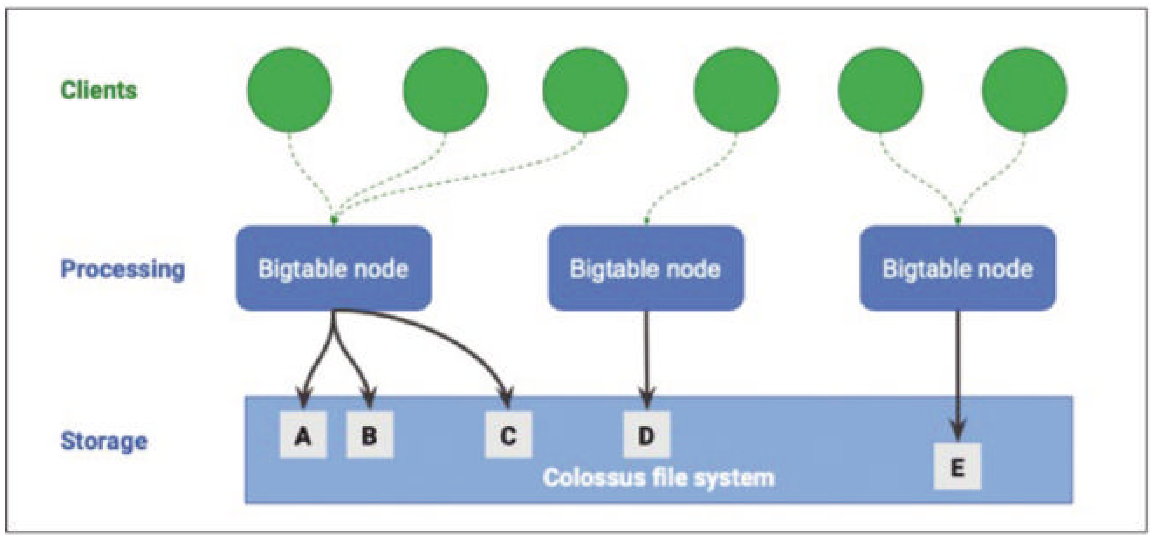
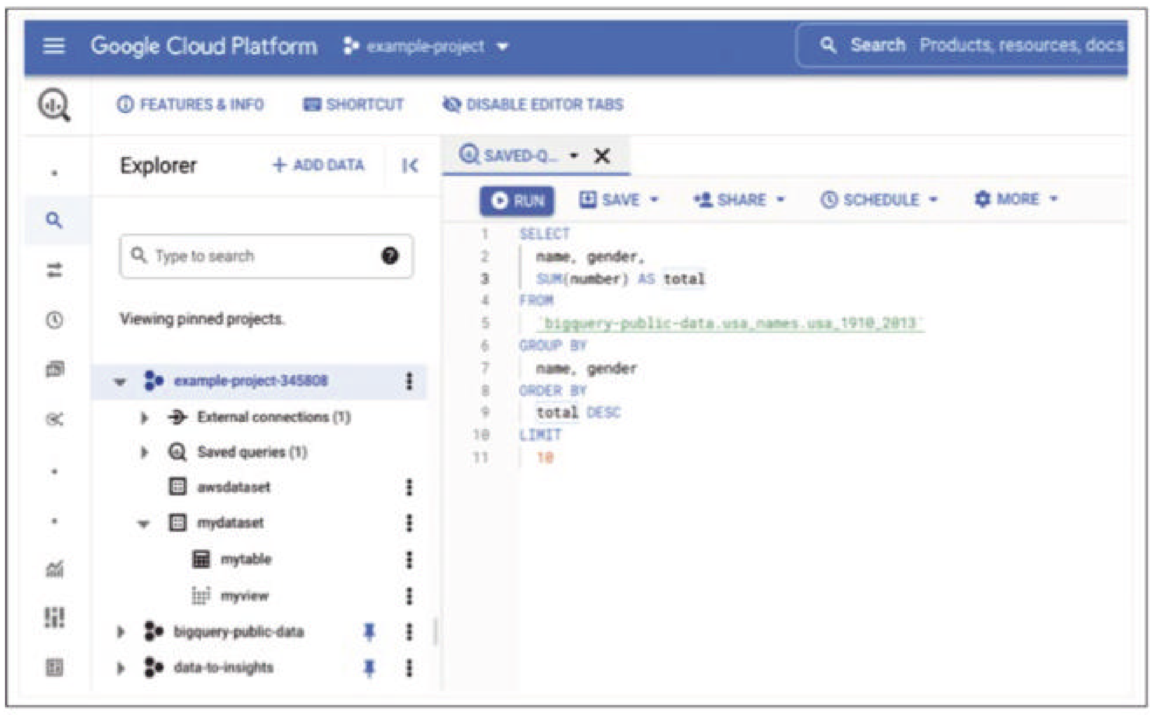


Figure 2: Isolation of processing from storage in Cloud Bigtable.



```
SELECT
  name, gender,
  SUM(number) AS total
FROM
  `bigquery-public-data.usa_names.usa_1910_2013`
GROUP BY
  name, gender
ORDER BY
  total DESC
LIMIT
  10
```

Figure 3: BigQuery can be managed in the Google Cloud console.

Kubernetes StatefulSet

Hide and Seek

Legacy databases are regarded as stateful applications and, theoretically, not a good fit for containers. We reveal how classic SQL can still work well on Kubernetes and the database options available to SMEs for scale-out environments. By Andreas Stolzenberger

Many enterprises are looking to migrate legacy monolithic applications to scale-out architectures with containers. Unfortunately, this step turns out to be far more complicated than many IT departments would like. A cloud-native architecture requires that various components of the applications communicate with each other on a message bus to allow for distribution and scaling to different nodes. The big challenge for the developer is: Where does the application back up its data files?

Up to now, SQL databases have mainly handled this task. However, very few SQL implementations can run as clusters in a scale-out architecture. Application developers now have a choice: Either run the existing database technology in a container or switch to a scale-out database. However, this choice is only available to large companies with their own software developers. Small and medium-sized enterprises (SMEs), on the other hand, tend to work with off-the-shelf tools and are forced to use one of the databases that their application supports.

Despite the steadily increasing number of NoSQL databases for scale-out architectures, many web applications still only support one or more classic SQL databases. The most popular candidates include MariaDB (MySQL), PostgreSQL, and Microsoft SQL. In this article, I discuss how to run these SQL classics reliably in container environments with Docker, Podman, or Kubernetes and present a few interesting NoSQL approaches.

Trusted Sources

In a genuine scale-out application, the failure of a single container should not affect the application as such. Therefore, early implementations had no function to provide persistent (i.e., non-volatile) mass storage for a single container. Unfortunately, the SQL classics run the database server in a single container: If the container fails, the database is lost, and it doesn't help that the container platform can restart the container in a fraction of a second.

The platform therefore needs to provide the container a reliable mass storage device on which it can store its data and that will survive the demise of a container. This storage is then usually mounted as an overlay on the container's filesystem at a specified location (e.g., `/var/lib/mysql`), and the container template for the respective database needs to provide the appropriate logic to evaluate correctly the content of the persistent storage.

If a container starts with empty persistent storage (and only then), the container template's housekeeping system has to create the appropriate directory structure. If the container starts up with populated persistent storage, the application reads the data and configuration from the overlay. In the process, the logic also needs to start update or repair processes in case of doubt. If a database container crashes and corrupts the database files in the process, the container needs to check for consistency on restarting.

On the other hand, you could stop a container with database version *x* and start a container with version *x* + 1. This container then checks the dataset for its version and carries out an update without corrupting any data.

You have to be very careful where you pick up your container templates and just as careful if you build them yourself. Most database server vendors offer an official container template through one of the popular Docker or Kubernetes registries and on GitHub. The documentation specifies exactly what is included in the template and how it handles update and recovery scenarios. If you want to build your own template, check the GitHub repositories of the official builds to see what their `entrypoint.sh` scripts do before starting the database.

Kubernetes Test with Microshift

The simple Microshift, a variety of OpenShift-Kubernetes that targets the niche between lean Linux edge devices and OpenShift-Kubernetes edge clusters, is useful as a Kubernetes environment for testing and development. To proceed, simply set up a Red Hat Enterprise Linux 8 (RHEL 8), CentOS Stream 8, or Fedora 35 VM with a minimal setup of two to four virtual (v)CPUs and 4 to 8GB of RAM and turn off the firewall. Then, you just need to type a few lines:

```
dnf module enable -y cri-o:1.21
dnf install -y cri-o cri-tools
systemctl enable crio -now
dnf copr enable -y @redhat-et/microshift
dnf install -y microshift
systemctl enable microshift -now
```

While the service starts and fetches the required container images from the Internet, you can download the `oc` and `kubectl` clients:

```
curl -O https://mirror.openshift.com/pub/open-shift-v4/$(uname -m)/clients/ocp/stable/
  openshift-client-linux.tar.gz
tar -xf openshift-client-linux.tar.gz -C /usr/local/bin oc kubectl
```

To check whether the back-end services are running, the command

```
crictl ps
```

should return a list of containers. If this is the case, copy the configuration file to your home directory:

```
mkdir ~/.kube
cat /var/lib/microshift/resources/kubeadmin/kubeconfig > ~/.kube/conf
```

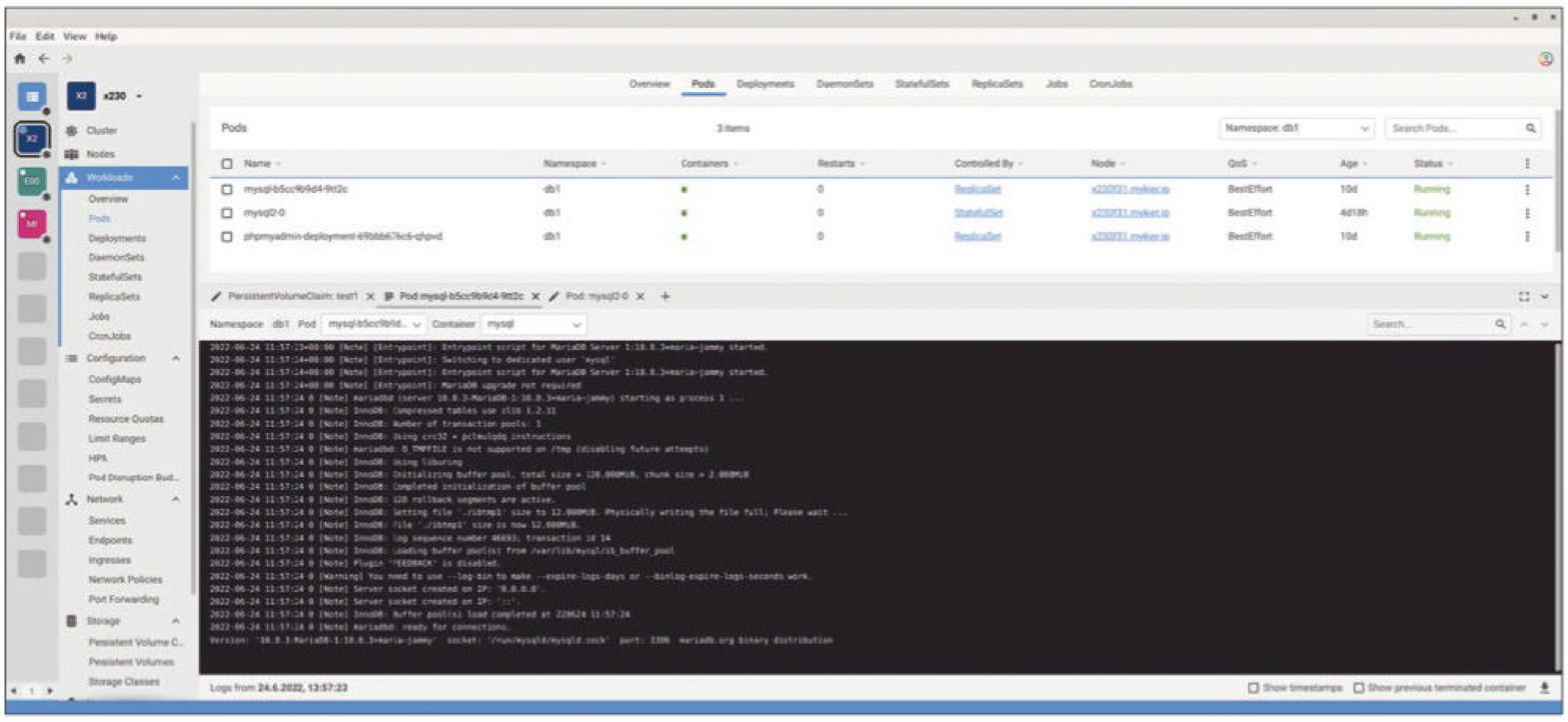
If you want to manage your Kubernetes server from another system (e.g., a Windows workstation), first copy the specified `kubeconfig` file to the client system, load the file into an editor, and look for the line:

```
server: https://127.0.0.1:6443
```

Replace `127.0.0.1` with the external IP address of your VM. The line occurs several times in the `kubeconfig` file. Next, install the `kubectl` and `oc` client tools you need for your workstation. You can then control your Microshift VM from your workstation.

Small Steps to the Container

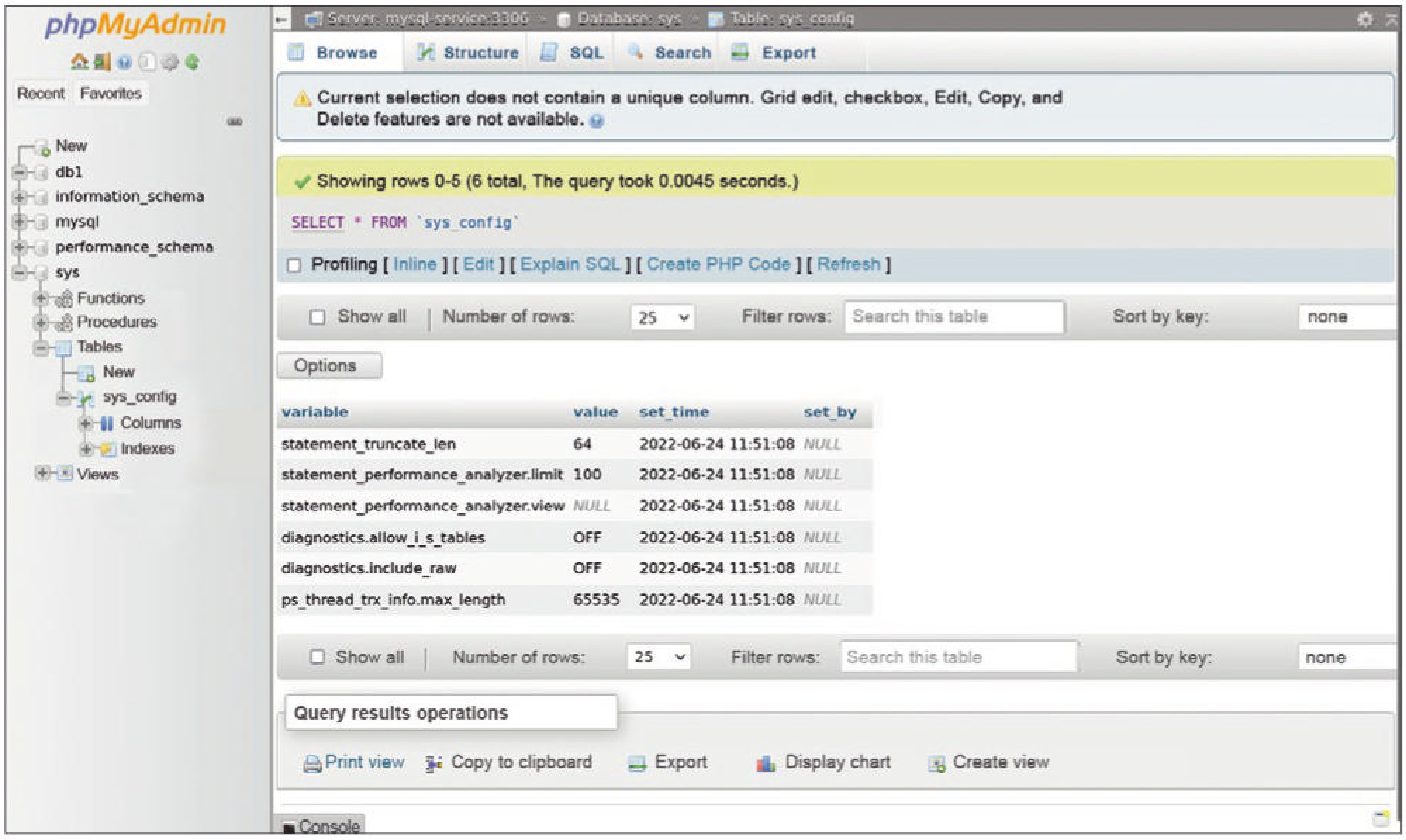
If you run containers on a single node with Docker or Podman, you will not usually assign separate IP addresses – at least not addresses that would be visible to other machines on the network. Port forwarding routes one or more IP ports of the container to ports of the host system. A MariaDB container can forward its internal port 3306 directly to host port 3306 or any other. Alternatively, you could provide a



Name	Namespace	Containers	Restarts	Controlled By	Node	QoS	Age	Status
mysql-b5cc9b9d4-9tt2c	db1	●	0	ReplicaSet	x230f31.mykier.io	BestEffort	10d	Running
mysql2-0	db1	●	0	StatefulSet	x230f31.mykier.io	BestEffort	4d18h	Running
phpmyadmin-deployment-69bbb676c6-qhpvd	db1	●	0	ReplicaSet	x230f31.mykier.io	BestEffort	10d	Running

```
2022-06-24 11:57:23+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:10.8.3+maria~jammy started.
2022-06-24 11:57:24+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2022-06-24 11:57:24+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:10.8.3+maria~jammy started.
2022-06-24 11:57:24+00:00 [Note] [Entrypoint]: MariaDB upgrade not required
2022-06-24 11:57:24 0 [Note] mariadbd (server 10.8.3-MariaDB-1:10.8.3+maria~jammy) starting as process 1 ...
2022-06-24 11:57:24 0 [Note] InnoDB: Compressed tables use zlib 1.2.11
2022-06-24 11:57:24 0 [Note] InnoDB: Number of transaction pools: 1
2022-06-24 11:57:24 0 [Note] InnoDB: Using crc32 + pclmulqdq instructions
2022-06-24 11:57:24 0 [Note] mariadbd: O_TMPFILE is not supported on /tmp (disabling future attempts)
2022-06-24 11:57:24 0 [Note] InnoDB: Using liburing
2022-06-24 11:57:24 0 [Note] InnoDB: Initializing buffer pool, total size = 128.000MiB, chunk size = 2.000MiB
2022-06-24 11:57:24 0 [Note] InnoDB: Completed initialization of buffer pool
2022-06-24 11:57:24 0 [Note] InnoDB: 128 rollback segments are active.
2022-06-24 11:57:24 0 [Note] InnoDB: Setting file './ibtmp1' size to 12.000MiB. Physically writing the file full; Please wait ...
2022-06-24 11:57:24 0 [Note] InnoDB: File './ibtmp1' size is now 12.000MiB.
2022-06-24 11:57:24 0 [Note] InnoDB: log sequence number 46693; transaction id 14
2022-06-24 11:57:24 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
2022-06-24 11:57:24 0 [Note] Plugin 'FEEDBACK' is disabled.
2022-06-24 11:57:24 0 [Warning] You need to use --log-bin to make --expire-logs-days or --binlog-expire-logs-seconds work.
2022-06-24 11:57:24 0 [Note] Server socket created on IP: '0.0.0.0'.
2022-06-24 11:57:24 0 [Note] Server socket created on IP: '::'.
2022-06-24 11:57:24 0 [Note] InnoDB: Buffer pool(s) load completed at 220624 11:57:24
2022-06-24 11:57:24 0 [Note] mariadbd: ready for connections.
Version: '10.8.3-MariaDB-1:10.8.3+maria~jammy'  socket: '/run/mysqld/mysqld.sock'  port: 3306  mariadb.org binary distribution
```

Figure 1: MariaDB `StatefulSet` for Kubernetes creates persistent storage and starts the database container. A web front end with phpMyAdmin runs in the namespace.



phpMyAdmin

Recent Favorites

- New
- db1
- information_schema
- mysql
- performance_schema
- sys
  - Functions
  - Procedures
  - Tables
    - New
    - sys_config
      - Columns
      - Indexes
  - Views

Server: mysql-service:3306 » Database: sys » Table: sys_config

Browse Structure SQL Search Export

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy, and Delete features are not available.

✓ Showing rows 0-5 (6 total, The query took 0.0045 seconds.)

```
SELECT * FROM `sys_config`
```

Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: none

Options

variable	value	set_time	set_by
statement_truncate_len	64	2022-06-24 11:51:08	NULL
statement_performance_analyzer.limit	100	2022-06-24 11:51:08	NULL
statement_performance_analyzer.view	NULL	2022-06-24 11:51:08	NULL
diagnostics.allow_i_s_tables	OFF	2022-06-24 11:51:08	NULL
diagnostics.include_raw	OFF	2022-06-24 11:51:08	NULL
ps_thread_trx_info.max_length	65535	2022-06-24 11:51:08	NULL

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: none

Query results operations

Print view Copy to clipboard Export Display chart Create view

Console

Figure 2: The phpMyAdmin front end in the Kubernetes main space of the database uses the service to communicate with the MariaDB pod.

bridge adapter to the container environment. The containers can then be managed directly with individual IP addresses like virtual machines (VMs). In such an environment, a "dedicated" MariaDB container could be run as a replacement for a MariaDB VM:

```
podman run ➥
  --name maria ➥
  --volume /var/pods/maria:➥
         /var/lib/mysql:Z ➥
  --net pub_net ➥
  --ip 192.168.1.10 ➥
  --mac-address 12:34:56:78:9a:bc ➥
  docker.io/mariadb:latest
```

The host system `/var/pods/maria` directory contains the database files of the container. If it crashes, the data is retained. The whole setup works without port forwarding. The `pub_net` defined by the administrator runs on a network bridge. All the systems on the local network can access the MariaDB container by way of its IP address.

The code points to the `mariadb:latest` image. Each restart of the container can trigger an update of the database – major versions, too. For production environments, you will therefore always want to specify the MariaDB version number and only update after appropriate testing. Image tags let you to specify only the major or minor releases (e.g., `mariadb:10` or `mariadb:10.8.3`). However, a setup like this with a "generic" database server that many clients in the local area network address directly is not used very often. In far more cases, a

Listing 1: `mariadb-state.yml`

```
---
apiVersion: v1
kind: Service
metadata:
   name: mariadb
   labels:
        app: mariadb
spec:
   type: NodePort
   ports:
   - port: 3306
      protocol: TCP
   selector:
      app: mariadb
---
apiVersion: v1
kind: ConfigMap
metadata:
   name: mariadb
   labels:
      app: mariadb
data:
   MYSQL_ROOT_PASSWORD: mysqlroot
   MYSQL_DATABASE: db1
   MYSQL_USER: mysqluser
   MYSQL_PASSWORD: mysqlpwd
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
     name: mariadb
spec:
   serviceName: "mariadb"
   replicas: 1
   selector:
        matchLabels:
          app: mariadb
---
volumeClaimTemplates:
- metadata:
     name: mariadb
    spec:
     accessModes: [ "ReadWriteOnce" ]
     resources:
      requests:
         storage: 10Gi
```

database server only supports a single application.

The second example shows how to run a MariaDB server on Kubernetes (see the box "Kubernetes Test with Microshift"). The example uses `StatefulSet` to ensure that an application with a state always has the appropriate persistent storage available (Figures 1 and 2). To begin, create the `mariadb-state.yml` file (Listing 1) that launches with the service.

Lines 1-14 declare the database port as a service. This definition can then be used as an easy way to connect other applications to the database server. Because this example is a single-node cluster and you want to address the MariaDB container directly, you need to create the service as a `NodePort`. Kubernetes then generates an automatic port mapping (Figure 3).

The file continues with a `ConfigMap` (lines 16-26). The values specified as `data` correspond to the environment variable declarations (`-e`) that you pass in at the Docker or Podman command line. However, the `ConfigMap` data is in plain text in the Kubernetes configuration. In a production environment, you would define the passwords separately as *Secret* for encryption and security.

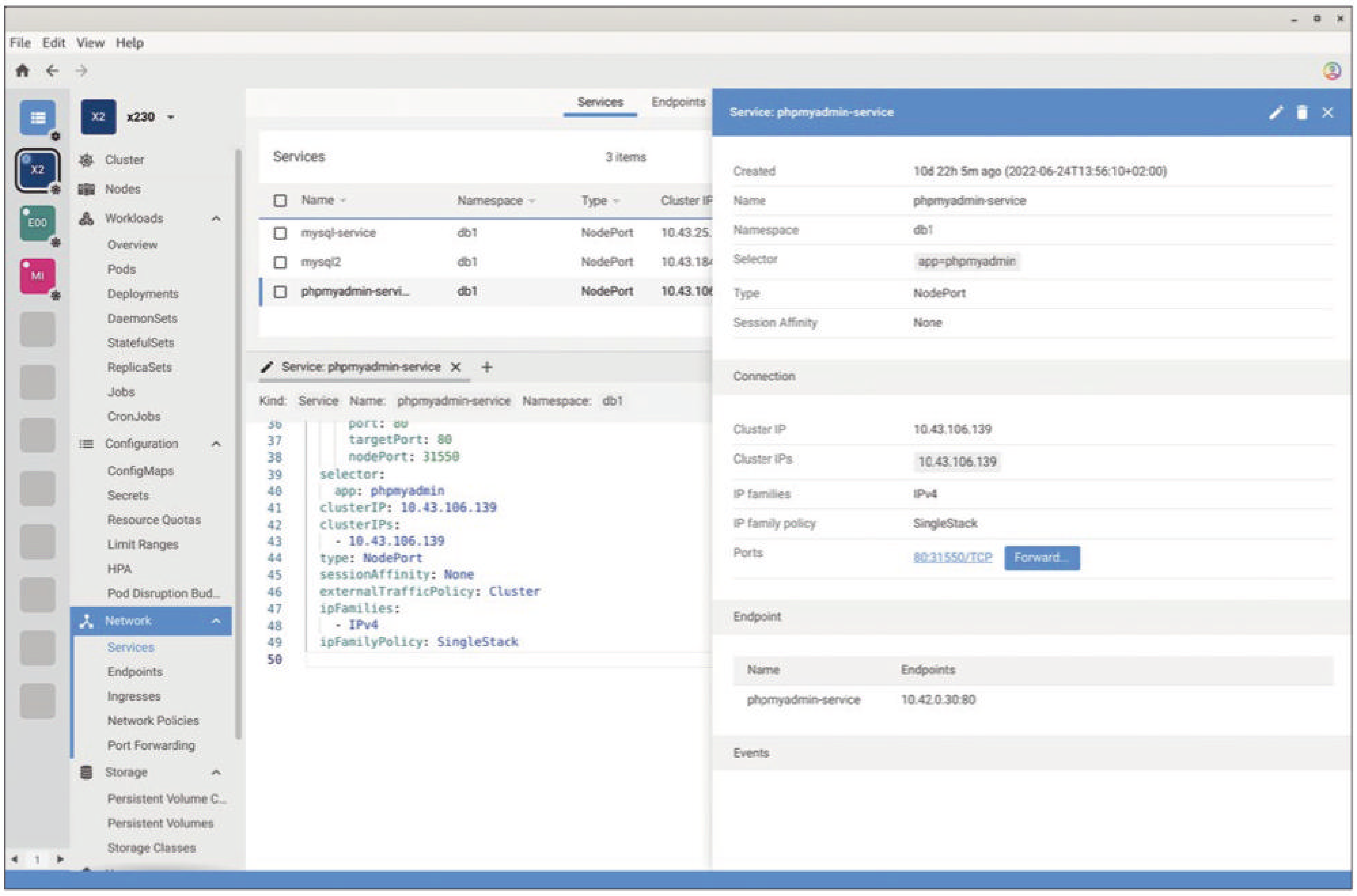
Now it's time for the `StatefulSet` itself (lines 28-37). This section declares the Kubernetes pod with its `name` and the number of `replicas`. Because MariaDB in this configuration does not support active-active mirroring, the pod only has one replica, so `StatefulSet` makes sure that exactly one pod is running at any given time. If it crashes for any reason, Kubernetes automatically starts a new pod within seconds (Listing 2).

A pod can contain one or more containers that always run together and cannot scale separately. Mostly, however, a pod comprises a single container (i.e., `mariadb:latest` here); alternatively, the version number might be stated. You could optionally specify quota rules at this point (i.e., the RAM size and CPU shares the container will be given) as maximum or minimum values. This pod retrieves its environment variables from the `ConfigMap` declared earlier.

At this point, the volume mount point for the persistent volume (PV) is important. With `volumeClaimTemplates` (lines 39-46), `StatefulSet` automatically generates a PV claim, which in turn creates the PV and binds it to the

Listing 2: MariaDB Pod Reboot

```
template:
  metadata:
    labels:
      app: mariadb
  spec:
    containers::
    - name: mariadb
      image: mariadb:latest
      ports:
      - containerPort: 3306
        name: mariadb
      volumeMounts:
      - name: mariadb
        mountPath: /var/lib/mysql
      envFrom:
      - configMapRef:
        name: mariadb
```



Services 3 items

Name	Namespace	Type	Cluster IP
mysql-service	db1	NodePort	10.43.25.
mysql2	db1	NodePort	10.43.18
phpmyadmin-servi...	db1	NodePort	10.43.10

Service: phpmyadmin-service

Kind: Service Name: phpmyadmin-service Namespace: db1

```
      port: 80
      targetPort: 80
      nodePort: 31550
  selector:
    app: phpmyadmin
  clusterIP: 10.43.106.139
  clusterIPs:
    - 10.43.106.139
  type: NodePort
  sessionAffinity: None
  externalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
```

Service: phpmyadmin-service

Created	10d 22h 5m ago (2022-06-24T13:56:10+02:00)
Name	phpmyadmin-service
Namespace	db1
Selector	app=phpmyadmin
Type	NodePort
Session Affinity	None

Connection

Cluster IP	10.43.106.139
Cluster IPs	10.43.106.139
IP families	IPv4
IP family policy	SingleStack
Ports	80:31550/TCP

Endpoint

Name	Endpoints
phpmyadmin-service	10.42.0.30:80

Events

Figure 3: In a single-node Kubernetes structure, NodePorts allow access to applications in containers over the local area network; Kubernetes clusters use routes instead.

pod. To run `StatefulSet`, simply create a namespace (projects) and run the specified YML file:

```
oc new-project mysql
oc create -f mariadb-state.yml
```

The command is

```
kubectl create namespace mysql
kubectl apply -f mariadb-state.yml
```

if you only use the `kubectl` client.

PostgreSQL and Microsoft SQL

You can also use exactly the same principle to create stateful sets for PostgreSQL or Microsoft SQL servers in containers. For PostgreSQL, use port 5432 instead of 3306 and the `postgresql:latest` or `postgresql:14` image and type the following lines into `ConfigMap`:

```
data:
  POSTGRES_DB: postgresdb
  POSTGRES_USER: admin
  POSTGRES_PASSWORD: test123
```

Microsoft delivers two different images for the SQL server. The link *mcr.microsoft.com/mssql/server* picks up the current SQL server image on an Ubuntu basis from the in-house registry, whereas the link *mcr.microsoft.com/mssql/rhel/server* gets a SQL server on an RHEL 8 (UBI8 = universal base image 8) basis. Microsoft (MS) SQL server uses port 1433. You have to pay attention to one small detail in the config map:

```
data:
  ACCEPT_EULA: "Y"
  SA_PASSWORD: "<LongPassword>"
  MSSQL_PID: "Developer"
```

MS SQL server requires a long password with eight digits or more. If you only specify *test123* here, as in the PostgreSQL example, the MS software will stop the container immediately after it is started and you end up in a crash loop. The `ACCEPT_EULA` variable is also mandatory.

Implementing Failover

With a `StatefulSet` or Kubernetes deployment, you don't have to worry too much about application availability. If a database pod crashes, Kubernetes starts a new one within seconds and connects it to the existing persistent storage. Of course, in a Kubernetes cluster, this also covers the crash of a host, with Kubernetes then launching the new pods on another node. Complicated active-passive constructs with quota and Pacemaker rules are no longer needed.

How you back up the databases themselves from the persistent volume depends on the storage back end and your backup strategy. For example, you can have a sidecar container (second container in the same pod) or a separate pod access the PV in read-only mode and back up the physical files from there. In that case, you would need to stop the database in the database pod itself at the right times to close the database files correctly on disk. Built-in tools work best, such as timed database dumps or a second pod that acts as a replication receiver to mirror all transactions asynchronously from the first database server. You can find online how-tos for MariaDB [1] and PostgreSQL [2].

Replication nodes are then useful for a rudimentary scale-out scenario. Replication means you can launch multiple read-only mirror pods of the SQL database, which are then accessed by application pods that require read-only access to the database, which lifts the load off the main database pod.

Scale-Out Dimensioning

One of the essential functions of a relational database is the relations between the tabulated data (queries with `JOIN`), which are queried in real time. For this reason, a SQL server needs to keep the complete inventory of the database on the queried node. Concepts such as scale-out and data sharding, wherein parts of the data are distributed across several nodes and no node has all the information, are difficult to implement here.

A little-known database named NuoDB [3] promises full SQL functionality with scale-out architecture. NuoDB relies on two types of nodes: the storage manager with persistent volumes and the transaction engine. In very simplified terms, the transaction nodes act as an in-memory cache for the storage nodes. If you want to test NuoDB, you will find on the project's website a suitable Helm chart for the community edition that supports a maximum of three nodes. However, it only worked in the test environment with Microshift running on a fairly powerful machine – NuoDB requires 8GB of RAM per node pod. Genuine scale-out is still the domain of NoSQL databases, which rely on completely different approaches and data and query structures.

Cassandra: NoSQL with a Structure

The open source Apache Cassandra project saves data in tables like a SQL server. It relies on its own Cassandra query language (CQL) for queries; as the name suggests, CQL is strongly oriented on SQL. Unlike SQL, however, CQL has no relations and therefore no `JOIN` queries that query the data from multiple linked tables.

Therefore, Cassandra can redundantly distribute the tables to multiple nodes. If a single node crashes, the system redistributes the lost datasets from the redundancies. If you add more nodes, the system also automatically rebalances the tables to match the number of nodes and redundancy specifications. However, much like a SQL database, Cassandra requires structured data and data types.

Cassandra is well suited as a SQL replacement for applications that handle data management in the application itself and do not require higher level SQL functions such as `JOIN` to do so. If you can handle

simple create, read, update, delete (CRUD) functions on your own, you can easily switch your application from SQL to CQL and change the database back end.
A Cassandra cluster can be easily launched with `StatefulSet` on Kubernetes [4]. With the right configuration, the nodes will automatically find each other; the replica count determines the size of the cluster. If you want to try Cassandra on a test cluster like the Microshift cluster referred to earlier, you need to have enough free RAM, because Cassandra is written in Java and is a little memory hungry. The setup for this article ran on an old laptop with a Core i7 and 16GB of RAM – and that was fine.

Document-Centric NoSQL

Document-centric databases such as Couchbase or MongoDB take a completely different approach. Instead of structured tables, these databases save information with keys and values. In principle, each document could use its own keys. These NoSQL databases focus on the dataset itself and are therefore considered document-centric, not table-centric. Index keys allow the database to find the stored data again. With simple calculated hash values, MongoDB and Couchbase can very easily distribute their documents across multiple nodes to achieve redundancy.
This solution basically works like GlusterFS or Ceph. Document databases use the vendor's API instead of a standardized query language. Cases in which application developers with an existing SQL connection convert their application to NoSQL are accordingly rare. The approaches are simply too different.
On the other hand, developers of modern web applications can handle this type of data storage quite quickly and easily. Simple function libraries exist for every modern programming language to simplify the handling of document databases greatly and eliminate the tedious process of elaborating a data table and variable structure up front.
Nevertheless, the lack of a unified query language for document-centric NoSQL databases is certainly one of the reasons these technologies have been slow to establish themselves as generic databases for off-the-shelf applications. If you want to take a closer look at these tools, you don't have to look far. Couchbase has a prebuilt Helm chart that makes installation on Kubernetes a breeze [5].

Special Services

NoSQL databases for general use are joined by a few modern scale-out databases for special applications, including Elasticsearch, which has a document structure similar to MongoDB and can be operated quite simply across several nodes. Elasticsearch, as the name suggests, provides a strong search function, which is why it is often used where comparatively unstructured data (e.g., logfiles) need to be searched for specific information.
Users are increasingly replacing databases with simple key-value stores in applications with low data volumes or simple datasets. Redis is certainly one of the most popular at the moment. Many scale-out applications use this data structure store as a communications hub, allowing pods to exchange data and configuration information. Redis runs in-memory but can also dump data to persistent storage at predetermined intervals to reload inventory data after a pod restart.

Conclusions

With concepts such as Kubernetes' `StatefulSet`, classic databases such as MariaDB, MS SQL, or PostgreSQL can be nicely accommodated in a container environment. Kubernetes delivers good, if not better, database availability than an elaborate active-passive failover construct with Pacemaker. However, this setup requires a reliable storage back end. NoSQL databases promise better integration in scale-out environments, especially for applications with large but not necessarily contiguous datasets. Migrating existing SQL applications to NoSQL would seem to be, with the possible exception of Cassandra, too expensive in most cases, because the data structures differ greatly. ■

Info

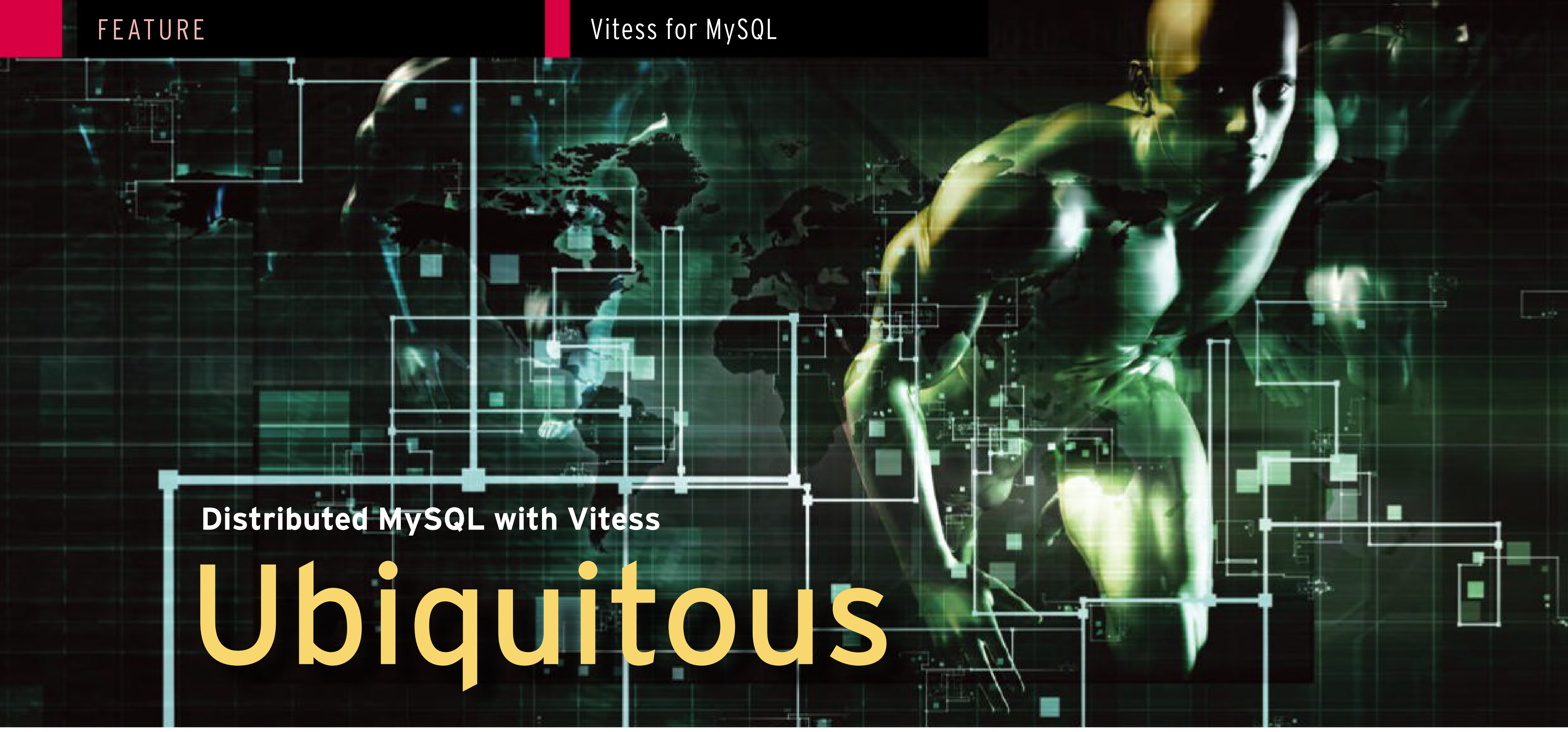
[1] MariaDB replication: [https://mariadb.com/kb/en/standard-replication/]

[2] PostgreSQL streaming replication: [https://wiki.postgresql.org/wiki/Streaming_Replication]

[3] NuoDB: [https://www.3ds.com/nuodb-distributed-sql-database/]

[4] Cassandra deployment with StatefulSet: [https://kubernetes.io/docs/tutorials/stateful-application/cassandra/]

[5] Helm chart for CouchDB: [https://artifacthub.io/packages/helm/couchdb/couchdb]



Distributed MySQL with Vitess

Ubiquitous

Vitess relies on various techniques to scale MySQL horizontally, while looking like the popular database from the outside. But does it deliver what its authors promise? By Martin Loschwitz

Whereas IT organizations in the past maintained individual setups with separate infrastructures, scalable platforms today almost inevitably set the tone. Because customers take it for granted that their providers will continually reduce the cost of IT infrastructure and its operation, today IT can only be operated efficiently if it relies on sheer mass.

In the past, admins might have had a few dozen systems in their care; however, today they are more likely to have hundreds of machines or even more. Of course, this also affects the applications that run on these platforms: They need to be able to grow along with the environments. Several examples do this impressively, including Ceph, which offers storage that grows with the usage scenario, or applications that adhere to a cloud-ready design and can internally scale seamlessly at every level.

The trend toward scaling does not stop at databases either. You might think this topic no longer needs further consideration. After all, you have Galera for MySQL and a cornucopia of solutions for PostgreSQL that retrofit scalability. Unfortunately, it's not that simple. Galera (Figure 1) has a fan base, but comes with architectural weaknesses that often make its use impossible. PostgreSQL is nowhere near as popular as MariaDB or MySQL; it is used in far fewer setups and is considered more complicated and cumbersome. Not to mention that quite a few of the replication solutions for PostgreSQL are also plagued with critical design flaws.

Vitess [1], a scalability solution for MySQL that has proven itself over the years at YouTube, deliberately does a few things differently from Galera. The authors promise seamless scalability with reliable consistency, as well as several additional features such as an internal engine that optimizes database queries to eke out more performance. At least according to the advertising, Vitess – in combination with MySQL – is the perfect database solution for scalable setups and, in turn, also for cloud setups.

Challenges

Before I turn directly to Vitess, it might be useful to briefly outline the basic challenges of a distributed database system to help understand why special software is needed at all to implement scaled databases. After all, the question of why a single MySQL instance on powerful hardware is not enough is quite legitimate and not at all easy to answer. The same applies to the question of why a single dataset from MySQL cannot be easily distributed to any number of MySQL instances on the back end.

Basically, databases in modern setups are one of the few places where persistent data is backed up. Today, modern cloud-ready applications in

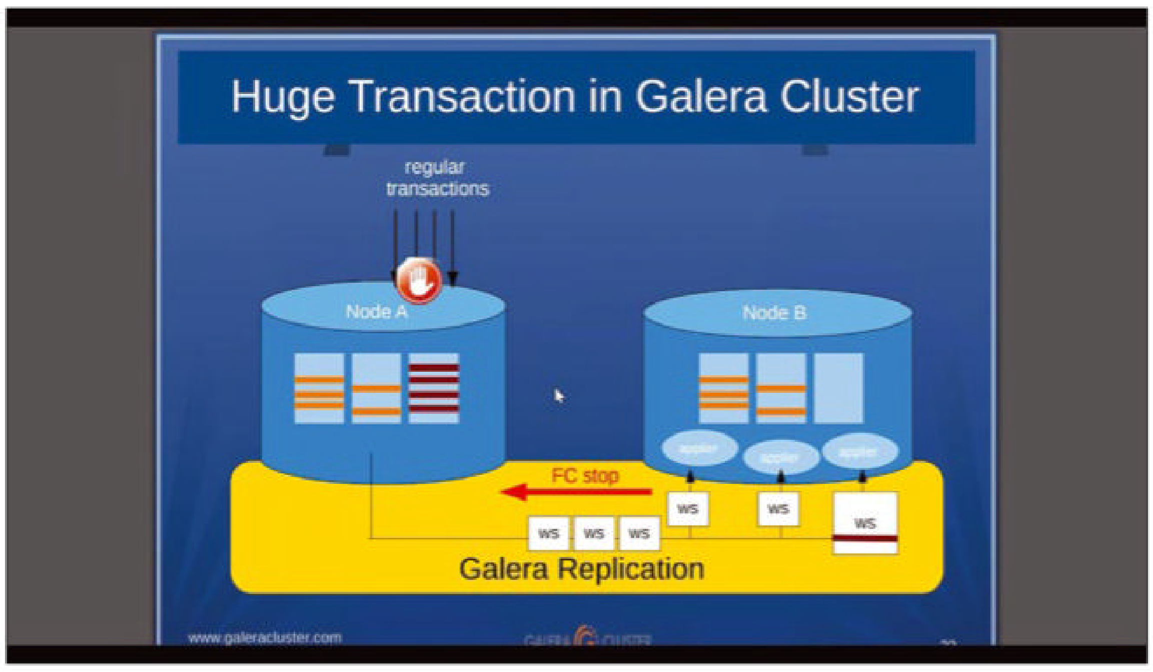


Figure 1: Galera is a first generation replication solution that suffers from some architectural restrictions. © Galera

particular often take a completely different approach to their data management: They logically isolate the data assets (i.e., static content such as images or videos) from the user data. Asset data in modern setups today often is outsourced to external storage, such as Amazon Simple Storage Service (S3), from where it is then mounted directly through an HTTP link. This process reduces the load on the application and is generally faster for clients because storage can be targeted geographically with the S3 protocol and, moreover, can be combined with caches to create a kind of mini content delivery network (CDN).
It used to be quite common to store asset data in databases, too, but those days are long gone. Why then is a single instance of MySQL not enough in cloud setups?
The short answer to this question is that a single MySQL would be a single point of failure and sooner or later could run into performance issues, depending on the size of the setup. Classic high availability (HA) solutions such as cluster managers are also anathema to the developers of cloud-ready setups. The crystal clear requirement for any application, simply put, is that it needs to scale seamlessly and run as an arbitrary number of connected instances. Solutions like MySQL are no exception to this rule.
Technically, though, a big challenge is in making a behemoth such as MySQL fit for horizontal scaling. In classic MySQL, everything is designed to have a central control authority. The master instance is the sole point of contact for write operations. It guarantees consistency without which a database can hardly be used in a meaningful way. What this looks like under the hood is clear to any admin who has ever had to deal with MySQL. Somewhere in `/var/lib/mysql/`, you have the databases with their tables, with simply no way to interconnect multiple MySQL instances so that they can handle queries from multiple instances and uphold consistency guarantees at the same time.

Many Instances and Sharding

This problem is exactly what Vitess addresses for MySQL. However Vitess is a totally different beast from Galera, which slots in as a MySQL plugin, implementing communication between nodes with its own consensus algorithm and keeping the contents of the cluster's database instances in sync.
The Vitess architecture (Figure 2) is nothing like that simple. Above all, Vitess can't really be called an add-on for MySQL anymore. Although MySQL still plays a role in the context of Vitess as a possible back end for storing data, the entire logic of data storage and data processing ultimately takes place at the Vitess level and therefore in services designed specifically for Vitess.
The linchpin of the solution is a component named VTGate, or Vitess Gateway, a pointer to the function of this component. All data that is directed to the database must pass through VTGate, which is supported by an additional Topology Service that manages and stores the metadata of a Vitess instance. To import changes to the metadata (e.g., when new databases or tables are created), you can use two tools: `vtctl`, which runs directly at the command line, or `vtctld`, which is the back end for a GUI that can also modify the data in the database.

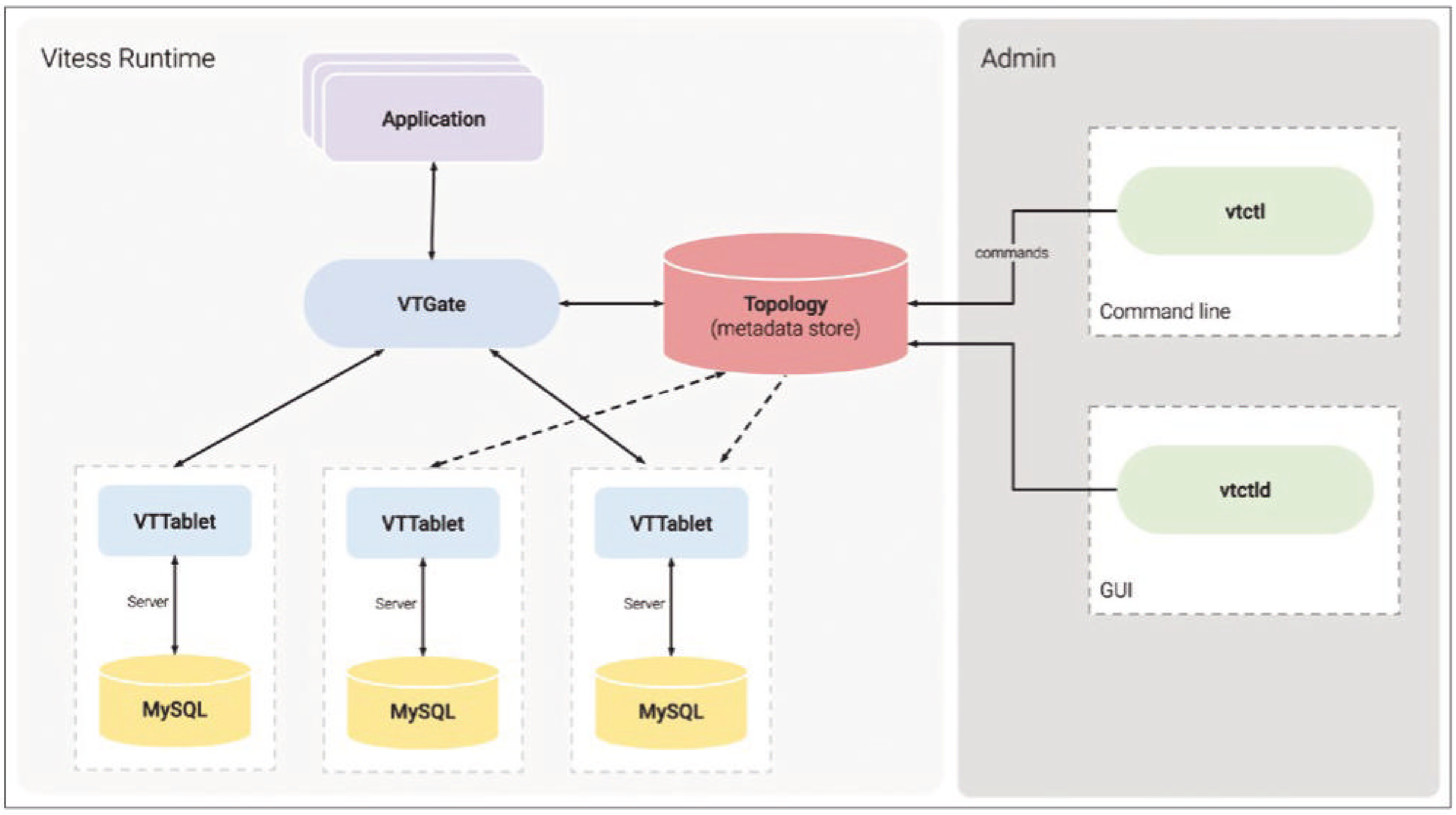


Figure 2: Vitess is a cloud-native database that relies on MySQL in the background and exposes the MySQL protocol to the outside world. However, it abstracts MySQL by means of an intermediate layer. © Vitess

When it comes to the database metadata, control occurs indirectly through the Topology Service, with which VTGate communicates constantly. To understand the way Vitess works, you must understand that its gateway implements its own MySQL interface in the form of the MySQL server protocol. Clients accessing the overall construct do not communicate with a genuine MySQL instance, but with a kind of abstraction layer that can handle the MySQL protocol (Figure 3).

Of course, a front end for storing data does not give you a database, because the information has to go somewhere. Vitess relies on MySQL, but in a different way than you might expect. A VTGate instance can have any number of VTTablet instances assigned to it, this being another service from the Vitess universe. A VTTablet instance always needs a real database on the back end where it can store its data locally, and Vitess relies on MySQL again for this task.

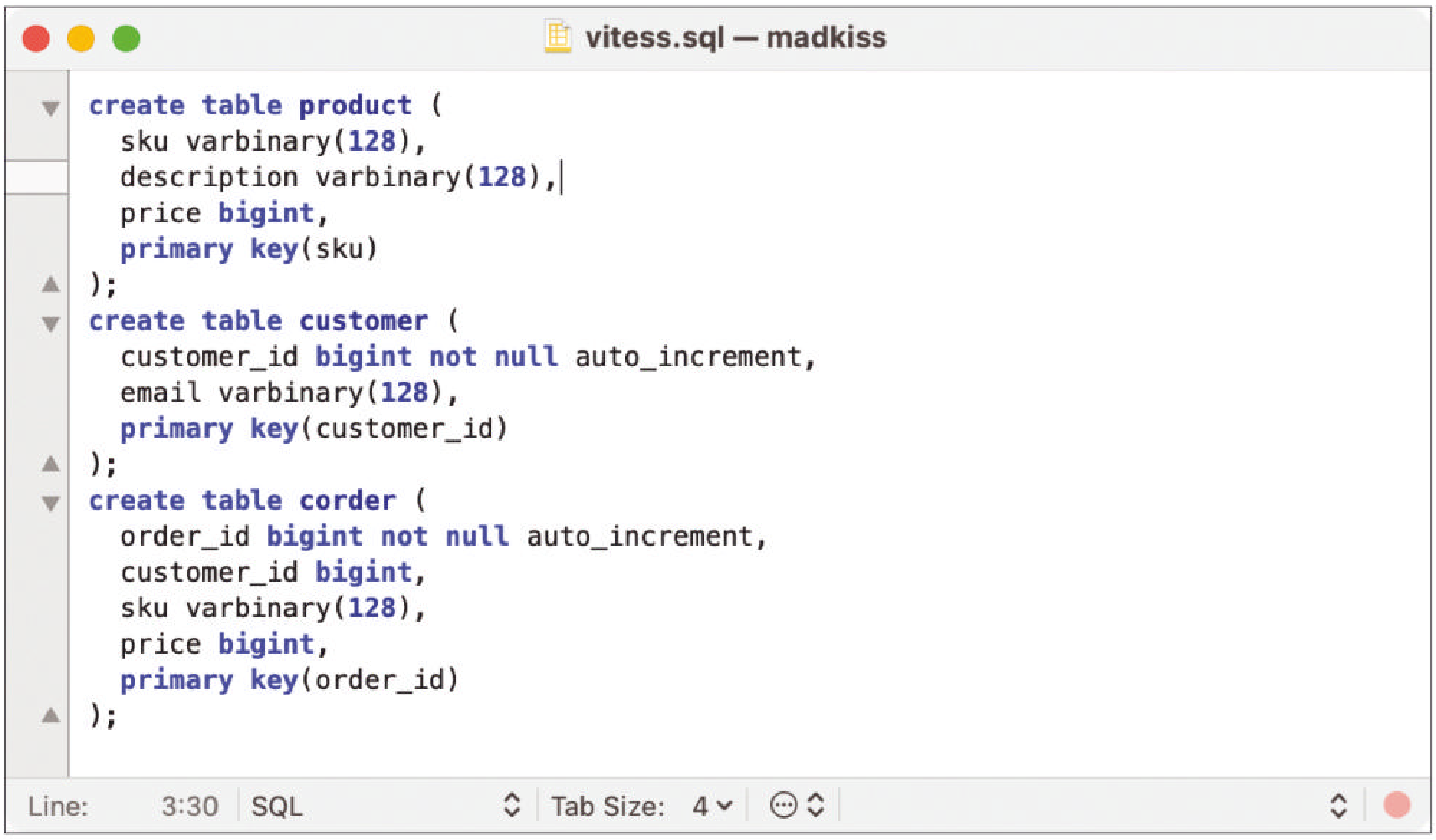
On the basis of this design alone, however, it is clear that the single MySQL instance here really just acts as a data store. The actual magic (i.e., the distribution of the data) is handled by a team of VTGate and VTTablet instances. A VTTablet instance can have several operating modes: In primary mode it allows write access, and in replica mode it's in a maintenance state.

Logical Databases

Administrators who want to run Vitess in the future need to understand fully the implications of this implementation. The classic division of MySQL into databases, tables, and rows and columns does not occur natively in Vitess. Also, in terms of the MySQL and VTTablet processes running in the background, there is no equivalent to what a client sees when accessing a MySQL instance of Vitess with VTGate. Instead, what a client sees through a VTGate instance is always the compiled, overall logical image of a widely distributed data set in the background on the individual VTTablet instances. Certain parallels to other distributed solutions such as Ceph are obvious. There, the Ceph Monitor (MON) servers implement the abstraction for the client, and the Object Storage Daemons (OSDs) store data on block storage devices in the background. Vitess works in a quite similar way, except the individual services go by the names VTGate and VTTablet.

The question then becomes: How does VTGate transpose the data in the background (i.e., how does Vitess implement the legacy MySQL data model)? When an arbitrary MySQL client connects to a VTGate instance, it will still see databases, tables, and rows and columns, but VTGate rewrites these structures internally. To make things slightly complicated for newcomers, the Vitess authors use a whole cornucopia of technical terms to describe their design, which prospective Vitess admins urgently need to study.

The analogy for the namespace, for example, is the keyspace, which is the counterpart to a classical database in MySQL at the Vitess level. In the Vitess documentation, the authors describe a keyspace as a logical database. Where the contents of such a database ultimately end up depends on its configuration and mode of operation – and that's the rub when it



```
create table product (
  sku varbinary(128),
  description varbinary(128),
  price bigint,
  primary key(sku)
);
create table customer (
  customer_id bigint not null auto_increment,
  email varbinary(128),
  primary key(customer_id)
);
create table corder (
  order_id bigint not null auto_increment,
  customer_id bigint,
  sku varbinary(128),
  price bigint,
  primary key(order_id)
);
```

Figure 3: From the client's point of view, nothing changes when working with MySQL. The commands from the Vitess documentation shown here would also be usable with a normal installation of MySQL.

comes to Vitess scalability. To achieve the desired scalability, Vitess relies on sharding, the splitting of a large data set into many smaller data sets that then end up on different storage back ends. Administrators are familiar with this principle from mail servers, among other things: If the space available on the back-end storage is not sufficient, the only solution is to acquire a second storage solution and distribute the mailboxes across the two storage instances.

Vitess follows quite a similar approach with sharding. The VTGate instance distributes data that belongs together across all VTTablet instances it knows. Vitess explicitly refers to the shard (i.e., a data fragment) as a subunit of the keyspace, which again reveals a little more about Vitess's functionality. What looks like a database and a table to the outside (i.e., when viewed through VTGate) is in reality several chunks of data located in the various VTTablet instances.

The operating modes of VTTablet, as previously mentioned, also play a role because they are inherited by the shards located on a VTTablet instance. Accordingly, shards can also be in the primary or replication state. The highlight is that the different roles of a shard are distributed to different VTTablet instances: VTTablet instance 1 can be the primary for one shard, instance 2 for another shard, and so on. This structure distributes load while allowing users to benefit from the many storage devices on different MySQL nodes in the background by simply logically merging their bandwidth.

Now you can see the core function of VTGate: It knows which shards are located where and cooperates with the Topology Service for this purpose. Earlier, I mentioned a compiled and logical view of the cluster that the clients see. It is VTGate in cooperation with the Topology Service that is responsible for compiling this totally logical view on the fly and delivering it to the client. Needless to say, VTGate is also scalable – many instances of VTGate can run in parallel.

The Vexed Subject of the Quorum

Anyone familiar with distributed systems – and specifically with distributed storage – will have started wondering how Vitess solves the problem of the quorum, which is always inherent in distributed systems. In fact, the quorum also plays a big role when working with shards, but the Vitess developers make it very easy for themselves. VTGate generates the view for clients with the help of the Topology Service. The Vitess developers have simply added an interface to cooperate with external consensus algorithms. In Vitess, this is Etcd by default, which runs in many Kubernetes instances anyway. Alternatively, Zookeeper can be used.

Performance and Redundancy

A running Vitess cluster must be thought of as a set of keyspaces spread wildly across all existing instances of VTTablet and MySQL in the form of shards. A shard always exists several times, and replica shards are used for read operations. If an instance fails, the missing shards are automatically reinstantiated on other instances of the cluster, which not only helps with performance but also guarantees application redundancy.

At the same time, VTGate ensures that the central consistency guarantees, such as those based on the atomic, consistant, isolated, durable (ACID) principle, continue to be upheld in Vitess. ACID is the catch-all question that comes up regularly in the context of distributed databases. Whether a database is reliably consistent ultimately drives the success or failure of the application in many areas. The Vitess developers have managed to achieve precisely this consistency by using VTGate as a proxy server with third-party language capabilities in the form of a reimplemented MySQL protocol, which is quite impressive from a technical point of view.

This is all the more true because Vitess also supports the cell as a logical unit, which is no less than the targeted distribution of shards across the boundaries of physical locations. Consequently, setups for disaster recovery are easy to implement with Vitess; again, this capability sets Vitess apart from other solutions such as Galera.

Tricks and More Tricks

The Vitess developers note at various points in their documentation that, although useful replication was a primary design goal of Vitess, it was far from the only one. The developers additionally looked to address some of the issues that were bothering them about MySQL but which remain unsolved to this day. One ongoing problem is the famous queries of death, which seem to take an eternity to execute. It's no secret that many companies would be better off investing money in a MySQL consultant than in increasingly powerful hardware for the database.

In many places, for example, database queries have been created in applications over the years that force MySQL to perform huge internal queries and that can, in the worst case, take down the entire database. The problem with MySQL is that, once such a query has been dispatched, you have no way to stop it prematurely from the outside and are forced to look on while MySQL maneuvers itself into a black hole or is eventually reprimanded by the kernel's out-of-memory killer.

Vitess obviously has a starting point for stopping this problem. The VTGate instance can also reach the limits of its performance when dealing with complex requests. Unlike real MySQL, however, a query can be terminated externally with `vtctl`, averting any danger to the functionality of the database. Prevention is also possible. `LIMIT` statements for certain query types can be stored at the VTGate level and, transparently from the client's point of view, terminate queries automatically after a certain wait or not execute them at all.

Another similarly useful VTGate capability targets performance optimization. Of course, if all queries to the database pass through VTGate, the VTGate instances also know what has been queried recently, which is where deduplication comes in handy. If VTGate or an instance from a VTGate team detects that several identical read operations are taking place at the same time, it forwards only one of them to the back ends in the background but delivers the results for all of the requests.

Another thing worth mentioning is the pooling of connections, which Vitess supports out of the box. Traditionally, each client connecting to a MySQL instance requires its own standalone TCP/IP connection with its own memory. However, as you know, clients in Vitess do not talk directly to MySQL, but only to VTGate. Because Vitess is written in Go, the connection pooling from this language is available: Instead of opening several self-sufficient connections to the individual VTTablet instances and their back ends in the form of MySQL in the background, it combines all of the incoming connections into significantly fewer open connections in the background, while still achieving higher bandwidth and more performance.

Made for the Cloud

Applications designed for the cloud today generally trigger certain expectations on the part of the user community. One assumption, for example, is that cloud-ready applications offer extensive interfaces out of the box for debugging and, more specifically, for analyzing performance. Vitess is no exception, and the developers of the software deliver. Vitess keeps elaborate records of practically everything it does at runtime, and it offers the possibility of parsing the data over standardized interfaces, which then benefits services such as Prometheus (Figure 4), for which Vitess even provides a native interface out of the box.

The service's data collection capability is not limited to individual applications like VTGate. Extensive metrics data also can be squeezed out of the individual VTTablets, allowing for classic event monitoring in addition to comprehensive performance monitoring. From the administrator's point of view, the most important

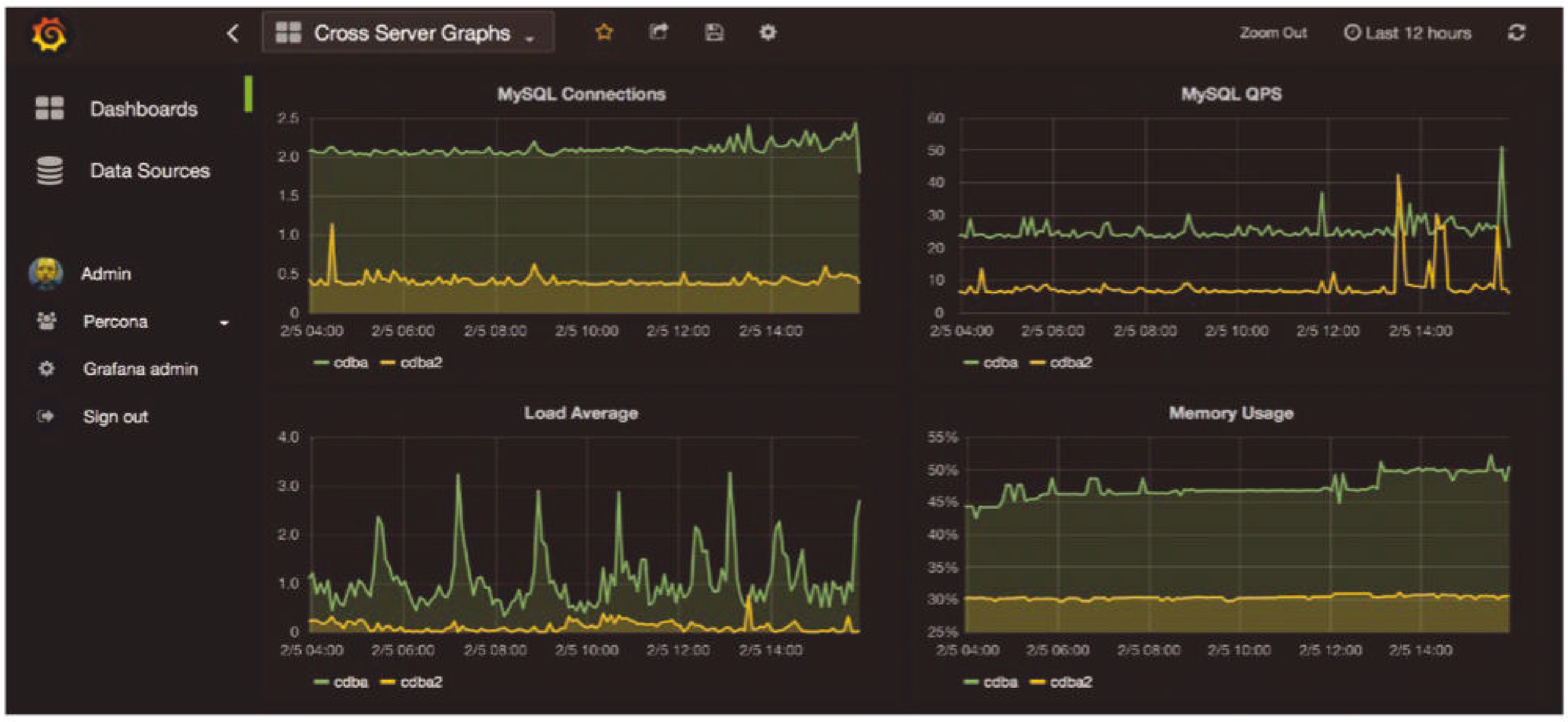
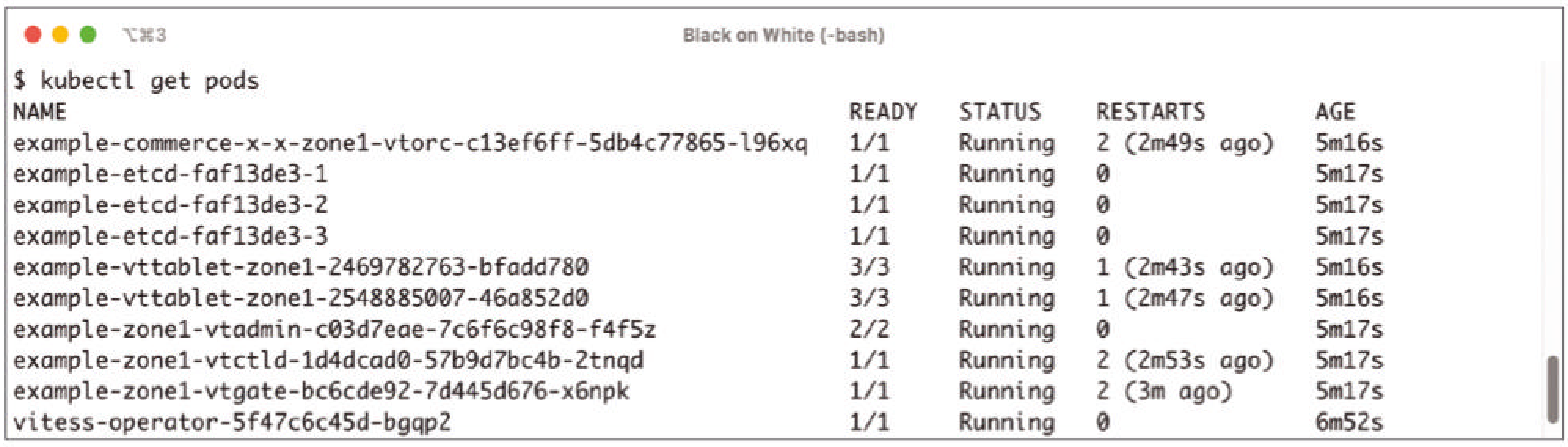


Figure 4: MySQL can only be integrated into Prometheus with its own exporter, as shown in this example. Vitess, on the other hand, comes with its own interface for Prometheus along with a dashboard for Grafana.© Percona



```
$ kubectl get pods
NAME                                                   READY   STATUS    RESTARTS        AGE
example-commerce-x-x-zone1-vtorc-c13ef6ff-5db4c77865-l96xq   1/1     Running   2 (2m49s ago)   5m16s
example-etcd-faf13de3-1                                1/1     Running   0               5m17s
example-etcd-faf13de3-2                                1/1     Running   0               5m17s
example-etcd-faf13de3-3                                1/1     Running   0               5m17s
example-vttablet-zone1-2469782763-bfadd780             3/3     Running   1 (2m43s ago)   5m16s
example-vttablet-zone1-2548885007-46a852d0             3/3     Running   1 (2m47s ago)   5m16s
example-zone1-vtadmin-c03d7eae-7c6f6c98f8-f4f5z        2/2     Running   0               5m17s
example-zone1-vtctld-1d4dcad0-57b9d7bc4b-2tnqd         1/1     Running   2 (2m53s ago)   5m17s
example-zone1-vtgate-bc6cde92-7d445d676-x6npk          1/1     Running   2 (3m ago)      5m17s
vitess-operator-5f47c6c45d-bgqp2                       1/1     Running   0               6m52s
```

Figure 5: As befits its status, Vitess offers perfect integration with Kubernetes and can be rolled out reliably in it in just a few seconds.

thing is to find a healthy monitoring balance – precisely because Vitess is implicitly redundant and offers certain self-healing capabilities. It no longer seems appropriate to drag the administrator out of bed at 3:00am on Saturday because a MySQL instance in a cluster of 20 Vitess nodes has died. Vitess will eventually recover all missing replicas of shards automatically, and in a short time, within the set recovery values.
However, the situation is different for central components: If a load balancer upstream of several VTGate instances fails, access to the database will eventually fail, triggering the need for manual intervention by the administrator.
Performance monitoring is likely to be almost more useful than event monitoring for Vitess admins in everyday life. After all, databases play a central role in many setups, with their performance regularly tipping the scales for the overall performance of an environment. This performance can be meticulously monitored down to the level of individual queries by reference to the logged metrics data. Vitess can do far more than MySQL, for example, out of the box.

Conclusions

The idea of re-implementing a part of MySQL on the basis of a different architecture understandably triggers skepticism among many, and particularly so with Vitess because the solution ultimately still needs a real MySQL instance in the background, acting as a back end for the VTTablet services.
Any doubts, however, are completely blown away on closer inspection. Unlike first-generation database replication solutions such as Galera, Vitess takes a far more comprehensive approach that translates to better performance, stronger redundancy, greater visibility of the details (e.g., in performance monitoring), and a far more pleasant user experience, all told.
Vitess as a state-of-the-art solution naturally also offers perfect Kubernetes integration and can be rolled out directly in a cluster with the Kubernetes Helm package manager, for example (Figure 5). In any case, anyone looking for a scalable database for the cloud should have Vitess on their radar. ■

Info

[1] Vitess: [https://vitess.io]

The Author

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

Scale-out with PostgreSQL

Load Up

The world of scale-out is stateless; unfortunately, databases are not. YugabyteDB solves this dilemma for PostgreSQL. By Franck Pachot

If you want to respond to higher load with more instances instead of a more powerful server, you typically need stateless applications. Horizontal scaling, whether in the cloud or with multiple servers, virtual machines, Kubernetes pods, or containers, creates the elasticity and resilience that business applications need – especially in the cloud, where every component needs to be replaceable in the event of failure and where elasticity ensures cost reductions without performance loss.

Hardly any application can do without a database to store data and ensure consistency, even with concurrent transactions. However, a database is likely to be stateful, and that presents a challenge when it comes to scaling. YugabyteDB is a cloud-native database, is fully open source, and aims to achieve PostgreSQL compatibility, which means the ability to run the same applications and tools with identical application behavior while handling complex online transactional processing.

Monolithic RDBMSs

You could ignore the modern challenges of relational database management systems (RDBMSs) if you just opened them on a single server. This restriction allows database processes simply to use the operating system's shared memory, which is protected by latches (spinlocks or mutexes).

Database users often think their database's shared memory pool is only useful as a cache for boosting performance, because it removes the need for I/O access to the data file in many cases. In fact, one of the main reasons for the existence of the shared memory pool is that it serves as a single source of truth when different sessions need read and write access to a given block of data without corrupting the block. However, a side benefit is that a larger shared memory pool also reduces read and write access to the hard disk for frequently used blocks. PostgreSQL in particular partially delegates this cache function to the Linux kernel, which is why the recommendation is to keep the shared memory pool smaller and leave more space for the filesystem buffers.

This mechanism limits the database to a single server that can share its RAM with all processes. One exception is Oracle's Real Applications Cluster (RAC), in which multiple RDBMS instances can open a database. However, this arrangement mandates complex intracluster synchronization, wherein the current version of a data block is only allowed to exist at a single location and therefore needs to be moved back and forth between physical machines. This global cache and lock management requires a network with exceptionally low latency, such as the InfiniBand network offered by Oracle's Exadata servers. On a conventional network, this setup will not work between different data centers or in the cloud. Additionally, RAC requires block storage that can be accessed by multiple virtual machines at the same time. This additional challenge in the cloud environment is unusual.

Over the years, monolithic RDBMSs have been beefed up with a variety of features to meet business needs or to make development and testing easier. Thanks to atomicity, consistency, isolation, and durability (ACID) guarantees, these applications deliver correct results even if race conditions or other issues occur. The key is SQL with its referential integrity (foreign keys) or unique constraints. What impresses most is that applications do not require any additional code. Once a level of isolation is established, anomalies that could result from concurrent access are prevented, again without additional code.

Another feature that is easy to implement in a monolithic architecture is the various indexes and their automatic maintenance, which means the same database can be used for multiple use cases with more flexibility as a bonus thanks to logical-physical independence. In short, the application queries the tables without worrying about the access path to the data. The database transparently and autonomously finds the optimal access path through the indexes.

Web Scale and NoSQL

Over the years, the pressure to serve the need for scaling across the board has become more intense. The reasons for this were the Internet, web-based business models, and the cloud. You cannot serve millions of users with a single database running at a single location somewhere in the world. This setup is impossible for three reasons.

First, the latency experienced by users when information is moved between different regions around the world is an unacceptable hundreds of milliseconds. No technical approach can alleviate this problem because information cannot be transmitted faster than the speed of light. The only option left is to keep the data in different locations near the users. Storing data at different locations can be a legal requirement, as well.

Second, the network between geographic regions is the Internet, where packets are routed through an infrastructure in which performance bottlenecks and errors can occur at any time. If the database only ran in one place, the application would become unavailable whenever the network went down. All popular RDBMSs implement active-passive replication and provide read replicas that can be used to output reports, even if the primary database is down. However, this is not equivalent to scaling across different nodes and implementing consistent cross-node read and write transactions.

Third, some workloads need high throughput with predictable response times even under peak load (e.g., Black Friday trading). The use of more CPUs and more RAM in a server (scale-up) requires downtime and always has limits. To ensure the availability of the application in all situations, the database needs to run on several servers, the number of which can be increased as needed and decreased later to save money. At the same time, this arrangement allows for updates during operation.

This need to scale out has spawned a new type of database that partitions data into shards, which it then stores on multiple servers. One solution has been to run transactions essentially in the way they were run before the advent of relational databases: with a hierarchical and distributed structure that does not allow for complex transactions or guarantees of consistency and makes do with very simple storage primitives. NoSQL databases do precisely this. They are similar to datastores before SQL was invented but are distributed across a resilient and elastic network of nodes. NoSQL is great for certain tasks for which the access patterns are known in advance. Some companies have migrated their data to these kinds of databases. In turn, however, it forces additional code on the application to replace the loss of SQL features. Of course, until some time ago, this was the only way to scale out.

NewSQL

The need to scale out arose for all kinds of workloads, including complex online transactional processing (OLTP); developers still wanted all the SQL features they enjoyed in the RDBMSs used previously but now with all the benefits of the NoSQL scale-out architecture. As a result, the popular SQL came back in a form called NewSQL.

Data warehouses and analytical applications simply used NoSQL's sharding technique to split their data, and each shard was its own RDBMS. However, this approach does not allow for transactions beyond the boundaries of a shard, no global indexes, no unique constraints, no foreign keys between shards, and no joins between partitions. This arrangement is acceptable for analytical applications, but OLTP applications need global transactions.

Distributed SQL databases are the latest development in this respect for the time being. They provide all the features of SQL, including cross-node transactions, foreign keys, and unique constraints in a scale-out architecture. Google's Cloud Spanner, CockroachDB, and TiDB belong to this group. Cloud Spanner provides SQL much like PostgreSQL, whereas TiDB resembles MySQL.

Vendor Yugabyte takes things a step further with YugabyteDB by reusing PostgreSQL to provide a full set of SQL features, then building the whole thing on a Spanner-style distributed engine foundation. In this way, the user gets the best of both worlds: You can connect to any node and always have read and write access to a single logical database (Figure 1).

Distributed SQL

YugabyteDB is based both on proven work, as described in a Google paper on Spanner [1], and on unique innovation. In a two-layer architecture, the reused PostgreSQL in the SQL layer builds on the underlying DocDB layer, which is the distributed transactional key-value data storage layer. Applications can connect to any node and see a single logical database from there. Although the Yugabyte structured query language (YSQL, the PostgreSQL-compatible layer) is

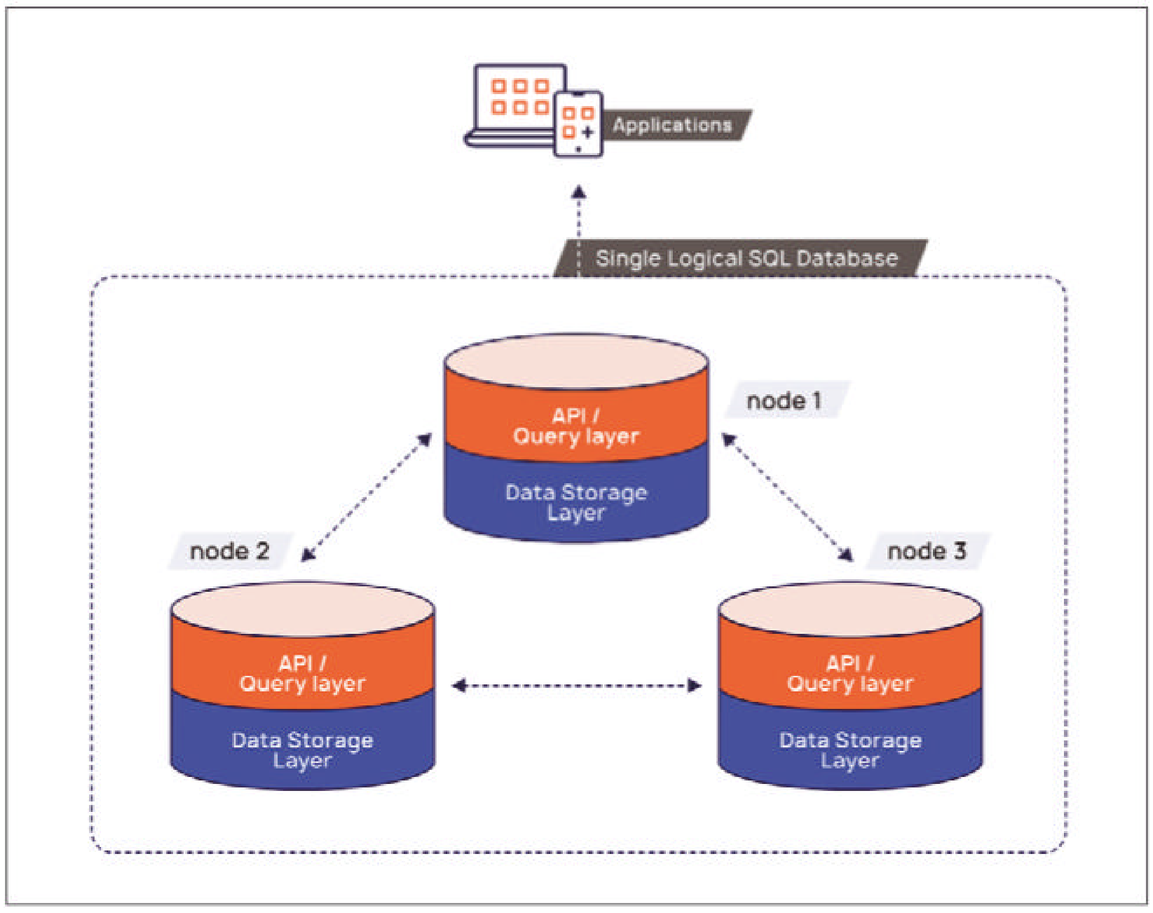


Figure 1: Users can connect to any node to gain read and write access to the database. © Yugabyte

the main interface, it can be replaced (Figure 2). For example, Yugabyte has a Cassandra-like API that can be used if the user does not need full SQL support and instead wants to use the syntax and special features of Cassandra (e.g., TTL, or time to live). The PostgreSQL back end, made ready for the cluster with additional code, parses the queries and executes them. The tables are split into shards by a hash or range function and distributed among the database nodes. The default sharding strategy includes a hash on the first column of the key and an automatic split as the table grows.

However, the PostgreSQL syntax has been extended to allow full administrative control. The statement

```
yugabyte=# create table emp (
  empno int, hiredate timestamp,
  primary key (empno hash)
  ) split into 42 tablets;
```

creates a table with hash sharding based on the `empno` column with 42 preset shards (known as `tablets`). The statement

```
yugabyte=# create index emp_hire
  on emp ( hiredate asc )
  split at values (
    ('2010-01-01'),
    ('2015-01-01'),
    ('2020-01-01')
  );
```

creates an index for the `hiredate` column that is split by range to allow a scan of the index range, for which the year ranges 2010 to 2020 are preset.

To avoid latency between nodes during joins, multiple tablets can be stored in the same shard so that all joins remain local. Some reference tables can be synchronized with multiple regions. With a replication factor (RF) = 3 across three data centers or availability zones, the database can be used even if one of the data centers fails. Nodes can also be distributed as part of a multicloud or hybrid cloud setup (Figure 3).

By default, the tables and indexes are distributed over the entire cluster. If latency or data management requires, distribution can be limited to a subset of the cluster, such as a specific region, achieved with the help of PostgreSQL partitioning functions and tablespaces. Tablespaces are used to map tables, indexes, and partitions to the cluster topology. The example in Listing 1 replicates a region (*eu-west*) across three availability zones. Afterward, the tablets are realigned with this topology (in the last line). This online process moves tablets as they are read from and written to, and it all works thanks to replication.

Architecture

Given the technical challenges, not many distributed SQL databases

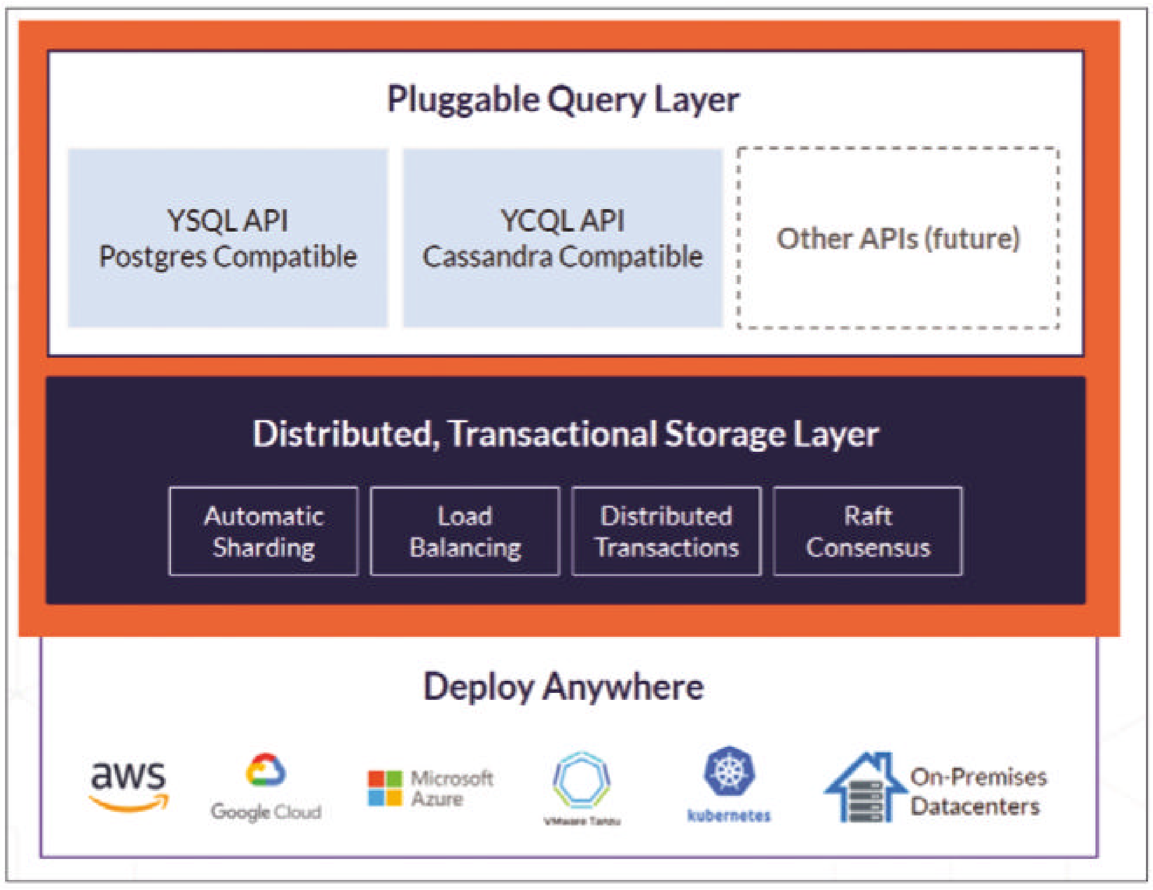


Figure 2: The exchangeable API layers on top of the distributed storage engine are a special feature of YugabyteDB. © Yugabyte

provide full SQL functionality without limiting transactions to a single server.
Google Spanner started with limited SQL but later added protocol and syntax compatibility with PostgreSQL. It is still restricted to the Google Cloud platform. CockroachDB does the same thing but opens up to other platforms and provides some core functionality as open source. Its PostgreSQL compatibility still requires code changes when porting PostgreSQL. TiDB is compatible with MySQL.
In this article on distributed SQL with YugabyteDB, I focus on the YSQL API, even though many Cassandra converts use the Cassandra-like Yugabyte cloud query language (YCQL).

PostgreSQL Compatibility

The decision in favor of compatibility with PostgreSQL was driven by its popularity for OLTP applications and its consistency with the SQL standard. Having the same protocol and syntax and providing the same functionality is useful – not only for the portability of existing applications, but also for familiarity. YugabyteDB is a new database, and in the event of a problem, searching for an answer online (e.g., Stack Overflow) would not yield many hits.
However, when it comes to SQL issues, the knowledge base in the documentation, forums, and blogs regarding PostgreSQL also applies to YugabyteDB. Compatibility in this case means the ability to run the same applications and tools used with PostgreSQL. YugabyteDB provides all isolation levels with the same semantics as PostgreSQL to ensure the same behavior in production. Other distributed databases only implement the serializable isolation level, which requires the application to implement repetition logic.
The YSQL code layer is based on a fork of PostgreSQL that comes with many SQL functions and includes not only all current PostgreSQL features, but also the use of numerous PostgreSQL extensions. The places

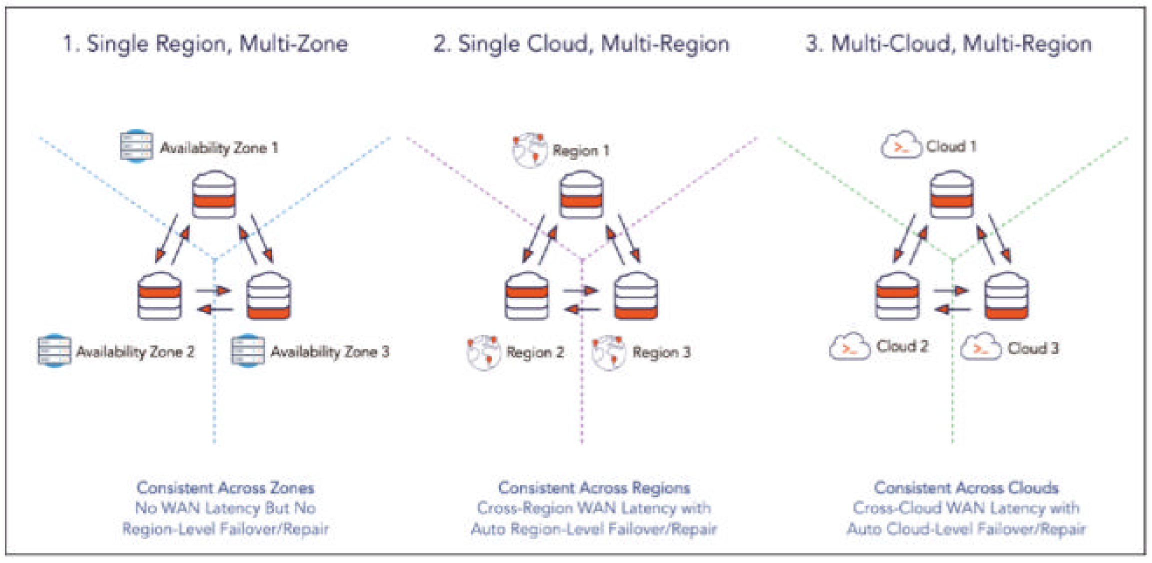

1. Single Region, Multi-Zone
2. Single Cloud, Multi-Region
3. Multi-Cloud, Multi-Region
Availability Zone 1
Availability Zone 2
Availability Zone 3
Region 1
Region 2
Region 3
Cloud 1
Cloud 2
Cloud 3
Consistent Across Zones
No WAN Latency But No
Region-Level Failover/Repair
Consistent Across Regions
Cross-Region WAN Latency with
Auto Region-Level Failover/Repair
Consistent Across Clouds
Cross-Cloud WAN Latency with
Auto Cloud-Level Failover/Repair

Figure 3: Several options for the distribution of nodes. © Yugabyte

Listing 1: Replication

```
create tablespace "eu-west" with ( replica_placement= $$
  {
  "num_replicas": 3, "placement_blocks": [
    { "cloud": "cloud", "region": "eu-west1", "zone": "eu-west1a", "min_num_replicas": 1 },
    { "cloud": "cloud", "region": "eu-west1", "zone": "eu-west1n", "min_num_replicas": 1 },
    { "cloud": "cloud", "region": "eu-west1", "zone": "eu-west1c", "min_num_replicas": 1 }
  ]
  } $$);
alter table emp set tablespace = "eu-west";
```

in the code that are specific to YugabyteDB are clearly marked. Look for `IsYBRelation()` in the code to get a good idea. This approach enables adaptation to the new features of future PostgreSQL versions that are released every year.

Open Source

Just like PostgreSQL, YugabyteDB does not have an Enterprise edition: YugabyteDB with all its database features is open source, giving you all the freedom you need and allowing for community contributions. For production, the company Yugabyte offers support and managed services, but the database and all of its features remain under the Apache 2 open source license. The decision to use free software was explained in detail by the founder and CTO of Yugabyte, Karthik Ranganathan [2].
Because YugabyteDB is open source, it is easy to test. It runs on any Linux platform – whether server, virtual machine, or container. You only need local storage and network access. The easiest way to test it on a laptop is to use Docker. A single node is all you need. If you add three or more nodes, you can test for resilience (by simulating a failure) and elasticity (by simply adding more nodes). The simple web console lets you monitor automatic data balancing and processing. The commands

```
$ docker network create -d bridge yb
$ docker run -d ➘
  --name yb1 --hostname yb1 --net=yb ➘
  -p5431:5433 ➘
  -p7001:7000 ➘
   yugabytedb/yugabyte:latest ➘
   yugabyted start ➘
  --daemon=false --listen yb1
```

create a network, start an initial node, and export ports 5433 for PostgreSQL and 7000 for the web console revealing the tablet server and tablets. Port 7000 can be accessed on *http://localhost:7001*. After startup, any PostgreSQL client can connect to the database with

```
postgres://yugabyte@localhost:5431/yugabyte
```

To add more nodes, use the `--join` option. You can build a three-node cluster with:

```
$ docker run -d ➥
  --name yb2 --hostname yb2 --net=yb ➥
  -p5432:5433 ➥
  -p7002:7000 ➥
   yugabytedb/yugabyte:latest ➥
   yugabyted start ➥
  --daemon=false --listen yb2 --join yb1
$ docker run -d ➥
  --name yb3 --hostname yb3 --net=yb ➥
  -p5433:5433 ➥
  -p7003:7000 ➥
   yugabytedb/yugabyte:latest ➥
   yugabyted start ➥
  --daemon=false --listen yb3 --join yb1
```

The user can then connect to any available node and read from and write to it, even if another node is down. The same database can also be created in YugabyteDB Managed, a fully managed database as a service (DBaaS) that runs on AWS and Google Cloud. A small machine can be used free of charge.

Challenges

The query layer is based on PostgreSQL code; the rows and index entries of the YugabyteDB tables (YSQL) are sharded. A range or hash function handles the distribution work, which can be specified with `create table` and `create index` by specifying `ASC` or `DESC` for range sharding or `HASH` for sharding with a hash function. The hash function is computed by default for the first column of the primary key of secondary indexes. The default values are used to facilitate migration from PostgreSQL without changing the data definition language (DDL).

Sharding occurs automatically by splitting the shards (or tablets) as the table grows. Additional syntax, or management commands, splits the tables and indexes manually (e.g., before a bulk load). The table rows and index entries are all distributed independently by default, but for performance reasons, it is possible to group small tables and indexes together into a single tablet. This decision depends on your operations. A cluster with only one region can accept a latency of one millisecond between availability zones; a cluster with multiple regions needs to reduce cross-node remote procedure calls (RPCs).

Yugabyte Tablet Server (YB-TServer) stores and manages data for client applications, which can connect to any node. Each node provides a PostgreSQL endpoint that forks a back-end process for each connection. Parsing and execution of the query relies on PostgreSQL code, which has been extended and optimized for cluster use. However, instead of reading or writing tuples from or to the shared buffer cache and local data files, as would be the case with PostgreSQL, the back end sends reads and writes in batches to tablet leaders distributed across nodes (TServers) in the cluster. The PostgreSQL back end is a client for DocDB.

Each tablet is replicated for high availability, with one of the replicas (tablet peers) acting as the tablet leader to which reads and writes are sent. The name "leader" comes from the Raft consensus algorithm, which guarantees exactly one leader per tablet, even in the event of a failure, as long as the quorum is present. The leader lease algorithm prevents split-brain situations. If a leader is unavailable, the quorum of followers selects a new leader, guaranteeing that no data is lost and that recovery time remains minimal (recovery time objective, RTO, is expressed in seconds and is configuration dependent on expected network latency).

All SQL layer reads and writes are sent to the tablet leader according to the tuple distribution method (by splitting on the primary key or index entry) and are optimized to reduce the effect of latency on response time. A single leader guarantees consistency of reads and writes. Data is distributed and processed by automatically balancing the leader and followers. Writes are synchronized with followers and wait for quorum confirmation. In a cluster with RF = 3, the change is usually executed locally and waits for confirmation from one of the two followers. If the leader is no longer available (e.g., because of a server failure), at least one of the followers will have received the last change and can be elected as the new leader. In terms of the consistency, availability, and partition tolerance (CAP) theorem, YugabyteDB satisfies the requirements for C and P: It always remains consistent with the best possible availability.

Each tablet peer is a log-structured merge (LSM) tree based on RocksDB that stores all changes as new versions of rows or columns. All isolation levels are supported with pessimistic or optimistic locking. The first level of the LSM tree, the MemTable, is located in RAM. It is swapped out to sorted sequence table (SST) files as soon as it reaches a certain size (128MB by default), creating many SST files with all intermediate versions. To limit the amount of data, these files are compressed in the background: The data from the SST files is concatenated in a new SST file, and deprecated entries are removed; then, the deprecated original SST files are deleted.

All changes are sequenced by a hybrid logical clock (HLC) to ensure the serialization capability of transactions despite the possible time difference between the physical times of each server [3]. The title is a reference to Google Spanner, which solved the time-shifting problem with atomic clocks but then limited the use to Google's own data centers. YugabyteDB solves this problem by synchronizing the hybrid logical clock with Lamport's algorithm. It only has to wait for the lock offset in the very rare cases of no messages between servers.

Roadmap

YugabyteDB is designed to enable high-performance OLTP that requires all the SQL features invented in the 30-year history of monolithic databases and, thus, the reason for reusing PostgreSQL. Few features are not supported in a distributed context [4].

OLTP applications also run some analytical queries for which YugabyteDB cannot be optimized. In cloud-native applications, these queries are usually processed by another database service developed in-house, which requires more code and components to define the extraction, transformation, and loading of data into the specialized database. YugabyteDB has implemented change data capture, which can stream data to other data services with the support of Debezium or Kafka. Event streaming, however, will never surpass the simplicity and performance of a converged database. Where possible, real-time analysis is easier to perform in an operational database.

SQL provides many features for running complex queries with window functions, secondary indexes, and materialized views. At this point, work is being done on YugabyteDB to optimize it in a distributed context. The main approach is to move the filters (`WHERE`), aggregations (`GROUP BY`), and sorts (`ORDER BY`) to the DocDB storage layer. The filters are processed on each database node instead of having to send a full set of rows to the SQL layer to perform these actions.

As already mentioned, besides the main PostgreSQL API (referred to as YSQL), another API is compatible with Cassandra (YCQL). With time series, for example, you can choose the interface to suit your needs. YCQL supports optimal data entry by taking advantage of Cassandra drivers that are threaded, are cluster-aware, and do not add overhead by providing complex transactional semantics. YSQL is better suited for queries because it offers many functions for processing data and many index types for optimizing access.

YugabyteDB is constantly being improved, and open source is not just about the code. The architecture design and roadmap are published on GitHub [5]. Many new features are designed to improve the user experience; support new cloud-native applications; or help migrate from PostgreSQL, Oracle, or other traditional databases to Yugabyte.

Although all benchmarks show good performance for OLTP, some complex queries, especially for analytics, require further development in YugabyteDB's query planner and executor. This work is ongoing, and the developers welcome community input. ■

Info

[1] Bacon, D. F., N. Bales, N. Bruno, et al. "Spanner: Becoming a SQL System." *In:* ACM SIGMOD, ed. *SIGMOD '17: Proceedings of the 2017 ACM International Conference on Management of Data* (Association for Computing Machinery, New York, 2017), pp. 331-343

[2] "Why We Changed YugabyteDB Licensing to 100 percent Open Source" by Karthik Ranganathan, YugabyteDB blog, July 2019: [https://www.yugabyte.com/blog/why-we-changed-yugabyte-db-licensing-to-100-open-source]

[3] Distributed transactions without atomic clocks: [https://vimeo.com/545130381]

[4] PostgreSQL compatibility: [https://docs.yugabyte.com/preview/explore/ysql-language-features/postgresql-compatibility/#un-Supported-postgresql-features]

[5] Design and roadmap: [https://github.com/yugabyte/yugabyte-db]

Verifying your configuration

Checkup

Automated acceptance testing is a powerful tool for catching problems related to misconfiguration. We'll show you how to implement your own acceptance testing environment with a free tool called goss. By Ankur Kumar

Misconfiguration has long been a major problem in the IT world. Any admin can recall a scenario in which Dev did something, QA did another thing, and other departments did their own thing to pass the software through its stages of delivery. Finally, the product fails because of unknown manual tricks and black magic. Who could forget the time wasted in painful war room sessions, always wondering and struggling to reason about the weird failures? In fact, misconfiguration has been identified as the root cause for many recent public cloud outages – even for the tech giants like Amazon and Google.

How do you solve config misconfiguration and parity issues in the modern cloud-driven IT world? Open source solutions come to the rescue. In this article I describe how to use a tool called goss (an acronym for Golang server spec) [1] [2] to detect standard configuration baseline drift. I'll describe how to establish the practice of Acceptance as Code so that the same configuration flows across the whole software development life cycle to maintain much-needed parity. I'll use the Footloose and Docker container tools to bring up container machines for a test setup. Everything was tested on my Ubuntu 18.04 LTS laptop, but the code examples should run on any modern GNU/Linux machine able to run Docker Engine and Golang static binaries.

Listing 1: Dockerfile_ServerBase

```
FROM ubuntu:22.04

ENV container docker

# Don't start any optional services except for the few we need.
RUN find /etc/systemd/system /lib/systemd/system -path '*.wants/*' -not -name '*journald*' -not -name
  '*systemd-tmpfiles*' -not -name '*systemd-user-sessions*' -exec rm \{} \;

RUN apt-get update && apt-get install -y dbus systemd openssh-server net-tools iproute2 iputils-ping
  curl wget vim-tiny sudo && apt-get clean && rm -rf /var/lib/apt/lists/*

RUN >/etc/machine-id
RUN >/var/lib/dbus/machine-id

EXPOSE 22

RUN systemctl set-default multi-user.target
RUN systemctl mask dev-hugepages.mount sys-fs-fuse-connections.mount systemd-update-utmp.service
  systemd-tmpfiles-setup.service console-getty.service
RUN systemctl disable networkd-dispatcher.service

# This container image doesn't have locales installed. Disable forwarding the
# user locale env variables or we get warnings such as:
#  bash: warning: setlocale: LC_ALL: cannot change locale
RUN sed -i -e 's/^AcceptEnv LANG LC_\*$/#AcceptEnv LANG LC_*/' /etc/ssh/sshd_config

COPY setup_goss.sh /usr/local/bin/setup.sh
RUN setup.sh && rm /usr/local/bin/setup.sh

# https://www.freedesktop.org/wiki/Software/systemd/ContainerInterface/
STOPSIGNAL SIGRTMIN+3

CMD ["/bin/bash"]
```

Server Validation

How do you verify a server configured to run a single service or a set of services? You get into the server and run a set of commands to dump and verify that the required packages are installed with proper versions and make sure the configuration files have correct values. You ensure that required services are configured to start automatically – and that those services are running and listening on the configured ports. You could also add more tests to cover more points

Listing 2: `setup_goss.sh`

```
#! /bin/bash
set -uo pipefail

GOSSVER='0.3.18'
GOSSCDIR='/etc/goss'
RQRDCMNDS="chmod
  echo
    sha256sum
    tee
  wget"

preReq() {

  for c in ${RQRDCMNDS}
  do
    if ! command -v "${c}" > /dev/null 2>&1
    then
      echo " Error: required command ${c} not found, exiting ..."
      exit 1
    fi
  done

}

instlGoss() {

  if ! wget -P /tmp "https://github.com/aelsabbahy/goss/releases/download/
    v${GOSSVER}/goss-linux-amd64"
  then
    echo "wget -P /tmp https://github.com/aelsabbahy/goss/releases/download/
      v${GOSSVER}/goss-linux-amd64 failed, exiting ..."
    exit 1
  fi

  if ! wget -P /tmp "https://github.com/aelsabbahy/goss/releases/download/
    v${GOSSVER}/goss-linux-amd64.sha256"
  then
    echo "wget -P /tmp https://github.com/aelsabbahy/goss/releases/download/
      v${GOSSVER}/goss-linux-amd64.sha256 failed, continuing ..."
  else
    pushd /tmp
    if ! sha256sum -c goss-linux-amd64.sha256
    then
      echo 'sha256sum -c goss-linux-amd64.sha256 failed, continuing ...'
    fi
    rm goss-linux-amd64.sha256
    popd

  fi

  if chmod +x /tmp/goss-linux-amd64
  then
    mv /tmp/goss-linux-amd64 /usr/local/bin/goss
  else
    echo 'chmod +x /tmp/goss-linux-amd64 failed, exiting ...'
    exit 1
  fi

}

cnfgrGoss() {
  mkdir "${GOSSCDIR}"
  tee "${GOSSCDIR}/goss.yaml" <<EOF
kernel-param:
  kernel.ostype:
    value: Linux

mount:
  /:
    exists: true
    filesystem: overlay
    usage:
      lt: 90

port:
  tcp:22:
    listening: true
    ip:
    - 0.0.0.0
  tcp6:22:
    listening: true
    ip:
    - '::'

user:
  sshd:
    exists: true

package:
  docker-ce:
    installed: true

service:
  sshd:
    enabled: true
    running: true
  docker:
    enabled: true
    running: true

process:
  sshd:
    running: true
  containerd:
    running: true

dns:
  localhost:
    resolvable: true
    addrs:
      consist-of: ["127.0.0.1","::1"]
    timeout: 500 # in milliseconds
EOF

}

main() {

  preReq
  instlGoss
  cnfgrGoss

}

main 2>&1
```

Listing 3: `footloose.yaml`

```
cluster:
  name: cluster
  privateKey: cluster-key
machines:
- count: 1
  spec:
    backend: docker
    image:  ubuntu2204basewgoss
    name: node%d
    privileged: true
    portMappings:
    - containerPort: 22
```

of misconfigurations (e.g., disk space, partitions and their filesystems, RAM, etc.). So, you are asking your server a set of questions and expecting the correct answers as per your specifications. The modern automated way to do this is to bake all the necessary acceptance tests in your server only, with the server providing an on-demand report showing which tests are passing or failing.

Goss is a lightweight, no dependencies, free and open source software (FOSS) solution designed to achieve machine-level acceptance testing. The solution comprises a Golang static binary (macOS and Windows binaries are in alpha currently), so you can just put it in your servers, define your acceptance tests in YAML files, and feed those tests to the running goss application – that's it.

Getting Started

The first step is to create a base image to invoke a container machine test setup by running the command

```
docker build ⮐
  -f Dockerfile_ServerBase . ⮐
  -t ubuntu2204basewgoss
```

in a directory where the Dockerfile (Listing 1) and setup script (Listing 2) are located.

Now you have a base image with the goss binary and its test configuration baked in. The Footloose configuration shown in Listing 3 brings up your test container machine with the command:

```
footloose create
```

The command should run in the directory where the Footloose config file is located.

To get into the created test node and initiate a goss acceptance test run, use the commands:

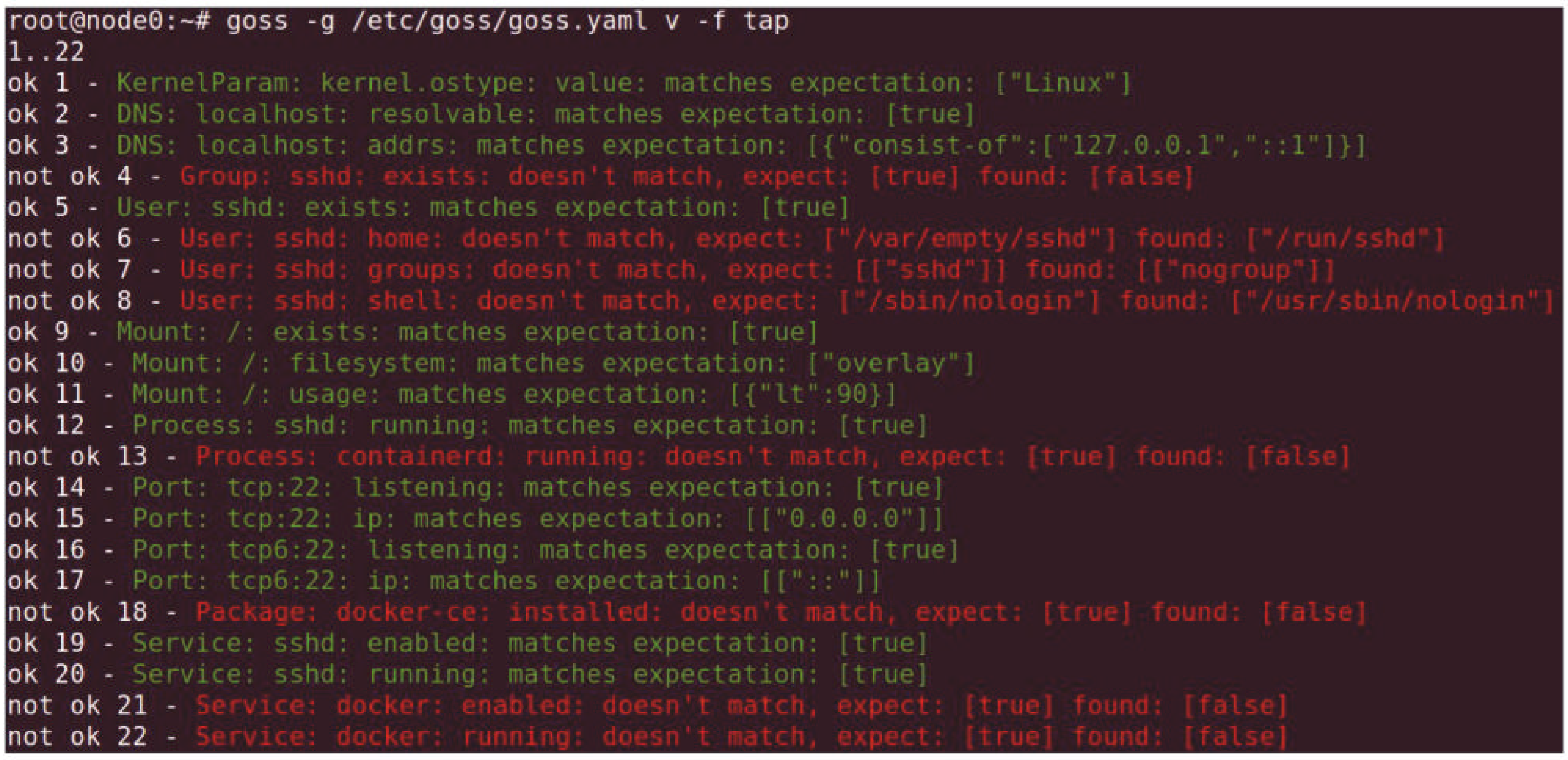
```
footloose ssh root@node0
goss -g /etc/goss/goss.yaml v -f tap
```

Congratulations, you have run your first Acceptance as Code for your test server, and you should see some red or green output (Figure 1).

The goss configuration is built around a set of resources and their properties, creating a set of test criteria. The goss run checks the current state of those resources and declares them pass (green) or fail (red). The default test configuration file is `goss.yaml`, but you can change the file name with the `-g` global option.

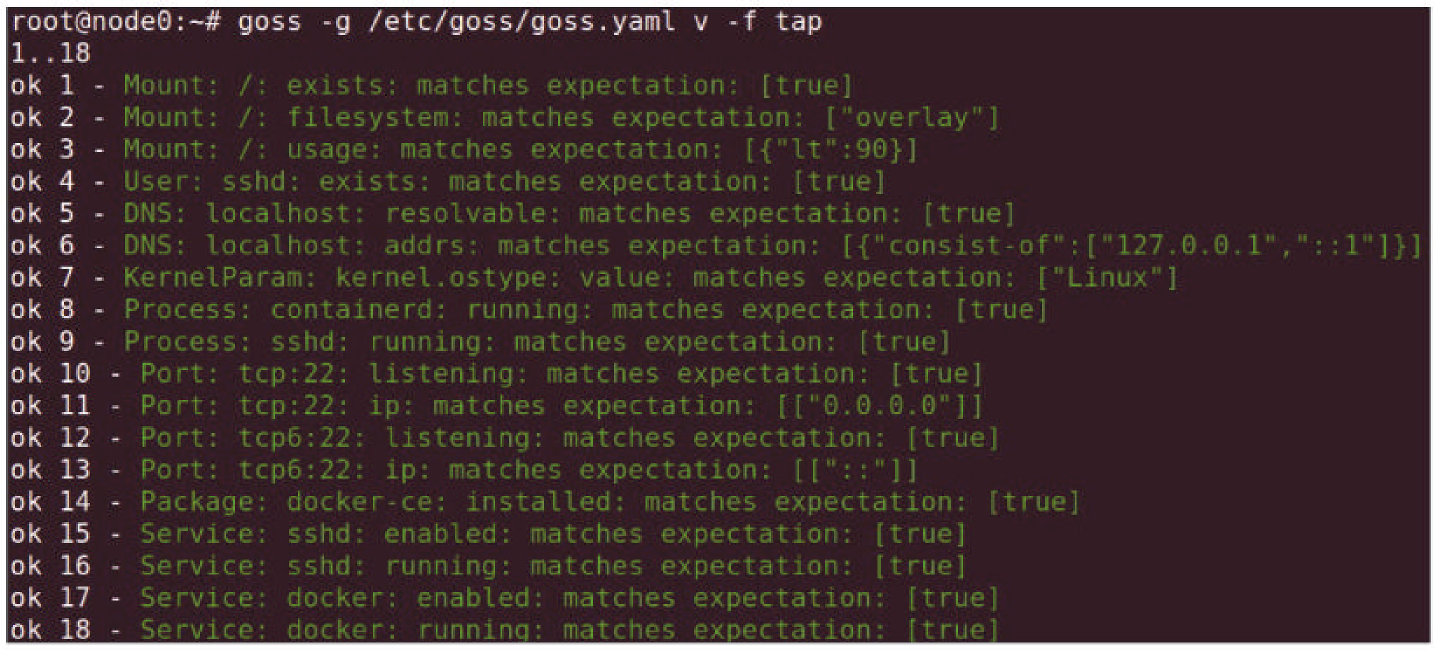
Goss requires an action argument, which is `-v` (`--validate`) in this case. Typing only `goss` on the command line dumps all the possible actions, as well as various global options. Every goss action takes a set of options that is dumped when you append the `-h` (`--help`) option. The `-f` (`--format`) option to `validate` dumps the test report in various formats, including `documentation`, `json`, `junit`, `nagios`, and others. A `silent` format doesn't dump anything but does indicates overall pass or fail for tests. You could use environment variables to set various options, as dumped in various help screens.

To summarize, I want to test for a GNU/Linux-based test server with at least working SSH login functionality, having root formatted with a particular filesystem and not filled up beyond 90 percent, with Docker Engine components properly set up and running. The goss test run verifies these requirements compared with reality in a report to catch a misconfiguration early, which can save you a lot of



```
root@node0:~# goss -g /etc/goss/goss.yaml v -f tap
1..22
ok 1 - KernelParam: kernel.ostype: value: matches expectation: ["Linux"]
ok 2 - DNS: localhost: resolvable: matches expectation: [true]
ok 3 - DNS: localhost: addrs: matches expectation: [{"consist-of":["127.0.0.1","::1"]}]
not ok 4 - Group: sshd: exists: doesn't match, expect: [true] found: [false]
ok 5 - User: sshd: exists: matches expectation: [true]
not ok 6 - User: sshd: home: doesn't match, expect: ["/var/empty/sshd"] found: ["/run/sshd"]
not ok 7 - User: sshd: groups: doesn't match, expect: [["sshd"]] found: [["nogroup"]]
not ok 8 - User: sshd: shell: doesn't match, expect: ["/sbin/nologin"] found: ["/usr/sbin/nologin"]
ok 9 - Mount: /: exists: matches expectation: [true]
ok 10 - Mount: /: filesystem: matches expectation: ["overlay"]
ok 11 - Mount: /: usage: matches expectation: [{"lt":90}]
ok 12 - Process: sshd: running: matches expectation: [true]
not ok 13 - Process: containerd: running: doesn't match, expect: [true] found: [false]
ok 14 - Port: tcp:22: listening: matches expectation: [true]
ok 15 - Port: tcp:22: ip: matches expectation: [["0.0.0.0"]]
ok 16 - Port: tcp6:22: listening: matches expectation: [true]
ok 17 - Port: tcp6:22: ip: matches expectation: [["::"]]
not ok 18 - Package: docker-ce: installed: doesn't match, expect: [true] found: [false]
ok 19 - Service: sshd: enabled: matches expectation: [true]
ok 20 - Service: sshd: running: matches expectation: [true]
not ok 21 - Service: docker: enabled: doesn't match, expect: [true] found: [false]
not ok 22 - Service: docker: running: doesn't match, expect: [true] found: [false]
```

Figure 1: Output from the goss test run.



```
root@node0:~# goss -g /etc/goss/goss.yaml v -f tap
1..18
ok 1 - Mount: /: exists: matches expectation: [true]
ok 2 - Mount: /: filesystem: matches expectation: ["overlay"]
ok 3 - Mount: /: usage: matches expectation: [{"lt":90}]
ok 4 - User: sshd: exists: matches expectation: [true]
ok 5 - DNS: localhost: resolvable: matches expectation: [true]
ok 6 - DNS: localhost: addrs: matches expectation: [{"consist-of":["127.0.0.1","::1"]}]
ok 7 - KernelParam: kernel.ostype: value: matches expectation: ["Linux"]
ok 8 - Process: containerd: running: matches expectation: [true]
ok 9 - Process: sshd: running: matches expectation: [true]
ok 10 - Port: tcp:22: listening: matches expectation: [true]
ok 11 - Port: tcp:22: ip: matches expectation: [["0.0.0.0"]]
ok 12 - Port: tcp6:22: listening: matches expectation: [true]
ok 13 - Port: tcp6:22: ip: matches expectation: [["::"]]
ok 14 - Package: docker-ce: installed: matches expectation: [true]
ok 15 - Service: sshd: enabled: matches expectation: [true]
ok 16 - Service: sshd: running: matches expectation: [true]
ok 17 - Service: docker: enabled: matches expectation: [true]
ok 18 - Service: docker: running: matches expectation: [true]
```

Figure 2: The goss test run after fixing.

headaches later. If you run the command pipeline:

```
wget "https://get.docker.com" ⤶
  -O get-docker.sh && ⤶
  sh get-docker.sh && ⤶
  systemctl enable --now docker
```

to install and set up Docker Engine, then the `goss` run will show all green (you could turn off the colors with the `--no-color` validation option) to indicate that everything is in place on the test server (Figure 2).

Goss provides additional resources to build tests covering almost every aspect of a modern running machine. You can browse through goss resources [3] [4] to discover all the tests it currently provides. Also, when you're done with the test server, don't forget to clean it up with the command:

```
footloose delete
```

You don't always need to create the goss test configuration from scratch. You can use the `add` command to append a test for a resource. If the mentioned resources are absent, then the added tests ensure they do not exist somewhere on the system. It's recommended you use the `add` command on a fully configured system matching your desired end state. An `autoadd` command automatically adds many, but not all, existing resources in your server by matching the provided argument. You should delve into the goss documentation to try your hand at these commands.

More Power to goss

So far I have used `goss` at the command line, which is good for basic uses; however, you can also serve the testing reports continuously on an HTTP endpoint by creating a self-acceptance-testing server with this feature and running goss as a system service.

To set up this scenario, I'll create another base image to test more goodies provided by goss in a directory where the Dockerfile shown in Listing 1 and the setup script shown in Listing 4 are located:

```
docker build ⤶
  -f Dockerfile_ServerBase . ⤶
  -t ubuntu2204wgossrvc
```

Note that Listing 4 shows only the additions made to Listing 2 that were revealed by a diff.

Now bring up your new test node by changing `footloose.yaml`, as shown in Figure 3, and executing

Listing 4: setup_goss.sh Diff Additions

```
5a6
> GOSSVFLE='/lib/systemd/system/goss.service'
9,10c10,12
<     sha256sum
<     tee
---
>     sha256sum
>     systemctl
>     tee
81a84,91
>   tcp:58080:
>     listening: true
>     ip:
>     - 0.0.0.0
>   tcp6:58080:
>     listening: true
>     ip:
>     - '::'
97a108,110
>   goss:
>     enabled: true
>     running: true
103a117,118
>   goss:
>     running: true
114a130,156
> setupGosSrvc() {
>
>   tee "${GOSSVFLE}" <<'EOF'
> [Unit]
> Description=GOSS - Quick and Easy server validation
> After=network.target
> Documentation=https://github.com/aelsabbahy/goss/
    blob/master/docs/manual.md
>
> [Service]
> ExecStart=/usr/local/bin/goss -g /etc/goss/goss.yaml
    s -l :58080 -f documentation -e /status
> ExecStop=/bin/kill -s QUIT ${MAINPID}
> Restart=on-abnormal
> StandardOutput=journal
> StandardError=journal
>
> [Install]
> WantedBy=multi-user.target
> EOF
>
>   if ! systemctl enable goss
>   then
>     echo ' systemctl enable goss failed, exiting
...'
>     exit 1
>   fi
>
> }
>
119a162
>   setupGosSrvc
```

the command `footloose create`. You should now see the test node port 58080 exposed on a local port when you run the `footloose show` command. If you hit that HTTP port with the command

```
curl localhost:⮐
  <localhost exposed 58080 port>/status
```

you'll see a dump of the goss test report. Now that you have a working self-acceptance test running on the machine, you could further integrate the goss HTTP endpoint with your monitoring and alerting system to roll out a configuration drift detection system.

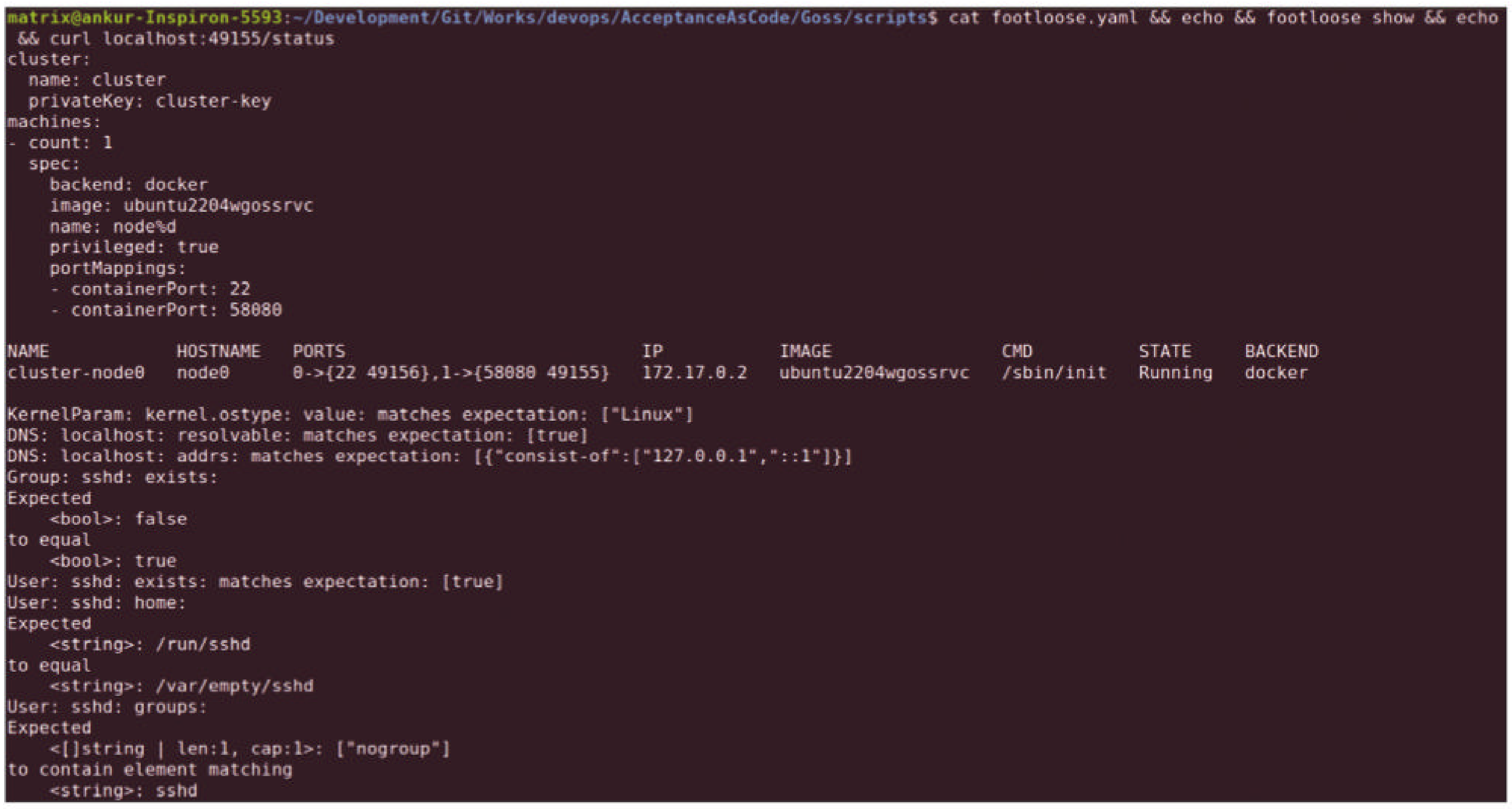
Until now, I have hard-coded properties and put the entire test configuration in a single file. This approach

Listing 5: Aggregation Test Files

```
tee /tmp/kernel-param.yaml <<EOF
kernel-param:
  kernel.ostype:
    value: Linux
EOF

tee /tmp/mount.yaml <<EOF
mount:
  /:
    exists: true
    filesystem: overlay
    usage:
      lt: 90
EOF

tee /tmp/port.yaml <<EOF
port:
  tcp:22:
    listening: true
    ip:
    - 0.0.0.0
  tcp6:22:
    listening: true
    ip:
    - '::'
EOF

tee /tmp/user.yaml <<EOF
user:
  sshd:
    exists: true
EOF

tee /tmp/package.yaml <<EOF
package:
  docker-ce:
    installed: true
EOF

tee /tmp/service.yaml <<EOF
service:
  sshd:
    enabled: true
    running: true
  docker:
    enabled: true
    running: true
EOF

tee /tmp/process.yaml <<EOF
process:
  sshd:
    running: true
  containerd:
    running: true
EOF

tee /tmp/dns.yaml <<EOF
dns:
  localhost:
    resolvable: true
    addrs:
      consist-of: ["127.0.0.1","::1"]
    timeout: 500 # in milliseconds
EOF

tee /tmp/goss.yaml <<EOF
gossfile:
  /tmp/kernel-param.yaml: {}
  /tmp/mount.yaml: {}
  /tmp/port.yaml: {}
  /tmp/user.yaml: {}
  /tmp/package.yaml: {}
  /tmp/service.yaml: {}
  /tmp/process.yaml: {}
  /tmp/dns.yaml: {}
EOF
```

Listing 6: Local Looping Template

```
addr:
{{- range mkSlice "kafka0" "kafka1"
"kafka2"}}

  tcp://{{.}}:2181:
  reachable: true
  timeout: 500

  tcp://{{.}}:9092:
  reachable: true
  timeout: 500
{{end}}
```



```
matrix@ankur-Inspiron-5593:~/Development/Git/Works/devops/AcceptanceAsCode/Goss/scripts$ cat footloose.yaml && echo && footloose show && echo
 && curl localhost:49155/status
cluster:
  name: cluster
  privateKey: cluster-key
machines:
- count: 1
  spec:
    backend: docker
    image: ubuntu2204wgossrvc
    name: node%d
    privileged: true
    portMappings:
    - containerPort: 22
    - containerPort: 58080

NAME            HOSTNAME   PORTS                            IP           IMAGE                CMD          STATE     BACKEND
cluster-node0   node0      0->{22 49156},1->{58080 49155}   172.17.0.2   ubuntu2204wgossrvc   /sbin/init   Running   docker

KernelParam: kernel.ostype: value: matches expectation: ["Linux"]
DNS: localhost: resolvable: matches expectation: [true]
DNS: localhost: addrs: matches expectation: [{"consist-of":["127.0.0.1","::1"]}]
Group: sshd: exists:
Expected
    <bool>: false
to equal
    <bool>: true
User: sshd: exists: matches expectation: [true]
User: sshd: home:
Expected
    <string>: /run/sshd
to equal
    <string>: /var/empty/sshd
User: sshd: groups:
Expected
    <[]string | len:1, cap:1>: ["nogroup"]
to contain element matching
    <string>: sshd
```

Figure 3: Modified `footloose.yaml` and HTTP endpoint.

could work for a small fleet of servers, but you need more dynamic behavior and modularity in configuration to work with server automation for a medium to large fleet of servers. Goss provides a gossfile resource to render the final test configuration by aggregating multiple separate configurations. For example, you could create separate test configuration files for each resource in your overall test suite and manage a large number of tests easily.

I use this functionality to derive the final test configuration, at runtime, from the tests common to each server along with server-specific tests. Listing 5 creates the necessary resource-specific test files.

To render a valid goss test configuration, type the command

```
goss -g /tmp/goss.yaml r
```

in your terminal, and you should see a configuration dumped to your screen similar to that used in the earlier section. Now you can validate your server with the command:

```
goss -g /tmp/goss.yaml v -f tap
```

Goss provides templating functionality to generate tests dynamically and interpolate values for various properties. Value interpolation could use environmental variables or values provided in YAML or JSON files with the `--vars` option. The test snippet in Listing 6 is a quick example demonstrating the loop functionality to ensure required interconnectivity in a three-node Kafka cluster.

Listing 7: Dockerfile_DgossDriver

```
FROM alpine:3.12
LABEL "com.richnusgeeks.vendor"="richnusgeeks"
LABEL version="latest"
LABEL description="dgoss test driver docker image"

ENV GOSS_VERSION 0.3.20

SHELL ["/bin/ash", "-o", "pipefail", "-c"]
WORKDIR /tmp
RUN apk add --no-cache --virtual=goss-deps ca-certificates curl && apk add --no-cache bash tini &&
  curl -sSLk -o /tmp/docker.tgz "https://download.docker.com/linux/static/stable/x86_64/$(curl -sSkL https://download.docker.com/
  linux/static/stable/x86_64/|grep '^ *<a'|grep docker|grep -v rootless|awk -F '\"' '{print $2}'|sort -nr|head -1)" &&
  tar zxf docker.tgz && mv docker/docker /usr/local/bin && rm -rf docker docker.tgz &&
  curl -sSLk "https://github.com/aelsabbahy/goss/releases/download/v${GOSS_VERSION}/goss-linux-amd64" -o /usr/local/bin/goss &&
  curl -sSLk "https://github.com/aelsabbahy/goss/releases/download/v${GOSS_VERSION}/dgoss" -o /usr/local/bin/dgoss &&
  chmod +x /usr/local/bin/*goss && mkdir -p /etc/goss && apk del goss-deps

WORKDIR /etc/goss/

ENTRYPOINT ["tini", "--"]
CMD ["dgoss", "-h"]
```

Container Validation with dgoss

In many cases you only have access to the physical or virtual servers at the application level. Containers have largely replaced running native applications. Packing the applications in container images and running them containerized makes everything immutable to upgrade and rollback on the fly, with fine-grained control over server resources. This scheme also greatly improves the packing density of applications to optimize the available server resources.

Containers present different kinds of challenges, though, because their true colors only show at runtime. Container images are always expected to be as complete as possible but have the minimum possible footprint to launch and keep containerized applications running dynamically. Acceptance testing becomes very important with containers because software version control and configuration drift are a problem. I myself have witnessed folks creating containers with images without any standards and then suffering major outages because of an unstable life cycle. The good news is that you can use your knowledge of goss to create Acceptance-as-Code gates of good quality for container workloads.

The `dgoss` shell wrapper over goss adds some extra functionality [5]. One benefit is that you can use dgoss to create an Acceptance-as-Code gate in your container image's build and release pipelines. The usual way to run `dgoss` is to download it and then launch it with your goss test configuration, but I'll use dgoss through a Docker driver image to run it containerized without downloading anything.

To begin, I'll create a dgoss driver image with all the necessary artifacts in the working directory where the Dockerfile (Listing 7) and `goss.yaml` file (Listing 8) are located:

Listing 8: goss.yaml

```
port:
  tcp:22:
    listening: true
    ip:
    - 0.0.0.0
  tcp6:22:
    listening: true
    ip:
    - '::'
user:
  sshd:
    exists: true
    uid: 101
    gid: 65534
    groups:
    - nogroup
    home: /run/sshd
    shell: /usr/sbin/nologin
process:
  sshd:
    running: true
```

```
docker build -f Dockerfile_DGossDriver . ↩
  -t dgoss0320driver
```

Now you can see dgoss in action by running the command:

```
docker run -it --rm ↩
  -v ${PWD}/:/etc/goss:ro ↩
  -v /var/run/docker.sock:↩
     /var/run/docker.sock:ro ↩
  -e GOSS_SLEEP=2 ↩
  -e GOSS_FILES_STRATEGY=cp ↩
  -e GOSS_OPTS='--color -format tap' ↩
  dgoss0320driver dgoss run ↩
  dgoss0320driver sleep infinity
```

Listing 9: spark.yaml

```
port:
  tcp:7077:
    listening: true
    ip:
    - 0.0.0.0
  tcp6:7077:
    listening: true
    ip:
    - ::
  tcp:8080:
    listening: true
    ip:
    - 0.0.0.0
  tcp6:8080:
    listening: true
    ip:
    - ::
process:
  java:
    running: true
```

The failures flashed in the run (Figure 4) are from a quick testing criterion of sshd running in containers, which is not true in the case of this image. The image is not a continuously running service, and that's why I supplied a `sleep` command, so that the launched service container is there during dgoss operations, such as copying goss and `goss.yaml` into it and launching a goss test run in the newly launched service container. The dgoss behavior is configurable by a number of environmental variables.

A more standard example of a popular Docker image is the high-performance Apache Spark analytics engine. To begin, pull the official Spark Docker image:

```
docker pull bitnami/spark:latest
```

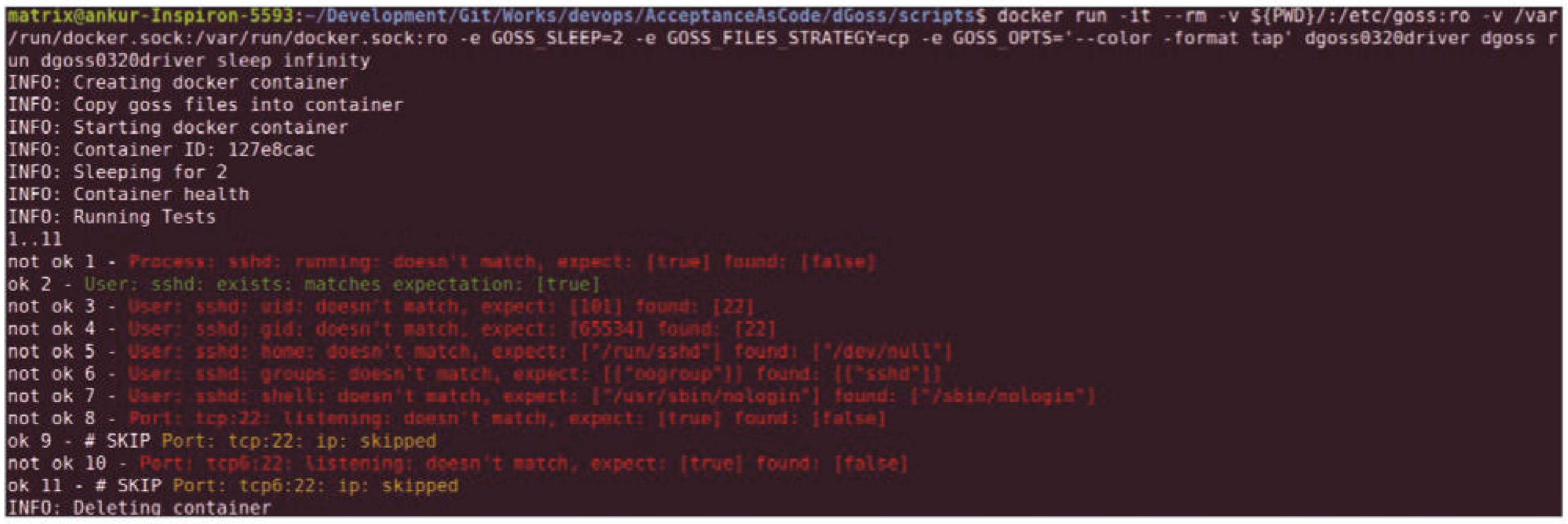
Next, prepare the `spark.yaml` goss test configuration file in your current working directory (Listing 9).

Now, run the command

```
docker run -it --rm ↩
  -v ${PWD}/:/etc/goss:ro ↩
  -v /var/run/docker.sock:↩
     /var/run/docker.sock:ro ↩
  -e GOSS_FILE=spark.yaml ↩
  -e GOSS_FILES_STRATEGY=cp ↩
  -e GOSS_OPTS='--color -format tap' ↩
  dgoss0320driver dgoss run ↩
  -e SPARK_MODE=master bitnami/spark
```

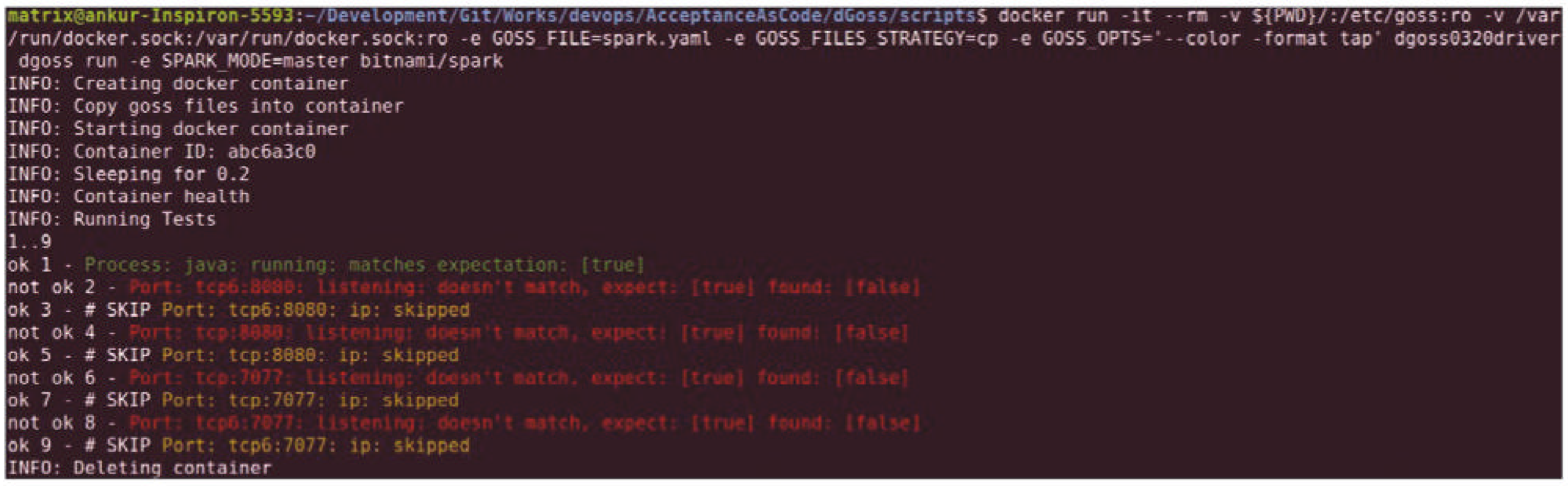
Listing 10: goss_wait.yaml

```
port:
  tcp:22:
    listening: false
    ip:
    - 0.0.0.0
user:
  sshd:
    exists: false
process:
  sshd:
    running: false
```



```
matrix@ankur-Inspiron-5593:~/Development/Git/Works/devops/AcceptanceAsCode/dGoss/scripts$ docker run -it --rm -v ${PWD}/:/etc/goss:ro -v /var
/run/docker.sock:/var/run/docker.sock:ro -e GOSS_SLEEP=2 -e GOSS_FILES_STRATEGY=cp -e GOSS_OPTS='--color -format tap' dgoss0320driver dgoss r
un dgoss0320driver sleep infinity
INFO: Creating docker container
INFO: Copy goss files into container
INFO: Starting docker container
INFO: Container ID: 127e8cac
INFO: Sleeping for 2
INFO: Container health
INFO: Running Tests
1..11
not ok 1 - Process: sshd: running: doesn't match, expect: [true] found: [false]
ok 2 - User: sshd: exists: matches expectation: [true]
not ok 3 - User: sshd: uid: doesn't match, expect: [101] found: [22]
not ok 4 - User: sshd: gid: doesn't match, expect: [65534] found: [22]
not ok 5 - User: sshd: home: doesn't match, expect: ["/run/sshd"] found: ["/dev/null"]
not ok 6 - User: sshd: groups: doesn't match, expect: [["nogroup"]] found: [["sshd"]]
not ok 7 - User: sshd: shell: doesn't match, expect: ["/usr/sbin/nologin"] found: ["/sbin/nologin"]
not ok 8 - Port: tcp:22: listening: doesn't match, expect: [true] found: [false]
ok 9 - # SKIP Port: tcp:22: ip: skipped
not ok 10 - Port: tcp6:22: listening: doesn't match, expect: [true] found: [false]
ok 11 - # SKIP Port: tcp6:22: ip: skipped
INFO: Deleting container
```

Figure 4: Output from a dgoss test run for the driver container.



```
matrix@ankur-Inspiron-5593:~/Development/Git/Works/devops/AcceptanceAsCode/dGoss/scripts$ docker run -it --rm -v ${PWD}/:/etc/goss:ro -v /var
/run/docker.sock:/var/run/docker.sock:ro -e GOSS_FILE=spark.yaml -e GOSS_FILES_STRATEGY=cp -e GOSS_OPTS='--color -format tap' dgoss0320driver
 dgoss run -e SPARK_MODE=master bitnami/spark
INFO: Creating docker container
INFO: Copy goss files into container
INFO: Starting docker container
INFO: Container ID: abc6a3c0
INFO: Sleeping for 0.2
INFO: Container health
INFO: Running Tests
1..9
ok 1 - Process: java: running: matches expectation: [true]
not ok 2 - Port: tcp6:8080: listening: doesn't match, expect: [true] found: [false]
ok 3 - # SKIP Port: tcp6:8080: ip: skipped
not ok 4 - Port: tcp:8080: listening: doesn't match, expect: [true] found: [false]
ok 5 - # SKIP Port: tcp:8080: ip: skipped
not ok 6 - Port: tcp:7077: listening: doesn't match, expect: [true] found: [false]
ok 7 - # SKIP Port: tcp:7077: ip: skipped
not ok 8 - Port: tcp6:7077: listening: doesn't match, expect: [true] found: [false]
ok 9 - # SKIP Port: tcp6:7077: ip: skipped
INFO: Deleting container
```

Figure 5: A dgoss test run for the Spark container.

to launch a test run against the official Spark image. It looks a bit surprising that, although the Spark Java service is running, the master and user interface port tests are failing (Figure 5).

I'll try once again by executing the command:

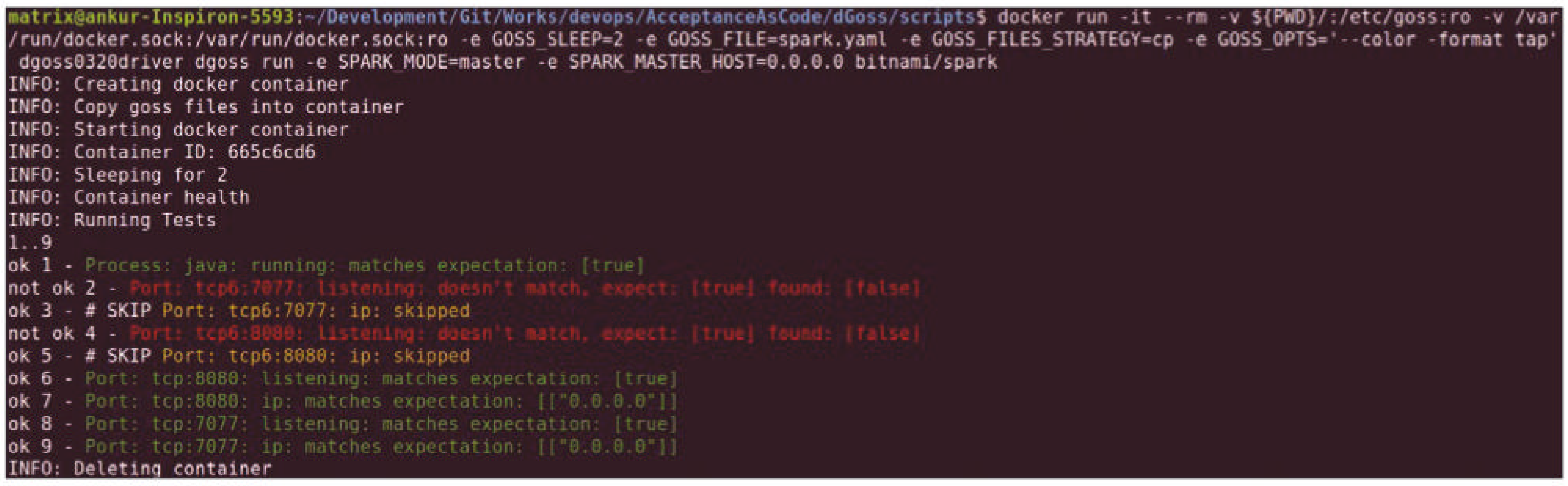
```
docker run -it --rm ↩
  -v ${PWD}/:/etc/goss:ro ↩
  -v /var/run/docker.sock:↩
    /var/run/docker.sock:ro ↩
  -e GOSS_SLEEP=2 ↩
  -e GOSS_FILE=spark.yaml ↩
  -e GOSS_FILES_STRATEGY=cp ↩
  -e GOSS_OPTS='--color -format tap' ↩
  dgoss0320driver dgoss run ↩
  -e SPARK_MODE=master bitnami/spark
```

Now things look better. If you have a Dockerized service that has startup delays to run and respond (e.g., Java, Ruby, etc.), then `GOSS_SLEEP` helps to delay before launching a goss run in the spawned service container. You should see all the IPv4 port tests passing now (Figure 6).

Another important dgoss feature is useful when you have a set of precondition tests. If you create the `goss_wait.yaml` file (Listing 10) in your working directory and execute the `docker` command shown in the previous section, it should wait for the tests in the `goss_wait.yaml` file to be passed. In this case, I ensured that no SSH functionality was part of the Spark image (Figure 7).

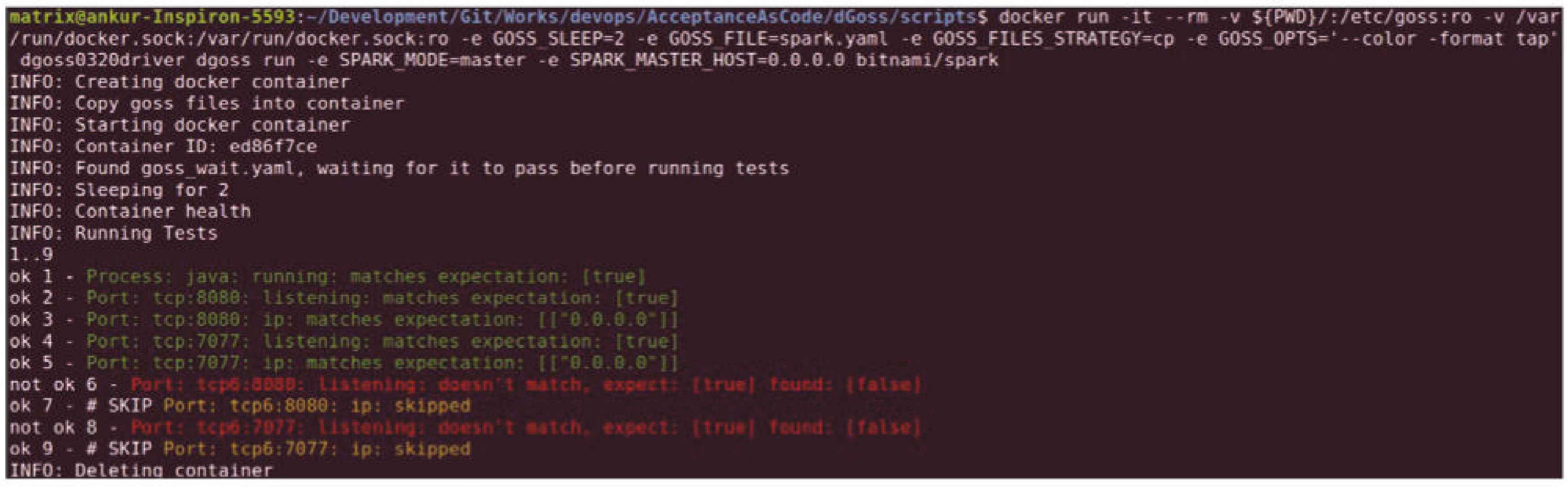
Listing 11: `Dockerfile_DCGossDriver`

```
FROM alpine:3.16
LABEL "com.richnusgeeks.vendor"="richnusgeeks"
LABEL version="latest"
LABEL description="dcgoss test driver docker image"

ENV DCPS_VERSION 2.12.2
ENV GOSS_VERSION 0.3.20

SHELL ["/bin/ash", "-o", "pipefail", "-c"]
WORKDIR /tmp
RUN apk add --no-cache --virtual=goss-deps ca-certificates curl && apk add --no-cache bash tini &&
  curl -sSLk -o /tmp/docker.tgz "https://download.docker.com/linux/static/stable/x86_64/$(curl -sSkL
  https://download.docker.com/linux/static/stable/x86_64/|grep '^ *<a'|grep docker|grep -v rootless|awk
  -F '\"' '{print $2}'|sort -nr|head -1)" && tar zxf docker.tgz && mv docker/docker /usr/local/bin &&
  rm -rf docker docker.tgz && curl -sSLk "https://github.com/docker/compose/releases/download/v${DCPS_
  VERSION}/docker-compose-linux-x86_64" -o /usr/local/bin/docker-compose && chmod +x /usr/local/bin/
  docker-compose && curl -sSLk "https://github.com/aelsabbahy/goss/releases/download/v${GOSS_VERSION}/
  goss-linux-amd64" -o /usr/local/bin/goss && curl -sSLk "https://raw.githubusercontent.com/aelsabbahy/
  goss/master/extras/dcgoss/dcgoss" -o /usr/local/bin/dcgoss && chmod +x /usr/local/bin/*goss && mkdir
  -p /etc/goss && apk del goss-deps

WORKDIR /etc/goss/

ENTRYPOINT ["/sbin/tini", "--"]
CMD ["dcgoss", "-h"]
```



```
matrix@ankur-Inspiron-5593:~/Development/Git/Works/devops/AcceptanceAsCode/dGoss/scripts$ docker run -it --rm -v ${PWD}/:/etc/goss:ro -v /var
/run/docker.sock:/var/run/docker.sock:ro -e GOSS_SLEEP=2 -e GOSS_FILE=spark.yaml -e GOSS_FILES_STRATEGY=cp -e GOSS_OPTS='--color -format tap'
 dgoss0320driver dgoss run -e SPARK_MODE=master -e SPARK_MASTER_HOST=0.0.0.0 bitnami/spark
INFO: Creating docker container
INFO: Copy goss files into container
INFO: Starting docker container
INFO: Container ID: 665c6cd6
INFO: Sleeping for 2
INFO: Container health
INFO: Running Tests
1..9
ok 1 - Process: java: running: matches expectation: [true]
not ok 2 - Port: tcp6:7077: listening: doesn't match, expect: [true] found: [false]
ok 3 - # SKIP Port: tcp6:7077: ip: skipped
not ok 4 - Port: tcp6:8080: listening: doesn't match, expect: [true] found: [false]
ok 5 - # SKIP Port: tcp6:8080: ip: skipped
ok 6 - Port: tcp:8080: listening: matches expectation: [true]
ok 7 - Port: tcp:8080: ip: matches expectation: [["0.0.0.0"]]
ok 8 - Port: tcp:7077: listening: matches expectation: [true]
ok 9 - Port: tcp:7077: ip: matches expectation: [["0.0.0.0"]]
INFO: Deleting container
```

Figure 6: The dgoss test run with `GOSS_SLEEP` for the Spark container.



```
matrix@ankur-Inspiron-5593:~/Development/Git/Works/devops/AcceptanceAsCode/dGoss/scripts$ docker run -it --rm -v ${PWD}/:/etc/goss:ro -v /var
/run/docker.sock:/var/run/docker.sock:ro -e GOSS_SLEEP=2 -e GOSS_FILE=spark.yaml -e GOSS_FILES_STRATEGY=cp -e GOSS_OPTS='--color -format tap'
 dgoss0320driver dgoss run -e SPARK_MODE=master -e SPARK_MASTER_HOST=0.0.0.0 bitnami/spark
INFO: Creating docker container
INFO: Copy goss files into container
INFO: Starting docker container
INFO: Container ID: ed86f7ce
INFO: Found goss_wait.yaml, waiting for it to pass before running tests
INFO: Sleeping for 2
INFO: Container health
INFO: Running Tests
1..9
ok 1 - Process: java: running: matches expectation: [true]
ok 2 - Port: tcp:8080: listening: matches expectation: [true]
ok 3 - Port: tcp:8080: ip: matches expectation: [["0.0.0.0"]]
ok 4 - Port: tcp:7077: listening: matches expectation: [true]
ok 5 - Port: tcp:7077: ip: matches expectation: [["0.0.0.0"]]
not ok 6 - Port: tcp6:8080: listening: doesn't match, expect: [true] found: [false]
ok 7 - # SKIP Port: tcp6:8080: ip: skipped
not ok 8 - Port: tcp6:7077: listening: doesn't match, expect: [true] found: [false]
ok 9 - # SKIP Port: tcp6:7077: ip: skipped
INFO: Deleting container
```

Figure 7: A dgoss test run with `goss_wait.yaml` for a Spark container.

Try to change a test in the `goss_wait.yaml` to `true`, and you should see dgoss abort the test run because of the test failure. The dgoss run after setting the `sshd` process running state to `true` in `goss_wait.yaml` is shown in Figure 8.

Listing 12: `docker-compose.yml`

```
version: "3"
services:
  zookeeper:
    image: 'bitnami/zookeeper:latest'
    ports:
      - '2181:2181'
    environment:
      - ALLOW_ANONYMOUS_LOGIN=yes
  kafka:
    image: 'bitnami/kafka:latest'
    ports:
      - '9092:9092'
    environment:
      - KAFKA_BROKER_ID=1
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://127.0.0.1:9092
      - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
      - ALLOW_PLAINTEXT_LISTENER=yes
    depends_on:
      - zookeeper
```

Container Stack Validation with dcgoss

Docker Compose is a popular tool that lets you define and run multicontainer applications. Another wrapper created over dgoss tests the containers of the entire Compose stack: Put the code in Listing 11 in your current working directory and run the command

```
docker build ➘
  -f Dockerfile_DCGossDriver . ➘
  -t dcgoss0320driver
```

Listing 13: `kafka.yaml`

```
port:
  tcp:2181:
    listening: true
    ip:
    - 0.0.0.0
  tcp:9092:
    listening: true
    ip:
    - 0.0.0.0

process:
  java:
    running: true
```

to build the driver image.

Next, create the `docker-compose.yml` file shown in Listing 12 and the `kafka.yaml` file shown in Listing 13 to bring up the ZooKeeper plus Kafka container stack over which the goss tests are run.

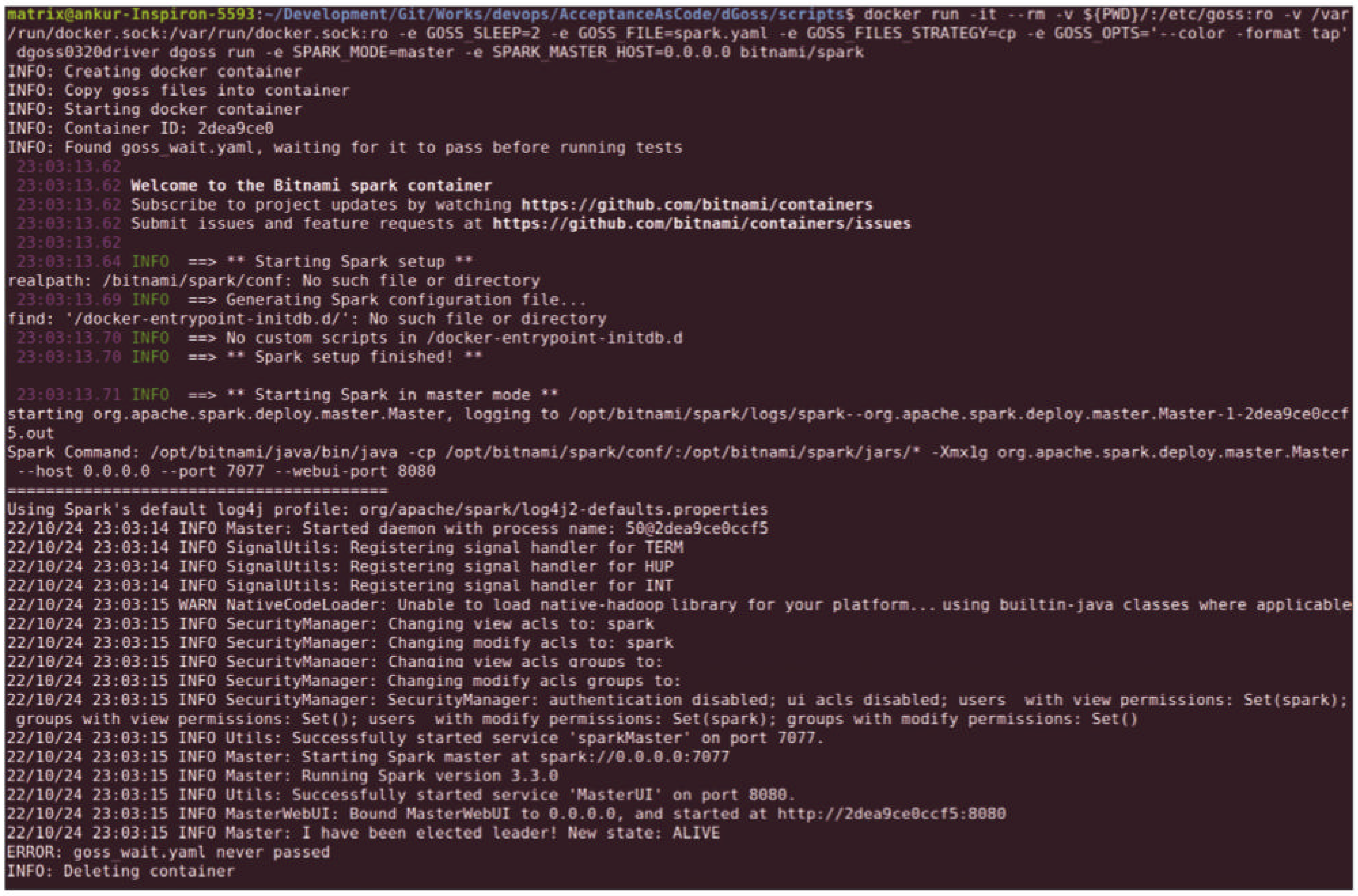
Now you can fire `dcgoss` to acceptance-test each service container,

```
docker run -it --rm ➘
  -v ${PWD}/:/etc/goss:ro ➘
  -v /var/run/docker.sock:➘
     /var/run/docker.sock:ro ➘
  -e GOSS_FILE=kafka.yaml ➘
  -e GOSS_SLEEP=5 ➘
  -e GOSS_FILES_STRATEGY=cp ➘
  -e GOSS_OPTS='--color ➘
                -format tap' ➘
  dcgoss0320driver dcgoss run zookeeper
```

and run all goss tests in the ZooKeeper service container. You should see that all the ZooKeeper tests pass (Figure 9).

To run an acceptance test for the Kafka container, use the command:

```
docker run -it --rm ➘
  -v ${PWD}/:/etc/goss:ro ➘
```



```
matrix@ankur-Inspiron-5593:~/Development/Git/Works/devops/AcceptanceAsCode/dGoss/scripts$ docker run -it --rm -v ${PWD}/:/etc/goss:ro -v /var
/run/docker.sock:/var/run/docker.sock:ro -e GOSS_SLEEP=2 -e GOSS_FILE=spark.yaml -e GOSS_FILES_STRATEGY=cp -e GOSS_OPTS='--color -format tap'
 dgoss0320driver dgoss run -e SPARK_MODE=master -e SPARK_MASTER_HOST=0.0.0.0 bitnami/spark
INFO: Creating docker container
INFO: Copy goss files into container
INFO: Starting docker container
INFO: Container ID: 2dea9ce0
INFO: Found goss_wait.yaml, waiting for it to pass before running tests
 23:03:13.62
 23:03:13.62 Welcome to the Bitnami spark container
 23:03:13.62 Subscribe to project updates by watching https://github.com/bitnami/containers
 23:03:13.62 Submit issues and feature requests at https://github.com/bitnami/containers/issues
 23:03:13.62
 23:03:13.64 INFO  ==> ** Starting Spark setup **
realpath: /bitnami/spark/conf: No such file or directory
 23:03:13.69 INFO  ==> Generating Spark configuration file...
find: '/docker-entrypoint-initdb.d/': No such file or directory
 23:03:13.70 INFO  ==> No custom scripts in /docker-entrypoint-initdb.d
 23:03:13.70 INFO  ==> ** Spark setup finished! **

 23:03:13.71 INFO  ==> ** Starting Spark in master mode **
starting org.apache.spark.deploy.master.Master, logging to /opt/bitnami/spark/logs/spark--org.apache.spark.deploy.master.Master-1-2dea9ce0ccf
5.out
Spark Command: /opt/bitnami/java/bin/java -cp /opt/bitnami/spark/conf/:/opt/bitnami/spark/jars/* -Xmx1g org.apache.spark.deploy.master.Master
 --host 0.0.0.0 --port 7077 --webui-port 8080
========================================
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
22/10/24 23:03:14 INFO Master: Started daemon with process name: 50@2dea9ce0ccf5
22/10/24 23:03:14 INFO SignalUtils: Registering signal handler for TERM
22/10/24 23:03:14 INFO SignalUtils: Registering signal handler for HUP
22/10/24 23:03:14 INFO SignalUtils: Registering signal handler for INT
22/10/24 23:03:15 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
22/10/24 23:03:15 INFO SecurityManager: Changing view acls to: spark
22/10/24 23:03:15 INFO SecurityManager: Changing modify acls to: spark
22/10/24 23:03:15 INFO SecurityManager: Changing view acls groups to:
22/10/24 23:03:15 INFO SecurityManager: Changing modify acls groups to:
22/10/24 23:03:15 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(spark);
 groups with view permissions: Set(); users  with modify permissions: Set(spark); groups with modify permissions: Set()
22/10/24 23:03:15 INFO Utils: Successfully started service 'sparkMaster' on port 7077.
22/10/24 23:03:15 INFO Master: Starting Spark master at spark://0.0.0.0:7077
22/10/24 23:03:15 INFO Master: Running Spark version 3.3.0
22/10/24 23:03:15 INFO Utils: Successfully started service 'MasterUI' on port 8080.
22/10/24 23:03:15 INFO MasterWebUI: Bound MasterWebUI to 0.0.0.0, and started at http://2dea9ce0ccf5:8080
22/10/24 23:03:15 INFO Master: I have been elected leader! New state: ALIVE
ERROR: goss_wait.yaml never passed
INFO: Deleting container
```

Figure 8: A dgoss test run with a failing `goss_wait.yaml`.

```
  -v /var/run/docker.sock:➘
     /var/run/docker.sock:ro ➘
  -e GOSS_FILE=kafka.yaml ➘
  -e GOSS_SLEEP=5 ➘
  -e GOSS_FILES_STRATEGY=cp ➘
  -e GOSS_OPTS='--color ➘
               -format tap' ➘
  dcgoss0320driver dcgoss run kafka
```

You should see all the Kafka tests passing now (Figure 10).

I have combined the ZooKeeper and Kafka goss tests in a common file for quick testing, showing *ok*, *not ok*, and *skip* test results, but you could separate the tests into their respective gossfiles and mention the corresponding file with the `GOSS_FILE` environment variable. Finally, to clean up the running service containers, use the command

```
docker ps -q|xargs -I % docker rm -f %
```

to say goodbye to dcgoss.

Last but not the least, Kubernetes has become a de facto platform to run and manage containerized services at scale, so the goss project also provides another wrapper in the form of `kgoss` to acceptance-test containers running in Kubernetes pods. You need to run `kgoss` on a machine with `kubectl` already installed and configured to talk to your local or remote Kubernetes cluster. The kgoss documentation [6] clearly describes the installation and usage.

Conclusion

Server misconfigurations can create a lot of havoc, resulting in big losses. Goss is an elegant modern solution to bake acceptance testing in your servers. The validation tool is highly dynamic, supporting templating and serving reports on an HTTP endpoint. The icing on the cake is its added value when integrated with modern cloud tools like Ansible, Molecule, Kitchen, and Packer. The goss wrappers created for services running in containers are very useful for quality gating in modern build-and-release pipelines. ■

Info

[1] goss: [https://github.com/goss-org/goss]

[2] goss documentation: [https://github.com/aelsabbahy/goss/blob/master/docs/manual.md]

[3] goss resources and usage: [https://github.com/aelsabbahy/goss/blob/master/docs/manual.md#available-tests]

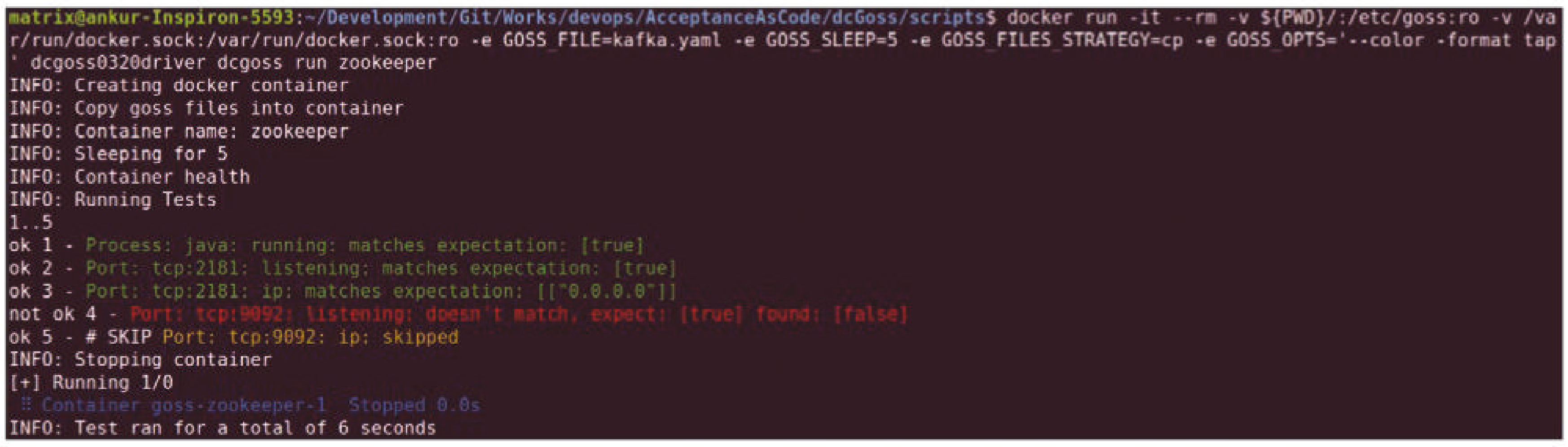
[4] goss add-ons from the community: [https://github.com/aelsabbahy/goss#community-contributions]

[5] dgoss documentation: [https://github.com/aelsabbahy/goss/tree/master/extras/dgoss]

[6] kgoss documentation: [https://github.com/aelsabbahy/goss/tree/master/extras/kgoss]

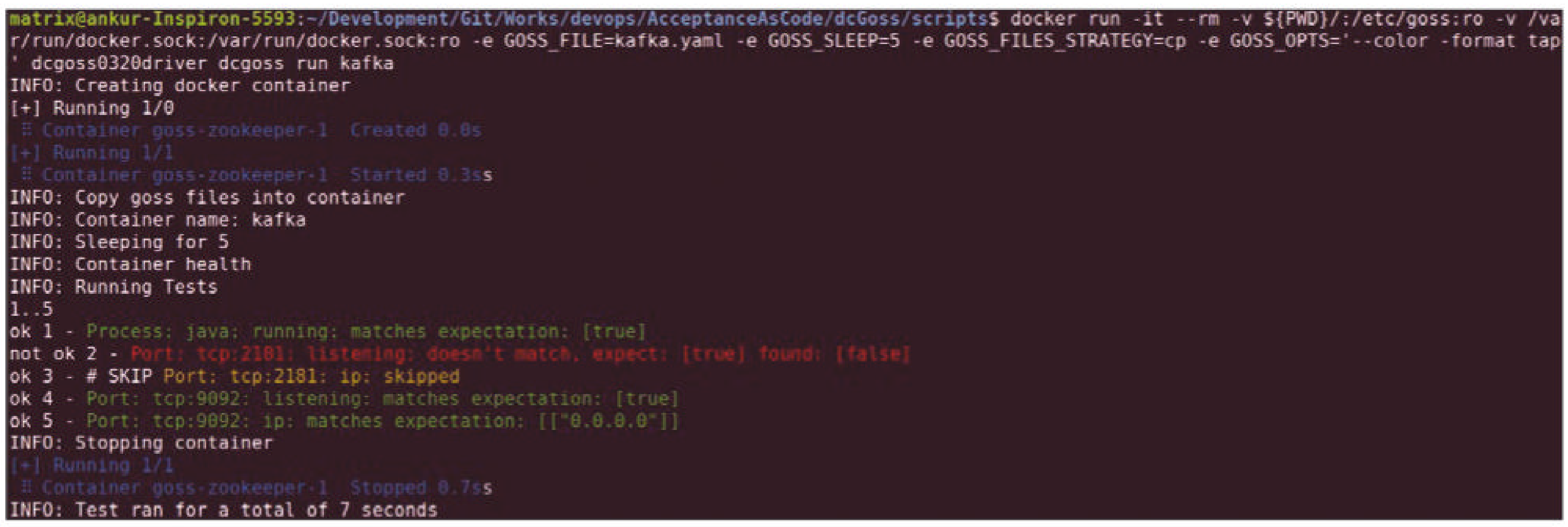
The Author

Ankur Kumar is a passionate, free open source hacker, researcher, and seeker of mystical life knowledge. He loves exploring cutting edge technologies, ancient sciences, quantum spirituality, various genres of music, mystical literature, and art. You can connect with Ankur on LinkedIn ([https://www.linkedin.com/in/richnusgeeks]) and explore his GitHub page ([https://github.com/richnusgeeks]) for other useful FOSS pieces.



```
matrix@ankur-Inspiron-5593:~/Development/Git/Works/devops/AcceptanceAsCode/dcGoss/scripts$ docker run -it --rm -v ${PWD}/:/etc/goss:ro -v /va
r/run/docker.sock:/var/run/docker.sock:ro -e GOSS_FILE=kafka.yaml -e GOSS_SLEEP=5 -e GOSS_FILES_STRATEGY=cp -e GOSS_OPTS='--color -format tap
' dcgoss0320driver dcgoss run zookeeper
INFO: Creating docker container
INFO: Copy goss files into container
INFO: Container name: zookeeper
INFO: Sleeping for 5
INFO: Container health
INFO: Running Tests
1..5
ok 1 - Process: java: running: matches expectation: [true]
ok 2 - Port: tcp:2181: listening: matches expectation: [true]
ok 3 - Port: tcp:2181: ip: matches expectation: [["0.0.0.0"]]
not ok 4 - Port: tcp:9092: listening: doesn't match, expect: [true] found: [false]
ok 5 - # SKIP Port: tcp:9092: ip: skipped
INFO: Stopping container
[+] Running 1/0
 ⠿ Container goss-zookeeper-1  Stopped 0.0s
INFO: Test ran for a total of 6 seconds
```

Figure 9: The dcgoss test run for the ZooKeeper service.



```
matrix@ankur-Inspiron-5593:~/Development/Git/Works/devops/AcceptanceAsCode/dcGoss/scripts$ docker run -it --rm -v ${PWD}/:/etc/goss:ro -v /va
r/run/docker.sock:/var/run/docker.sock:ro -e GOSS_FILE=kafka.yaml -e GOSS_SLEEP=5 -e GOSS_FILES_STRATEGY=cp -e GOSS_OPTS='--color -format tap
' dcgoss0320driver dcgoss run kafka
INFO: Creating docker container
[+] Running 1/0
 ⠿ Container goss-zookeeper-1  Created 0.0s
[+] Running 1/1
 ⠿ Container goss-zookeeper-1  Started 0.3ss
INFO: Copy goss files into container
INFO: Container name: kafka
INFO: Sleeping for 5
INFO: Container health
INFO: Running Tests
1..5
ok 1 - Process: java: running: matches expectation: [true]
not ok 2 - Port: tcp:2181: listening: doesn't match, expect: [true] found: [false]
ok 3 - # SKIP Port: tcp:2181: ip: skipped
ok 4 - Port: tcp:9092: listening: matches expectation: [true]
ok 5 - Port: tcp:9092: ip: matches expectation: [["0.0.0.0"]]
INFO: Stopping container
[+] Running 1/1
 ⠿ Container goss-zookeeper-1  Stopped 0.7ss
INFO: Test ran for a total of 7 seconds
```

Figure 10: The dcgoss test run for the Kafka service.

Photo by Erda Estremera on Unsplash

Package management tools for Windows

Unboxing

Chocolatey and WinGet offer full-fledged package management on Windows, but which is best for your environment? By Evgenij Smirnov

Installing and uninstalling applications in Windows traditionally involves running a graphical wizard. Unattended installs that depend on this type of installer are often troublesome. In comparison, installing an application on Linux is usually a piece of cake: Package managers such as Apt, Yum, and Zypper make sure that the sources are always up to date and resolve dependencies, and they usually do not require any parameters apart from the package name. However, Windows users also have some practical command-line interface (CLI) based package management tools.

In this article, I look at two solutions for full-fledged package managers on Windows: Chocolatey and Microsoft's WinGet.

Unique Installs

If you want to install a simple Windows program such as 7-Zip, GreenShot, or Notepad++, you usually download and launch the installer (EXE or MSI) and click through the graphical wizard, even though the same information is typically provided every time. All of the programs support unattended installation, but hardly anyone is familiar with the parameters, and it takes longer to look them up than to click through the installer. To keep applications up to date, you usually just install a newer version. Some programs, such as Notepad++ or KeePass, notify the user that an updated version is available, but downloading and installing are manual tasks.

Alternatives to the unique installation and updating of applications have been around for a long time. Since the introduction of the iPhone, proprietary app stores have become the standard for mobile devices, and Apple offers a similar solution for its macOS desktop operating system. Likewise, Microsoft has created a central platform for installing and updating applications in the form of the Microsoft Store, although the platform does not enjoy the same popularity among software providers as Apple's AppStore or Google Play – for many reasons. The bottom line is that even Windows end users tend to be reluctant to purchase software from the Microsoft Store.

Practical Repositories

For decades, Linux has successfully relied on a non-graphical approach for the large mass of open source and freeware applications and operating system-related tools: text-based package managers that access an extensible list of repositories to locate applications and libraries, determine their mutual dependencies, and ultimately enable installs, uninstalls, and upgrades. The packages listed in the repositories each have a specific format that varies from distribution to distribution.

The repositories are maintained in three ways:

- Distribution maintainers offer repositories that they pre-install when they ship their distribution. Most system-specific tools and libraries can be found here, but application programs are also sometimes included in the mainstream repositories to increase the versatility of the distribution. The maintainers vouch for the quality and mutual compatibility of the packages on offer, which means some serious overhead for testing new versions.
- Software vendors who develop applications for Linux, including Microsoft, often offer their own

repositories that contain only in-house products. If common libraries are required as dependencies, they need to be obtainable from another repository registered on the same system. Vendor-owned repositories usually contain the latest versions of the products, but the compatibility of the packages offered with individual distributions and versions can vary.

- Projects or special interest groups within a community often establish and maintain repositories for packages that distribution maintainers have dropped from mainstream feature sets – even though the software in the packages is still used by end users or other products need them as prerequisites.

This architecture, complex at first sight, results in a very satisfying user experience. With just a few commands, you can search for available packages and install or uninstall them. When doing so, the package manager takes into account the dependencies of the desired application and installs them, as well, if necessary, or cancels the installation if you do not agree to install or update one of the required packages. Although the command syntax is not identical for the different package managers, the logic behind it is very similar, which means that you can quickly find your way around the new package landscape when changing the distribution you use.

A similar user experience that also supports scripting of complex installations has been on many a Windows administrator's wish list for years. Over the years, several developer communities have tackled the task of providing a full-fledged package manager for Windows. Two of these projects deserve special mention:

- Chocolatey [1] has a development history going back more than 10 years and has become the de facto standard for command-line-based package management on Windows.
- Microsoft's open source WinGet [2] has an interesting history. Developer Keivan Beigi originally set out to solve both the technical challenges of package management on Windows and to overcome the organizational hurdles in the package review and release process with his AppGet project [3]. In early 2020, Microsoft expressed its intention to acquire the author's business and product, but they decided against it in the same year. Meanwhile, the ideas underlying AppGet formed the basis for WinGet. After a media outcry in the community, Microsoft admitted [4] to having used AppGet's ideas directly.

The Classic Choice: Chocolatey

Chocolatey was developed in 2011 by Rob Reynolds. Today a whole team works on the product. The original idea was to extend the functionality of NuGet (installing and including components in software projects) to entire application packages, thus addressing the needs of end users and system administrators. According to the company's website, the name "Chocolatey" is a culinary analogy to "NuGet" (nougat), a package manager for reusable code. The `choco` program, which offers all of Chocolatey's functions, is itself delivered as a NuGet package.

Getting started with the Chocolatey Community Edition is simple. You can install the product on all Windows versions from Server 2003 and Windows 7 in 32 and 64 bit, provided .NET Framework 4 is available. The installation website shows some setup instructions and offers a PowerShell one-liner (Listing 1) that downloads and executes a script.

The `install.ps1` script in turn downloads the NuGet package and installs the program components in the designated paths, which, however, end up in `%ProgramData%` instead of `%ProgramFiles%`, as intended by Microsoft. The Chocolatey website also provides playbook definitions for organizations already using Ansible, Chef, or Puppet. However, this assumes that you have already set up an internal NuGet repository to which you've uploaded the Chocolatey package.

Once Chocolatey is installed, you have access to the community repository, which contains several thousand packages for various applications. You can search the repository for (and install) a desired package on your system using:

```
choco search <part of package name>
choco install <package name>
```

If administrative rights are required, the calling account must have appropriate authorizations, and depending on the configuration, you will need to confirm the user account control (UAC) prompt.

You can upgrade to the latest version available in the repository or remove an application installed by Chocolatey with:

```
choco upgrade <package name>
choco uninstall <package name>
```

The Underlying Repository Is Crucial

Repositories make a significant contribution to the success of a package manager because only a comprehensive and up-to-date repository guarantees a good user experience when it comes to searching for and installing packages. A poorly maintained repository is not only unattractive, but it can put its users' systems at risk if the present version of a package is out of date.

Listing 1: PS Chocolatey Setup

```
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::
  SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

The Chocolatey community repository contains several thousand packages, and some Chocolatey users exclusively rely on it to populate their systems. However, the packages in the community repository are not subject to any real quality control, which means that someone could maliciously release a package bearing the name of a popular application that damages the target systems when installed (Figure 1). This fact was also criticized by Beigi in one of his posts [3].

If you want to use only trusted sources for your installations, you need to set up your own repository. The options to do so are described on the Chocolatey website [5] and range from a simple Universal Naming Convention (UNC) share to commercial products such as Artifactory Pro.

If you trust the source, you can add packages from a public repository to your own repository, which is known as internalizing. The instructions are included in the Chocolatey documentation. Depending on the edition, various tools are available for this purpose, but the community edition also supports the manual internalization of packages that originate from external repositories. Likewise, all editions of Chocolatey support addressing multiple repositories simultaneously. You can build a more complex repository landscape, for example, by having individual departments maintain their own repositories and make them available to the entire organization.

Licensed Chocolatey Editions

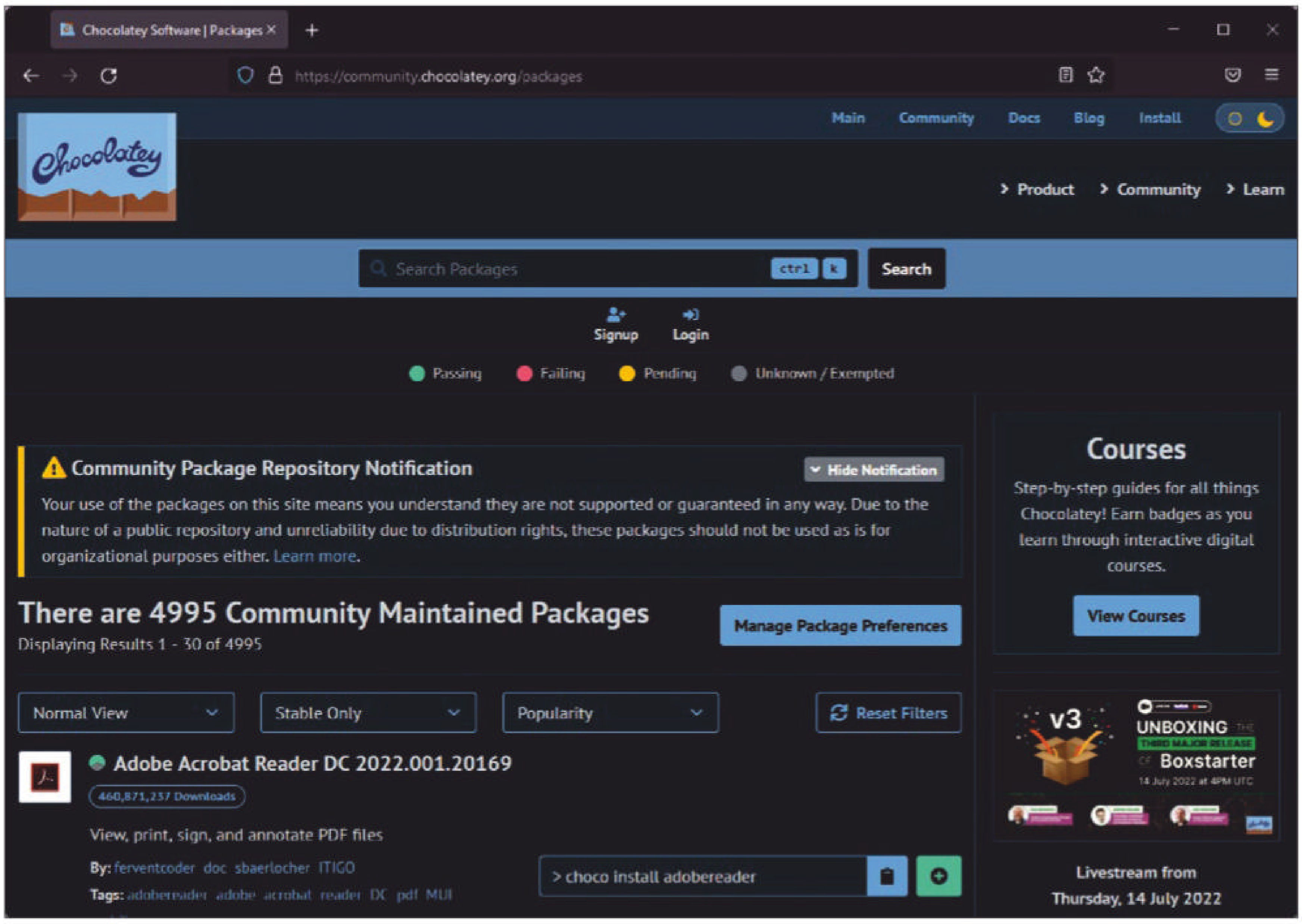
The community edition of Chocolatey (called the Open Source edition on the website) already includes everything you need to install and keep packages up to date on Windows systems; however, Chocolatey can do far more. Additional functions come with the two commercial editions.

The big advantage of the Pro edition is automatic synchronization of packages installed with Chocolatey and by other means. Therefore, systems set up before you rolled out Chocolatey can be included under Chocolatey's management umbrella. The Pro edition also comes with some additional PowerShell features to improve packaging further. Anti-malware integration is another useful feature of the licensed edition. You can use your installed antivirus software, an upload to VirusTotal, or both variants at the same time.

At the top of Chocolatey's package management tree is the Business (C4B) edition. In addition to numerous features of the installation engine, C4B includes a central management server that lets you track, say, which machines successfully installed a package and which did not. You can install the central management infrastructure in your data center or in Azure, where Chocolatey offers it as a pre-built Azure service.

Chocolatey and AppLocker

As mentioned previously, the Chocolatey package installs the executables in



● Passing ● Failing ● Pending ● Unknown / Exempted

Community Package Repository Notification

Hide Notification

Your use of the packages on this site means you understand they are not supported or guaranteed in any way. Due to the nature of a public repository and unreliability due to distribution rights, these packages should not be used as is for organizational purposes either. Learn more.

There are 4995 Community Maintained Packages

Displaying Results 1 - 30 of 4995

Manage Package Preferences

Normal View | Stable Only | Popularity | Reset Filters

Adobe Acrobat Reader DC 2022.001.20169

460,871,237 Downloads

View, print, sign, and annotate PDF files

By: ferventcoder doc sbaerlocher ITIGO

Tags: adobereader adobe acrobat reader DC pdf MUI

```
> choco install adobereader
```

Courses

Step-by-step guides for all things Chocolatey! Earn badges as you learn through interactive digital courses.

View Courses

Livestream from Thursday, 14 July 2022

Figure 1: You need to take the warning on Chocolatey's website quite seriously.

`%ProgramData%`. According to the default AppLocker rules, `choco.exe` would not be executable because the program is located in an unauthorized path, so you need to add an exception or an explicit permission rule for Chocolatey in your AppLocker configuration.

Another peculiarity of Chocolatey in the context of AppLocker is shimming. When you install Chocolatey, its installation path is included in the system's `%PATH%` environment variable. Not every Chocolatey package installer does this. To ensure that each program installed by Chocolatey can be reliably started by calling its name, Chocolatey creates an executable file of the same name in its own installation directory, which directs the input directly to the "real" program, receives the output from it, and returns it to the caller. Of course, these files are not digitally signed and are not known to the system in advance. In other words, the calls will probably be blocked by AppLocker, too, if you have not allowed the entire Chocolatey directory for execution. Incidentally, the shims are not removed when you uninstall packages.

As of Build 1607: WinGet

Since 2021, Microsoft has offered its own package manager for the command line: WinGet. Unlike Chocolatey, the condition for installing this tool is the operating system's ability to install apps from the Microsoft Store, which is why WinGet is only installable on Windows 10 as of build 1607, with Windows Server 2019 and older left out in the cold. Although you can theoretically install WinGet on Server 2022 (e.g., as described by Andreas Nick [6]), WinGet is not officially supported there.

WinGet is already pre-installed on the current builds of Windows 10 and 11, which gives you access to two repositories: *winget* and *msstore*. WinGet can download and install apps from the *msstore* repository in the same way as the graphical Microsoft Store app, whereas the *winget* repository contains various programs that are available in different formats (e.g., MSI installers or MSIX packages, ready-made store apps, or scripts).

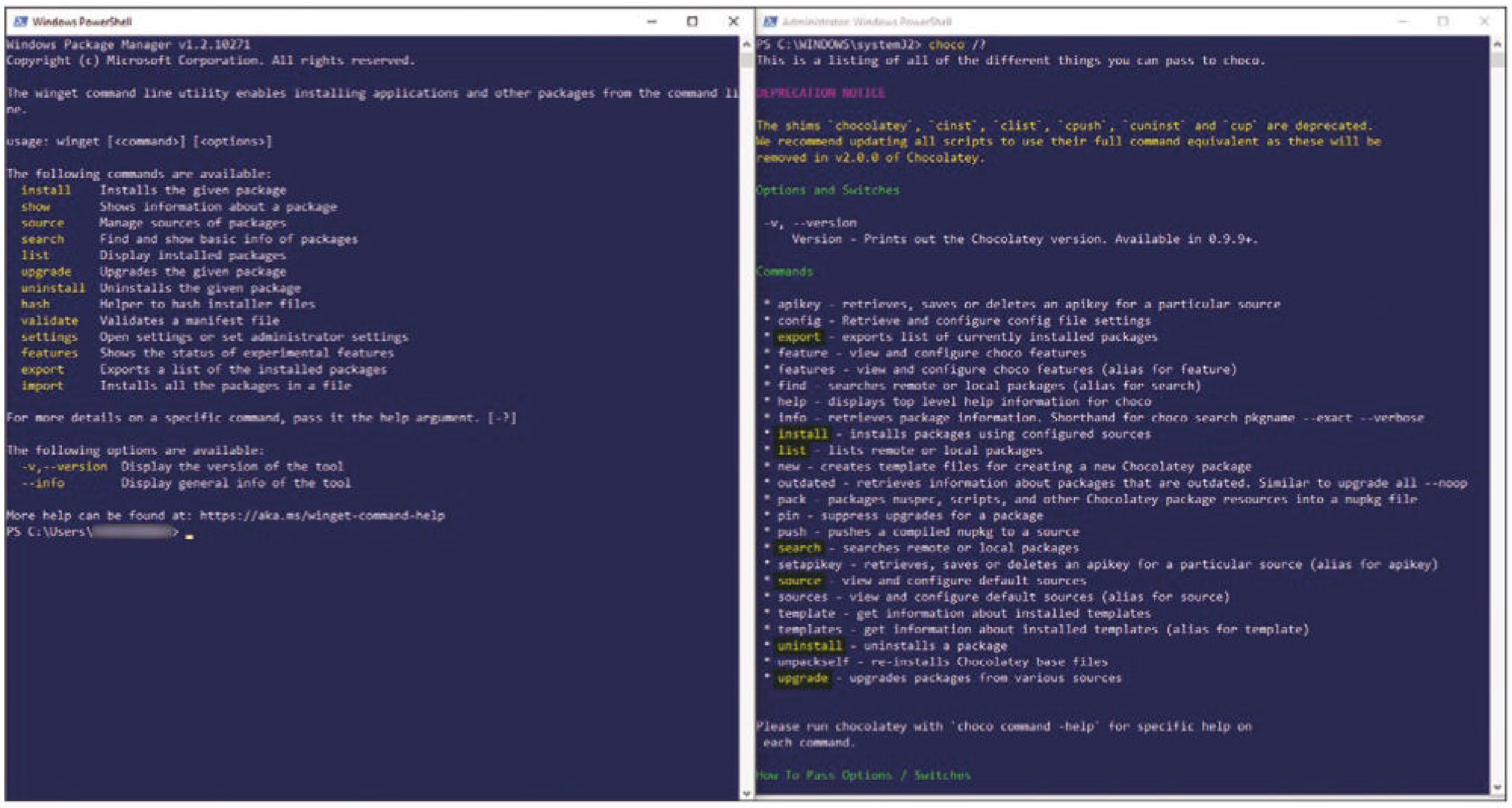
A WinGet package is represented by a YAML manifest that can include one or more files. For example, if a package is offered in multiple languages, its manifest comprises at least four files: version, installer, default locale, and one additional locale file per supported language. You can submit a finished manifest to the community repository *microsoft/winget-pkgs* with a pull request on GitHub. The pull request triggers a validation process, after which your package is released and can be found and installed by the WinGet instances of all Windows users connected to the Internet.

This process is fundamentally different from submitting apps to the Microsoft Store, which requires a dedicated developer contract and imposes far stricter content guidelines.

Initially, WinGet's command structure looks very similar to that of Chocolatey (Figure 2), which is hardly surprising, because both package managers are oriented toward the Linux role models and seek to offer people switching from Linux administration as familiar a user experience as possible. Beyond the basic range of functions, however, WinGet and Chocolatey differ considerably, as you can discern from the command set.

Operating Private Repositories

Like AppGet, WinGet follows a philosophy different from Chocolatey regarding the distribution of binary installation sources. Whereas a



```
Windows Package Manager v1.2.10271
Copyright (c) Microsoft Corporation. All rights reserved.

The winget command line utility enables installing applications and other packages from the command li
ne.

usage: winget [<command>] [<options>]

The following commands are available:
  install    Installs the given package
  show       Shows information about a package
  source     Manage sources of packages
  search     Find and show basic info of packages
  list       Display installed packages
  upgrade    Upgrades the given package
  uninstall  Uninstalls the given package
  hash       Helper to hash installer files
  validate   Validates a manifest file
  settings   Open settings or set administrator settings
  features   Shows the status of experimental features
  export     Exports a list of the installed packages
  import     Installs all the packages in a file

For more details on a specific command, pass it the help argument. [-?]

The following options are available:
  -v,--version  Display the version of the tool
  --info        Display general info of the tool

More help can be found at: https://aka.ms/winget-command-help
PS C:\Users\[redacted]> _
```

```
PS C:\WINDOWS\system32> choco /?
This is a listing of all of the different things you can pass to choco.

DEPRECATION NOTICE

The shims `chocolatey`, `cinst`, `clist`, `cpush`, `cuninst` and `cup` are deprecated.
We recommend updating all scripts to use their full command equivalent as these will be
removed in v2.0.0 of Chocolatey.

Options and Switches

 -v, --version
     Version - Prints out the Chocolatey version. Available in 0.9.9+.

Commands

 * apikey - retrieves, saves or deletes an apikey for a particular source
 * config - Retrieve and configure config file settings
 * export - exports list of currently installed packages
 * feature - view and configure choco features
 * features - view and configure choco features (alias for feature)
 * find - searches remote or local packages (alias for search)
 * help - displays top level help information for choco
 * info - retrieves package information. Shorthand for choco search pkgname --exact --verbose
 * install - installs packages using configured sources
 * list - lists remote or local packages
 * new - creates template files for creating a new Chocolatey package
 * outdated - retrieves information about packages that are outdated. Similar to upgrade all --noop
 * pack - packages nuspec, scripts, and other Chocolatey package resources into a nupkg file
 * pin - suppress upgrades for a package
 * push - pushes a compiled nupkg to a source
 * search - searches remote or local packages
 * setapikey - retrieves, saves or deletes an apikey for a particular source (alias for apikey)
 * source - view and configure default sources
 * sources - view and configure default sources (alias for source)
 * template - get information about installed templates
 * templates - get information about installed templates (alias for template)
 * uninstall - uninstalls a package
 * unpackself - re-installs Chocolatey base files
 * upgrade - upgrades packages from various sources


Please run chocolatey with `choco command -help` for specific help on
 each command.

How To Pass Options / Switches
```

Figure 2: The command structures of WinGet and Chocolatey are similar.

Chocolatey package always contains the installation files, the artifact (a deployable component of an application) in a WinGet repository is typically just a manifest containing the download path for the installation sources and the instructions for installation.

Microsoft provides instructions and sample code [7] that you can use to create a private WinGet repository in the form of an Azure resource. The resulting REST API is documented and can be implemented on an on-premises web server. A few community projects make this possible, but none have made it to production maturity yet. If you want to deploy a private repository for WinGet, Azure is your best bet right now.

PowerShell Support

As nice as the original command line is in Windows, many Windows admins now expect to be able to operate the system of their choice with PowerShell. The success of Foil [8] – a wrapper for the `choco` command-line program created with Microsoft's Crescendo technology – shows how great the interest in this functionality is in the Chocolatey user community. Although this module, created by Ethan Bergstrom, was still classified as version 0.1.0 by the author at the time of writing, it has been downloaded almost five million times since its release in September 2021.

Another PowerShell artifact category that is gaining popularity is Chocolatey-based Desired State Configuration (DSC) resources, which let you keep both your Chocolatey installation and the packages installed with Chocolatey up to date. The official resources from both Chocolatey and community developments have six-figure download numbers.

At its core, Chocolatey is even more intertwined with PowerShell, with PowerShell commands providing the basis for installing Chocolatey packages. You also have access to a great abundance of PowerShell modules and scripts for creating packages.

PowerShell support for managing packages with WinGet is not as advanced, although WinGet is a Microsoft product. The highest download numbers in the PowerShell Gallery so far are for two projects also penned by Bergstrom – the Cobalt Crescendo wrapper and the *WinGet* module on which it is based.

The reason no DSC resources have yet been submitted could be because WinGet is reserved for clients, whereas servers are usually configured and managed by DSC. When it comes to creating WinGet manifests, Microsoft has so far only offered a `winget-create` [9] command-line tool, but not a PowerShell module.

Conclusions

One solution I did not discuss in this article is Scoop [10], which takes the approach of allowing installations in the user context (and therefore also in the user profile) and mutually isolating the packages to the extent possible. This naturally limits the selection of available applications slightly, but at the same time it helps the potential user base grow.

If you want to establish full-fledged package management on Windows, you basically have a choice between Chocolatey and Microsoft's own WinGet. If you want to manage the application zoo on your own servers, you are far better off with Chocolatey, but you need your own repository and package quality assurance procedure. WinGet, on the other hand, gives your clients access to Microsoft-approved repositories and can address a private repository, if needed, but you have to build it yourself, which involves substantial overhead. If you want to track the lifecycle of packages across your entire infrastructure, Chocolatey for Business is the best choice. ■

Info

[1] Chocolatey: [https://chocolatey.org]

[2] WinGet in Docs: [https://learn.microsoft.com/en-US/windows/package-manager/]

[3] AppGet announcement: [https://keivan.io/appget-what-chocolatey-wasnt/]

[4] Microsoft comments on WinGet and AppGet: [https://devblogs.microsoft.com/commandline/winget-install-learning/]

[5] In-house repositories for Chocolatey: [https://docs.chocolatey.org/en-us/features/host-packages]

[6] WinGet on Server 2022: [https://www.andreasnick.com/112-install-winget-and-appinstaller-on-windows-server-2022.html]

[7] Microsoft: Private WinGet resources: [https://github.com/microsoft/winget-cli-restsource]

[8] GitHub Foil repository: [https://github.com/ethanbergstrom/Foil]

[9] winget-create: [https://github.com/microsoft/winget-create]

[10] Scoop: [https://scoop.sh]

Get started with

SysAdmin
JOB HUB

Top jobs for IT professionals
who keep the world's
systems running

SysAdminJobHub.com

Microsoft 365 DSC

Outliers

The declarative PowerShell Desired State Configuration extension supports easy and transparent configuration of systems and applications. We describe the fairly complex initial setup and use of the Microsoft 365 Desired State Configuration. By Nico Thiemer

In the Microsoft 365 environment, with its variety of configurations and multiple tenants to manage, Desired State Configuration (DSC) is an ideal way to track changes and reset a configuration to the desired state. Microsoft provides a great deal of information about the project [1], but in my experience, some of it is outdated. This article refers to the version available March 2022.

Preparations for DSC

Before setting up Microsoft 365 DSC (M365DSC), you need to create an Azure Active Directory (Azure AD) application that you will use later to authenticate the PowerShell script. Alternatively, you could log in with a username and password; I will look at the advantages and disadvantages of these methods in detail.

When authenticating with an Azure AD application, you use either a certificate you create yourself or a client secret. This example uses a certificate because it is the approach currently recommended by Microsoft. In the course of creating the Azure AD application, the certificate is stored in the certificate store of the currently logged in user – authentication only works for this user.

Working with SharePoint Online is essential in Microsoft 365, so most administrators use the PowerShell Office PnP module. Other approaches use the Azure AD web portal. However, I'll take the Office PnP route to create the Azure AD application. If you don't have the module installed, the following commands install it and grant the necessary rights to the PnP management shell in an administrative PowerShell:

```
Install-module PnP.PowerShell
Register-PnPManagementShellAccess
```

Then, create the Azure AD application with:

```
Register-PnPAzureADApp ⮧
  -ApplicationName DSCAuthApp ⮧
  -Tenant <tenant name on>.microsoft.com ⮧
  -OutPath c:\DSC ⮧
  -CertificatePassword (⮧
   ConvertTo-SecureString ⮧
  -String "<SecretPassword>" ⮧
  -AsPlainText -Force) ⮧
  -store CurrentUser ⮧
  -scopes "SPO.Sites.FullControl.All" ⮧
  -DeviceLogin
```

A certificate is automatically created in `C:\DSC`, and the password of the private key matches the `SecretPassword` parameter.

The certificate ends up in the user's certificate store and is uploaded to the Azure AD application. Therefore, the application is given full permissions in SharePoint (`Sites.FullControl.All`, `Group.ReadWrite.All`, and `User.Read.All`).

If multifactor authentication is enabled for the executing user, which should be a matter of course for an administrator in Microsoft 365, you also need to use the `-Device Login` parameter. Now a code is generated in PowerShell, which you then enter in the browser window that opens before logging in with the username and password. For more information, see the PowerShell PnP GitHub page [2]. After this preparation, install DSC with:

```
Install-Module Microsoft365DSC -Force
```

The first command after a successful installation should be:

```
Update-M365DSCDependencies
```

Schedule some time here, because the command can take some time to complete.

Permissions need to be set in Microsoft Graph to match the components you want to access in Microsoft 365. If you want to use DSC for all components in Microsoft 365, the corresponding rights can be queried:

```
Get-M365DSCCompiledPermissionList  ⤶
  -ResourceNameList (⤶
   Get-M365DSCAllResources)
```

To use only individual components or simply check which ones exist, use the command:

```
Get-M365DSCAllResources
```

All permissions for individual components, along with read permission for the `SharedMailbox` and `TeamsUser` components, are set by the cmdlets:

```
Update-M365DSCAllowedGraphScopes
Update-M365DSCAllowedGraphScopes ⤶
  -ResourceNameList @(⤶
   "EXOSharedMailbox", "TeamsUser") ⤶
  -Type Read
```

The list of components is passed as an array in the `ResourceNameList` parameter; however, it is not possible to change (update) the components with the command.

This action requires active consent during execution. The commands

```
Update-M365DSCAllowedGraphScopes ⤶
  -All -Type Read
Update-M365DSCAllowedGraphScopes ⤶
  -All -Type Update
```

set the `Read` and `Modify` permissions for all components, if you so desire.

Login for Tenant Snapshot

The first step exported the configuration of a Microsoft 365 tenant and saved it as a text file (snapshot). Before continuing, the topic of authentication needs to be brought up again, because it's important to understand the risks involved.

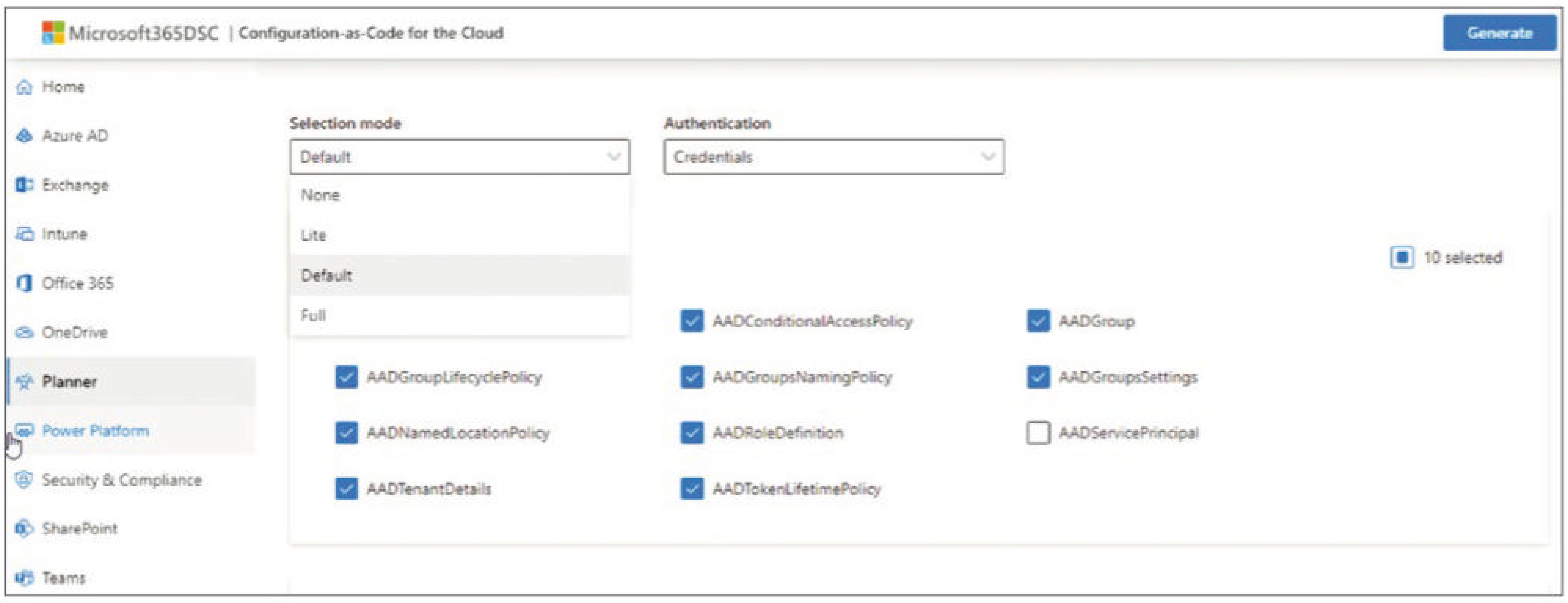
As mentioned earlier, an application, including a PowerShell script, can authenticate to Azure AD, and therefore to Microsoft 365, in multiple ways. The two that are relevant here are logging in as a user account with a username and password or logging in as a service principal with an Azure AD application ID. You also have to distinguish between a login with a client secret and one with a certificate. Logging in with an Azure AD application ID and a certificate is always preferable to the username-password solution. On the one hand, this reduces the attack surface, and the maintenance overhead (disabled accounts, password changes) is lower, as well. Unfortunately, however, M365DSC does not let you log in to all components with an Azure AD application ID.

Although you can log in to Azure AD with an Azure AD application ID and a certificate, if you want to configure, for example, conditional access, you have to use a username and password. To take a snapshot of an entire tenant, your only approach is to log in with a username and password.

You can export the data with

```
Export-M365DSCConfiguration
```

and either put together the command with its options at the command line or launch it with the `-LaunchWebUI` parameter. A redirection in the browser to *https://Export.Microsoft365DSC.com* lets you conveniently compose the export in a GUI, but note that not everything is selected by default (Figure 1). In Microsoft Planner task management software, for example, nothing is exported at all. If you want to include all data, select *Full* in the drop-down



Selection mode: Default

Authentication: Credentials

10 selected

	✓ AADConditionalAccessPolicy	✓ AADGroup
✓ AADGroupLifecyclePolicy	✓ AADGroupsNamingPolicy	✓ AADGroupsSettings
✓ AADNamedLocationPolicy	✓ AADRoleDefinition	☐ AADServicePrincipal
✓ AADTenantDetails	✓ AADTokenLifetimePolicy	

Figure 1: The `-LaunchWebUI` switch for `Export-M365DSCConfiguration` starts exporting a configuration in the GUI.

menu at top left and then press *Generate* at top right.
This action creates a script that is copied and then passed to PowerShell for execution. The process can take several minutes or even hours. You might also need to reauthorize the PnP Management Shell module, depending on the previous configuration and usage.
Once the configuration is complete, you have a snapshot of the tenant at the time of execution. Although that's all well and good, it's not really practical. An unsupervised export that runs, say, once a week would be preferable. This arrangement would require an interactive login, because the copied script always prompts for a username and password.
One alternative is to log in with the Azure AD application. If you do not want to export all components, but only those that are compatible with a login to an Azure AD application, this approach is best. You just need to test something. The script generated on the website requires `ApplicationId`, `CertificateThumbprint`, and `TenantId` to log in. The `ApplicationId` and `TenantId` identifiers can be found in the Azure AD application created previously under *App Registration*. If you go there and click on the app name, the header will give you the data.
You will find *CertificateThumbprint* in *Certificates & secrets* in the app. I imported the private key into the certificate store earlier.
Because not everything works with an Azure AD application, you have no alternative to using a personal login. The challenge here is to encrypt the user password in the script, because the password can be changed to plain text in the script:

```
$password = "<Test1234>" | ↩
  ConvertTo-SecureString ↩
  -AsPlainText -Force
```

However, this solution is not ideal. It would be better to encrypt the password with an AES key. To do so, you need to store the password and the AES key in separate files and reference them in the script:

```
$AESKey = New-Object Byte[] 32
[Security.Cryptography.↩
 RNGCryptoServiceProvider]::↩
  Create().GetBytes($AESKey)
$AESKey | Out-File C:\DSC\<aes.key>
$password = Read-Host -prompt ↩
  "<Please enter password>" -AsSecureString
$password | ConvertFrom-SecureString ↩
  -Key $AESKey | ↩
  Out-File C:\DSC\<password.txt>
```

The code

```
$username = "<Microsoft-365-Login>"
$AESKey = Get-Content C:\DSC\<aes.key>
$password = ↩
  Get-Content C:\DSC\<password.txt> | ↩
  ConvertTo-SecureString -Key $AESKey
$credentials = New-Object System.↩
  Management.Automation.↩
  PSCredential ("$username", $password)
```

handles the login in the script.

Exporting Configurations

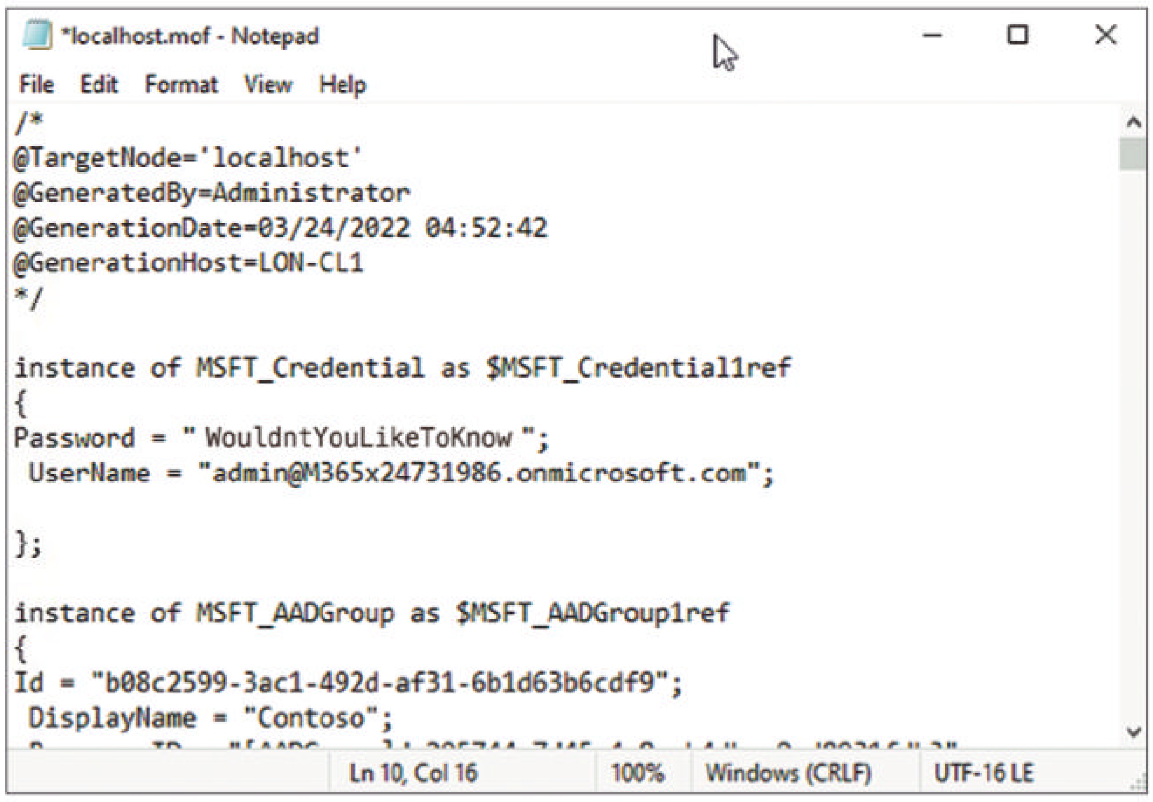
The previously mentioned script from the website is exported and executed in PowerShell. Once it has completed, it prompts you for a storage location for the configuration. If you do not set anything here, the export file will always use the same name. To avoid this – or if you want to run the script periodically – add the command `Export-M365DSCConfiguration` with the `-Path` and `-FileName` parameters.
Alternatively, you could leave out the component list and use the `-Mode` parameter with the `Lite|Default|Full` arguments instead. A call that exports the entire tenant configuration and stores it in the `C:\DSC` path would be:

```
Export-M365DSCConfiguration ↩
  -Mode 'Full' -Credential $Credentials ↩
  -Path 'C:\DSC\'
```

Besides exporting, it is also possible and important to be able to import existing configurations again. To do this, the generated PS1 file must first be compiled into what is known as a managed object format (MOF) file by executing the `M365TenantConfig.ps1` file in PowerShell.
However, the MOF file contains the password in plain text (Figure 2).
Even if you used encryption or an encrypted user object previously, at this point it is always in plain text.
Fortunately, you can use an on-board DSC tool to solve the problem:

```
Set-M365DSCAgentCertificateConfiguration
```

The command generates a certificate that DSC uses to encrypt the credentials in the MOF file. Be careful: The operation will fail if you have not configured Windows remote management (WinRM) with



```
/*
@TargetNode='localhost'
@GeneratedBy=Administrator
@GenerationDate=03/24/2022 04:52:42
@GenerationHost=LON-CL1
*/

instance of MSFT_Credential as $MSFT_Credential1ref
{
Password = " WouldntYouLikeToKnow ";
 UserName = "admin@M365x24731986.onmicrosoft.com";

};

instance of MSFT_AADGroup as $MSFT_AADGroup1ref
{
Id = "b08c2599-3ac1-492d-af31-6b1d63b6cdf9";
 DisplayName = "Contoso";
```

Figure 2: **The MOF file contains the password as plain text by default.**

the `winrm quickconfig` command. If the configuration is good, the encrypted password is displayed in the file.

Also important to note is that you must export a configuration after the certificate has been created. It makes most sense to run the command before the first export. When exporting, the corresponding certificate is then also stored in the folder and referenced accordingly in the `ConfigurationData.psd1` file.

Working with Configurations

In contrast to exporting, importing a configuration is comparatively simple. After all, everything is already configured. The command is `Start-DSCConfiguration` and as an argument, you will pass in the path where the MOF file is stored. Use the `Verbose` parameter here to have more options in case of errors. Again, it can take hours to complete the process:

```
Start-DSCConfiguration ➘
  C:\DemoM365DSC\M365TenantConfig ➘
  -Wait -Verbose -Force
```

You might see an error message stating that the request size was exceeded in WinRM. In this case, increase the size with the command:

```
winrm set winrm/config ➘
  @{MaxEnvelopeSizekb="8192"}
```

DSC also lets you check a given configuration against a default every 15 minutes. If the default is not met, DSC can restore the desired state from the MOF file. This process is adjustable, and instructions can be found online [3].

An option to clone configurations is very useful, especially if you manage several tenants that have the same configurations. You can create all the settings on a master tenant and then export the configuration to other tenants.

Generating Evaluations

The ability to compare different configurations with each other is also interesting and would encompass exports of the same tenants at different times, as well as exports from different tenants. M365DSC lets you generate evaluations in Excel and HTML formats. An Excel file only documents one tenant, whereas HTML reports let you make comparisons. To generate a simple report, use:

```
New-M365DSCReportFromConfiguration
```

You can also pass in the `-Type` parameter as `Excel` or `HTML` and use `ConfigurationPath` to define the location of the configuration file – not the MOF file, but the `M365TenantConfig.ps1` file (if you keep the default name).

The `-OutputPath` parameter specifies the location of the report. You must specify the location and file name. Without a file name, if you choose the `HTML` option, you will see the output in PowerShell, but the file will not be created on the filesystem. The `Excel` option opens the file in Excel, but `-OutputPath` has no meaning, because a file can only be saved in Excel. For a comparison use the command

```
New-M365DSCDeltaReport
```

with `-Source` and `-Destination` parameters, which also point to the respective configuration files. In this way you create an HTML file; again, you need to specify the location with `-OutputPath`. If you omit the parameter, the HTML output is displayed on the PowerShell console.

Conclusions

DSC for Microsoft 365 can be used to archive configurations and restore them when needed. You can also compare different tenants – and don't forget the possibility to keep the settings of multiple tenants in sync easily and reliably. The administrative gains more than make up for the comparatively costly setup overhead. ■

Info

[1] Microsoft 365 Desired State Configuration: [https://microsoft365dsc.com]

[2] Register-PnPAzureADApp cmdlet: [https://pnp.github.io/powershell/cmdlets/Register-PnPAzureADApp.html?q=Register-PnPAzureADApp]

[3] Setting up the Local Configuration Manager: [https://learn.microsoft.com/en-us/powershell/dsc/managing-nodes/metaconfig?view=dsc-1.1&viewFallbackFrom=dsc-1.1.]

Delegate and restrict authorizations in Azure AD

Temporary Admin

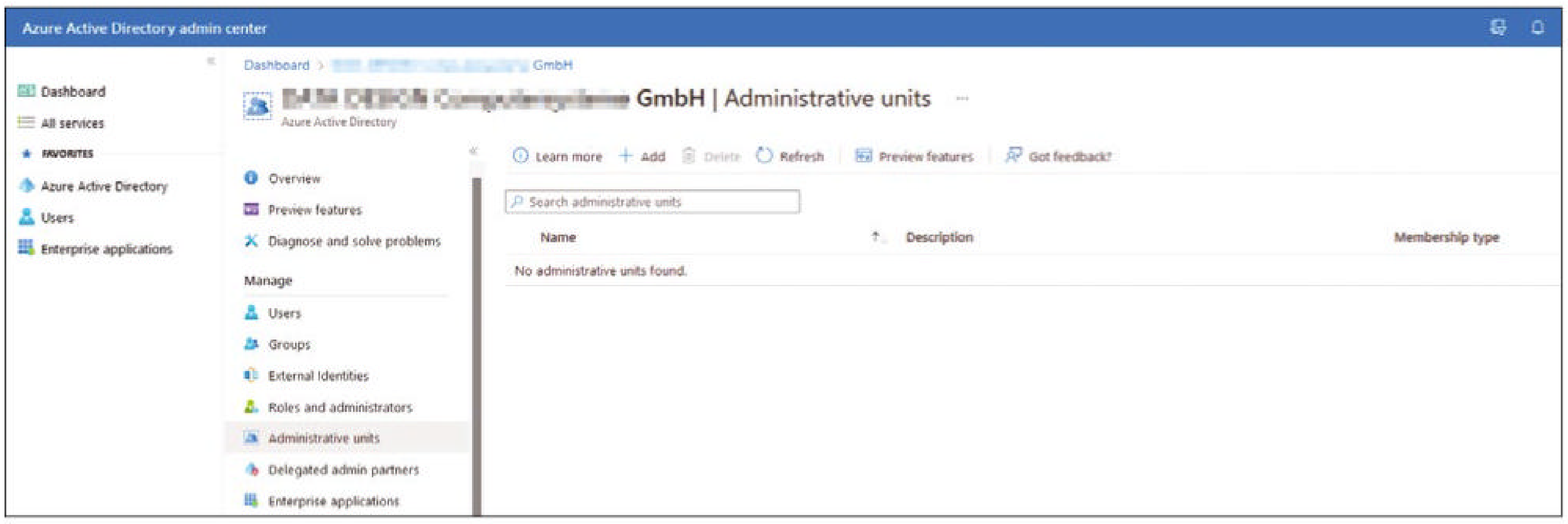
Azure AD is one of the most important authentication services for cloud environments. We show you how to delegate authorizations in Azure AD to ensure better security. By Thomas Joos

In the Microsoft world of Azure and Microsoft 365, especially, Azure Active Directory (AD) is an important component for authenticating users. By synchronizing with Active Directory, organizations can also synchronize on-premises credentials to the cloud, enabling single sign-on (SSO) scenarios. As with Active Directory, you need to keep accounts in Azure AD organized and delegate the management of various tasks. Organizational units (OUs) are used for this purpose in Active Directory; Azure AD has something similar to OUs called administrative units (AUs). In this article I'll show you how to work with AUs for a better way to delegate cloud directory authorizations. Although in general the AUs in Azure AD correspond to the OUs in Active Directory, the two differ significantly. In contrast to AD, the authorization structures in Azure AD are very flat, and restricting them is a complex process. Administrative units and role-based authorizations can be the solution.

Security with Roles in Azure AD

Administrative units are intended to help improve the structure in Azure AD in a similar way that OUs do in Active Directory. Administrative units are available on the Azure portal under *Azure Active Directory*. They can also be configured in the Azure Active Directory admin center by selecting *Azure Active Directory* | *Administrative units* (Figure 1).

In Azure, authorizations for all resources can be mapped with a role-based authorization structure. You need to restrict the authorizations for administrators so that only those who are genuinely necessary are allowed, which complicates the configuration to some extent but significantly improves security. Administrative units work in combination with role-based access control (RBAC), which means you can assign roles to the AUs and then map them to users, groups, and devices.



No administrative units found.

Figure 1: Administrative units in Azure AD correspond to the organizational units of Active Directory. Management is through the Azure portal.

Table 1: Microsoft Portals

Portal	URL
Management Portal	
Microsoft Azure	[https://portal.azure.com]
Azure AD admin center	[https://aad.portal.azure.com]
Microsoft 365 admin center	[https://admin.microsoft.com]
Microsoft Teams admin center	[https://admin.teams.microsoft.com]
Microsoft Exchange admin center	[https://admin.exchange.microsoft.com]
SharePoint admin center	[https://admin.microsoft.com/sharepoint]
Microsoft Endpoint Manager admin console	[https://endpoint.microsoft.com]
Azure Cloud Shell	[https://shell.azure.com]
Azure subscriptions	[https://portal.azure.com/#view/Microsoft_Azure_Billing/SubscriptionsBlade]
User Portal	
Azure AD user self-service	[https://myaccount.microsoft.com]
Microsoft user account	[https://account.microsoft.com]

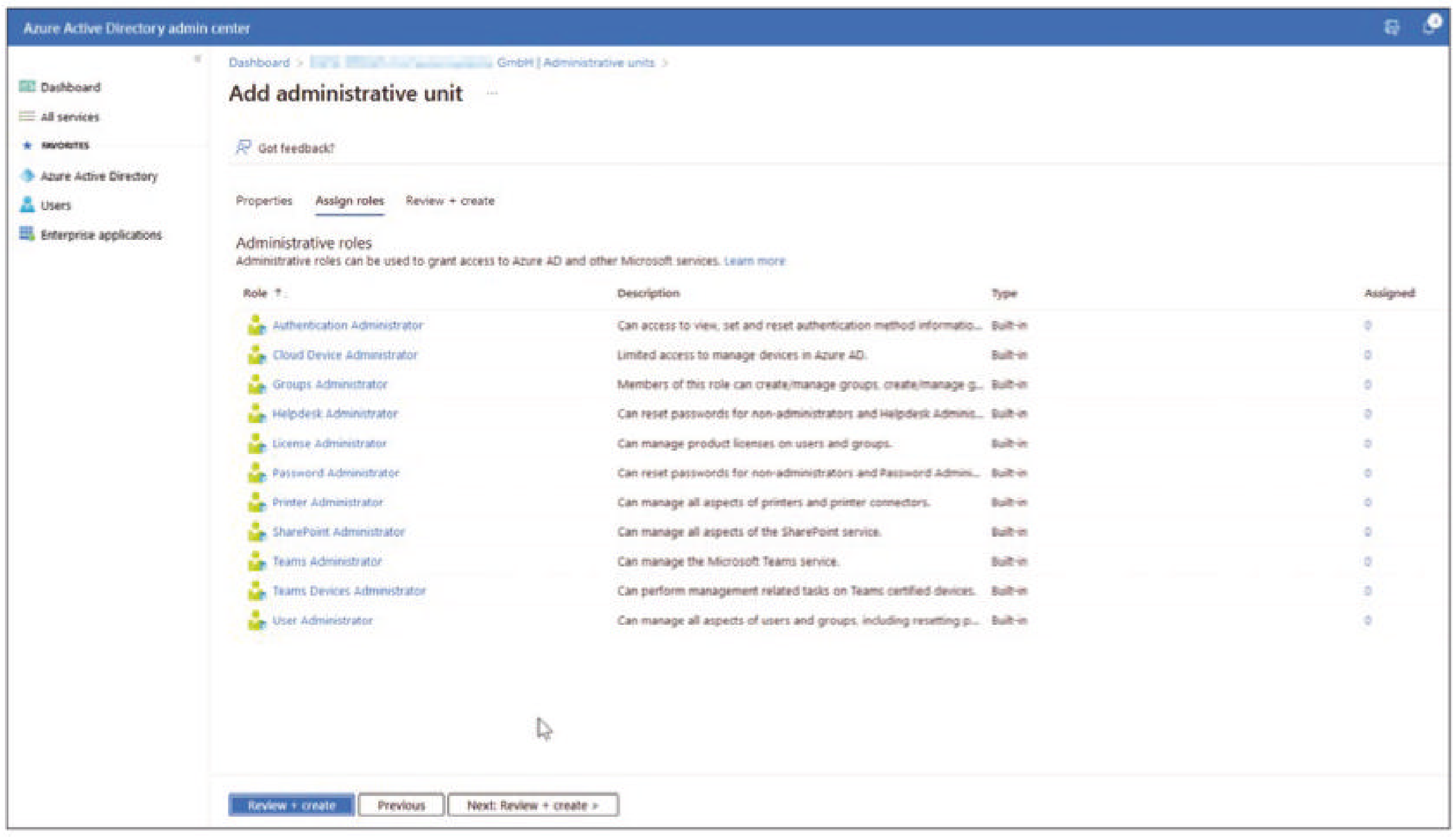
The Azure AD objects to which the AU is linked can be managed by the users who are members of the roles, which in turn are linked to the AU.
After clicking on a user account, you can go to the Users section and click *Assigned Roles* to control which authorizations belong to the subscription. If you click on a role when managing the assigned roles, you then see all the user accounts assigned to this role. The possibility of working with Privileged Identity Management in Azure AD is also interesting. Doing so lets you designate users who are authorized to perform administrative tasks only for a certain period of time. You can then assign these roles to administrative units.

Isolating Users

Administrative units basically help you control and restrict the type of administrative access for admins. The purpose is to isolate specific users and groups and their devices from the admin groups. Administrative units let you create administration containers and a logical structure of authorizations in Azure AD. The scope of the admins' permissions can be flexibly controlled with AUs.
Administrative units let you control authorizations for users and groups, and you can even configure access to devices in the preview. In the Azure AD admin center, you can use the *Devices* menu item to check which devices are currently logged in to Azure AD, as well as devices that are connected to, but not managed by or compliant with, the stored policies. *All devices* lets you see whether all the devices are still required at any time.
For security reasons, it may make sense to remove devices that are no longer needed. At this point, you can also adjust settings of the devices and define who is allowed to connect how many devices to your Azure AD. To do so, call up the *Device settings* menu item where you can link the devices found there to create new AUs and delegate the management tasks after doing so. It is also possible to connect computers dynamically (i.e., on the basis of their attributes).

Creating, Customizing, and Managing AUs

To create and control management entities and their associated objects, you need to familiarize yourself with the various management tools in Azure. They also play an important role for Microsoft 365. Web portals are used to control most options for managing Azure, Azure AD, and therefore the management entities.
To manage Azure AD, Microsoft 365, and Azure, you need to know the various URLs, and maybe even save them as favorites, to access the

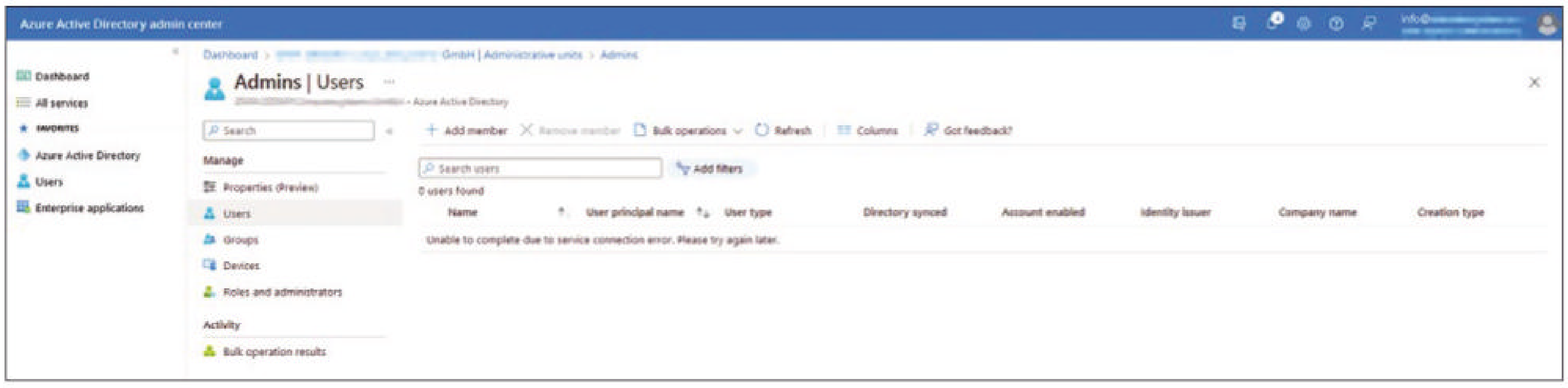


Add administrative unit

Administrative roles
Administrative roles can be used to grant access to Azure AD and other Microsoft services. Learn more

Role ↑	Description	Type	Assigned
Authentication Administrator	Can access to view, set and reset authentication method informatio...	Built-in	0
Cloud Device Administrator	Limited access to manage devices in Azure AD.	Built-in	0
Groups Administrator	Members of this role can create/manage groups, create/manage g...	Built-in	0
Helpdesk Administrator	Can reset passwords for non-administrators and Helpdesk Adminis...	Built-in	0
License Administrator	Can manage product licenses on users and groups.	Built-in	0
Password Administrator	Can reset passwords for non-administrators and Password Admini...	Built-in	0
Printer Administrator	Can manage all aspects of printers and printer connectors.	Built-in	0
SharePoint Administrator	Can manage all aspects of the SharePoint service.	Built-in	0
Teams Administrator	Can manage the Microsoft Teams service.	Built-in	0
Teams Devices Administrator	Can perform management related tasks on Teams certified devices.	Built-in	0
User Administrator	Can manage all aspects of users and groups, including resetting p...	Built-in	0

Figure 2: **Assigning the administrator roles for a new administrative unit.**



0 users found

Unable to complete due to service connection error. Please try again later.

Figure 3: Customizing the management entities in Azure AD.

various management portals directly. The most important portals are shown in Table 1.

For AUs, you would either use the Azure portal and call up *Azure Active Directory* there or use the Azure AD admin center from the outset, where *Add* gives you a quick and easy approach to creating new management entities. The first step is to define the name of the administrative unit. You can then specify which administrator roles you want to assign to the AU under *Assign roles* (Figure 2). The process is not complicated and you can always customize the roles assigned to an AU.

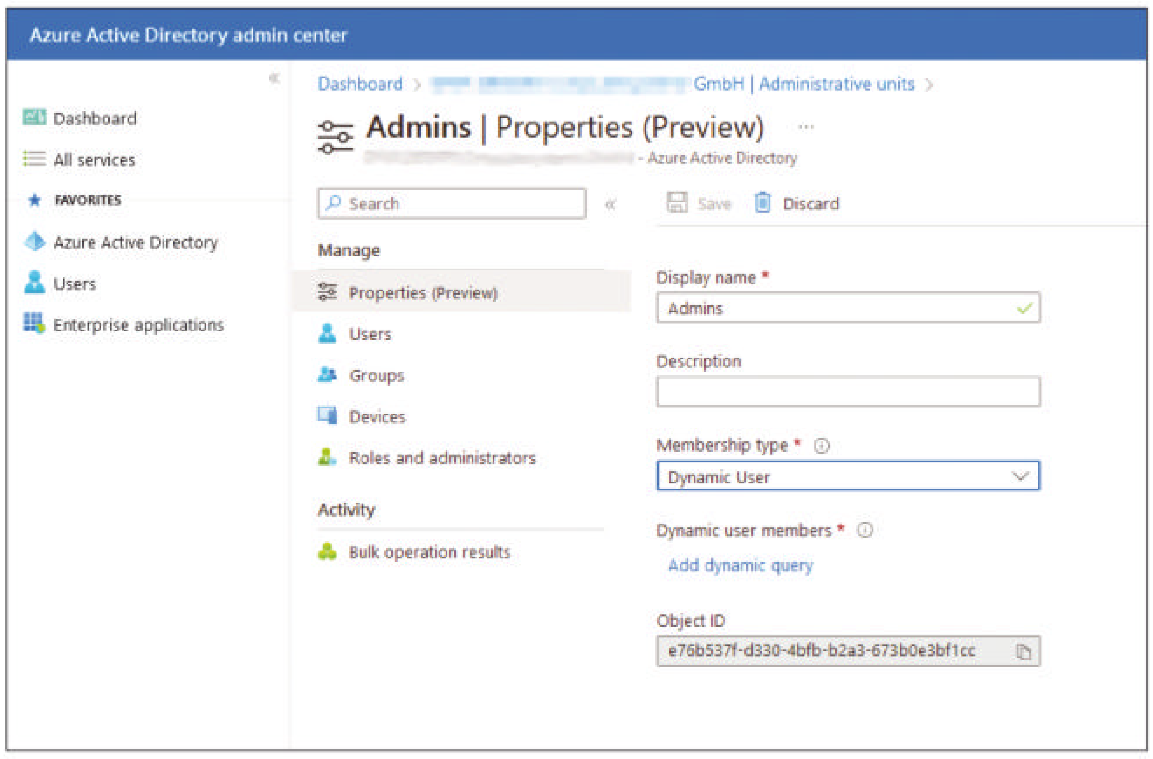
For each role, the *Description* column shows which authorizations it has and the tasks for which it can be used. You can then assign user accounts to the individual roles, which are created in the respective client. After assigning roles to the new administrative unit, you can then create them.

The AU can then be viewed from the *Administrative units* menu item, where you can adjust the settings at any time and assign users, groups, devices, and roles (Figure 3). These objects can then be managed by the users who are members of the administrative roles; in turn, these roles are part of the management entity. Static assignments are possible at this point, but you can also manage AUs dynamically.

Dynamic AUs

Dynamic AUs support queries for membership in an administrative unit on the basis of user and device attributes. Therefore, membership can be automated to a large extent. When you click on a management entity in the Azure portal, you can customize its settings and assignments (Figure 4). Besides an *Assigned* membership type, you can also choose between *Dynamic User* and *Dynamic Device* in the Properties pane.

After setting up an administrative unit with dynamic membership, the *Add dynamic query* link lets you define your own rules for membership. You can also access the attributes of the user accounts or devices. Once you have defined and saved the rule, you can view the members on the main page of the AU and access the objects. You can also configure a membership rule at this point.



Display name * Admins

Description

Membership type * Dynamic User

Dynamic user members *

Object ID e76b537f-d330-4bfb-b2a3-673b0e3bf1cc

Figure 4: Configuring administrative units dynamically.

Configuration with PowerShell

Administrative units are not a complex product. So that they can contribute to improved security in Azure AD, you need to configure the rest of the environment to allow the management entities to achieve their full potential, which includes managing Azure and the AUs in PowerShell. To manage administrative units in PowerShell, use the cmdlet

```
New-AzureADMSAdministrativeUnit ⮯
  -Description "West Coast region" ⮯
  -DisplayName "West Coast"
```

Before you can work with PowerShell in Azure, you first need to do some preparatory work. Almost all areas in Azure AD can be managed with PowerShell, too, not just the administrative units. Besides managing Azure with the Azure Cloud Shell, you can install the module for managing Azure AD in PowerShell, which lets you administer sections of Azure AD locally and, incidentally, parallel to managing Active Directory. To begin, install the Azure AD PowerShell module on your computer and open a connection to Azure AD with

```
install-module azuread
Connect-AzureAD
```

To view all of the cmdlets for Azure AD management, use

```
get-command -module azuread
```

For example, you can display the commands for managing administrative units by running the cmdlet

```
Get-Command *AzureAD*AdministrativeUnit*
```

When assigning administrative units to specific users or groups, user logons should be monitored. This also applies to the admin accounts that are members of the administrative units. Changing memberships often requires a new registration at the AAD. Whether this login was successful can be monitored in the Azure portal.

When assigning administrative units to specific users or groups, as well as to admin accounts that are members of the administrative units, you need to monitor user logins. When changing memberships, it is often necessary to re-register with Azure AD. You can use the Azure portal to discover whether the login worked. In the Azure AD admin center, click *Users* | *< user >* | *Sign-in logs* to check when users have logged in and where. The portal also displays the IP address from which the login took place and the use of multifactor authentication. Multiple unsuccessful logins indicate an attack on a user account.

If you type `Get-AzureADUser` in the Azure Cloud Shell or Azure PowerShell, you can view information about the user accounts in an Azure subscription. If you want to see more detailed information about a user or group in Azure AD, use the cmdlet with the `ObjectID` parameter

```
Get-AzureADUser -ObjectID ➘
  3b47a729-16c1-4be1-9ee2-7f3bd00a346b
```

You can retrieve the `ObjectID` for each user after entering `Get-AzureADUser`, and you can use `UserPrincipalName`, typically the email address the user uses to log in, as the `ObjectID`. To view all the data for a user account, pass the results from `Get-AzureADUser` to the select cmdlet and use the `*` wildcard to output all the data:

```
Get-AzureADUser -ObjectID 3b47a729-16c1-➘
  4be1-9ee2-7f3bd00a346b |select *
```

Administrative units can also be mapped to groups that you manage in PowerShell, as well (e.g., to supervise administrative tasks). The individual groups in Azure AD can be displayed with the `Get-AzureADGroup` cmdlet. Additionally, you can create new groups, control group memberships, delete groups, and much more in PowerShell. To retrieve all the cmdlets, use

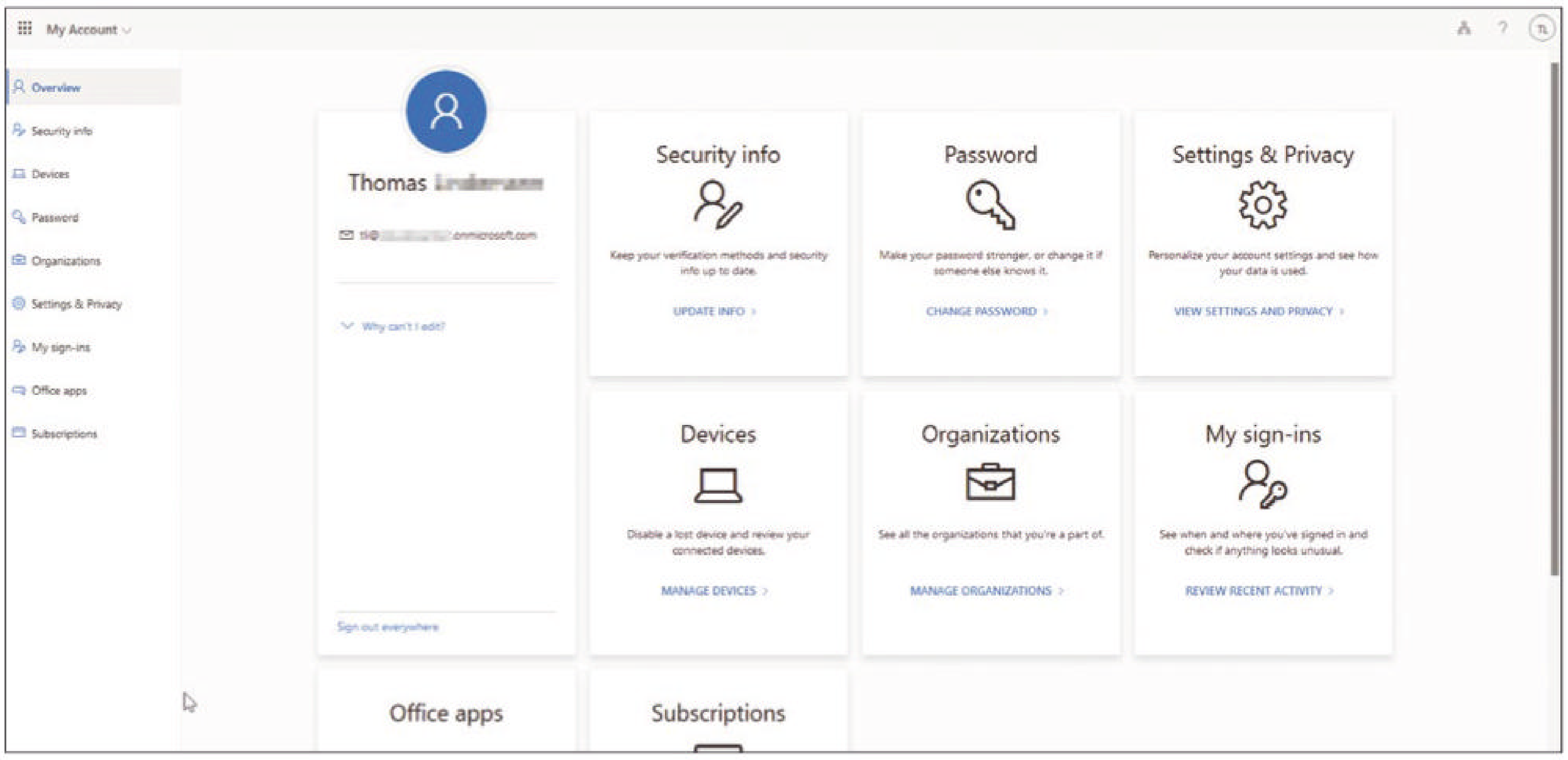
```
Get-Command *ADGroup*
```

User accounts that are members of an administrative role can manage a group. In turn, the role can be associated with an administrative unit to which the group has been assigned.

DIY Administrative Tasks

After assigning admins to administrative units and associating these AUs with users, groups, and devices in Azure AD, admins in the AU can also perform various tasks to manage the objects. However, users can also perform many tasks themselves given sufficient authorization. The focus of the AUs is on delegating tasks for managing the linked objects.

Azure AD provides a self-service portal where all users can log in to manage their own user accounts (Figure 5). If users also have a Microsoft account, it can be found on a separate portal. Both accounts can be accessed as shown in Table 1 under User



Keep your verification methods and security info up to date.

Make your password stronger, or change it if someone else knows it.

Personalize your account settings and see how your data is used.

Disable a lost device and review your connected devices.

See all the organizations that you're a part of.

See when and where you've signed in and check if anything looks unusual.

Figure 5: Users can handle many administrative tasks themselves.

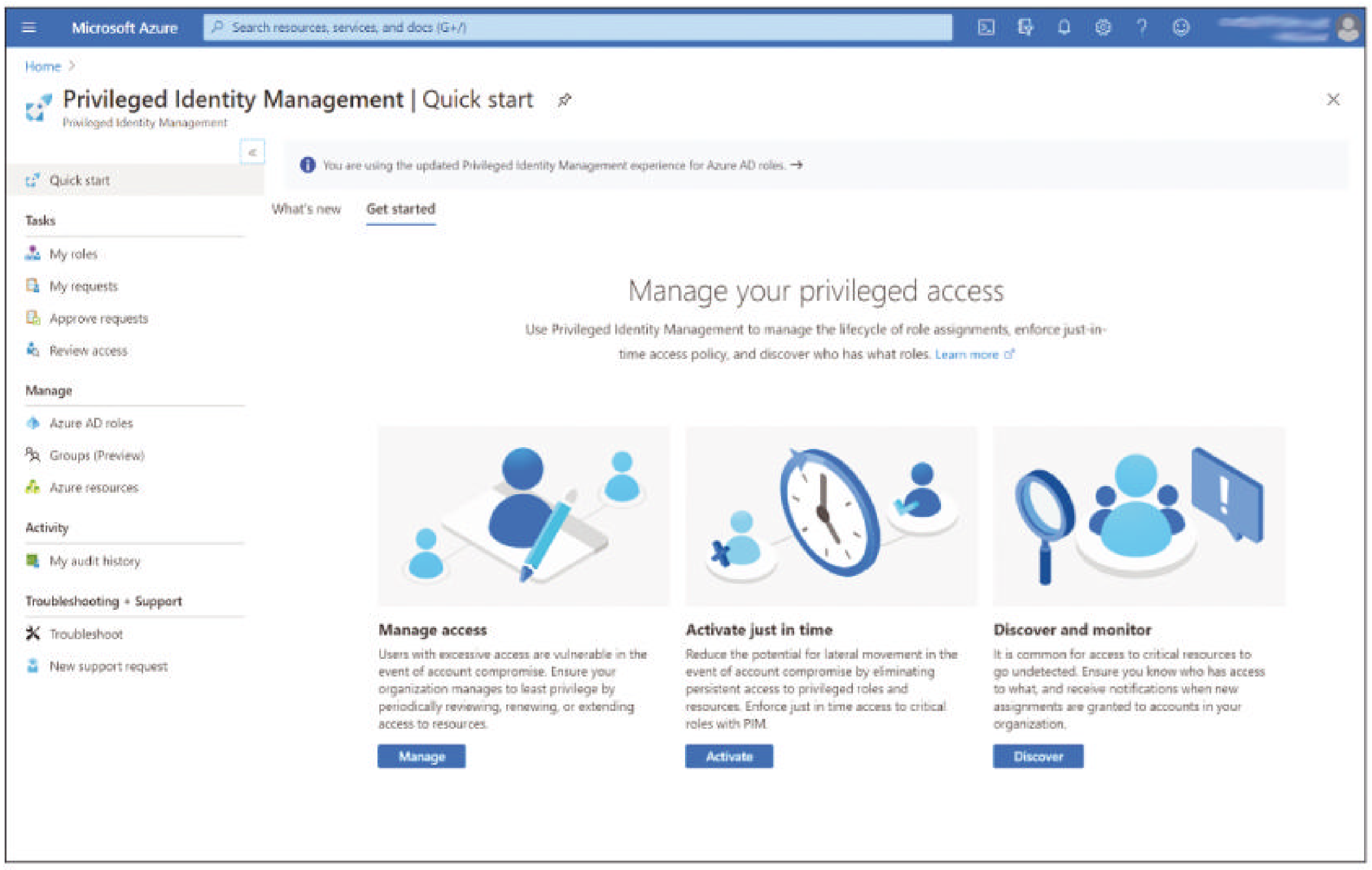
Portal. Users can change or reset their passwords, for example, and also adjust settings and optimize data protection. You can also review your own logins here (e.g., to discover whether an account has been hacked). If Microsoft Office licenses are on file for an account, users can also download the installation files through the portal. Administrative units are not mandatory for this action.

Privileged Identity Management

In the context of administrative entities, Privileged Identity Management (PIM) is an essential component for delegating user authorizations. PIM protects resources and user accounts in Azure AD – and by extension in Microsoft 365 and Endpoint Manager. The technology can help prevent attacks on privileged user accounts.
Managing user accounts in Azure AD is important, of course, especially in the context of protecting privileged accounts. In particular, these are the accounts that have more rights in the environment than normal user accounts and that are members of an administrative unit through management roles: admin accounts or accounts belonging to support staff and users with far-reaching permissions.
Protecting user accounts with elevated authorizations is an important factor for higher security in the cloud, and it boosts the security of the various resources in Microsoft 365 and other cloud services. Every environment has users who have access to a particularly large number of resources or complete applications. The question is whether the respective user accounts need these rights permanently or only for certain tasks and at specific times. In most cases, comprehensive authorizations are rarely necessary. It makes sense to restrict these authorizations for user accounts at times when they are not needed so that even if an admin account is hijacked successfully by an attacker, the resources are protected because the account cannot exercise its privileges.
Protecting admin accounts is the job of Azure AD Privileged Identity Management (Figure 6). This service protects accounts in Azure, Microsoft 365, and environments such as Microsoft Endpoint Manager. If attackers hijack accounts in infrastructures with these services, they can usually carry out far-reaching, damaging actions – even as far as penetrating into the local data center. To use Azure AD PIM, organizations need a license for Azure AD Premium P2. Microsoft spells out the license requirements exactly online [1].

Managing Azure AD PIM

Azure AD PIM is managed on the Azure portal. Searching for *Azure AD Privileged Identity Management* in the portal opens the management interface for the service, which can be used to control access, in parallel with administrative units, in the various environments already mentioned.
On the service management page, *Azure AD roles* shows the different roles and the options for protecting these roles. Clicking on *Roles* takes you to the individual roles that are allowed to perform administrative tasks in the environment. These are also the roles you assign in the administrative unit. PIM in turn protects user memberships of these roles and therefore user authorizations. By way of an example, the *Exchange Administrator* role can manage almost all settings in Exchange Online.
After clicking on the role, the Azure portal displays the various management options. To add user accounts to a role in Azure AD and therefore also in Microsoft 365 and Endpoint Manager, all you need to do is click *Roles | Add assignments*. You can then choose the desired role under *Select role* and then select a user account to which you want to assign the role; the users will be able to manage Exchange Online or other resources in the future but are protected by PIM.



You are using the updated Privileged Identity Management experience for Azure AD roles. →

Manage your privileged access

Use Privileged Identity Management to manage the lifecycle of role assignments, enforce just-in-time access policy, and discover who has what roles. Learn more

Manage access
Users with excessive access are vulnerable in the event of account compromise. Ensure your organization manages to least privilege by periodically reviewing, renewing, or extending access to resources.

Activate just in time
Reduce the potential for lateral movement in the event of account compromise by eliminating persistent access to privileged roles and resources. Enforce just in time access to critical roles with PIM.

Discover and monitor
It is common for access to critical resources to go undetected. Ensure you know who has access to what, and receive notifications when new assignments are granted to accounts in your organization.

Figure 6: Azure AD PIM works with the management entities for assigning user rights.

Controlling Access Times and Scenarios

Once you have selected the role and added the members, the next step is to set the access type for the role. In addition to permission for administrative access, at this point it is also possible to control the times at which user accounts will be granted administrative authorizations. However, you first need to establish the membership by clicking *Assign*. New members can be added and removed at any time in the role settings. When added, members are notified automatically by email of the details of the role.

When you call up the role, you can see its members and configure the details in *Settings*. In the upper area of the details, press *Edit* to start making changes. At this point you can, for example, define the user account that will authorize administrative access for the users of the role. In the role management settings, you can configure how long access will be allowed once the approver has allowed access (Figure 7). You can also stipulate multifactor authentication for accessing the role's authorizations. However, this is something you would generally want to do for all user accounts in Azure AD and Microsoft 365 anyway.

The buttons in the lower area can then be used to configure further settings, such as whether a user is given permanent authorization or when authorizations expire. You can also choose to have email sent when a user requests and receives a role.

Role Requests by Admins

Once the customizations are in place, you can request access to the role by clicking *Activate* in the Azure AD portal when managing PIM. You can also see the roles for which you are authorized and enable access. When doing so, admins must enter a reason. Once access is requested, the user will receive an email to approve access. However, you can also unlock access directly on the Azure portal when managing Azure AD roles.

Once access is approved, the admin will receive an email message that their request has been approved. (All operations are traceable in Azure and the monitoring system.) After that the admin is a member of that role, which in turn can be a member of an administrative unit.

Conclusions

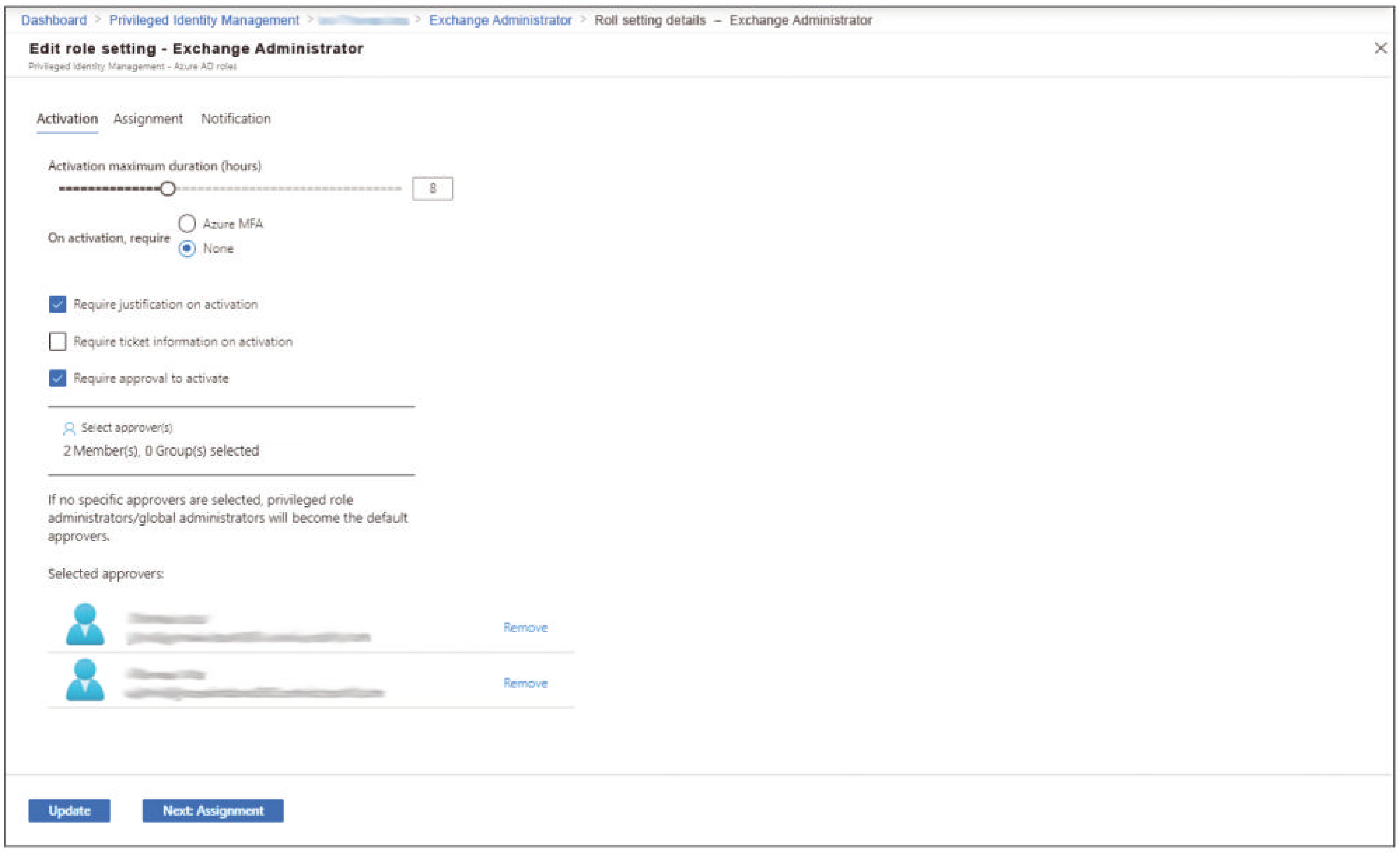
Administrative units enable structured delegation of authorizations in Azure AD. In the current version, it is also possible to manage devices or make memberships dynamic. Together with role-based entitlement management and privileged identity management, the management entities can be used to create effective structures for managing authorizations in Azure AD. ■

Info

[1] Licensing requirements to use PIM: [https://learn.microsoft.com/en-us/azure/active-directory/privileged-identity-management/subscription-requirements]

The Author

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


Dashboard > Privileged Identity Management > [redacted] > Exchange Administrator > Roll setting details – Exchange Administrator

Edit role setting - Exchange Administrator
Privileged Identity Management - Azure AD roles

Activation | Assignment | Notification

Activation maximum duration (hours): 8

On activation, require: ( ) Azure MFA [✓ None]

- [x] Require justification on activation
- [ ] Require ticket information on activation
- [x] Require approval to activate

Select approver(s)
2 Member(s), 0 Group(s) selected

If no specific approvers are selected, privileged role administrators/global administrators will become the default approvers.

Selected approvers:

[redacted] Remove
[redacted] Remove

Figure 7: PIM lets you control membership in Azure AD management roles.

Data security and data governance

New Gold

Protecting data becomes increasingly important as the quantity and value of information grows. We describe the basics of data security and governance and how they intertwine. By Martin Kuppinger

Data has a central function in the digital economy. It is collected in gigantic quantities so that automated or autonomous decisions can be made on the basis of data analysis. In this context, data security is achieved by technologies such as encryption, access protection, and tools that prevent attacks (e.g., SQL injection) on information. In contrast, data governance involves tools that encourage correct handling of data (e.g., managing personal and other sensitive information) and that implement the associated risk management with the use of, for example, metrics and control instruments.

The Basis: Data Overview

To implement data security, not just as a local solution for certain databases but to secure all or at least a large amount of relevant information, you first need to establish the basis. Data management informs you of the information you have, which is important because you can't use or protect what you don't know about.
And a positive side effect is that this knowledge is needed not only for data security, but also for the efficient use of data and for data governance. More specifically, it's about metadata management and data catalogs. Metadata management is the functional approach in data catalogs that acts as a repository of information across the various data storage systems. This helps you identify data worthy of protection, as well as the best data sources for the efficient and targeted use of information.
Data catalogs and metadata management have grown in importance, especially in the wake of the European Union General Data Protection Regulation (GDPR), with its stricter requirements for handling personal data. Some of the vendors in this market started with products for privacy management and then implemented data catalogs and metadata management as core components to serve requirements other than "simple" data protection.
The cost of data usage can be reduced if users access the most suitable sources in a targeted way. The potential added value increases to match. On the other hand, better data governance and data security give a boost to security or help mitigate security risks and avoid potential compliance violations.
In an age of looking not only at increasing volumes of data on increasing numbers of systems but also at an increasingly larger number of physical systems for data storage and management, consolidation tools (more specifically, metadata management and data catalogs) and integration systems are required to keep track of information. The days when IT managers primarily thought of relational databases and SQL when it came to data are long gone. Massively growing volumes of data add complexity to information handling and raise the bar for a comprehensive data architecture.

Data Fabric

Data security and governance are central components in a comprehensive data architecture, or data fabric (Figure 1). Data architecture in this context explicitly refers to managing and using systems and their architecture and interaction, not to the information itself and its structures. Whereas data architecture at the information level is conceivable for isolated use cases, organization-wide approaches in the sense of an enterprise data model failed more than a decade ago.

A model for data management builds on the various sources ranging from traditional databases to business applications and their data storage systems to analytic applications. All of these sources often generate information themselves, which, in turn, can serve as a source for other usage scenarios.

The first integrating layer is metadata management and the resulting data catalogs, which provide an overview of what data (and of what quality) can be found where. However, state-of-the-art tools also provide more detailed information on the data lineage (e.g., the origin of the data) and enable evaluation and collaboration for the information.

A level above is data integration and quality. Data integration products such as extract, transform, load (ETL) and extract, load, transform (ELT) support the integration of information from different sources and the implementation of data formats.

Data quality tools check the quality of information to supplement or correct the data as needed. External sources such as address databases are often accessed for this purpose. Master data management (MDM) builds on this foundation and delivers function- and industry-specific applications for handling information such as product data.

Another level above are the analytical applications and functions for data usage (e.g., for serving up user-specific content in digital services or for decision support).

Specific Data Protection

The functions of data governance act as an overall theme across the layers of a data fabric. One central topic in this context is data security, which in recent years has developed beyond individual technical solutions for security of classic, relational databases.

Database security continues to be an important subarea in this context, protecting databases against breaches of integrity, confidentiality of information, and availability. Security primarily involves functions for the information itself stored and processed on database systems, as well as the underlying server and network infrastructure and access to the information.

However, as infrastructures and technologies for processing and storing data change – especially given cloud-native tools and the resulting hybrid infrastructures of modern and legacy approaches – the requirements change. The core functions of modern data security products include the following functional areas:

- Vulnerability assessment: identifying potential points of attack, configuration errors, and other dangers.
- Data discovery and classification: knowledge of the data and classification of the data in terms of sensitivity (e.g., personal information); tools ideally build on existing infrastructures for metadata management and data catalogs.

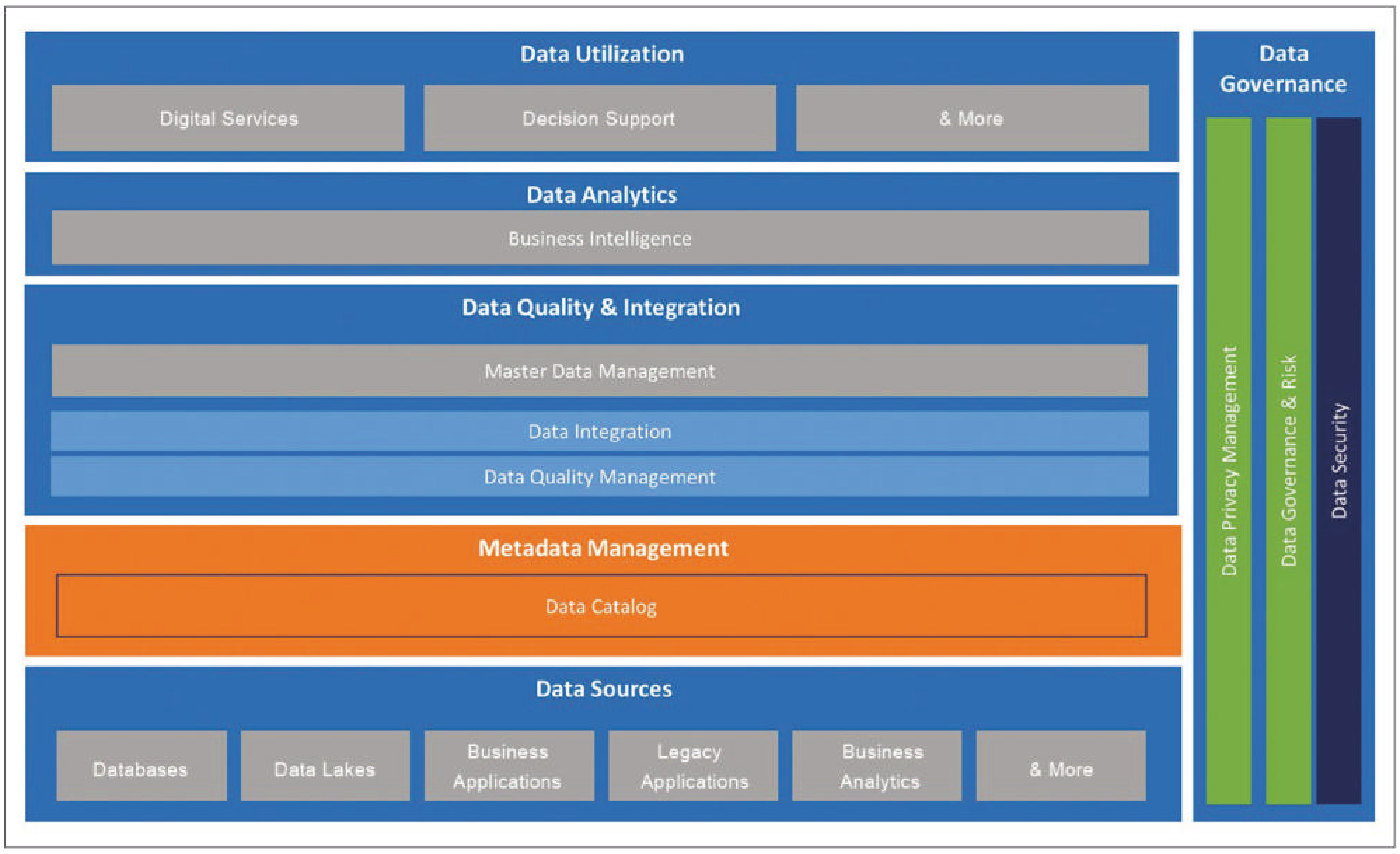


Figure 1: Enterprises need a comprehensive data architecture (or data fabric) to make the most of and protect information.

- Data protection: encryption, tagging, and other technologies for both storage and data transfer.
- Monitoring and analysis: continuous monitoring of access to and the use of data, and analysis to detect and respond to anomalies, including integration with security information and event management (SIEM) tools.
- Threat prevention: guarding against targeted attacks such as SQL injection.
- Access management: targeted protection of privileged user accounts and dynamic, policy-driven access control; often handled by specialized applications.
- Audit and compliance reporting: automatically generated and ad hoc reports and dashboards for an overview of the current security status.

These tools are fundamental building blocks of a modern, secure data fabric and must be designed to support complex hybrid environments and multicloud platforms.

Data Governance

As mentioned earlier, data governance is more than just data protection and is best defined as an umbrella term covering various functions of protecting and controlling data and data usage. Two other important functional areas in addition to data security are:

- Privacy management for handling information that falls under the scope of the GDPR. Like the entire topic of data management, it is no longer just about structured data, but also about unstructured data that needs to be analyzed, managed, and protected.
- Data governance and risk is a subarea that focuses on concrete metrics and control functions that can be used to monitor and improve compliance with defined rules for handling data. On the one hand, this sphere includes regulations in the area of data protection such as the GDPR; on the other hand, it encompasses other requirements for handling sensitive information, as well as internal rules for handling and protecting particularly critical and valuable data. Such tools typically dovetail with IT governance, risk, and compliance (GRC) products to deliver data into a higher level of risk management.

A good strategy for data governance is always built on integration of the specific functions with data security, as well as the underlying metadata management and data catalogs.

Flexibility and Coordination

Perhaps the biggest challenge in implementing a data fabric – and in sub-areas such as metadata management or data governance – is that people work with data everywhere in organizations. The multiple areas of use and numerous stakeholders make it difficult to avoid a proliferation of initiatives and technical approaches.

However, precisely here, well-thought-out and comprehensive approaches can help, including an architecture with associated operating models (e.g., target operating models, TOMs) for the data fabric and service-oriented approaches for providing the technical implementation. Experience shows that very few divisions in the corporate environment actually want to implement their own tools if they can turn to a functionally useful service with a suitable operating model. In other words, with a correctly implemented data fabric as an internal service, a corporation has a good chance of containing a significant amount of undesirable growth.

Communication is important so that the different divisions in the company know which services are available. Because the users and areas involved come from both IT and business, advisory support is required on top of technical services. In many cases, today's tools in the wider data management environment support functions for collaboration; assessing data usability; and collaboration among users, data stewards, and administrative and technical users.

Conclusions

The importance of data security means corporations need to move away from isolated, incomplete tools that are expensive, often fail to cover important security and data governance requirements, or do not do a complete job of providing coverage. An additional consideration is that continually introducing new local solutions simply takes too much time. However, solutions must also serve the quite different requirements of stakeholders from business, IT security, data protection, and other areas.

To deal with data efficiently and effectively, corporations need a strategy in which a holistic view of the required elements (e.g., a data fabric) plays a central role. Individual approaches in the area of analysis are not enough. IT managers need to implement both the foundations with metadata management and data catalogs and the interdisciplinary functions of data governance and data security correctly to be able to work optimally with data and generate the desired added value. ■

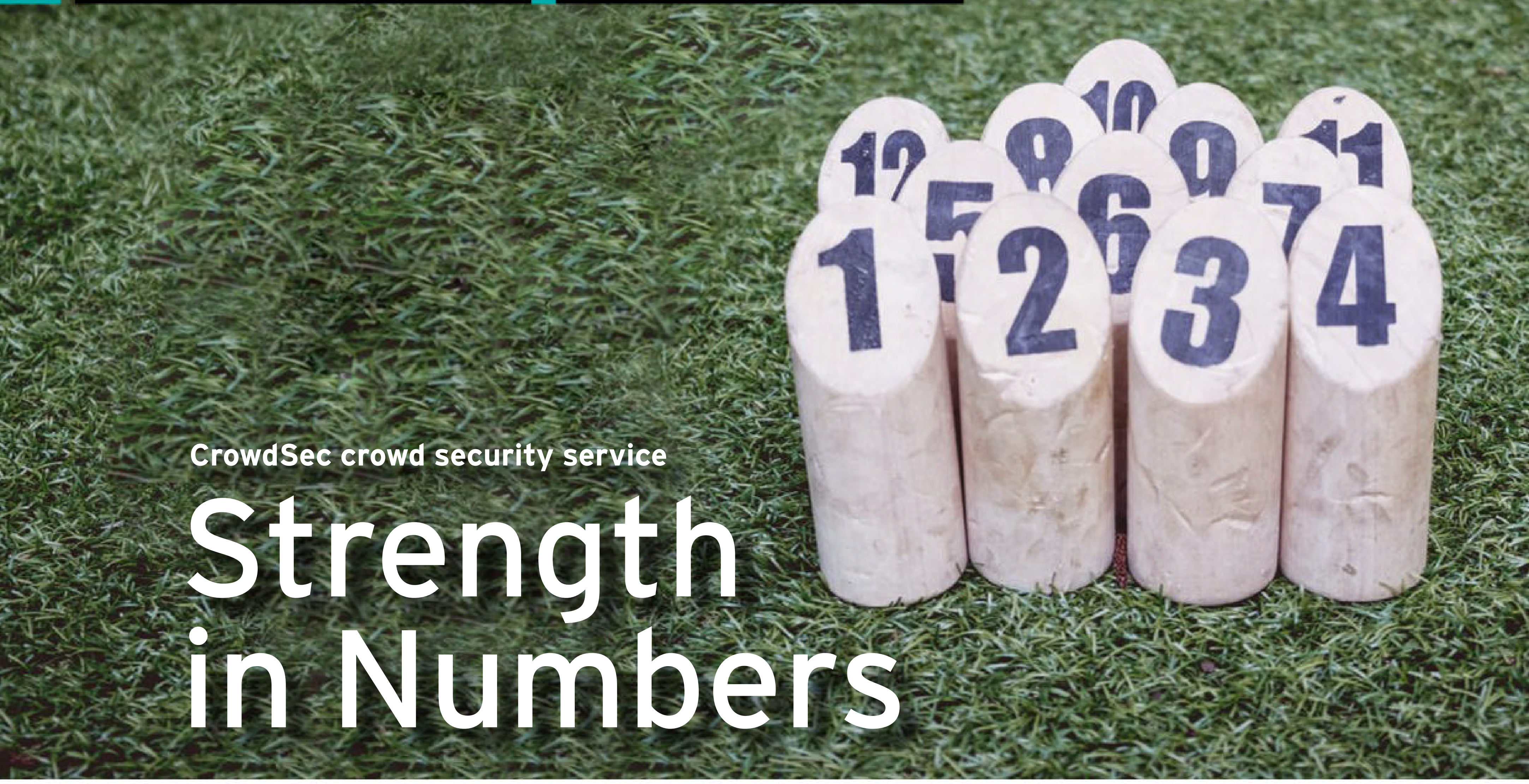
Author
Martin Kuppinger is the founder and Principal Analyst of KuppingerCole Analysts AG.

FOSSLIFE

Open for All

News • Careers • Life in Tech
Skills • Resources

FOSSlife.org



CrowdSec crowd security service

Strength in Numbers

Threats can be detected and averted at an early stage with crowd security, in which organizations form a community to take concentrated action against cyberattacks by sharing attack data. We explain how this strategy works with the CrowdSec cloud service. By Thomas Joos

Cyberattacks are constantly on the rise, and ransomware is spreading rapidly. As a result, corporations also need to update their security strategies constantly. And it is better to fight against aggressors together than go it alone, according to CrowdSec [1], an open source cloud service and participative intrusion protection system (IPS) capable of analyzing the behavior of systems and providing a customized response to attacks. The tool acts as a community, sharing attack intelligence and fighting cyber criminals together. In this way, corporations can rely on data from the entire community to protect their servers, and not just on information obtained from their enterprise.

Information can come from syslogs, CloudTrail events, security information and event management (SIEM) systems, and other sources (e.g., from firewalls or the event viewer of Windows servers). Community members can access the details of the analyzed data and build their own intrusion detection systems (IDSs). The process of sending and receiving information can also be fully automated. After the initial setup, the system is autonomous. You can check the cloud service web console to discover whether your servers have been attacked and whether you need to take any action. The software used in a CrowdSec network runs locally, but it can access community data offline, which means the software agent at the local data center can quickly identify unfriendly IP addresses drawn from community information. If your installation discovers new, unfriendly IP addresses itself, it in turn can upload that data to the cloud. After verification, these new addresses are published in the community.

Agent-Based Flexible Use

Corporations do not need to replace their entire security setup when they start using CrowdSec. Because the functionality resides in the cloud, you don't even have to operate your own servers. You just need to install agents on the individual servers. CrowdSec's fields of application can be broken down into different scenarios (e.g., the exploitation of the Log4j vulnerability).

Adding your own scenarios is no problem. Organizations can decide for themselves which potential attacks they want to defend against through CrowdSec and what information of their own they want to contribute to the community's common fight. The scenarios are often defined as YAML files and can be easily integrated into your environment. CrowdSec also works with honeypots to attract attackers. The reputation databases created in the process are also available to the general public.

Crowd security tools have notably been successful in defending against the Log4j vulnerability. Many applications and server operating systems contain undocumented components from the Java library. As a result, almost all networks are at risk. Because of the huge volume of data available for analysis, CrowdSec detects where exploits occur, and if other members of the community are using the same product, the exploit probably exists there, too.

Installing the Agent on Linux and Windows

Installing CrowdSec is a three-step process: (1) Set up the web console by creating an account with the provider (free of charge). (2) Install the agent on the servers you want to include and connect the servers to the console through the agent. (3) Use "bouncers" to prevent attacks

Photo by Dan Burton on Unsplash

actively, wherein the system simply blocks certain IP addresses.

Typically, you can use the package manager to install CrowdSec on Linux. On Debian systems, use the commands:

```
curl -s
https://packagecloud.io/install/⮐
  repositories/crowdsec/crowdsec/⮐
  script.deb.sh | sudo bash
sudo apt-get update
sudo apt-get install crowdsec
```

CrowdSec's agent is lightweight open source software that detects peers with aggressive behavior to prevent them accessing your systems. Although the Linux client is more mature, you can connect Windows servers to CrowdSec with an agent available on GitHub [2]. After downloading the MSI file, which is about 40MB in size, you can proceed to install the agent on Windows. The installation doesn't require any configuration steps and can be easily automated.

CrowdSec relies on collections to defend against threats. The install wizard automatically applies a collection for Windows servers to the systems. The installer also drops the matching system service into place on the Windows device. Servers connected in this way can be managed from *https://app.crowdsec.net* on the web console.

Configuring CrowdSec

Unlike Linux, CrowdSec on Windows does not yet support automatic configuration at install time. Some work is required after installing the agent. The default configuration intercepts brute force attacks against the remote desktop protocol (RDP), server message block (SMB) protocol, or any type of remote authentication that Windows uses, which already offers a modicum of protection. Whenever the CrowdSec service is updated or customized on Windows, you need to restart the service. You can do this in PowerShell by typing:

```
restart-service crowdsec
```

The required files and functions are placed in the Windows `C:\Programs\CrowdSec` directory. The agent's executable (`crowdsec.exe`) and command-line (`cscli.exe`) files also reside there. The various configuration files are stored in the `config` directory, and CrowdSec's internal logfiles are stored in the `logs` folder.

More advanced configuration relies on the `cscli.exe` command (Figure 1). The installation wizard also includes the agent executables in the Windows path, which means you can run the commands from any location in PowerShell, in Windows Terminal, or from the command prompt.

In the same way, you install different collections on a Windows server (Figure 2). To do this, run the command:

```
cscli collections install ⮐
  crowdsecurity/windows
```

Another collection for protecting SQL servers is installed with:

```
cscli collections install ⮐
  crowdsecurity/mssql
```

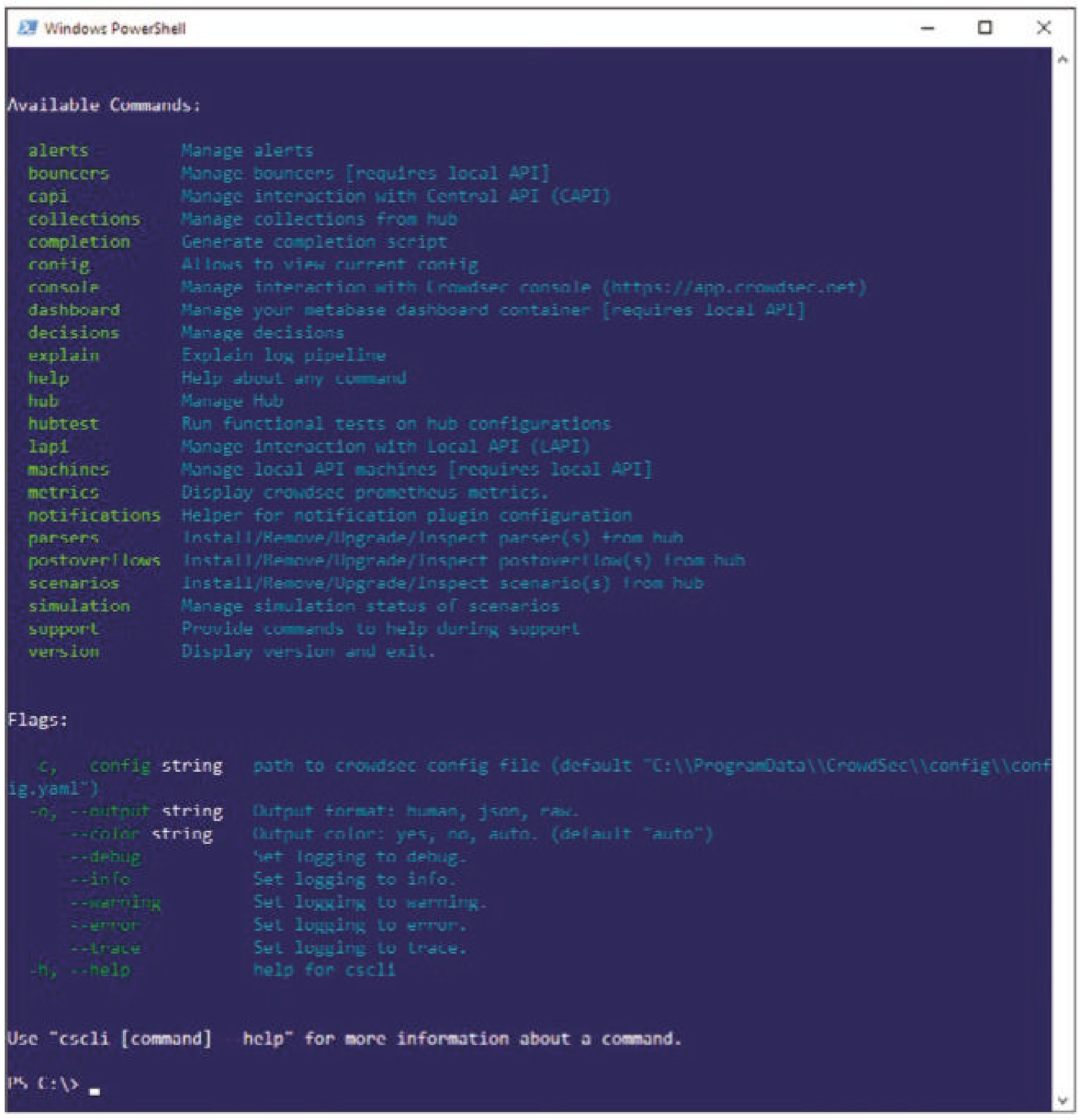
The collection detects attack attempts on the authentication of SQL servers. To monitor SQL servers you now need to edit the `C:\ProgramData\CrowdSec\config\acquis.yaml` file:

```
source: wineventlog
event_channel: Application
event_ids:
  - 18456
event_level: information
"label":
  type: eventlog
```

A separate collection is available for protecting Internet Information Services (IIS)-based web servers:

```
cscli collections install crowdsecurity/iis
```

Again, you need to edit a file – in this case, `C:\ProgramData\CrowdSec\config\acquis.yaml`:



```
Available Commands:
  alerts          Manage alerts
  bouncers        Manage bouncers [requires local API]
  capi            Manage interaction with Central API (CAPI)
  collections     Manage collections from hub
  completion      Generate completion script
  config          Allows to view current config
  console         Manage interaction with Crowdsec console (https://app.crowdsec.net)
  dashboard       Manage your metabase dashboard container [requires local API]
  decisions       Manage decisions
  explain         Explain log pipeline
  help            Help about any command
  hub             Manage Hub
  hubtest         Run functional tests on hub configurations
  lapi            Manage interaction with Local API (LAPI)
  machines        Manage local API machines [requires local API]
  metrics         Display crowdsec prometheus metrics.
  notifications   Helper for notification plugin configuration
  parsers         Install/Remove/Upgrade/Inspect parser(s) from hub
  postoverflows   Install/Remove/Upgrade/Inspect postoverflow(s) from hub
  scenarios       Install/Remove/Upgrade/Inspect scenario(s) from hub
  simulation      Manage simulation status of scenarios
  support         Provide commands to help during support
  version         Display version and exit.

Flags:
  -c,   config string   path to crowdsec config file (default "C:\\ProgramData\\CrowdSec\\config\\conf
ig.yaml")
  -o, --output string   Output format: human, json, raw.
      --color string    Output color: yes, no, auto. (default "auto")
      --debug           Set logging to debug.
      --info            Set logging to info.
      --warning         Set logging to warning.
      --error           Set logging to error.
      --trace           Set logging to trace.
  -h, --help            help for cscli

Use "cscli [command]  help" for more information about a command.

PS C:\> _
```

Figure 1: CrowdSec is managed on a server with the `cscli` tool, available on both Linux and Windows servers.

```
use_time_machine: true
filenames:
  - C:\\inetpub\\logs\\LogFiles\\*\\*.log
"label":
  type: iis
```

To include the IIS event viewer entries from Windows in CrowdSec, use:

```
source: wineventlog
event_channel:
  Microsoft-IIS-Logging/Logs
event_ids:
  - 6200
event_level: information
"label":
  type: iis
```

Windows firewall monitoring can be enabled by typing:

```
cscli collections install ⮐
 crowdsecurity/windows-firewall
```

Here, too, you need to make a few changes to the YAML file mentioned earlier:

```
filenames:
  - C:\\Windows\\System32\\LogFiles\\⮐
    Firewall\\pfirewall.log
"label":
  type: windows-firewall
```

After adding these collections for Windows, CrowdSec offers the following protections:

- RDP/SMB: brute force detection
- IIS: HTTP attacks
- SQL Server: brute force detection
- Windows Firewall: network scan detection

The Windows collections mainly analyze Windows logfiles and various scenarios for protection against password attacks. After installing CrowdSec on a server and integrating the desired collections, you need to register each server as an instance – more about that later. You can then use the web interface for administration. Typing

```
cscli collections list
```

accesses the settings on the current server.

Instances and Bouncers

Once the local CrowdSec instance is configured, you can connect it to the service by typing something like:

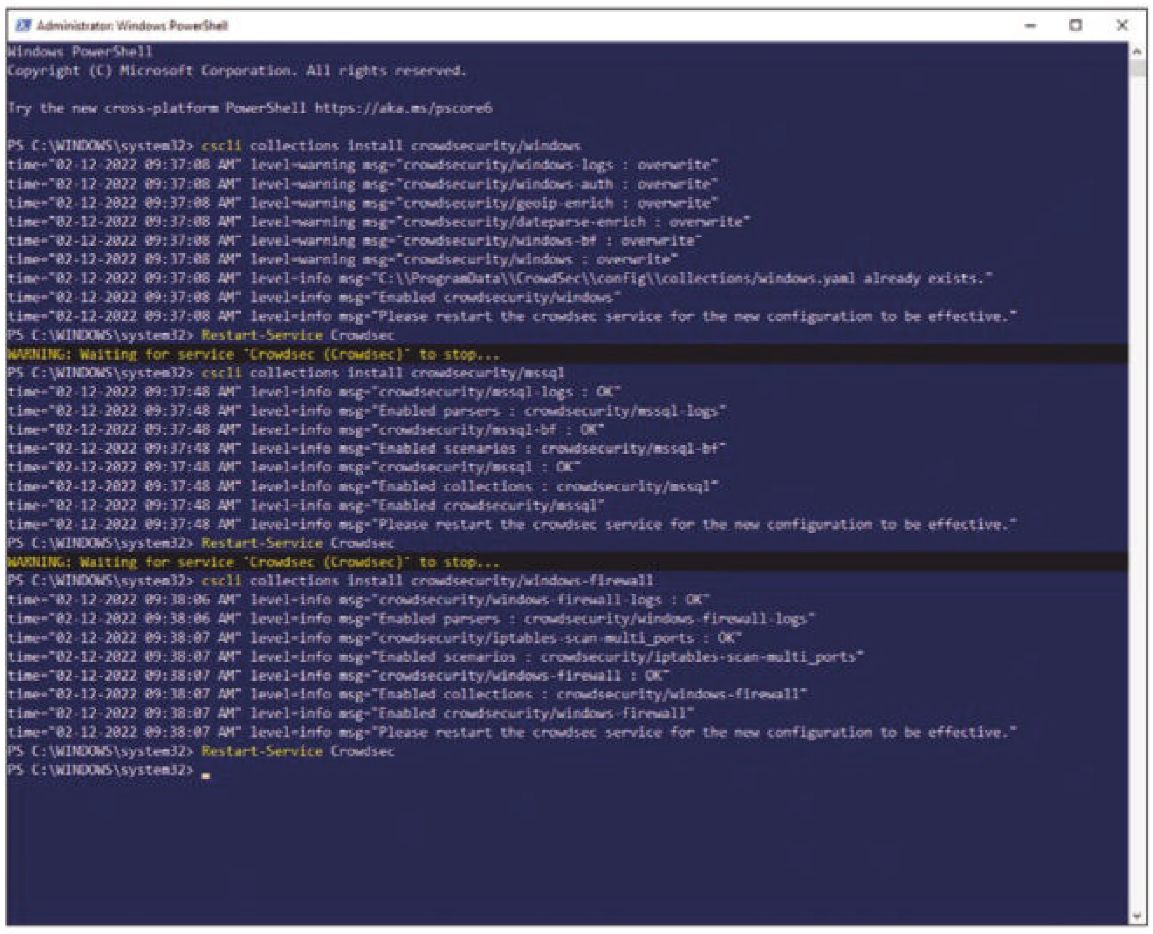
```
cscli console enroll ⮐
 cl5zgf4qs00 030wjqmvrt7s30
```

The web console must be running on the computer, and you need to be authenticated. After that, you will see that instance under *Instances* in the web interface. Clicking *Accept* adds it to the interface. From this moment on you can see the status of the server. Security information about the connected server can be obtained with *Alerts*, and *Activity* shows the last actions you have performed (e.g., to which servers you connected).

Bouncers block the attacks detected by CrowedSec. You must install these on the server. With Windows, for example, the bouncer for Windows Firewall manages and automatically updates rules for blocking suspicious IP addresses. Windows also requires the .NET6 framework. The full installation files are on the GitHub page [3]. The bouncer configuration is described in more detail online [4].

Conclusions

Cybercriminals often act as a group. One way of combating attackers is to join a community yourself and leverage the information gathered by all of its members. Systems such as CrowdSec support most Linux distributions and Windows. As shown here, however, some manual reconfiguration work is required. ■



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> cscli collections install crowdsecurity/windows
time="02-12-2022 09:37:08 AM" level=warning msg="crowdsecurity/windows-logs : overwrite"
time="02-12-2022 09:37:08 AM" level=warning msg="crowdsecurity/windows-auth : overwrite"
time="02-12-2022 09:37:08 AM" level=warning msg="crowdsecurity/geoip-enrich : overwrite"
time="02-12-2022 09:37:08 AM" level=warning msg="crowdsecurity/dateparse-enrich : overwrite"
time="02-12-2022 09:37:08 AM" level=warning msg="crowdsecurity/windows-bf : overwrite"
time="02-12-2022 09:37:08 AM" level=warning msg="crowdsecurity/windows : overwrite"
time="02-12-2022 09:37:08 AM" level=info msg="C:\\ProgramData\\CrowdSec\\config\\collections/windows.yaml already exists."
time="02-12-2022 09:37:08 AM" level=info msg="Enabled crowdsecurity/windows"
time="02-12-2022 09:37:08 AM" level=info msg="Please restart the crowdsec service for the new configuration to be effective."
PS C:\WINDOWS\system32> Restart-Service Crowdsec
WARNING: Waiting for service 'Crowdsec (Crowdsec)' to stop...
PS C:\WINDOWS\system32> cscli collections install crowdsecurity/mssql
time="02-12-2022 09:37:48 AM" level=info msg="crowdsecurity/mssql-logs : OK"
time="02-12-2022 09:37:48 AM" level=info msg="Enabled parsers : crowdsecurity/mssql-logs"
time="02-12-2022 09:37:48 AM" level=info msg="crowdsecurity/mssql-bf : OK"
time="02-12-2022 09:37:48 AM" level=info msg="Enabled scenarios : crowdsecurity/mssql-bf"
time="02-12-2022 09:37:48 AM" level=info msg="crowdsecurity/mssql : OK"
time="02-12-2022 09:37:48 AM" level=info msg="Enabled collections : crowdsecurity/mssql"
time="02-12-2022 09:37:48 AM" level=info msg="Enabled crowdsecurity/mssql"
time="02-12-2022 09:37:48 AM" level=info msg="Please restart the crowdsec service for the new configuration to be effective."
PS C:\WINDOWS\system32> Restart-Service Crowdsec
WARNING: Waiting for service 'Crowdsec (Crowdsec)' to stop...
PS C:\WINDOWS\system32> cscli collections install crowdsecurity/windows-firewall
time="02-12-2022 09:38:06 AM" level=info msg="crowdsecurity/windows-firewall-logs : OK"
time="02-12-2022 09:38:06 AM" level=info msg="Enabled parsers : crowdsecurity/windows-firewall-logs"
time="02-12-2022 09:38:07 AM" level=info msg="crowdsecurity/iptables-scan-multi_ports : OK"
time="02-12-2022 09:38:07 AM" level=info msg="Enabled scenarios : crowdsecurity/iptables-scan-multi_ports"
time="02-12-2022 09:38:07 AM" level=info msg="crowdsecurity/windows-firewall : OK"
time="02-12-2022 09:38:07 AM" level=info msg="Enabled collections : crowdsecurity/windows-firewall"
time="02-12-2022 09:38:07 AM" level=info msg="Enabled crowdsecurity/windows-firewall"
time="02-12-2022 09:38:07 AM" level=info msg="Please restart the crowdsec service for the new configuration to be effective."
PS C:\WINDOWS\system32> Restart-Service Crowdsec
PS C:\WINDOWS\system32> _
```

Figure 2: On Windows, you first need to configure CrowdSec with the `cscli.exe` file after installing the agent.

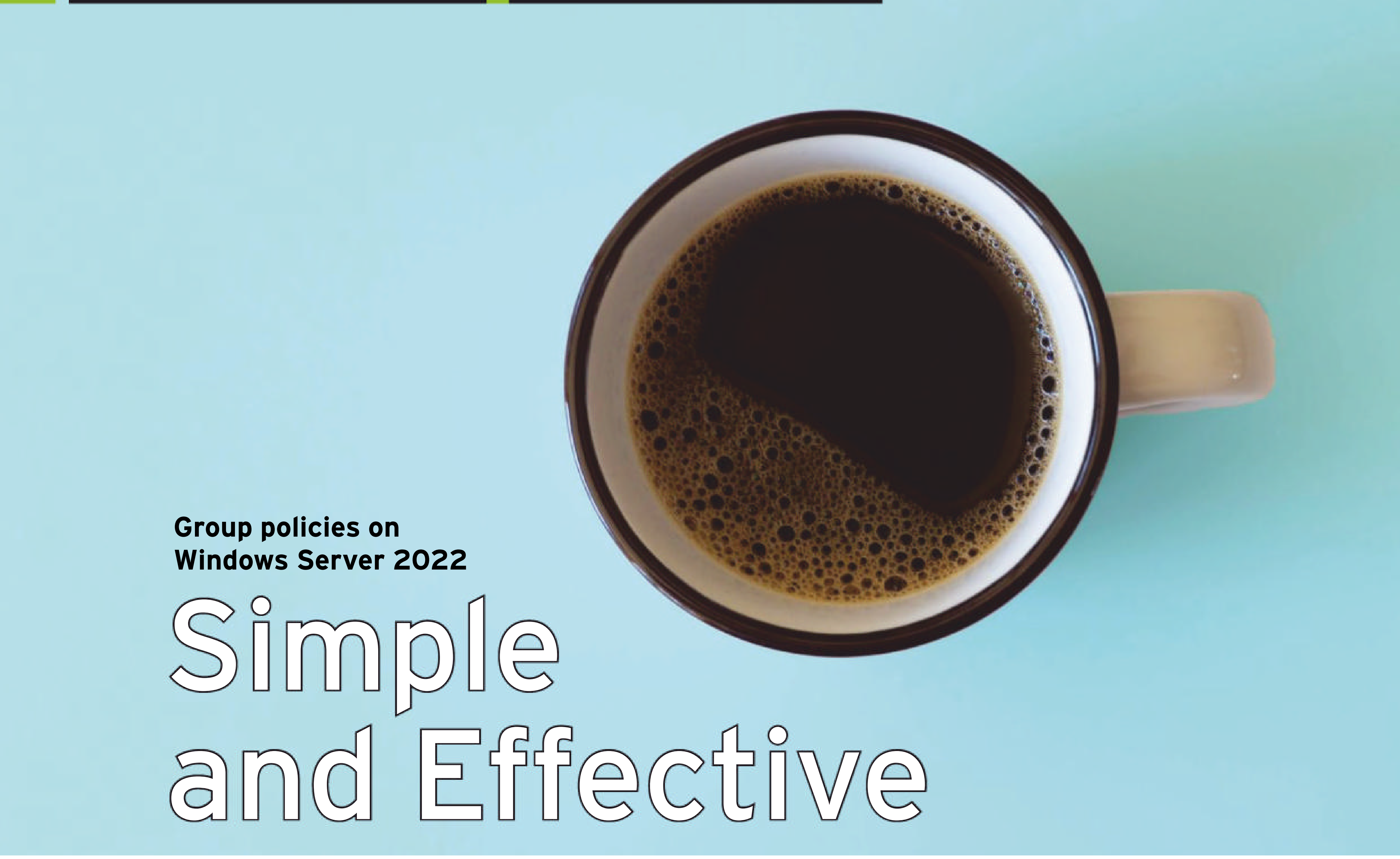
Info

[1] CrowdSec: [https://www.crowdsec.net]
[2] CrowdSec agents on GitHub: [https://github.com/crowdsecurity/crowdsec]
[3] Windows Firewall bouncer: [https://github.com/crowdsecurity/cs-windows-firewall-bouncer/releases]
[4] CrowdSec on Windows: [https://docs.crowdsec.net/docs/getting_started/install_windows/]

The Author

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

ALGEBRA
2 VEGAN
architecture
architecture
ANCIENT ART
MODERN ART



Group policies on Windows Server 2022

Simple and Effective

We discuss how to manage and secure clients with group policy object templates and look at some recommendations from various governmental and non-governmental security advocates. By Mark Heitbrink

Every admin knows group policies inside out. Group policy objects (GPOs) are still the most effective means of centrally managing and, above all, protecting clients. They play a crucial role, especially when it comes to protection against ransomware.

Generally speaking, Windows Server 2022 reveals no technical innovations in the area of group policies for local installations. The GPO infrastructure and the available feature set are static, with no more development in this area. However, some innovation and change is emerging in the cloud, but only if you sign up for the right plan. Still, there's nothing out of the box in a plain vanilla Azure Active Directory (AD) that controls the client as extensively as group policies can locally. With this in mind, I look at new or changed approaches that have emerged over 20 years of group policy design and structure, with an emphasis on rules that should always be implemented, even if nothing has changed technically and the best practices have been valid for years.

GPOs in the Light of Ransomware

Where admins struggled with desktop configurations in the early days of AD, today the greatest effort is put into defending the environment against ransomware and the sword of Damocles called the General Data Protection Regulation (GDPR) in the event of a possible data loss. Surprisingly, very few ADs are up to these requirements in terms of delegation and group policies.

The companies affected by ransomware attacks often have many things in common, but the main similarity is an IT structure built 20 years ago. Admins have neglected to change fundamentals and keep structures current, instead focusing on issues other than AD as a core login resource. User login works fine, why should there be a problem? This way of thinking is now getting in the way. After ransomware has made it into an organization, it spreads with PsExec and a batch file distributed by the admin share.

This Windows NT4 technology is still commonplace today and many administrators use it in their daily work. The doors are open internally, there is no firewall strategy between the clients, no software allowlist, and no delimitation of the administrative accounts in terms of their spheres of action. The passwords in use may be "secret" literally but not by nature. Instead, you see password recycling, identical passwords for various accounts, and a domain administrator password that is often 15 years old. The last time I tried to change the password, so many services failed that it was easier to keep the old one than to find all the places it was stored. At any rate, this and similar excuses are often sought when things come to light during an audit or health check – or simply when talking to a service provider. In the worst case, ransomware will exploit these flaws.

One Setting Does Not a Policy Make

Admins have spent a long time and much brain power worrying about configuration and customizing group policies. Today, the focus is more on security requirements, as well as reducing telemetry data. However, you do not have to reinvent the wheel, because the technology is available and guidelines exist.

The challenge facing an administrator today when it comes to group policy is answering the question, "What do I need to configure?" The answer, as always, depends on your own capabilities: As a policy, some configurations are just a single switch and extremely trivial from a policy perspective. Not much more happens than toggling a registry value, but the function behind it often requires the right hardware, as in the case of virtualization-based code integrity protection; or the case of a certificate infrastructure and processes that are required to implement code signing; or whether you are confronted with organizational issues when you have to handle three, four, or even more accounts for administrative tasks (depending on the security group or level).

In the search for "what" predefined templates can help, vendors already offer ready-made policy sets and make them available for download. Others charge money, or they provide a fully documented PDF that you have to convert into a GPO yourself.

GPO Innovations in Windows Server 2022

Before I talk about the general methods and rules independent of Windows Server 2022, I'll offer some resources that look at what's new:

- The Windows Server 2022 ADMX and GPO settings in an Excel sheet [1]
- Administrative templates (ADMX) for Windows Server 2022 [2]
- Group policy settings for Windows Server 2022 [3]

Windows Server 2022 comes with 47 new policy settings, for a total of 4,442 policies. More key data include:

- 39 new policies for the computer object and eight for the user
- 13 policies in the privacy and telemetry section
- Six policies for Windows Defender
- Another six GPOs that control the Windows sandbox

Looking at the numbers and distribution alone, you can see that no major changes are required when it comes to controlling the operating system. Microsoft has added functionality to some components, but it's not as big a leap as going from Windows Server 2003 to 2008 R2 or 2008 R2 to 2016. Few companies used the intermediate versions (e.g., Server 2019). Most moved from Server 2003 to 2008 R2 to 2016, which means Server 2022 is now the next step in AD for many IT organizations.

Static Windows as a Problem

Depending on the region of the world, different institutions claim to be able to configure a system more securely than when it was delivered. This claim poses the question of why the system was not already secure in the as-purchased state. The answer lies in the Microsoft ecosystem with its ancient roots. Software that is 20 to 30 years old is still being used in industry, as well as banks and insurance companies. It has to carry on working. Companies expect to be able to continue using their expensive, customized enterprise resource planning (ERP) systems.

You could also tell the sorry tale of Internet Explorer at this point. Techniques that were new when it was established are still in use. Unfortunately, these are not so easy to replace and adapt to today's conditions. Sometimes a solution presents intself – bug fixing, hardening, and more security courtesy of Microsoft – if you simply integrate a patch. However, security-related changes are often only possible after announcements made in advance, because of dependencies on the third-party ecosystem (e.g., if you disable SMBv1, LDAP signing, or the default setting for Excel macro execution). Microsoft itself defined in an RFC at some point the rules of the game and cannot change them dynamically. Other applications rely on the set of rules defined back then. Consider, if you will, the access token with its maximum size of 65,536 bytes. When the size was set, no administrator believed that a user would ever be in more than 1,000 security groups, exceeding the token's maximum size. Now that some large companies are experiencing their third or even fourth AD migration, including taking the security identifier (SID) history with them, space is suddenly at a premium.

SMBv1 can be considered an example of a change for which a long period of notice was given. Every Microsoft operating system from Vista onward speaks SMBv2, but SMBv1 was not removed from the system until Windows 10 v1903, and it can still be enabled in any build – even in Windows 11 – because various devices (e.g., multifunction printers with scan-to-UNC interfaces) still only speak SMBv1.

GPO Templates

Once a Microsoft technology was released or allowed, people started using it, and getting rid of it was very difficult or required individual testing and assessment on the part of an organization. In this dilemma, Microsoft unfortunately cannot act as they would like. Although the limit of functionality is defined by the ecosystem, that does not mean that it would not be better to configure things correctly, which is where templates and recommendations come in. Examples include:

- Microsoft security baselines from the Microsoft Security Compliance Toolkit [4]
- SiSyPHuS, a study of system design, logging, hardening, and security features in Windows 10 from the German Federal Office for Information Security [5]
- Center for Internet Security benchmarks [6]
- US Department of Defense Security Technical Implementation Guide [7]
- Australian Centers for Cyber Security Guidelines for system hardening [8]
- gp-pack Privacy and Telemetry from Mark Heitbrink [9]

Before I look at the individual templates with their advantages and disadvantages, I can issue a blanket statement for all of them: The newer templates go back to the Microsoft baselines and the older ones align with it. Microsoft itself has gone on a major offensive and has started making recommendations on how to make its own system better, in the sense of "harder."

The focus of the templates, with the exception of gp-pack Privacy and Telemetry, is on operating system hardening. In contrast, the reduction of transmitted data (silencing) is usually only rudimentary. It's about security – nothing more, nothing less. Data that Windows sends to a manufacturer can improve security; however, it can contradict the GDPR under certain circumstances, and guidelines as to what is permitted or prohibited are not clear. I recommend sending as little data as possible to ensure continued secure operation. Unfortunately, this plan is defined individually, and as long as there are no legally effective rulings on what is allowed and what is not, admins operate in an unsatisfactory gray area.

Security Template Integration

My recommendation in terms of GPO templates is: Import the values for settings without much to-do and then investigate in the test scenario what you can and cannot implement. The test shows the values that cause errors. If you try to resolve everything up front, you will never get done. Often admins even take out some values for fear that something bad might happen. Of course, you can do a rough review in advance but avoid addressing all the details, especially when many settings are not self-explanatory or refer to algorithm or encryption techniques that you cannot evaluate because the basic technical understanding is missing.

The approach first uses values without any attempt at evaluation, and the test shows whether the value should be kept in the recommended setting. In the end, you have a construct that is genuinely best defined for your company. All values are implemented, and only the requirements from your own infrastructure brought about changes. If an external audit occurs with such a set of rules, only a few results will show up as deficiencies because they were not implemented, and you had no knowledge of these facts. External auditors only apply one of the known yardsticks and check for the existence of rules.

It is important to note that implementation is an ongoing process. It does no good to enable the rules on a cutoff date and then just leave them be for the next few years without making any further adjustments. Rules and regulations are not static and need to be reviewed every six or 12 months. Where vendors provide updates, you will want to check and integrate the updates as soon as possible. The overhead of making small changes on a regular or permanent basis is far less than that of making massive adjustments every two years.

Another important point, quickly clarified, is the selection of the appropriate template, because – in fact – it is simply all those mentioned. There is no debate as to which of the templates is the better or more comprehensive. Each provider has their own ideas that are important in their own value systems, but it would be a mistake not to adopt all of these points, assuming they only generate work and do not cause any problems.

Time and labor are unfortunately still the biggest arguments against template integration, which I look at in detail below.

Microsoft Security Baseline

The Microsoft Security Compliance Toolkit has been at version 1.0 since it was released – the version number has never been incremented. The toolkit provides configuration defaults for any current operating system. For each build, a baseline is released for both the server and the client. Moreover, Microsoft provides suggestions for configuring the Chromium-based Edge browser and the Office 365 package outside of a fixed cycle. This coverage comprises operating systems, browsers, and Office, the three most important areas for Microsoft in the enterprise.

The suggestions are delivered as a ZIP file and always with the same structure. The file contains Excel spreadsheets with the new settings and a PDF that explains the included Microsoft security baseline (MSB) blog post with the changes. The policies come as a Group Policy Management Console backup and can be imported in bulk by a script, but the single values can also be integrated into existing policies with the local GPO (LGPO) utility [10]).

Other administrative templates (ADMX files) have the guidelines as an HTML report, meaning you can take a look at the content before importing. The policies are broken down into user and computer objects and their use cases. BitLocker and Defender are separate policy objects, although they are part of the operating system, because these two tasks are often handled by third-party software. Microsoft saves you a lot of the work by removing unnecessary and unused settings from the policy.

This approach illustrates an important point. Microsoft wants MSB to be used and actively removes any stumbling blocks that might prevent its integration. They have deliberately left out some settings because they (probably) can never be realized in practice. For example, some configurations are set to monitor instead of deny or force. You may well be shocked to discover how easy it is to integrate the policy in practical terms.

The name "Baseline" is a good description of the underlying concept. It is the minimum that must be defined, the lowest common denominator that can be introduced without spending a great deal of effort and time on long integration tests.

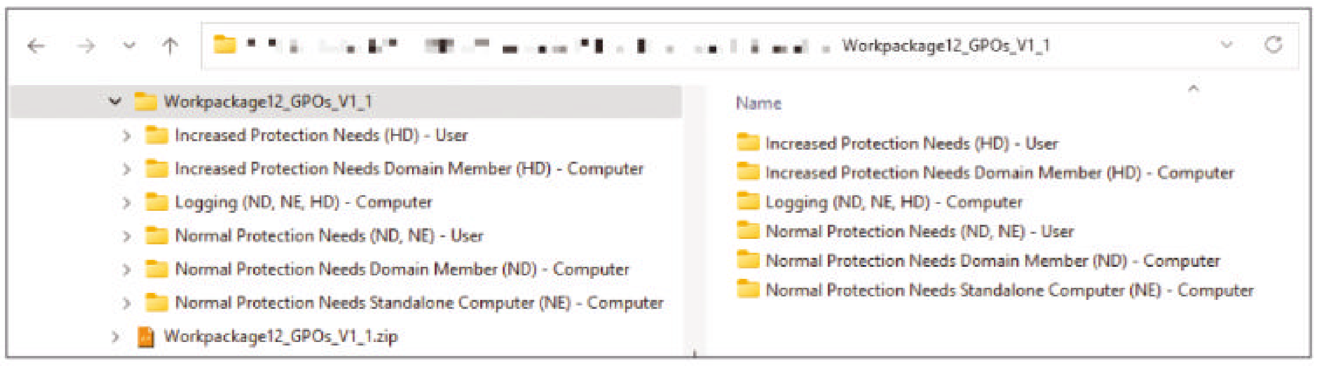
Microsoft does not prevent telemetry or cloud access in the baseline; some criticism may be justified from a European perspective. Because Microsoft sorts the baseline by topic, it often seems a bit smaller than other rulesets. Microsoft also tries to define MSB for the as-delivered state of the current build and only tries to specify values that are not already implemented. It does not try to reconfigure default values that are already correct. Unfortunately, this process is not implemented consistently. I found definitions for various values in the baseline that have not been necessary since Windows 8 (e.g., WDigest settings). However, because other templates check for the existence of this value, Microsoft has continued to include it to avoid errors in an audit report.

BSI SiSyPHuS

The German Federal Office for Information Security (BSI) expands the Microsoft baseline to include items such as the cloud. However, it has some potential for improvement in the area of telemetry. At the time of writing this article, a new version of SiSyPHuS was in the works; when it would be released was uncertain.
Conceptually, in contrast to the MSB, the BSI rules and regulations were created to build on each other. Each object has a "normal" and an "increased" protection requirement (Figure 1). The BSI distinguishes between individual computers and domain members, except for the "logging" policy configuration, which is regulated identically for all systems. This feature reduces triple configurations with identical settings in each policy.
If you want to realize a high level of protection for your computers or users, you need to apply at least three group policy objects: *Normal Protection* plus *Increased Protection* plus *Logging*. If you add up the rulesets, the result is then rated high.
You find this type of evaluation in other templates, too. The problem is the evaluation itself, which has no rules and has its own value system. The bar for grading each setting is ultimately something you define yourself; there are no real-world arguments. This categorization dates back to the time when the major concern was reducing the administrator's workload. Normal protection has less potential for error than increased protection, so applying the normal level of protection across the board in AD should be the minimum. If you use other adjectives such as "poor" and "good" instead of "normal" and "increased," suddenly things don't sound so cozy. Why should an administrator settle for a normal (poor) configuration when they can implement an increased (good) protection requirement?
If an incident occurs, accusing fingers will point at the administrator, who should always try to configure the network to the best of their ability. Administrators will not want to implement a weak configuration with the knowledge that there is a better way. Merging the two templates into one would take a lot of political hassle out of the system, especially because you will always have debates about every setting as to whether the value is now normal or increased. Just consider the screensaver. To which of the two categories does the value belong? Is it normal because by now all users have gotten used to a screensaver coming on after a certain amount of time, or is it more of a special factor because a password is needed to unlock it again? Depending on the value at stake, discussing how to classify this could well be just a waste of time. The far easier approach here is to integrate without evaluating.
One minor point of criticism relates to the way SiSyPHuS sorts the folders in the ZIP file. Instead of using a single backup folder with a central `manifest.xml` file such as MSB, BSI stores each policy backup in its own folder. This arrangement might look neatly sorted and structured, but it makes importing the policies more complex, because several sources need to be read instead of just one.
If you slip SiSyPHuS over the Microsoft baseline, you will have no conflicts whatsoever with regard to the set values. The overlapping settings are defined identically. The thinking on the question of what is considered secure is unanimous. Unfortunately, SiSyPHuS is defined for 1809 LTSC, so no practical ruleset has emerged here. The Long-Term Servicing Channel (LTSC) version was never intended for normal operation, and most companies have long since switched to the semiannual channel versions. Moreover, build 1809 is already three and a half years old and lagging far behind the current requirements.

CIS Benchmarks

The Center for Internet Security (CIS) has been providing suggestions for hardening IT systems for many years, and Microsoft is just one of many operating systems covered. Microsoft's Aaron Margosis and Rick Munk are involved in the CIS benchmarks; they also happen to be the two leads on MSB. CIS offers security configurations from Apache to Zoom. These evaluations can be assessed retroactively and compared with specifications.
CIS is a complete system: In the case of Microsoft policies, it provides GPO backups and delivers a build kit that automatically integrates and redeploys version changes and updates; alternatively, it can make changes to existing policies. It also comes with comparison tools and can work out the score that a system achieves on


[redacted] › Workpackage12_GPOs_V1_1

Workpackage12_GPOs_V1_1
- Increased Protection Needs (HD) - User
- Increased Protection Needs Domain Member (HD) - Computer
- Logging (ND, NE, HD) - Computer
- Normal Protection Needs (ND, NE) - User
- Normal Protection Needs Domain Member (ND) - Computer
- Normal Protection Needs Standalone Computer (NE) - Computer

Workpackage12_GPOs_V1_1.zip

Name
- Increased Protection Needs (HD) - User
- Increased Protection Needs Domain Member (HD) - Computer
- Logging (ND, NE, HD) - Computer
- Normal Protection Needs (ND, NE) - User
- Normal Protection Needs Domain Member (ND) - Computer
- Normal Protection Needs Standalone Computer (NE) - Computer

Figure 1: Increased or normal protection requirements in GPOs? BSI SiSyPHuS knows the answer.

the basis of the benchmark's internal rating system. Both GPO backups and the build kit are commercial products. Available free of charge is a 1,300-page PDF that contains all settings. This tome can seem extremely daunting at first sight because of the sheer mass, but if you frequently use the GPO editor and feel at home there, you can convert the PDF into a GPO with just two hours of hard work. CIS understood years ago that the best way to document group policies is to follow the structure of the Group Policy Editor. In the editor, you start at the top in *Policies* and end up in the *Administrative Templates | Windows Update* area. This is exactly the order that CIS uses for the PDF.

The sheer volume of the PDF results from the excellent documentation of each value. Each setting has a short explanation describing the effect. Additionally, the document names each value with its policy and shows the registry entry behind it so that its existence can be checked on the target to enable compliance.

Also interesting is that the default value of the system is documented. The CIS benchmark often sets values that confirm the default in a GPO, which helps orient the actual state on the build if the system was not installed by the customer (e.g., by an OEM hardware partner). By the way, third-party installation is never a good idea, because manufacturers also earn money from advertising. You tend to end up with too much software garbage on the systems, and nobody knows what the software manipulates.

The benchmark is similar in structure to BSI SiSyPHuS, defining *Level 1* and *Level 2*, which are equivalent to *normal* and *increased*. More recently, CIS has attempted to provide further assistance in the form of the implementation groups construct (IG1 to IG3). The groups reference other CIS tools and attempt to create an assessment outside the terms of the Levels. Approximately 150 evaluation categories meet a varying number of criteria, from IG1, with some 50 criteria met, to IG3, with all criteria met. IG1 is defined as essential cyber hygiene. Just as with BSI, what this variant is good for remains a mystery. Money might be the prime motivation. Again, administrators will want to avoid delivering a system with a poorer configuration than they could easily achieve. The requirement is that the system meet the best possible configuration. However, this can mean something different for a machine controller in production than for an office PC in administration.

It is a good idea to aim for the best possible approach and not make concessions that mean less work for the time being but are likely never to be further hardened because no one is around to do the work.

DoD Security Technical Implementation Guide

The US Department of Defense (DoD) delivers a ZIP file with its Security Technical Implementation Guide (STIG), covering everything from the operating system to the browser to Office and even Adobe. In the field of operating system hardening, the browser is often overlooked. However, it – and email – offer the biggest attack vectors in the enterprise. Browsers and Office have been neglected thus far in this series, but they are worth discussing when you look at hardening.

The DoD provides just south of 25 templates that also cover various legacy operating systems. STIG conforms to the Microsoft baseline, with just one policy per object and no rating by levels or threat scenarios. No assessment tells you which value is more secure than any other. As in the previously mentioned templates, STIG does not have any massive collisions with the other templates; it is simply another assessment of the state of affairs, with a broader perspective.

Administrative templates especially have configuration options that do not just cover security issues but also prevent calls to the help desk when a user has problems – which can be a valid argument to hide or not provide this function in the first place. The DoD regularly works on the settings and publishes updates several times a year with no fixed cycle. The DoD delivers STIG as a GPO backup, each policy residing in its own folder in the ZIP file. Importing policies is a bit tedious because you have to select the `manifest.xml` file from each folder.

System Hardening by ACSC

The Australian Centers for Cyber Security (ACSC) works with *LowPrio*, *MediumPrio*, and *HighPrio* ratings. The website has recently been rebuilt, and you cannot currently download a ready-to-run GPO backup from the site. Unfortunately, at the moment, the website only provides configurations, plus a DOCX and PDF file; at least they give you the content for the three categories offline.

The ACSC did not keep to the order of the GPO editor, but defined the order itself, which makes the learning curve for familiarization with a GPO very tedious. (One hopes the GPO backup will be available online some time in the future.) The ACSC has historically hardened systems somewhat more rigorously than other template suppliers.

When using the ACSC guide, two things stand out in daily work.

When you call a user account control (UAC) dialog, you also need to press Ctrl + Alt + Del; the run keys of the registry are cleared by the GPO and must be controlled by it as well. Although the additional keyboard shortcut is just annoying, controlling programs in Autostart with GPOs causes a massive amount of work.

gp-pack PaT

gp-pack PaT has group policy backups and PowerShell scripts for privacy and telemetry (PaT) and is the only candidate in these template collections solely designed for silencing. Hardening has been covered by many others, which prompted these suppliers to ignore it. What the package aims to do is reduce the exchange of communication with Microsoft. All of the policy settings are available free

of charge as an HTML report online. The GPO backup for import, including various scripts to customize the client (e.g., removing apps and Windows features), is available for a fee. After integrating gp-pack, you might experience more problems in day-to-day operations compared with the previous templates. Because communication with Microsoft is prevented wherever possible, features that may be desirable are also disabled. For example, the Network Connectivity Status Indicator (NCSI) calls up a Microsoft web page and checks its availability every time a user logs on or sets up a network. If the website is accessible, the system network icon reports that it is connected to the Internet. If this information is missing, some products could malfunction. Outlook and Edge respond to the icon and don't even connect to the Internet – because it doesn't exist, according to the network display.
The big issue with silencing is that it involves functionality that is desirable but could collide with the GDPR. Security and silencing with gp-pack are also available for the three major browsers (Chrome, Edge, Firefox), Office, and Defender.

Policy Analyzer

Policy Analyzer [11] displays in a table of security settings from a policy's `gpttmpl.inf` file and administrative templates from the `registry.pol` file (Figure 2). Gray fields show values that are not included in the guideline, and conflicts show up in yellow. Policy Analyzer is a very good way of assessing that any template mentioned can be combined in any order because no critical contradictions exist. Whether the security event log is 192MB or 1GB is not critical for the time being. Depending on event forwarding, central logging, and monitoring, the size of the client log may or may not be relevant. Policy Analyzer also shows you how uniformly various manufacturers act in their evaluation of security-relevant settings, and it becomes clear that you need to combine them for the best possible approach, because each manufacturer defines slightly different specifications.

Templates Combined

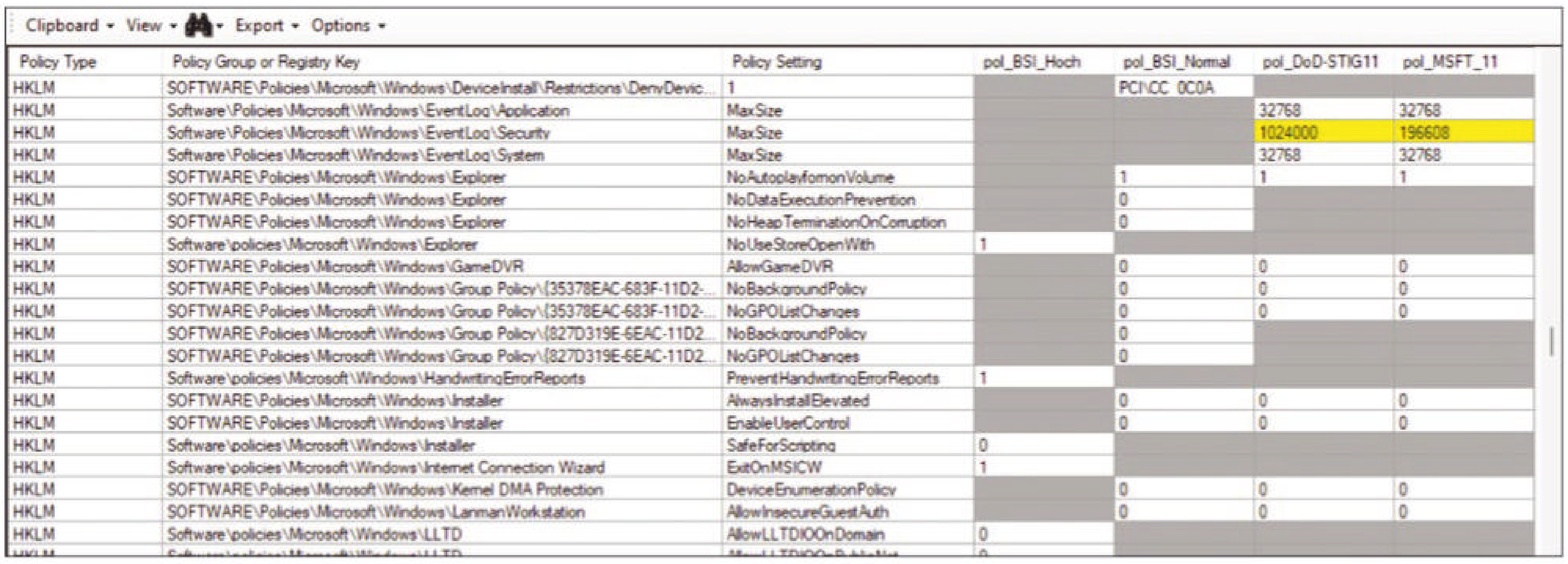
Microsoft Security Baseline is a must on every network; it forms the basis for all further configurations. You might want to add BSI SiSyPHuS next to take German requirements into account. However, because of its age, you will definitely want to add another template. The structured template and good documentation of CIS is recommended, even if it means some manual work.
The DoD STIG and ACSC configurations are maybe just nice to have, and gp-pack PaT is somewhat isolated in this list because it focuses on a different aspect; however, in addition to hardening, data reduction should be a consideration in your operating system architecture and security strategy.

Handling ADMX Files

When a new variant of Windows Server arrives, updating the ADMX files is one of the first items on your to-do list. The use of a Central Store [12] for the ADMX files has become established: Copy the complete folder `C:\Windows\Policydefinitions` (or its contents) to `\<name of domain>\SYSVOL\Policies\Policydefinitions`.
The Group Policy Management Console (GPMC) is designed to search on this path and use the ADMX files stored there to display the administrative templates in the editor. The path is hard coded and cannot be changed. The only exception is a computer on which the administrator manually sets a registry entry to tell the GPMC on that system to use the local store in `C:\Windows\Policydefinitions`, despite the existence of a central store:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\↩
  Microsoft\Windows\Group Policy
EnableLocalStoreOverride=1 (Reg_Dword)
```

Whether a central store or a local store, at some point you will face the challenge of extending or renewing the ADMX files. One thing I need to point out is that you only replace or extend the templates for configuring settings; don't make changes to existing policies at any point – nothing dire can happen, nothing will ever break. The complete `\Policydefinitions` folder can be deleted and rebuilt; nothing is lost. Any file



Policy Type	Policy Group or Registry Key	Policy Setting	pol_BSI_Hoch	pol_BSI_Normal	pol_DoD-STIG11	pol_MSFT_11
HKLM	SOFTWARE\Policies\Microsoft\Windows\DeviceInstall\Restrictions\DenyDevic...	1		PCI\CC_0C0A		
HKLM	Software\Policies\Microsoft\Windows\EventLog\Application	MaxSize			32768	32768
HKLM	Software\Policies\Microsoft\Windows\EventLog\Security	MaxSize			1024000	196608
HKLM	Software\Policies\Microsoft\Windows\EventLog\System	MaxSize			32768	32768
HKLM	SOFTWARE\Policies\Microsoft\Windows\Explorer	NoAutoplayfornonVolume		1	1	1
HKLM	SOFTWARE\Policies\Microsoft\Windows\Explorer	NoDataExecutionPrevention		0		
HKLM	SOFTWARE\Policies\Microsoft\Windows\Explorer	NoHeapTerminationOnCorruption		0		
HKLM	Software\policies\Microsoft\Windows\Explorer	NoUseStoreOpenWith	1			
HKLM	SOFTWARE\Policies\Microsoft\Windows\GameDVR	AllowGameDVR		0	0	0
HKLM	SOFTWARE\Policies\Microsoft\Windows\Group Policy\{35378EAC-683F-11D2-...	NoBackgroundPolicy		0	0	0
HKLM	SOFTWARE\Policies\Microsoft\Windows\Group Policy\{35378EAC-683F-11D2-...	NoGPOListChanges		0	0	0
HKLM	SOFTWARE\Policies\Microsoft\Windows\Group Policy\{827D319E-6EAC-11D2...	NoBackgroundPolicy		0		
HKLM	SOFTWARE\Policies\Microsoft\Windows\Group Policy\{827D319E-6EAC-11D2...	NoGPOListChanges		0		
HKLM	Software\policies\Microsoft\Windows\HandwritingErrorReports	PreventHandwritingErrorReports	1			
HKLM	SOFTWARE\Policies\Microsoft\Windows\Installer	AlwaysInstallElevated		0	0	0
HKLM	SOFTWARE\Policies\Microsoft\Windows\Installer	EnableUserControl		0	0	0
HKLM	Software\policies\Microsoft\Windows\Installer	SafeForScripting	0			
HKLM	Software\policies\Microsoft\Windows\Internet Connection Wizard	ExitOnMSICW	1			
HKLM	SOFTWARE\Policies\Microsoft\Windows\Kernel DMA Protection	DeviceEnumerationPolicy		0	0	0
HKLM	SOFTWARE\Policies\Microsoft\Windows\LanmanWorkstation	AllowInsecureGuestAuth		0	0	0
HKLM	Software\policies\Microsoft\Windows\LLTD	AllowLLTDIOOnDomain	0			

Figure 2: Policy Analyzer provides a GPO overview as an Excel sheet.

Additional Registry Settings

You have probably already come across the Additional Registry Keys section [13] in the HTML report of the guidelines. The GPMC HTML report resolves the values backward; that is, the `registry.pol` file in the policy runs through a parser or discovery process: For each existing entry, the GPMC tries to find a display text in the system's `\Policydefinitions` folder. If it finds a match, the registry value appears in a named category along with the display name and an explanation from the ADMX file. If the ADMX entry is missing, the report can only document the existence of the value in the file as a registry key and value.

contained in it can be copied from a running computer or downloaded off the web. ADMX files are swappable objects that create a file in the policy to store the settings that import the object. How this value makes its way into the configuration file (`registry.pol`) does not matter to the importing object. (See also the "Additional Registry Settings" box.)

In addition to being able to write entries to the `registry.pol` file via ADMX, Microsoft provides PowerShell cmdlets that act through the GPMC interface and can modify, delete, or add registry values. It is important to remember this technical possibility, because this is your lifeline if no suitable ADMX file is available. Two useful PowerShell commands are `Set-GPRegistryValue` and `Remove-GPRegistryValue`; the syntax is almost self-explanatory. For example, to set or add a value and delete a value, use:

```
Set-GPRegistryValue -name <NameOfGPO> ➘
  -key HKLM\Software\<ThirdPartySoftware> ➘
  -valuename <DBServer> ➘
  -value <SRV-SQLDB-01> -type <STRING>
Remove-GPRegistryValue -name <NameOfGPO> ➘
  -key <HKLM\Software\ThirdPartySoftware> ➘
  -valuename <DBServer>
```

All in all, the structure of a `\Policydefinitions` folder only focuses on the configuration options and how they are displayed or documented. It can be updated and exchanged without any problems – or could be, if it weren't for Microsoft's development chaos.

Long-Standing Encumbrances

The development strands of the ADMX files are, to put it kindly, less than perfect. Before Windows 10, so-called "Super Sets" stipulated that all previous settings are always included in the latest version. It didn't matter whether the policies came from a server or from a client service pack, all the legacy values were always included in every new version. This status was frozen in Windows 10, so there will be no further changes or adjustments of the legacy settings (2000/XP/2003/Vista/7/8/8.1). The current Windows 10 policies still contain settings that were only applied under Windows 2000 when they were distributed to a client or user – a cleanup process never took place.

In Windows 10, the idea arose that policies would be removed from the ADMX settings if they belonged to an unsupported operating system and would no longer take effect in the new build. A review was conducted every six months (this is now annual) to determine what to keep and what to remove.

To its own chagrin, Microsoft itself has broken the rule that an operating system only has a half-life of three builds. Enterprise editions starting with 1809 have a lifetime of five builds in the fall releases. The exception is 1803, which was supported for five years despite being a spring release. On top of this is LTSC (formerly LTSB) with a 10-year lifetime. Now specific 1607, 1809, and 21H2 guidelines need to be kept "forever" because of an LTSB/C edition.

Windows 11 now joins this chaos as a completely new system – at least in terms of nomenclature and the ADMX development thread. Support for Windows 10 ends in October 2025, and Windows 11 has existed in parallel since fall 2021. Different ADMX files are available for download for the two operating system versions, with some settings from version 10 missing from version 11, and vice versa.

Administrators now have to deal with two ADMX variants and may wonder what to do with Windows Server 2022 and its ADMX files. You will find no easy solution for this messy system.

The Problem with the PolicyDefinitions Store

You can only establish a versioned `PolicyDefinitions` store if you set up complex scripts that include permanent renaming of existing folders for each version. Incidentally, editing a script appropriate for the operating system so that it chooses the "correct" `PolicyDefinitions` folder is not recommended. This action leads to more chaos and errors than to opportunities for the script to improve the situation.

In this confusion, it is difficult for an administrator to find their way through the jungle. Best practices now dictate simply using the latest templates at all times, mixing server and client ADM files in a huge merge, and replacing existing ADMX files with newer ones. For clients, you need to agree on a version. The recommendation is to use the version (Windows 10 or 11) used by the majority of clients in your organization. Remember that nothing dire can happen; there are no wrong ADMX files. However, you could have a better fit for the current use case. If this happens, make sure you use PowerShell before rebuilding the entire `PolicyDefinitions` store just to change a registry value or make the HTML report look nicer.

Overrated Client

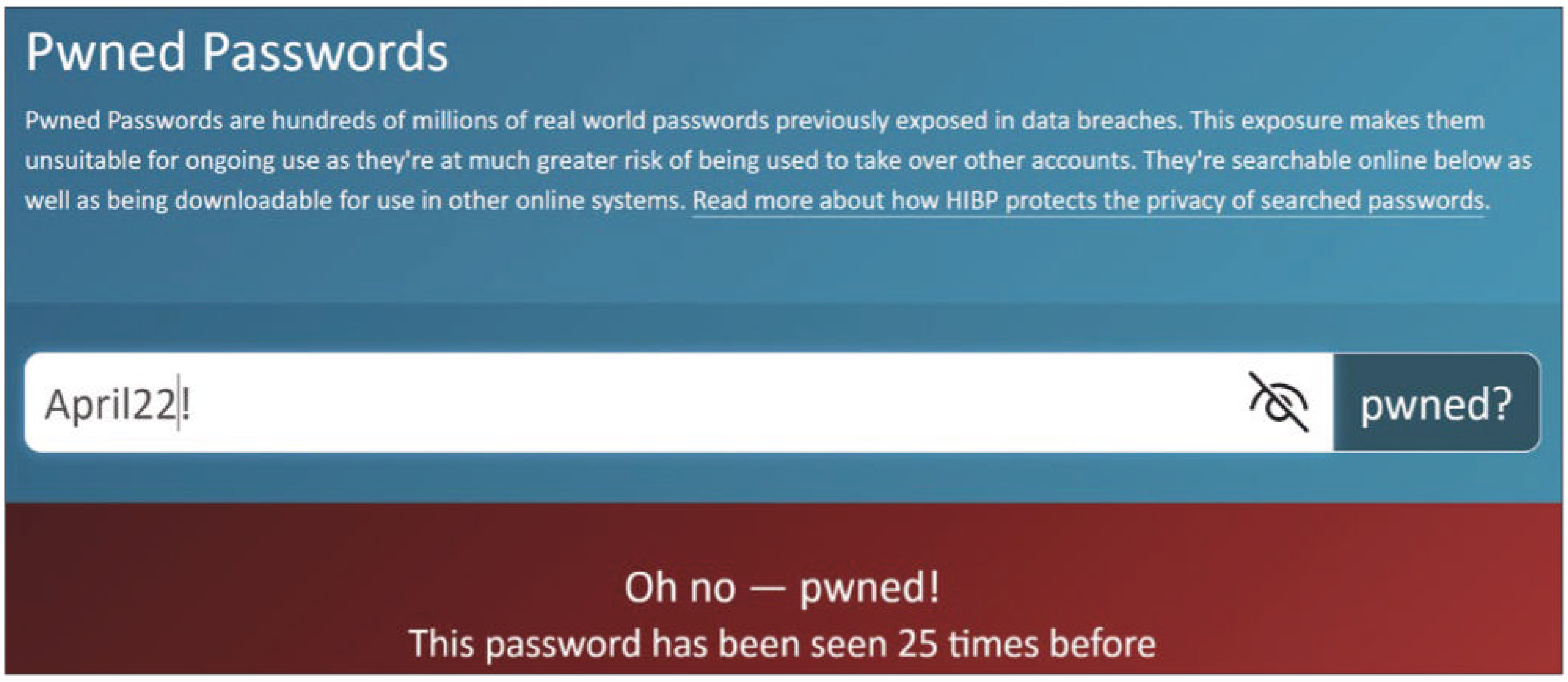
Many administrators still see the issue of group policies strongly anchored to the client, and it is the first system they control and monitor with group policies. The terminal device is the weakest link in the chain and is always assumed to be the first to break when an attack occurs. The question you need to answer, though, is: What is the method that hackers are most likely to use to attack a system these days? The route into the enterprise currently tends to rely on email, web portals, or the cloud, because they provide publicly available resources that are often secured by a far-too-weak and simplistic username-password construct. However, by the time the malware reaches the client, other systems have long since failed: Users use a password that appeared in a data leak. The anti-spam filter on the mail server forwarded the email. The proxy server has not prohibited access to the spoofed "company website" where the user data is stored. These and similar cases are the rule today – the theory of the client as the weakest link in the chain is no longer tenable. The client is more likely to be the link in the chain through which distribution occurs at the end, but it is other systems that failed first. Sadly, the focus here is on the user.

Outdated Password Policy

Now is the time to talk about the skeletons in everyone's closet that every administrator has had on their mind for years, but – for whatever reason – failed to implement. To begin: password policy.
The whole world is talking about two-factor authentication (2FA) and passwordless authentication and is horrified to find that the enterprise still allows the use of six- or eight-digit passwords without complexity rules. The 2003 rulebook from the National Institute of Science and Technology (NIST) was a good idea but has proven to be a mistake over time. Permanent changes to passwords have forced users to add a counter or append the month as a number. All security experts (and even the GPO templates) now agree on the length of a good password. It should be at least 14 characters – more is better.
The BSI has responded to this realization and now says that a password change is no longer necessary if it is a long, complex password that has not yet been compromised. This development is interesting because it means that admins need to check the password quality of their user accounts regularly.
A very good tool for this purpose is provided by DSInternals [14]. Its PowerShell cmdlets let you integrate to an offline "Have I Been Pawnd" database (Figure 3), which means storing a text file of about 30GB on the system that contains most of the currently known password hashes that have already been compromised by attackers. The DSInternals cmdlets now check whether the password hashes appear in the database in your own AD.
If you combine DSInternals cmdlets with the *PwnedPass-Check* PowerShell module [15], you can efficiently determine how often a password has already appeared in various databases. Armed with this knowledge, you can evaluate how urgent the need is to change the password. A password that occurs only 25 times in a database is certainly not one of the first to be used in an attack, unlike a password that appears 46,000 times (e.g., *secret*).

Fewer Rights for Admins

Reducing administrator authorizations is one of the easiest tasks to implement with group policy. The organizational aspects are the problem, starting with the local administrators of a client or member server and ending with the number of domain admins in the enterprise. Only in the rarest of cases will you have a clear overview of who has to manage which systems and who needs administrative authorizations for the system.
If you look at it, everyone knows this is one of the biggest problems in any organization and that some people absolutely need to be removed from specific groups. Unfortunately, especially in small to mid-size enterprises (SMEs), the potential for political pressure is high if a veteran colleague insists on keeping their authorizations.


Pwned Passwords

Pwned Passwords are hundreds of millions of real world passwords previously exposed in data breaches. This exposure makes them unsuitable for ongoing use as they're at much greater risk of being used to take over other accounts. They're searchable online below as well as being downloadable for use in other online systems. Read more about how HIBP protects the privacy of searched passwords.

April22!

Oh no — pwned!
This password has been seen 25 times before

Figure 3: Websites deliver statistics on how often a password has been compromised by attackers; these are important facts to drive password changes in the enterprise.

Many people feel that something is being taken away from them, when in fact it is no more than self-protection.

Defender Firewall vs. GPOs

Microsoft's Defender firewall is still disabled in many companies. Although various group policies and processes are used to prevent an attack propagating on the network, if credentials such as a local admin account are compromised, an attacker can log in to any machine in the organization because the password is often identical on every machine.

With Defender, you can block access from the network to a computer when assigning user authorizations for any local account. When WannaCry attacked the client via the SMBv1 protocol, Microsoft delivered patches, whereas the far easier way would have been to disable SMBv1, if there were no dependencies on it. The PetitPotam attack became active again in April 2022 with a new New Technology LAN Manager (NTLM) attack vector, and PrintNightmare attacked systems by way of the print queue, which is required by every machine that needs to print.

The administrator is quite helpless. With the multitude of attack options, they need a solution for every single attack. It would be far easier to prevent incoming communication from a client. If the door is closed (i.e., the firewall is enabled), the client terminates the contact as soon as it registers a communication attempt. The network needs segmentation, allowing for machines that can access the system for administrative tasks; this alone could help avoid mass risk and potential wildfire spread. The technology has existed since Windows XP SP2 – now it's time to use it.

Allow Only Known Applications

Software allowlisting is an administrative monster waking up from a deep sleep. An allowlist of applications would virtually eliminate the problem of ransomware. Ransomware launches an executable when the attack occurs. The simple logic is that if only known and documented software is allowed in the company, and all other software is forbidden, the ransomware cannot do its job. Although it is a trivial approach, it works worlds better than trying to detect and prevent ransomware from launching with behavior detection.

In my security proposals, AppLocker and Software Restriction Policies are always included as mandatory components. However, this solution increases workload and costs time, and the system needs maintenance once it is set up. As with the firewall, the technology comes from the age of Windows XP SP2.

The latter four security ideas mentioned here can be wonderfully implemented locally with group policies, but they share issues with time, maintenance, and overhead. Microsoft's desired solution is to move to the cloud. In Microsoft 365, you can force the user to use 2FA, with no ancient protocols that an attacker can exploit. From Microsoft's point of view, the system is secure if local resources are no longer important. If the crown jewels of the company are in the cloud, then admins can encrypt all data locally. The response to a security incident would then simply be to reinstall computers and continue working. The time and downtime are still annoying and cost money, but they are inexpensive compared with a genuine data loss by encryption for ransom.

Conclusions

Even if Windows Server 2022 does not deliver that much new functionality in terms of GPO, it is still important to ensure maximum security for clients. Numerous GPO templates help with this job. Server 2022 does not establish any new rules. It's just that the requirements for your own network now need to adapt to the 2022 server version. ■

Info

[1] Windows Server 2022 ADMX and GPO settings: [https://techcommunity.microsoft.com/t5/windows-server-for-it-pro/windows-server-2022-admx-and-gp-settings-now-available/m-p/3254928]

[2] ADMX Templates forWindows Server 2022: [https://www.microsoft.com/en-us/download/details.aspx?id=104003]

[3] Excel spreadsheet with GPOs: [https://www.microsoft.com/en-us/download/details.aspx?id=104005]

[4] Security Compliance Toolkit: [https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-security-configuration-framework/security-compliance-toolkit-10]

[5] BSI SiSyPHuS: [https://www.bsi.bund.de/EN/Service-Navi/Publikationen/Studien/SiSyPHuS_Win10/SiSyPHuS.html?nn=1022786]

[6] Center for Internet Security: [https://www.cisecurity.org/cis-benchmarks/]

[7] STIGs by the DoD: [https://public.cyber.mil/stigs/]

[8] ACSC guide to system hardening: [https://www.cyber.gov.au/acsc/view-all-content/advice/guidelines-system-hardening]

[9] gp-pack PaT: [https://www.gp-pack.com/gp-pack-pat-privacy-and-telemetry/] (in German)

[10] LGPO utility: [https://techcommunity.microsoft.com/t5/microsoft-security-baselines/lgpo-exe-local-group-policy-object-utility-v1-0/ba-p/701045]

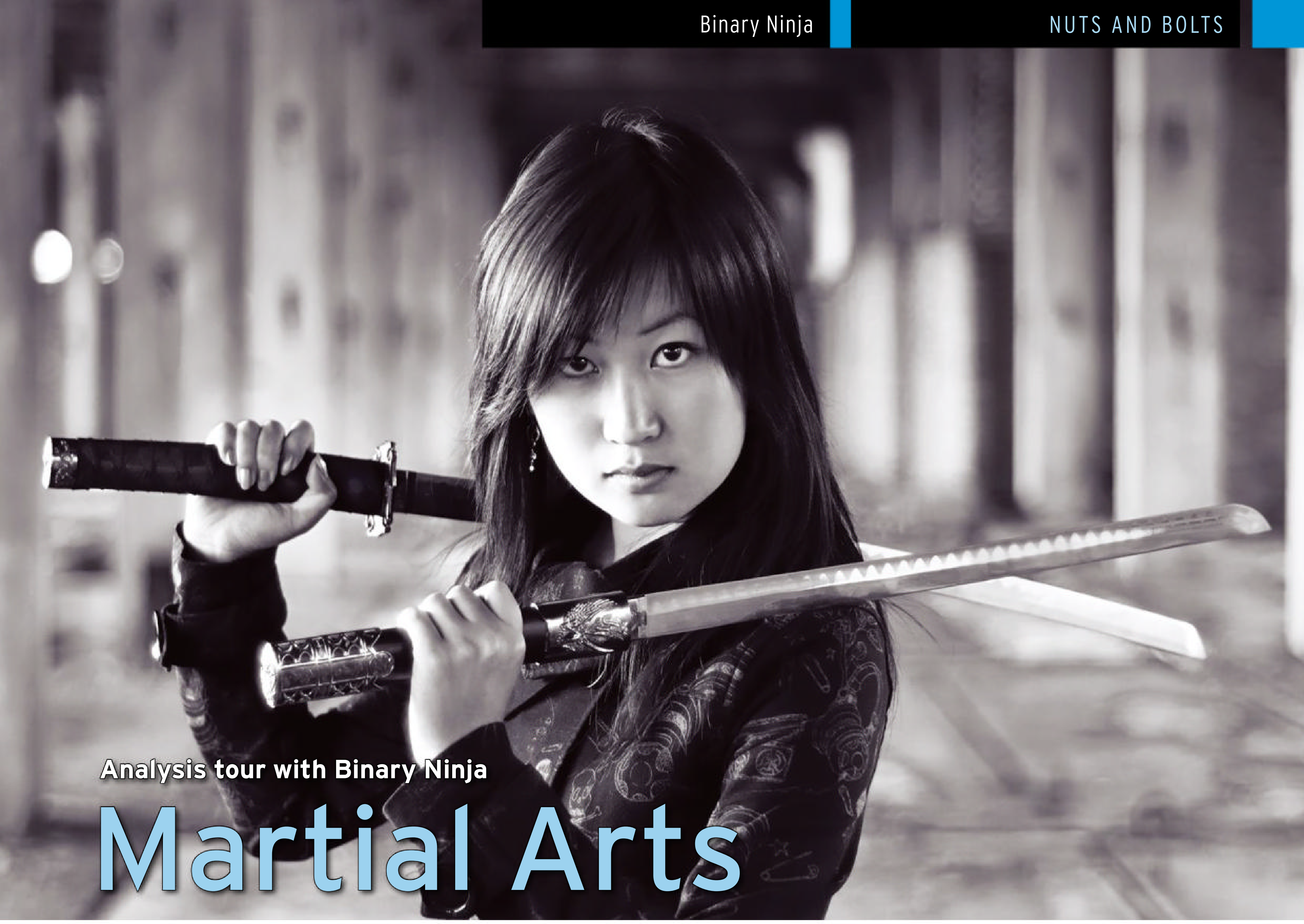
[11] Policy Analyzer: [https://techcommunity.microsoft.com/t5/microsoft-security-baselines/new-tool-policy-analyzer/ba-p/701049]

[12] Central Store for Administrative Templates: [https://learn.microsoft.com/en-us/troubleshoot/windows-client/group-policy/create-and-manage-central-store]

[13] Editing additional registry settings: [https://sdmsoftware.com/tips-tricks/removing-extra-registry-settings-from-gpos/]

[14] DSInternals: [https://www.dsinternals.com/en/]

[15] PwnedPassCheck: [https://www.dvolve.net/blog/2019/08/new-module-pwnedpasscheck/]



Analysis tour with Binary Ninja

Martial Arts

Binary analysis is an advanced technique used to work through cyberattacks and malware infestations and is also known as reverse engineering. We show you how to statically analyze binary programs with Binary Ninja, an interactive binary analysis platform. By Matthias Wübbeling

If you want to know exactly what operations a program on your computer performs, you have several ways to find out. One obviously simple method is to investigate the source code of the file, if available. You can do so easily with scripting languages like Python or open source software, but if programs are translated into bytecode before delivery, your job is a little more complicated.

Various tools are involved in converting code into an executable program: in most cases, at least a compiler and a linker. From the commands in a programming language, the compiler creates the machine language. In doing so, it optimizes the execution sequence or individual operations, depending on the configuration, to a greater or lesser extent, which can ultimately have a major effect on the resulting machine code. During linking, the libraries are statically or dynamically linked to the program. Static linking adds the code from the libraries to the resulting program. If libraries are linked dynamically, the code is located in external files and is only added to the process's working memory when the program is started.

Binary Ninja [1] is a tool for static program analysis. Originally designed for use in capture-the-flag competitions, Binary Ninja is now being developed commercially. For initial insights into the functionality, as covered by this article, it's fine to use the free trial version; download and install the version for your operating system to try it out.

Create a Test Program

To get started with a program that is as simple as possible for analysis, it's a good idea to write your own. Name the following code `admin.c`:

```
#include <stdio.h>
int main(int argc, char **argv)
{
printf("Hello World!");
return 0;
}
```

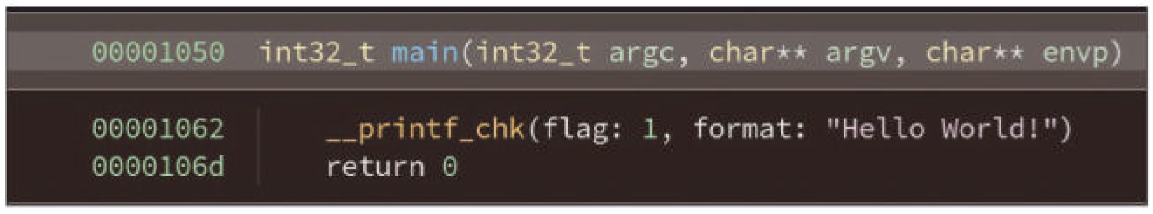
You can compile the source code with:

```
gcc -O0 admin.c -o admin
```

The `-O0` entry switches off the compiler optimizations to keep the machine code closer to the source code, which is especially useful for debugging during development.

Analyzing Applications

To analyze the program you created, launch Binary Ninja and open the `admin` file. The analysis interface shows an overview of the symbols included in the binary, a code view,



```
00001050  int32_t main(int32_t argc, char** argv, char** envp)

00001062      __printf_chk(flag: 1, format: "Hello World!")
0000106d      return 0
```

Figure 1: Underlying `printf` function.

and the structure of the program in a feature map, among other options. Binary Ninja offers different levels of the generated program code, which can significantly facilitate the analysis. The default *High Level IL* output shows generated code that already looks very similar to the original C code. IL stands for "intermediate language"; Binary Ninja offers different language levels up to C code. In this example, the `main` function will look very similar to the C code.

The call to `printf` is replaced by the underlying function `__printf_chk` (Figure 1), which is called at the code level and checks the format string for possible stack overflows before outputting. If you want to get closer to the machine code, click through the different language layers and look at the different derivations.

Note that the displayed code is generated from the machine code. It's more of an approximation of programming in a high-level language like C or C++ than an actual representation of the underlying code. This is also evident in the generic, and mostly not very meaningful, variable names, which are oriented on the CPU registers or memory locations. You cannot compile this code again without further editing.

Now select *Disassembly* in the top bar, and you will see the program in assembler code. The `main` function then contains the commands. In addition to operations on the stack pointer `rsp`, you can see how `lea` is used to load the address of the *Hello World!* string into the `rsi` register (Figure 2) – after all, the format string resides in the Data area of the binary. Next is initializing the registers `edi` with `1` and `eax` with `0`. Instead of `xor eax, eax`, you could also use `mov eax, 0x0`. In fact, this is already a compiler optimization, because the opcode is two bytes shorter, which prepares the arguments for `__printf_chk`, whereas `call` calls the function. Afterward, a `0` return value is stored in `eax`, the stack pointer is reset, and the function is exited.

You are currently in executable and linkable format (ELF) mode in the code display. ELF is the Linux binary format that is evaluated directly by Binary Ninja and displayed accordingly. If you open the menu where *ELF* is currently selected and switch to *RAW* mode, you can view the ELF headers themselves, which is where you will find, for example, the pointers to dynamic libraries or the string and symbol tables.

For example, to display the sequence graph instead of the linear representation of the code, select the *Graph* representation in the menu to the right – currently still with the *Linear* display. In this small example, only the `main` function is initially displayed. For example, if you select `deregister_ tm_clones` from the icons on the left to manage memory transactions, the graph becomes a little larger. If you open a real program with Binary Ninja, you can better understand the structure and relationships of the processes with the flow graph.

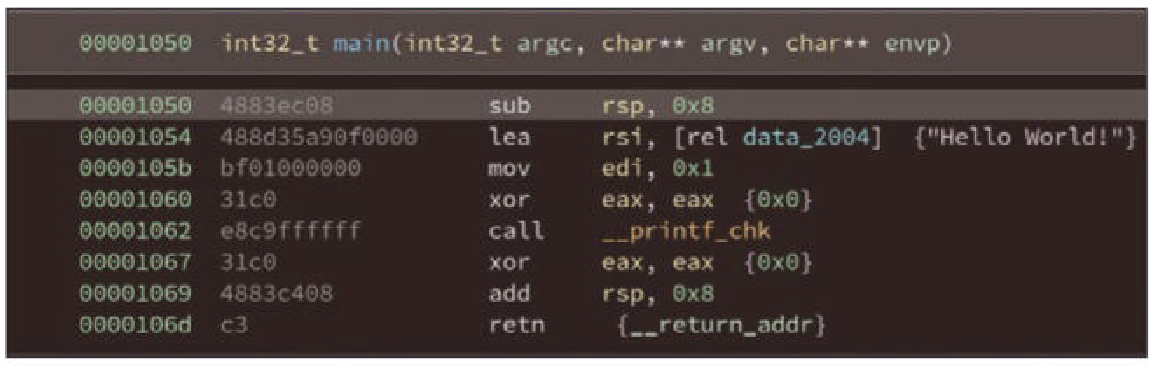
If you select *Hex* as the display format instead, the program file will be displayed in a hex editor. Besides the address on the left and the hexadecimal representation in the middle, you can see the printable characters on the right. For example, here you can find the hex values of the string *Hello World!*. However, changes to the program are not possible in this mode.

Besides ELF programs, Binary Ninja supports many other executable file formats. You can analyze the portable executable (PE) binaries common on Windows systems, as well as the Mach-O binaries used by Apple's operating system. You are not limited to x86 or x64 platforms and can disassemble programs compiled for architectures such as ARM, MIPS, or PowerPC. Because Binary Ninja is not a debugger or decompiler, but disassembles binary data in line with the assembler for the respective platform, interpreted languages or those with an intermediate representation of the code, such as with the Java Virtual Machine (VM) or .NET, cannot be meaningfully analyzed.

Extensions and Legal

If you have a valid license, you can use the Binary Ninja Python API. With its help, you can automate operations that you perform regularly or control the program (e.g., change settings or displays, move around the generated code, or launch plugins). If you are missing a function, Binary Ninja lets you add plugins. Of course, you can also develop these yourself. Interfaces to C/C++, Rust, and Python are available for this purpose. The Plugin Manager also lets you install extensions written by other users.

To analyze malware, experts regularly use tools such as Binary Ninja or Ghidra [2], developed by the US National Security Agency (NSA), to convert binary code into other representations. Copyright law sets narrow limits for this activity. For example, you are only allowed to



```
00001050  int32_t main(int32_t argc, char** argv, char** envp)

00001050  4883ec08        sub     rsp, 0x8
00001054  488d35a90f0000  lea     rsi, [rel data_2004]  {"Hello World!"}
0000105b  bf01000000      mov     edi, 0x1
00001060  31c0            xor     eax, eax  {0x0}
00001062  e8c9ffffff      call    __printf_chk
00001067  31c0            xor     eax, eax  {0x0}
00001069  4883c408        add     rsp, 0x8
0000106d  c3              retn     {__return_addr}
```

Figure 2: Hello World! program assember code.

convert the code if you hold the rights to it yourself or if it is necessary to ensure the interoperability of an existing program. Analyzing malware to protect your own infrastructure is presumably not critical here, but whether parts of the code may be used in the context of public reporting is something that needs to be examined on a case-by-case basis. Of course, it seems very unlikely that the malware developers, who are themselves criminals, will enforce their rights in a court of law in this case.

Conclusions

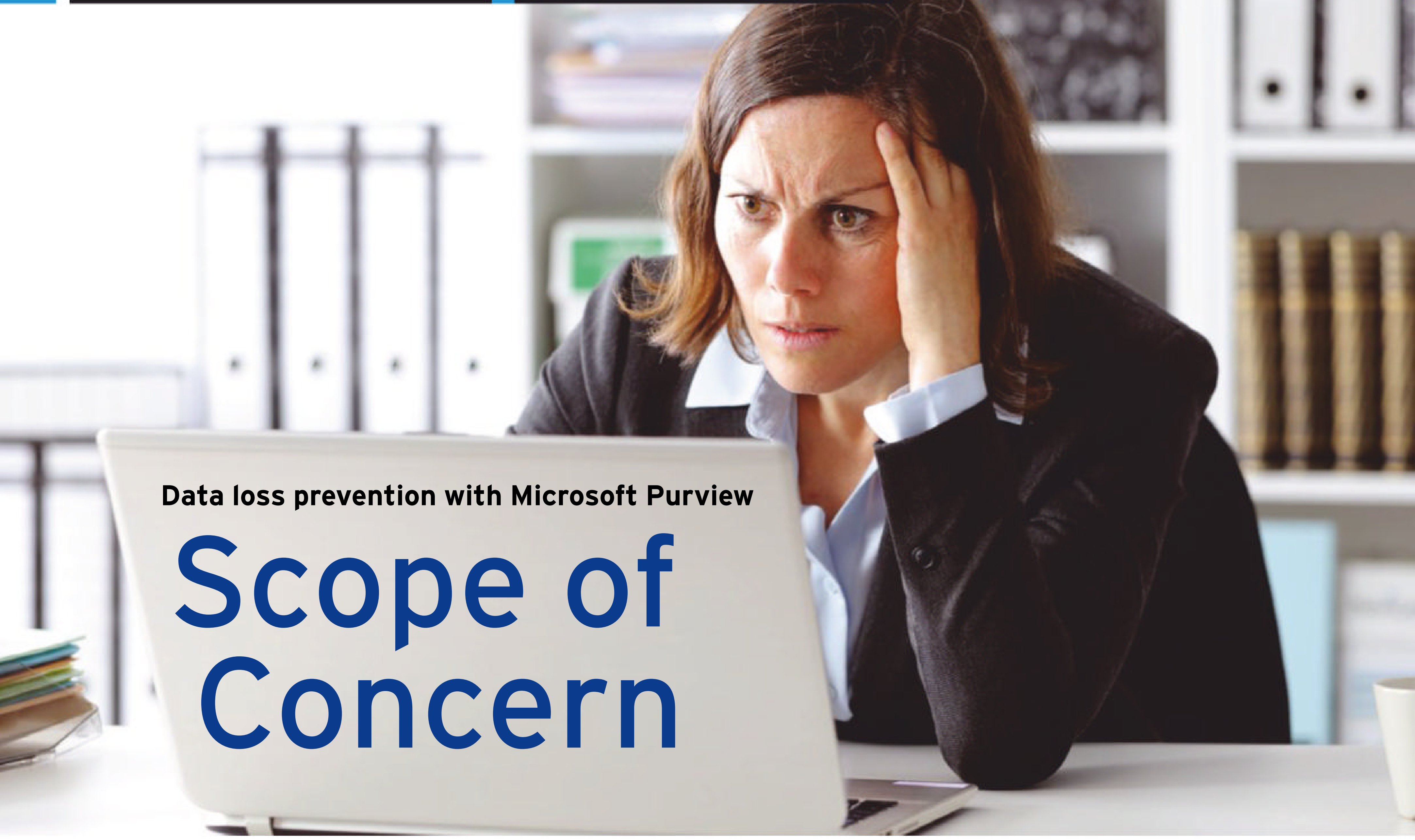
Binary Ninja is a neat tool for analyzing binary data and programs. With a little practice, you will easily find your way around, although analyzing third-party code can be a little tricky at times. If you just want to take a quick look at a binary, Binary Ninja is great choice with comprehensive display options. ■

Info

[1] Binary Ninja: [https://binary.ninja]
[2] Ghidra: [https://ghidra-sre.org]

The Author

Dr. Matthias Wübbeling is an IT security enthusiast, scientist, author, consultant, and speaker. As a Lecturer at the University of Bonn in Germany and Researcher at Fraunhofer FKIE, he works on projects in network security, IT security awareness, and protection against account takeover and identity theft. He is the CEO of the university spin-off Identeco, which keeps a leaked identity database to protect employee and customer accounts against identity fraud. As a practitioner, he supports the German Informatics Society (GI), administrating computer systems and service back ends. He has published more than 100 articles on IT security and administration.



Data loss prevention with Microsoft Purview

Scope of Concern

Microsoft Purview combines compliance and data governance to address the security of confidential data in the new hybrid working world. By Christian Schulenburg

Companies spend huge amounts of money on preventing threats and ensuring data confidentiality caused by home office and distributed working practices. Microsoft has also identified this market and, in addition to Endpoint Protection in Microsoft 365, offers the comprehensive Purview Compliance Manager designed to prevent the outflow of confidential data through data loss prevention.

Microsoft Purview

The Microsoft Compliance Center has been around for a while, but it was renamed Microsoft Purview in April 2022. With Purview, Microsoft combines compliance with data governance, primarily to accommodate the new hybrid working world, where connectivity is required all the time, wherever people happen to be. Documents are exchanged flexibly and are no longer simply transmitted by email, but by instant messaging tools such as Teams or OneDrive. The result is data fragmentation and data spread across applications and devices. Purview offers a number of approaches to protect and manage data by combining the functions of Azure Purview and Microsoft 365 Compliance.

Purview's scope is broad and includes information protection and governance, data protection, insider risk management, and investigation and response. Information protection and governance includes the dataset management, information protection, data life-cycle management, and data loss prevention tasks. In this article, I take a closer look at data loss prevention. For an overview of the various services in Microsoft Purview, see the Microsoft documentation [1].

Confidential Data at a Glance

The data loss prevention (DLP) module is designed to prevent the leakage of confidential data. In most cases, Exchange administrators are already familiar with DLP. This consolidation in Microsoft Purview also means that DLP has migrated from Exchange Online to the new Compliance Manager and can no longer be found in the current Exchange Administration Center. In the old view, the link is still there, but you will be redirected to Microsoft Purview.

In addition to Exchange, the DLP function also monitors other services. For Microsoft 365, these include Teams, SharePoint, and OneDrive. Also on the table are the Office applications such as Word, Excel, and PowerPoint. In addition to monitoring Windows 10 and Windows 11 endpoints, macOS is covered starting with Catalina 10.15, as well as local shares and local SharePoint instances. To identify the data, extensive content analysis takes a closer look – and not just a simple text review; the process includes machine learning algorithms.

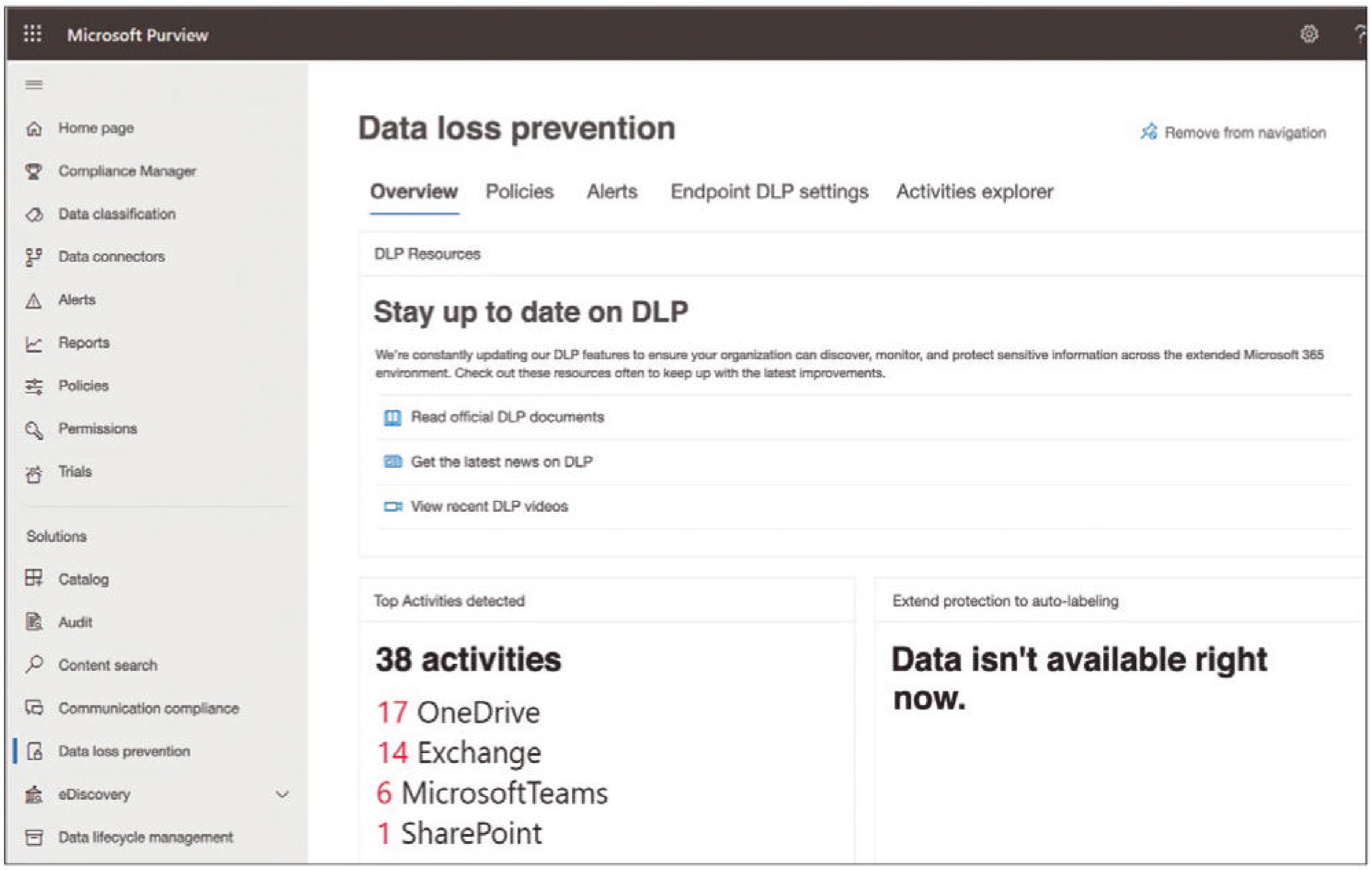
Predefined DLP policy templates for the different areas that work with confidential data take into account data protection legislation, personal data, or financial data in various countries. More than 250 predefined confidential data types are available and can be broken down to 38 countries. Custom data types can also be created in Microsoft Purview; the procedure is described in the documentation. You can find an excerpt of the templates provided by Microsoft in

Table 1 and a detailed overview of the DLP policy templates online [2].

The DLP Cockpit at a Glance

DLP configuration takes place in the browser in Microsoft Purview. You can access the page from the Microsoft 365 Admin Center or directly by the compliance URL [3]. In Purview, a menu for the various services is on the left and is also where you will find the *Data loss prevention* section (Figure 1). Go there to complete the configuration. The DLP toolbar at the top comprises the entries:

- *Overview*: information and resources for DLP
- *Policies*: existing policies are edited or deleted and new policies are created
- *Alerts*: notifications originating from DLP actions and detailed information about an event
- *Endpoint DLP settings*: detailed settings for monitoring content on Windows and Mac devices



Data loss prevention

Stay up to date on DLP

We're constantly updating our DLP features to ensure your organization can discover, monitor, and protect sensitive information across the extended Microsoft 365 environment. Check out these resources often to keep up with the latest improvements.

38 activities
17 OneDrive
14 Exchange
6 MicrosoftTeams
1 SharePoint

Data isn't available right now.

Figure 1: DLP policy configuration in Microsoft Purview.

Table 1: Purview DLP Templates

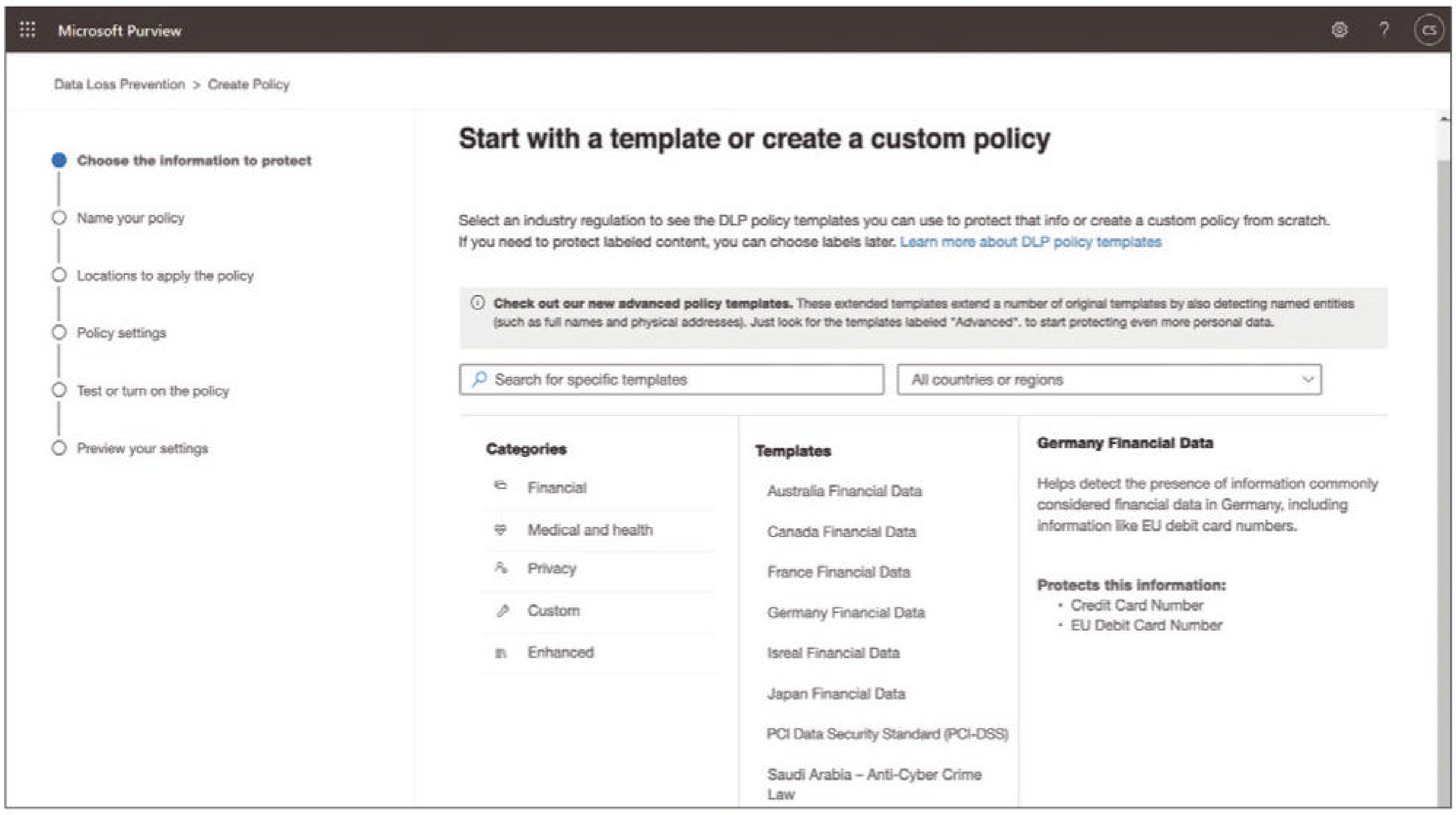
Templates Integrated in Purview	Description
France: Data Protection Act	Information that is usually covered by data protection law in France, such as the French Carte nationale d'identité (CNI) and the French social security number (INSEE).
Germany: personally identifiable information (PII data)	Data that is usually considered personal information in Germany, such as driver's license and passport numbers.
Israel: PII data	Information that is normally considered personal information in Israel, such as ID card information.
Saudi Arabia: PII data	Personal data in Saudi Arabia, such as information on national IDs.
UK: Access to Medical Reports Act	Information covered by the Access to Medical Reports Act in the United Kingdom, such as the UK National Health Service number and the UK National Insurance Number (NINO).
UK: Data Protection Act	Data covered by the Data Protection Act in the UK, such as the UK passport number.
UK: Online Code of Conduct for personal information	Information covered by the Personal Information Online Code of Practice in the United Kingdom, such as the UK NINO and the UK National Health Service number.
US: Health Insurance Portability and Accountability Act (HIPAA)	Information that falls under HIPAA in the US, such as the US Social Security Number (SSN) and Drug Enforcement Agency (DEA) number information.
US: Patriot Act	Data that falls under the Patriot Act in the US, such as credit card number, bank account number, US tax identification number (ITIN), and US SSN.
US: state social security number confidentiality laws	Information covered by state social security number confidentiality laws, such as the US SSN.

- *Activities explorer*: a history of the activities related to the data

The Policies window shows the existing policies. The name, sequence, date of change, and status are displayed first. The order is important, because a further check of subsequent rules can be disabled as soon as a rule has taken effect. In the case of the status, it is important to check whether a rule is enabled. Your options for a policy after creating it are *Turn it on right away*, *Keep it off*, or *Test it out first*, and this can be done with or without policy tips. Testing a policy extensively before going live is useful to avoid having to activate a policy directly during the initial implementation phase.

Setting Up Policies

If you want to create a new policy, press *Create policy* (Figure 2). First, select the data you want to protect. You can create policies based on an existing template set or choose to create a *Custom* entry, for which you will then need to define your own rules during the configuration. The


Microsoft Purview

Data Loss Prevention > Create Policy

- Choose the information to protect
- Name your policy
- Locations to apply the policy
- Policy settings
- Test or turn on the policy
- Preview your settings

Start with a template or create a custom policy

Select an industry regulation to see the DLP policy templates you can use to protect that info or create a custom policy from scratch.
If you need to protect labeled content, you can choose labels later. Learn more about DLP policy templates

Check out our new advanced policy templates. These extended templates extend a number of original templates by also detecting named entities (such as full names and physical addresses). Just look for the templates labeled "Advanced". to start protecting even more personal data.

Search for specific templates | All countries or regions

Categories
- Financial
- Medical and health
- Privacy
- Custom
- Enhanced

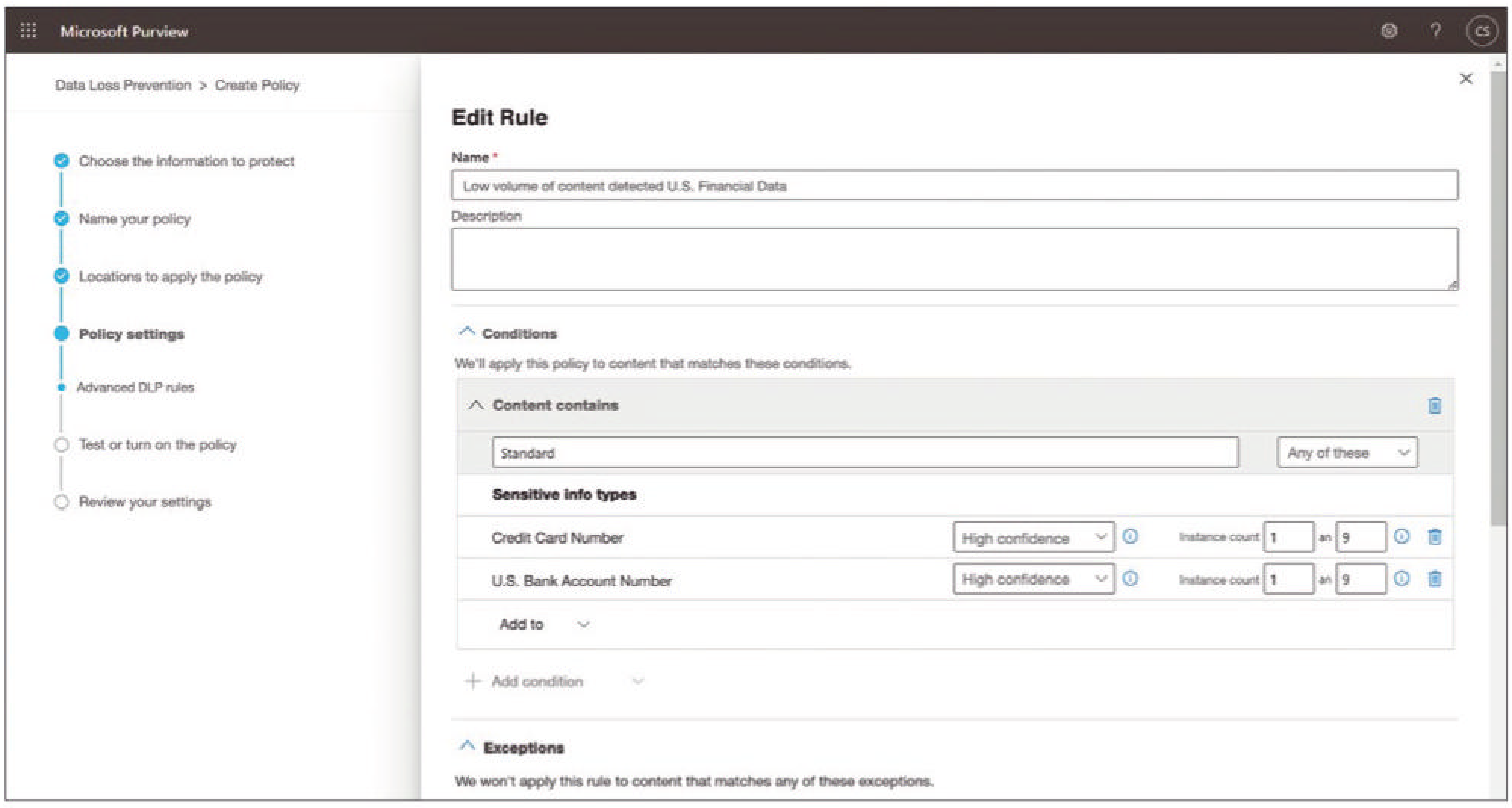
Templates
- Australia Financial Data
- Canada Financial Data
- France Financial Data
- Germany Financial Data
- Isreal Financial Data
- Japan Financial Data
- PCI Data Security Standard (PCI-DSS)
- Saudi Arabia – Anti-Cyber Crime Law

Germany Financial Data

Helps detect the presence of information commonly considered financial data in Germany, including information like EU debit card numbers.

Protects this information:
- Credit Card Number
- EU Debit Card Number

Figure 2: Microsoft Purview comes with a very extensive template set that already takes many use cases into account.


Microsoft Purview

Data Loss Prevention > Create Policy

- Choose the information to protect
- Name your policy
- Locations to apply the policy
- Policy settings
- Advanced DLP rules
- Test or turn on the policy
- Review your settings

Edit Rule

Name *: Low volume of content detected U.S. Financial Data

Description:

Conditions

We'll apply this policy to content that matches these conditions.

Content contains

Standard | Any of these

Sensitive info types

Credit Card Number	High confidence	Instance count 1	an 9
U.S. Bank Account Number	High confidence	Instance count 1	an 9

Add to

Add condition

Exceptions

We won't apply this rule to content that matches any of these exceptions.

Figure 3: A policy comprises rules that define the confidential data, but also exceptions, actions, notifications, and damage reports.

templates let you define the details of the policies up front. A DLP policy comprises rules, which in turn comprise *Conditions* describing the confidential data, *Actions*, and *Exceptions* (Figure 3).

This example looks at German financial data and needs the appropriate template. The next step of the wizard stores the name and a short description. By default, the name and description of the template are entered automatically. The places where the policy will apply are then selected. Various areas, such as Exchange, OneDrive, end devices with Windows 10/11 and macOS, and local file shares are available for selection. Note that some locations are monitored directly after selection, but others have requirements that need to be met first. For example, monitoring local shares requires the Azure Information Protection (AIP) Scanner, and some preliminary work is also required for endpoints. I go into more detail about the requirements in the course of the article.

After choosing the locations, policy settings are defined in one of two ways. In the simpler case, you just accept the default settings from the template. Here, the wizard briefly prompts you for the data to be protected, the protections, and access settings again. A policy definition is quickly finished in this way. Alternatively, you can customize rules in more detail with the *Create or customize advanced DLP rules* item. A wizard is then used to configure conditions, exceptions, actions, notifications, and override rules and to create damage reports (Figure 4).

In most cases, you will create two content-checking rules that take specific actions according to the severity of a data breach (low or high number). By default, a low count is reached for between one and nine positive hits of a data type (e.g., credit card or social security numbers), whereas a high count is more than 10 hits.

The last step is to specify whether a rule should be enabled automatically, run in test mode first, or remain disabled. Next, you see a summary and can check and save the DLP policy, after which the policy is ready for action.

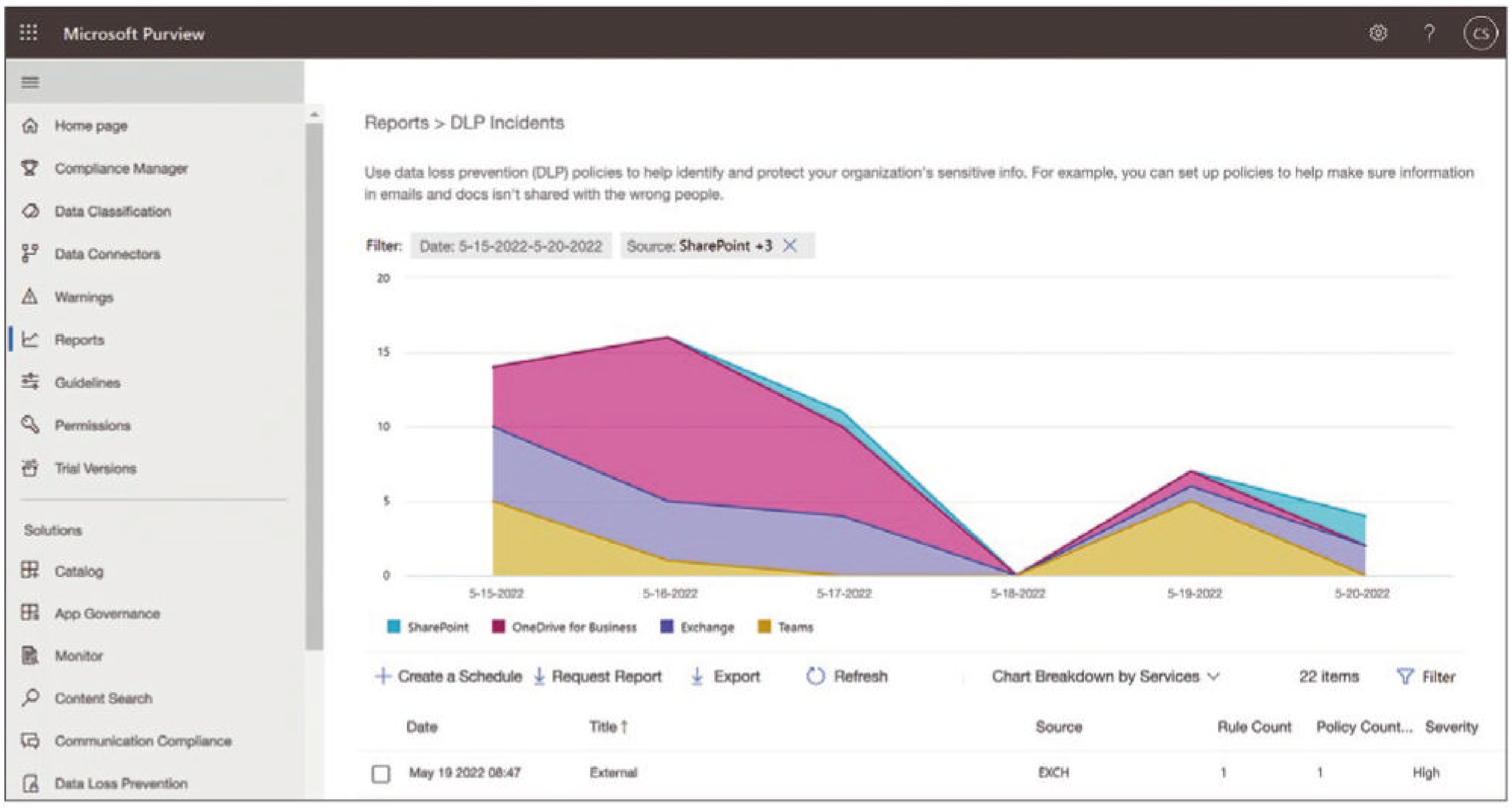
User Device and Scanner Overhead

The big advantage of Microsoft DLP is that it can be configured in one place for a wide variety of locations, including local user devices. A separate license might be required for this, as described in the "Keeping Track of Licensing" box, but that's not all, because the user devices first need to be integrated. You have several ways to do this, such as Group Policy, Intune, Endpoint Configuration Manager, a virtual desktop infrastructure (VDI) onboarding script, or a local script. I will use a local script for onboarding available under the *Settings* section in Purview.

The device appears in the device overview after a while. Details of the Endpoint DLP settings can be found

Keeping Track of Licensing

As with all cloud services, if you want to use a feature, you need a matching license – and the devil is in the details, because different licenses are required depending on which location you want to monitor. Office 365 and Microsoft 365 E3 already include DLP protection for SharePoint Online, OneDrive, and Exchange Online. However, an E5 license is required to monitor team chats. If necessary, you can order add-ons for existing licenses to use the full functionality. When monitoring local services, note that all users at the location require a license, not just the user of the scanner. A detailed check is necessary, and Microsoft provides a license overview for the required compliance services [4].



Use data loss prevention (DLP) policies to help identify and protect your organization's sensitive info. For example, you can set up policies to help make sure information in emails and docs isn't shared with the wrong people.

Date	Title ↑	Source	Rule Count	Policy Count...	Severity
May 19 2022 08:47	External	EXCH	1	1	High

Figure 4: In addition to the warning overview in the DLP section, the *Reports* item offers reports with filters on DLP information.

in the matching tab on the Data Loss Prevention page. You can enable the *Advanced classification scanning and protection* feature here, too. Because content is sent from the local device to cloud services for scanning and classification, a bandwidth limit can be configured here. Additionally, you can set file path exclusions and detail restrictions.

Setting up scanning for local directories is a little more complicated. First, the Azure Information Protection Unified Labeling Client needs to be installed and configured. It then connects to the Azure Information Protection extension on the Azure portal and adds the repositories to the content scan job. The procedure is described in detail in the documentation [5].

Policies in Action

Before activating a DLP policy, you need to *Test it out first*. In this way, you can check the policy by looking at the policy tips, alerts, and activities. Outlook online and Outlook as of version 2013 directly show you notices relating to rule violations (Figure 5). On the one hand, you get to see the unauthorized recipient, and on the other, you can view references to the confidential information.

If an employee attempts to send a sensitive email message from an older Outlook client, Exchange Online will respond with an undeliverability report, or at least document the email, depending on the configuration and severity of the violation. For documents in OneDrive and SharePoint, policy tips show up as warning icons on the individual entry. Teams directly adds a notice to the message. Not only in the applications, but also on the end device, hints can be displayed for various file actions.

The Compliance Manager can be notified directly of the incident by email, as mentioned earlier. You can define the scope of the information in a status notification in the rule settings. If required, the original email can also be sent. Alerts also appear in the DLP alert management dashboard. If you click on an alert there, an area opens up on the right making it very easy to see what happened, who was responsible for the action, and what policy applied.

The *Show details* link opens the alert and shows the content in more detail. Additionally, detailed information about the incident is displayed, and actions can be performed as a function of the incident. For example, data found on OneDrive can be deleted or sharing of the data blocked. The DLP alerts overview also acts as a simple case management system, and you can assign alerts to a user. You can manage the entries with the status *Active*, *Investigating*, *Dismissed*, or *Resolved*.

Purview also has an area for reports, including *DLP incidents*, *DLP policy matches*, and *False positives and overrides*. The reports can be requested by email and exported in Excel format, and you can create a schedule and have reports sent to you on a weekly or monthly basis.

PowerShell Alternative

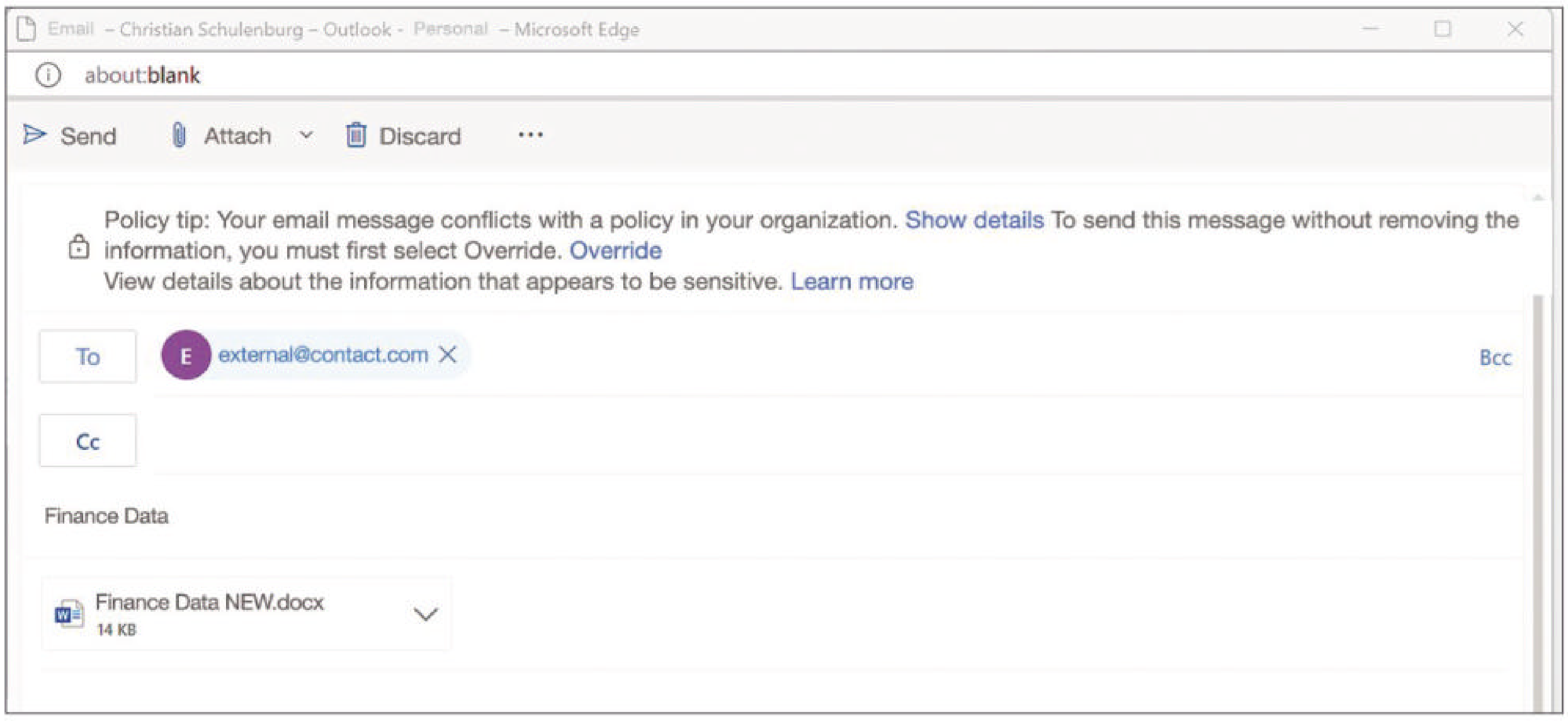
DLP policies can be managed and configured by PowerShell with the Exchange Online PowerShell V2 Module, which you can install and import with the commands:

```
Install-Module ⮧
  -Name ExchangeOnlineManagement
Import-Module ⮧
  -Name ExchangeOnlineManagement
```

Afterward, the commands

```
Connect-IPPSSession ⮧
 -UserPrincipalNamechristian@schulenburg.co
Get-DlpCompliancePolicy
Get-DlpComplianceRule
New-DlpCompliancePolicy
New-DlpComplianceRule
```

let you connect to Microsoft Online and use the DLP commands, gives you all the policy information at a glance, reveals more about the


Email – Christian Schulenburg – Outlook - Personal – Microsoft Edge

about:blank

Send   Attach   Discard   ...

Policy tip: Your email message conflicts with a policy in your organization. Show details To send this message without removing the information, you must first select Override. Override
View details about the information that appears to be sensitive. Learn more

To   external@contact.com   Bcc

Cc

Finance Data

Finance Data NEW.docx
14 KB

Figure 5: Policy tips alert the user to violations at an early stage and offer the option of overrides, like this one in Outlook.

existing rules, and creates new policies and rules. For more commands that let you manage DLP policies with PowerShell, see Table 2.

Creating Exceptions

Not every email with confidential information should be blocked outright. Employees have several ways to send messages or store data. For example, you can define a policy for the locations stating to whom they apply or do not apply. The filters have a different effect depending on the location: Exchange uses distribution groups to control adding and blocking, whereas SharePoint uses sites to differentiate. Exceptions can also be created directly in a rule. Note that each location can offer different exceptions. If multiple locations are selected, only exceptions that apply to all locations can be configured. For example, the recipient, file extensions, and document name can be selected here. Once you have selected all locations for monitoring, don't be surprised to see the option to add exceptions grayed out.

If you do not have an exception from the outset, you can configure an override for the end user. To do so, enable the *Allow overrides from M365 services* item in the rule settings. Optionally, a business justification can be requested in the process. A policy can be overridden if an employee has reported it as a false positive. Overriding is done from the policy tip client-side.

An option in Outlook and Teams lets you bypass the policy when composing a message. In the window, you specify the reason for overriding to enable sending, which means that users in Exchange, SharePoint, OneDrive, and Teams can override DLP policies, if needed. The Compliance Manager is, of course, informed about the exception in the *Justification* text in the status report. DLP policies provide a sufficient choice of exceptions for senior management or specialist departments that have to work with sensitive data all the time so that they are not hindered in their daily tasks.

Conclusions

DLP policies provide a quick way to check the daily flood of data from various Microsoft services for compliance with on-board tools. On the positive side, the variety of locations that can be included in a single policy makes the setup fast and clear-cut. ■

Table 2: **PowerShell for DLP Policies**

Cmdlet	Function
`Get-DlpCompliancePolicy`	Displays information about existing data loss prevention policies
`Get-DlpPolicyTemplate`	Displays existing DLP policy templates in an Exchange organization
`Get-DlpDetailReport`	Lists details of DLP rule matches for Exchange Online, SharePoint Online, and OneDrive for Business for the last 30 days
`Get-DlpDetectionReport`	Displays a summary of DLP rule matches for Exchange Online, SharePoint Online, and OneDrive for Business for the last 30 days
`New-DlpCompliancePolicy`	Creates a DLP policy in an Exchange organization
`Remove-DlpCompliancePolicy`	Removes an existing DLP policy
`Remove-DlpComplianceRule`	Removes an existing DLP rule
`Set-DlpPolicy`	Modifies a DLP policy in an organization

Info

[1] Microsoft Purview overview: [https://learn.microsoft.com/en-us/microsoft-365/compliance/microsoft-365-compliance-center?view=o365-worldwide]

[2] DLP policy templates: [https://learn.microsoft.com/en-us/microsoft-365/compliance/what-the-dlp-policy-templates-include?view=o365-worldwide]

[3] Microsoft compliance: [https://compliance.microsoft.com/homepage]

[4] License overview of the required compliance components: [https://learn.microsoft.com/en-us/office365/servicedescriptions/microsoft-365-service-descriptions/microsoft-365-tenantlevel-services-licensing-guidance/microsoft-365-security-compliance-licensing-guidance]

[5] Local M365 scanner: [https://learn.microsoft.com/en-us/microsoft-365/compliance/dlp-on-premises-scanner-use?view=o365-worldwide]

■

The end of SHA-1

Pulling the Plug

The SHA-1 cryptographic hash function has been considered insecure for a long time. Some Linux distributions have banned or no longer use it – with some consequences, though. By Thorsten Scherf

Cryptographic hash functions have a number of properties. For example, one requirement is that a unique checksum with a fixed length can be generated from arbitrary data. Collisions – where two different datasets have the same checksum – must be practically impossible. Finally, cryptographic hash functions are one-way functions, which ensures that it is virtually impossible to reconstruct the original data from the checksum.

These properties are ideal for integrity checking arbitrary data, because it is practically impossible to change a message without also changing the message's checksum. Typical applications for digital signatures include X.509 certificates, PGP/GPG keys, software packages, and DNS entries. Of course, signatures like this are also used when you transmit data over a network to ensure the integrity of the transmitted data.

Insecure Hash Functions

If one of the aforementioned properties no longer applies to a cryptographic hash function in the public domain, it is considered broken and should no longer be used. However, you need to make a distinction between theoretical and practical attacks on the algorithm. Nowadays it is possible to break Message-Digest Algorithm 5 (MD5) [1] with very little effort, which is why it has not been used for some time. The situation is somewhat different with the Secure Hash Algorithm (SHA). The first version of SHA was published by the National Institute of Standards and Technology (NIST) in 1993. The algorithm generates a 160-bit checksum. However, because of a design flaw, a corrected version of the algorithm, commonly known as SHA-1 [2], was released just two years later.

Even for this version, an attack method was presented by Chinese scientists back in 2005, demonstrating that a collision is theoretically possible. The same year saw the publication of an optimized version of this theoretical attack, where a collision was possible with far fewer operations. However, the required computational overhead would have been so massive that an implementation was hardly practical.

From the findings of the initial attacks on SHA-1, NIST nevertheless promptly recommended that SHA-2, a SHA-1 successor, be used from that point on. SHA-2 was published back in 2001, and the hash algorithms belonging to this family have a far longer checksum than SHA-1 – 256 bits (SHA-256) or 512 bits (SHA-512). In 2011, NIST even officially classified the SHA-1 algorithm as "deprecated."

Stubborn

Despite NIST's recommendation to discontinue SHA-1 and despite the availability of a secure successor in the SHA-2 and, later, the SHA-3 family, SHA-1 continued to be used virtually everywhere without a care. Then, in 2017, Shattered [3], a new variant of an attack on SHA-1, was introduced to the general public. Scientists were able to generate two different PDF files with the same checksum but with relatively little overhead compared with previous attacks.

SHA-1 can be regarded as definitely broken since 2020, when chosen-prefix collision attacks succeeded for the first time. This attack makes it possible to generate the same checksum for any two datasets with relatively little overhead and inserts additional byte sequences into the data [4]. Currently, hardware worth less than $35,000 is required for an attack of this type, and it can be assumed that the costs will drop considerably in the coming years.

Software Projects Without SHA-1

In the meantime, many software projects have completely abandoned SHA-1 or have at least severely restricted its use. For example, OpenSSL has not allowed X.509 certificates to be signed with a SHA-1 signature for some time now [5], and OpenSSH has also not allowed RSA keys that use SHA-1 for signing since version 8.8 [6].
If you still want to use the RSA algorithm for the host key of an SSH server, it must be used together with a hash algorithm from the SHA-2 family (e.g., SHA-256 or SHA-512). However, OpenSSH does not support these algorithms below version 7.2 [7]. Alternatively, of course, another algorithm (e.g., the elliptic curve digital signature algorithm, ECDSA) can be used instead. This option is especially interesting for older OpenSSH servers that do not yet support SHA-2. You can create a new host key quite quickly with:

```
ssh-keygen -t ecdsa -b 384 ↵
  -f /etc/ssh/ssh_host_ecdsa_key
```

Then, use the `HostKey` option and enter

```
HostKey /etc/ssh/ssh_host_ecdsa_key
```

in the `/etc/ssh/sshd_config` OpenSSH server configuration file.

Fedora Drops SHA-1

The Fedora Linux distribution and related distributions (e.g., CentOS Stream, Red Hat Enterprise Linux) have completely banned SHA-1 from their current versions. Crypto policies are used to regulate which algorithms are permitted on the system and can be used by the individual cryptographic components. However, the `DEFAULT` policy disables SHA-1 so that crypto components on a system of this type can no longer use SHA-1 to generate or verify digital signatures.
In addition to the problems with older SSH implementations already mentioned, users especially notice this change when they use software packages that still have a SHA-1 signature: You can no longer install them on a Fedora 36 system [8] because the system cannot verify the package signature. The correct approach is to ask the package vendor to sign the software with an algorithm other than SHA-1. In the short term, you can also turn off signature verification for individual transactions:

```
dnf install --setopt=tsflags=↵
  nocrypto foo.rpm
```

If you prefer to use the RPM package manager to install the package instead, the command is:

```
rpm -Uhv --nosignature foo.rpm
```

However, I need to point out explicitly at this point that installing software packages without signature verification is not recommended and endangers the security of the entire system.

Fallback to SHA-1

In a few cases, it may be necessary to make the SHA-1 algorithm available again on a system, at least temporarily. To do so, you use the SHA-1 crypto policy to load the policy in addition to the default:

```
update-crypto-policies --set DEFAULT:SHA1
```

However, this process also jeopardizes the security of the entire system because, from now on, all crypto components have access to SHA-1 again.

Conclusions

The funny thing about hash functions is that the original input cannot be calculated from a hash value and, moreover, two different sets of inputs will never result in the same hash value. To be certain, cryptographic methods rely on complex mathematics. However, it is not only errors in the algorithms that invalidate the two basic requirements in some hash methods. Greater compute power can also help crack weak algorithms.
SHA-1 has long been considered insecure, and practical attacks that break the algorithm with relatively little effort have existed for several years. Avoiding SHA-1 and using SHA-2 or SHA-3 instead is therefore highly advisable. Logically, Fedora and other Linux distributions have now completely disabled SHA-1, while still giving users the option to revive the algorithm if needed; however, you seriously need to consider whether you have genuinely compelling reasons for doing so. ■

Info

[1] MD5: [https://de.wikipedia.org/wiki/Message-Digest_Algorithm_5]

[2] SHA-1: [https://de.wikipedia.org/wiki/Secure_Hash_Algorithm]

[3] Shattered: [https://shattered.io]

[4] Shambles: [https://sha-mbles.github.io]

[5] OpenSSL and SHA-1: [https://www.openssl.org/news/changelog.html#openssl-30]

[6] OpenSSH disables SHA-1: [https://www.openssh.com/txt/release-8.8]

[7] OpenSSH support for SHA-2: [https://www.openssh.com/txt/release-7.2]

[8] Fedora and RHEL drop SHA-1: [https://www.redhat.com/en/blog/legacy-cryptography-fedora-36-and-red-hat-enterprise-linux-9]

The Author

Thorsten Scherf is the global Product Lead for Identity Management and Platform Security in Red Hat's Product Experience group. He is a regular speaker at various international conferences and writes a lot about open source software.

ESXi ransomware attacks

New Targets

Files encrypted by ransomware have been the nightmare scenario of IT departments, and even specialized operating systems like the ESXi server are not immune. We look at how to mitigate risk and prepare for recovery if hypervisor protection fails. By Matthias Wübbeling

Today, it hardly matters which operating systems are used on servers; the malware developers working in the background cover all the popular systems. Even specialist operating systems such as the VMware ESXi hypervisor have repeatedly been targeted by criminals. This article sheds light on the damage potential, pointing out ways to mitigate risk and actions to help prepare for an incident.

In many cases, you will hear about the benefits of virtualization, the added security that isolating individual machines can provide, and how easy it is to revert to previous versions at any time with snapshots. Modern ransomware and the behavior of the groups behind it have adapted to this kind of reasoning and the technology behind it. Today, malware is installed well ahead of the attack. The overhead required to analyze attacked infrastructures gives the attackers a clear advantage: They already know all the systems; the deployed software, including the security suites and backup applications; the login data; and areas of responsibility of the employees and their vacation planning.

Attacks on Hypervisors

Attempting to fight this professionalization on the part of the criminals are IT departments in small to large enterprises. Besides handling security, they are primarily responsible for the continuous operation of the infrastructure. In addition to the operating systems of the virtual machines (VMs), the hypervisors on which the VMs run have long been the focus of attackers. Most recently, ransomware named Cheerscrypt [1] grabbed the limelight about the middle of last year. It is based on the Linux variant of the Babuk malware and attacks VMware ESXi servers through known vulnerabilities and successively encrypts the files used by VMware.

In this case, the attack usually occurs by way of the hypervisor guests and a vulnerability in the hypervisor's hardware abstraction or isolation. ESXi also offers some attack vectors itself, though. Although the list of CVEs is not as long as for other software, it also includes vulnerabilities of the highest severity and the ability to execute arbitrary code in the context of the server software as a remote attacker.

Greater Potential for Damage

In contrast to attacking individual VMs, controlling the hypervisor can cause even greater damage immediately. Also, basically no malware protection products are available for ESXi on the hypervisor itself. VMware also does not think that antivirus programs are necessary on the ESXi server [2]. This attitude not only leads to inadequate protection against attacks, but also to failures in detecting successful attacks. It also follows that, from the perspective of ESXi cluster administration teams, security only plays a minor role in day-to-day operations.

Running VMs securely is usually handled by dedicated administrators who are responsible for individually protecting the operating systems used by the VMs. After all, the VMs primarily offer services over the network and are therefore naturally considered vulnerable. Unfortunately, this always leaves you one massive step behind the attackers, and when the active VMs become unreachable, the ESXi administrators need to respond.

If an ESXi hypervisor is directly affected by malware, if even material parts of the active VMs are encrypted, these VMs will usually be inoperable. If you run ESXi as a cluster in your company, the other servers in the cluster and the connected storage are also affected in the event of a successful attack.

Photo by Afif Ramdhasuma on Unsplash

Immediate Response to Attacks

When responding to a security incident, the initial focus in administration is on getting services up and running again on the affected VMs. Depending on the dimensions of the cluster, several VMs could fail more or less simultaneously. In the best case, you will have comprehensive and secure backups and be able to have the machines themselves up and running again quickly. Whether checking and backing up the hypervisor is covered in this situation depends on an administrator's response.

To get the system up again and, in case of doubt, to ensure the survival of the enterprise, the people responsible might even consider paying ransom to the criminals. At the end of June, a public letter from IT security experts from academia and industry discussed and criticized the payment of ransom across society [3]. Although many insurance companies will pay the ransom today, some providers have already explicitly ruled out payment.

Of course, criminals are also responding to these developments and to better backup strategies in the enterprise, sometimes demanding double ransom payments: Before the data is encrypted, it is copied in plain text to the attacker's servers. Companies are then asked to pay once to be able to decrypt the encrypted data – if the criminals actually planned to do this – and again to prevent sensitive company data being published.

Reconstructing the Cluster

Even if you can get critical VMs up and running again from existing backups on other servers, the ESXi cluster needs to be completely rebuilt. This is sometimes a challenge because infrastructures often grow dynamically. In some cases, the team member who created the initial configuration is no longer with the company, or the documentation is sparse. However, if you have access to scripts prepared for automatic installation of the cluster, also known as kickstart files, the reconstruction work is far easier to handle and faster [4].

If you have an Enterprise Plus license, you can back up your cluster's configuration profiles and the configurations of the virtual switches automatically. If not, you can run a backup manually or automate it with your own tools. Use the following commands at the ESXi command line to synchronize the configuration first and then create a `tar.gz` file that you can download in your browser:

```
vim-cmd hostsvc/firmware/sync_config
vim-cmd hostsvc/firmware/backup_config
```

The output from the second command contains the URL for downloading the file. If you prefer to use the vSphere command-line interface (CLI), or if it is easier for you to automate, open the CLI and navigate to `C:\Program Files\VMware\VMware vSphere CLI\bin`. Launch the backup program there with the command:

```
vicfg-cfgbackup.pl --server=esxi ⮐
  --username=root -s latest_backup.tgz
```

You will then be prompted to enter the root password. To work around this prompt (e.g., in your automated scripts), you can pass in the password directly with `--password=<password>`. You will then find the backup file in your current working directory.

To restore the configuration, you first need to install a version with the same build ID as the one used to make the backup. Copy the backup file to the system or the attached datastore and then connect to the console. When you get there, switch the system to maintenance mode before starting the recovery:

```
vim-cmd hostsvc/maintenance_mode_enter
vim-cmd hostsvc/firmware/ restore_config ⮐
  /<directory>/latest_backup.tgz
```

If the universally unique identifier (UUID) of the host system has changed in the meantime, you need to add 1 before specifying the backup file. This means that the UUID is overwritten, but this only works if the backup was not encrypted. Encryption was introduced in vSphere version 7.0 U2. However, the key remains in the hardware's trusted platform module (TPM), and the file can only be recovered on the same host system. Afterward, you will of course need to install all the recommended updates and restore the backups of the VMs.

Conclusions

This article illustrates the consequences of being hit by a specialized ransomware variant on the ESXi server and how you can at least prepare for recovery, if you can't protect yourself reliably. ■

Info

[1] Linux Cheerscrypt ransomware: [https://www.trendmicro.com/en_us/research/22/e/new-linux-based-ransomware-cheerscrypt-targets-exsi-devices.html]

[2] Antivirus protection on hypervisors: [https://kb.vmware.com/s/article/80768]

[3] Ransom letter: [https://www.databreaches.net/ransomware-ransom-payments-a-geostrategic-risk/]

[4] Kickstart files for VMware vSphere: [https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.esxi.install.doc/GUID-00224A32-C5C5-4713-969A-C50FF4DED8F8.html]

The Author

Dr. Matthias Wübbeling is an IT security enthusiast, scientist, author, consultant, and speaker. As a Lecturer at the University of Bonn in Germany and Researcher at Fraunhofer FKIE, he works on projects in network security, IT security awareness, and protection against account takeover and identity theft. He is the CEO of the university spin-off Identeco, which keeps a leaked identity database to protect employee and customer accounts against identity fraud. As a practitioner, he supports the German Informatics Society (GI), administrating computer systems and service back ends. He has published more than 100 articles on IT security and administration.

Network performance in the world's favorite data center

Head in the Cloud

We launch a network performance test on EC2 to answer the question: Is the cloud as fast as expected?

By Federico Lucifredi

In the Performance Tuning Dojo article two issues ago [1], I determined the importance of baselines to every performance analysis. This month, I measure a baseline on the cloud and confirm it meets my expectations – and discuss what to do in light of unexpected surprises. Network performance is a complex problem with many variables. To make the problem tractable, you start by looking at specific scenarios, as opposed to the entire problem space in abstract. What does network performance look like in AWS Elastic Compute Cloud (EC2) when using familiar tools?

Amazon's Network

At its simplest, networking in AWS is straightforward: Launching an instance with the default networking configuration will give you an instance with a valid IP address. At a minimum, an AWS instance has one network device attached. The maximum number of network devices that can be attached depends on the instance type, but it is safe to assume that current instance types (except T2 and M3) all support enhanced networking for Linux. That is where the easy part ends, with available instance configurations reaching to 170Gbps with a single card (and 200Gbps with two cards or the Elastic Fabric Adapter) [2]. Verifying that the expected configuration throughput matches actual wire speed is where benchmarking comes in. Following the advice in Table 1, launch two `m6in.2xlarge` instances located in the same AWS availability zone for testing (see the "AWS Testing Policy" box). Rated at "up to 40 Gbps" by AWS [6] and equipped with an Elastic Network Adapter (ENA), they should perform well with any operating system providing appropriate hardware support [7]. I am using Ubuntu 22.04 (Jammy) for this test in the original `us-east-1` availability zone.

```
$ aws ec2 run-instances ↩
  --image-id ami-0ea1c7db66fee3098 ↩
  --region us-east-1 ↩
  --key federico ↩
  --instance-type m6in.2xlarge ↩
  --count 2 ↩
  --output text
```

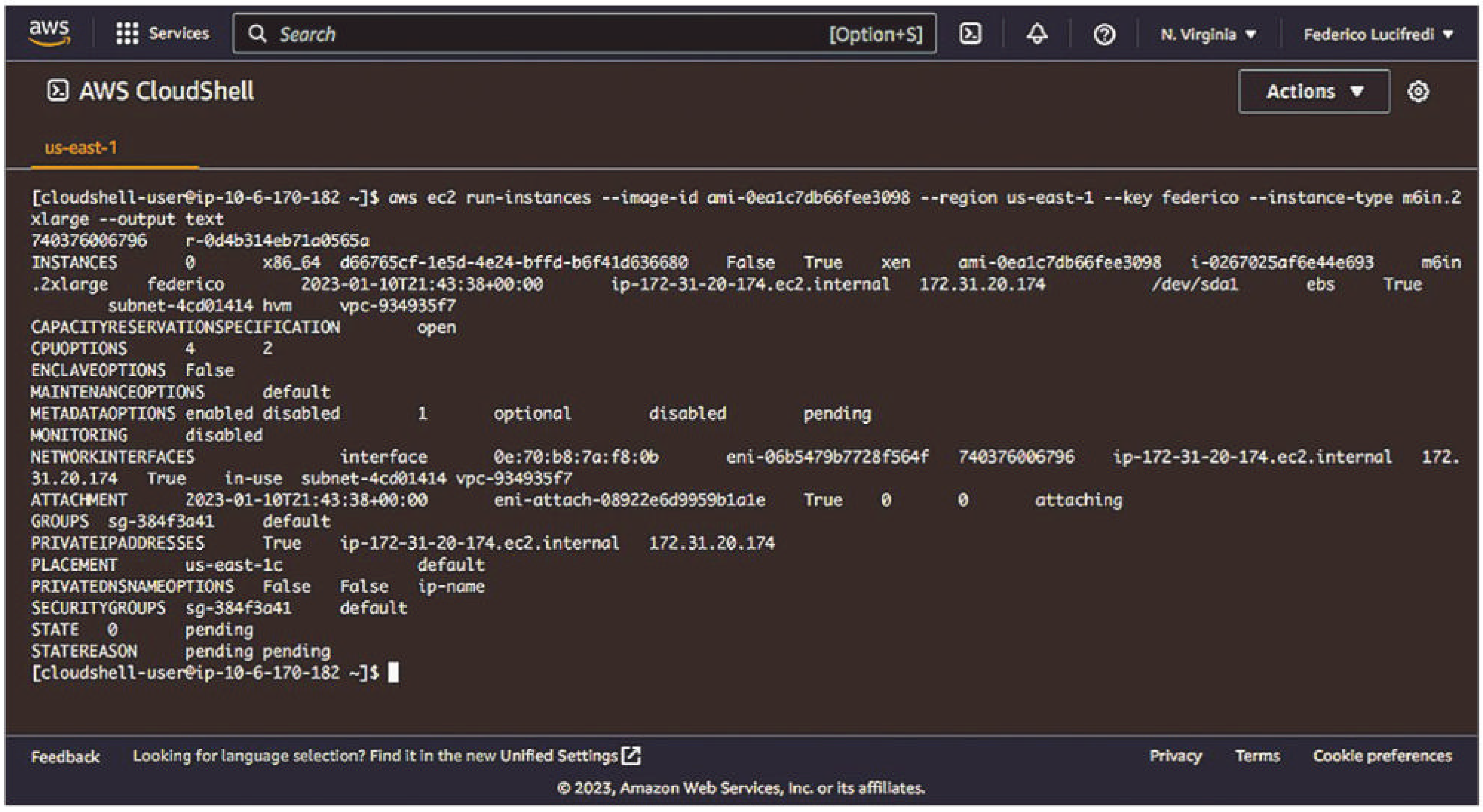
Table 1: Network Performance Factors*

Enhanced networking [4]		SR-IOV [5] and better hardware
Elastic network adapter (ENA)		
Path MTU	Enable jumbo frames	Improve throughput, not latency
Direct connection		No intervening gateways
Physical distance	Same availability zone	The closer the faster
	Same region	
	Same continent	
Instance type		Bigger is better (CPU bound)
		Credit-IO instances (like T4g) vary performance
Multithreaded application		
Elastic Fabric Adapter		

* Major cloud network performance factors, roughly in order of precedence.

AWS Testing Policy

Network performance testing on EC2 may inadvertently resemble a distributed denial of service (DDoS) attack. Although the differences between a benchmark and DDoS attack are clear, their similarity lies in the attempt to reach a target node's limits with external traffic. Nothing in this two-node testing should cause AWS to get upset with you, but when performing testing with a fleet of client nodes, it is worth remembering to review their testing policy [3] and determine whether advance notice of your plans is in order. AWS may have suggestions for your testing plan and will be standing by to intervene during your test should the availability zone's network be negatively affected.



```
[cloudshell-user@ip-10-6-170-182 ~]$ aws ec2 run-instances --image-id ami-0ea1c7db66fee3098 --region us-east-1 --key federico --instance-type m6in.2
xlarge --output text
740376006796    r-0d4b314eb71a0565a
INSTANCES       0       x86_64  d66765cf-1e5d-4e24-bffd-b6f41d636680    False   True    xen     ami-0ea1c7db66fee3098   i-0267025af6e44e693     m6in
.2xlarge    federico        2023-01-10T21:43:38+00:00       ip-172-31-20-174.ec2.internal   172.31.20.174           /dev/sda1       ebs     True
        subnet-4cd01414 hvm     vpc-934935f7
CAPACITYRESERVATIONSPECIFICATION        open
CPUOPTIONS      4       2
ENCLAVEOPTIONS  False
MAINTENANCEOPTIONS      default
METADATAOPTIONS enabled disabled        1       optional        disabled        pending
MONITORING      disabled
NETWORKINTERFACES               interface       0e:70:b8:7a:f8:0b       eni-06b5479b7728f564f   740376006796    ip-172-31-20-174.ec2.internal   172.
31.20.174    True    in-use  subnet-4cd01414 vpc-934935f7
ATTACHMENT      2023-01-10T21:43:38+00:00       eni-attach-08922e6d9959b1a1e    True    0       0       attaching
GROUPS  sg-384f3a41     default
PRIVATEIPADDRESSES      True    ip-172-31-20-174.ec2.internal   172.31.20.174
PLACEMENT       us-east-1c              default
PRIVATEDNSNAMEOPTIONS   False   False   ip-name
SECURITYGROUPS  sg-384f3a41     default
STATE   0       pending
STATEREASON     pending pending
[cloudshell-user@ip-10-6-170-182 ~]$
```

Figure 1: Launching a couple of `m6in.2xlarge` instances from the AWS CloudShell.

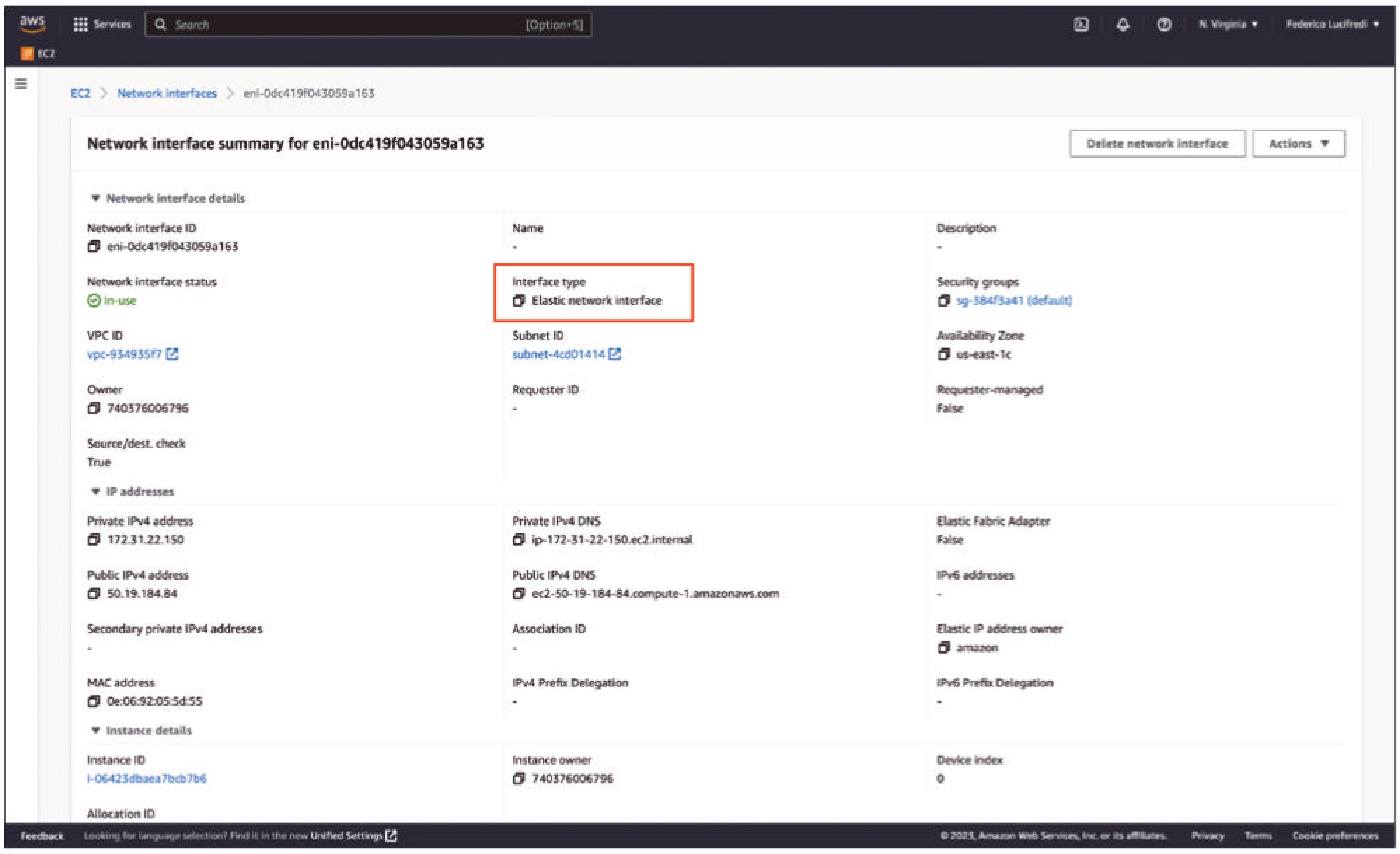
The results are shown in Figure 1. Here I used AWS's CloudShell service, which provides convenient access to the client-side command-line interface (CLI) in a pre-configured environment hosted in a web browser terminal.

Dirty Hands

Logging in to the AWS Console, you should double-check first that ENA is correctly enabled (Figure 2). At the instance's shell, you can verify the same by checking that the corresponding driver is loaded (Figure 3):

```
$ modinfo ena
```

The same could have been ascertained through the AWS CLI with a



aws
Services
Search
[Option+S]
N. Virginia
Federico Lucifredi
EC2
EC2 > Network interfaces > eni-0dc419f043059a163
Delete network interface
Actions
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2023, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Network interface summary for eni-0dc419f043059a163

Network interface details

Network interface ID eni-0dc419f043059a163	Name -	Description -
Network interface status In-use	Interface type Elastic network interface	Security groups sg-384f3a41 (default)
VPC ID vpc-934935f7	Subnet ID subnet-4cd01414	Availability Zone us-east-1c
Owner 740376006796	Requester ID -	Requester-managed False
Source/dest. check True		

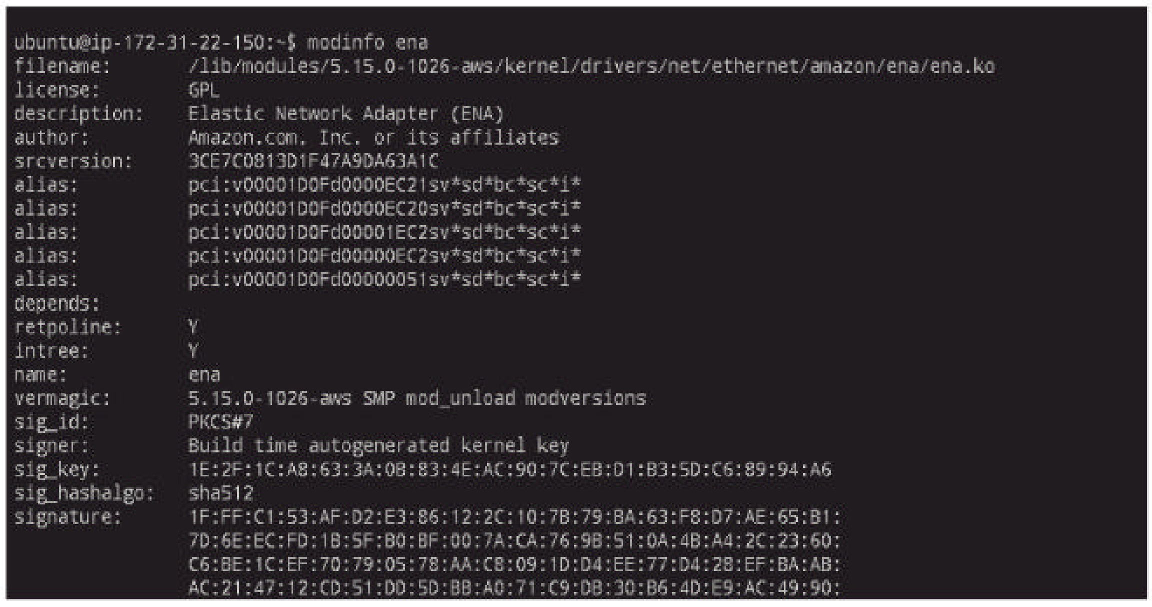
IP addresses

Private IPv4 address 172.31.22.150	Private IPv4 DNS ip-172-31-22-150.ec2.internal	Elastic Fabric Adapter False
Public IPv4 address 50.19.184.84	Public IPv4 DNS ec2-50-19-184-84.compute-1.amazonaws.com	IPv6 addresses -
Secondary private IPv4 addresses -	Association ID -	Elastic IP address owner amazon
MAC address 0e:06:92:05:5d:55	IPv4 Prefix Delegation -	IPv6 Prefix Delegation -

Instance details

Instance ID i-06423dbaea7bcb7b6	Instance owner 740376006796	Device index 0
Allocation ID		

Figure 2: Browsing the AWS Console to verify the instances indeed have an Elastic network interface.



```
ubuntu@ip-172-31-22-150:~$ modinfo ena
filename:       /lib/modules/5.15.0-1026-aws/kernel/drivers/net/ethernet/amazon/ena/ena.ko
license:        GPL
description:    Elastic Network Adapter (ENA)
author:         Amazon.com, Inc. or its affiliates
srcversion:     3CE7C0813D1F47A9DA63A1C
alias:          pci:v00001D0Fd0000EC21sv*sd*bc*sc*i*
alias:          pci:v00001D0Fd0000EC20sv*sd*bc*sc*i*
alias:          pci:v00001D0Fd00001EC2sv*sd*bc*sc*i*
alias:          pci:v00001D0Fd00000EC2sv*sd*bc*sc*i*
alias:          pci:v00001D0Fd00000051sv*sd*bc*sc*i*
depends:
retpoline:      Y
intree:         Y
name:           ena
vermagic:       5.15.0-1026-aws SMP mod_unload modversions
sig_id:         PKCS#7
signer:         Build time autogenerated kernel key
sig_key:        1E:2F:1C:A8:63:3A:08:83:4E:AC:90:7C:EB:D1:B3:5D:C6:89:94:A6
sig_hashalgo:   sha512
signature:      1F:FF:C1:53:AF:D2:E3:86:12:2C:10:7B:79:BA:63:F8:D7:AE:65:B1:
                7D:6E:EC:FD:1B:5F:B0:BF:00:7A:CA:76:9B:51:0A:4B:A4:2C:23:60:
                C6:BE:1C:EF:70:79:05:78:AA:C8:09:1D:D4:EE:77:D4:28:EF:BA:AB:
                AC:21:47:12:CD:51:DD:5D:BB:A0:71:C9:DB:30:B6:4D:E9:AC:49:90:
```

Figure 3: Checking the network interface at the operating system level.

clever query (Figure 4). You don't actually need to repeat this check – pick the interface style that most suits you.

Onward and Upward

After updating the package caches (`sudo apt update`), you can install `iperf3` [8], the standard tool to test a network path's performance end to end. Found in the Ubuntu Universe repository (install it with `sudo apt install iperf3`), it measures the effective bit rate between two systems by taking multiple samples and averaging the results. The `iperf3` tool defaults to uploads from client to server, but the reverse is also possible (`-R` option) for bi-directional testing. Launch the server first:

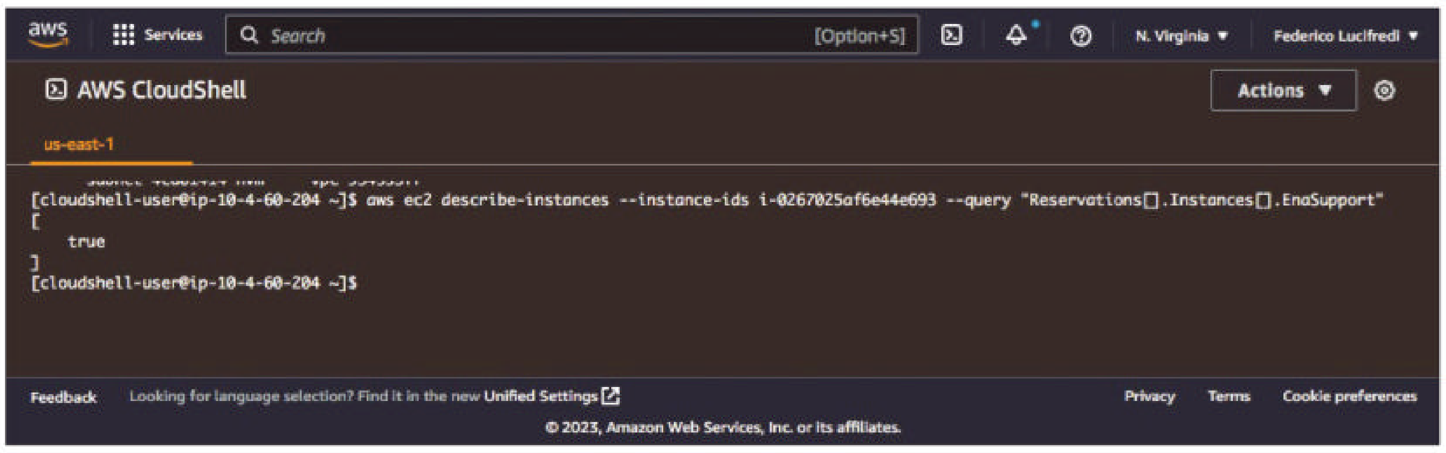
```
$ iperf3 -s
-------------------------
Server listening on 5201
-------------------------
```

Note that the server will listen on all the IP addresses of the instance, which will come in handy later. The results are lackluster; you are paying for 40Gbps connectivity, yet the benchmark is showing only 4.7Gbps were achieved (Figure 5)! What is going on?

Inspecting the connection with `tracepath` [9] shows that although the client is configured with the "jumbo frame" maximum transmission unit (MTU) of 9001 bytes [10], an intervening node is providing the standard Ethernet MTU of 1500 bytes (Figure 6). The explanation for this is straightforward: You used public IP addressing for your test, and 5Gbps is the highest bandwidth the gateway node of the availability zone (AZ) can supply. Switching the test to the private address of your instances (remember that the two instances are on the same AZ), you can achieve a more respectable 9.5Gbps (Figure 7). As you are now hitting the limits of a single network socket, your next test ramps up to 20 parallel connections:

```
$ iperf3 -c 172.31.20.174 -i 2 -P 20
```

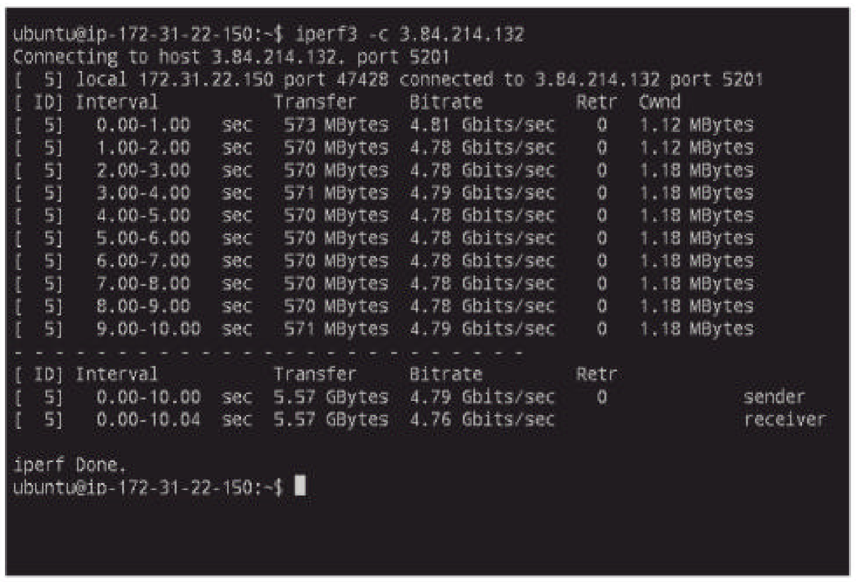
The log is somewhat unwieldy, but the final tally reached 34Gbps



```
[cloudshell-user@ip-10-4-60-204 ~]$ aws ec2 describe-instances --instance-ids i-0267025af6e44e693 --query "Reservations[].Instances[].EnaSupport"
[
    true
]
[cloudshell-user@ip-10-4-60-204 ~]$
```

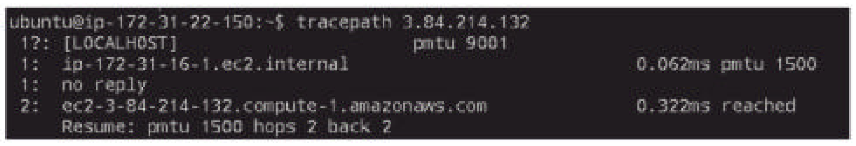
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences
© 2023, Amazon Web Services, Inc. or its affiliates.

Figure 4: Querying the network interface from the AWS CLI.



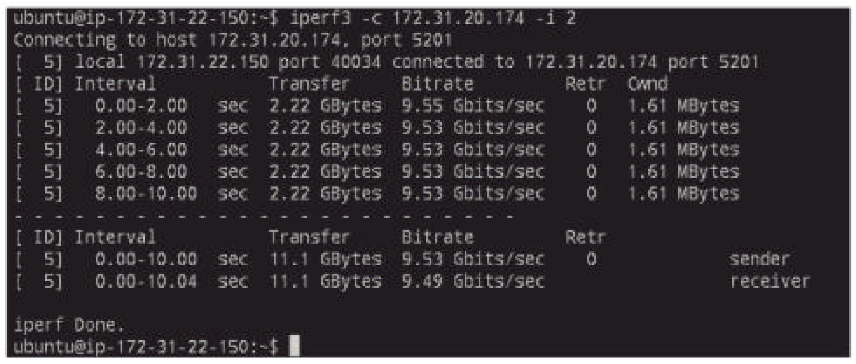
```
ubuntu@ip-172-31-22-150:~$ iperf3 -c 3.84.214.132
Connecting to host 3.84.214.132, port 5201
[  5] local 172.31.22.150 port 47428 connected to 3.84.214.132 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   573 MBytes  4.81 Gbits/sec    0   1.12 MBytes
[  5]   1.00-2.00   sec   570 MBytes  4.78 Gbits/sec    0   1.12 MBytes
[  5]   2.00-3.00   sec   570 MBytes  4.78 Gbits/sec    0   1.18 MBytes
[  5]   3.00-4.00   sec   571 MBytes  4.79 Gbits/sec    0   1.18 MBytes
[  5]   4.00-5.00   sec   570 MBytes  4.78 Gbits/sec    0   1.18 MBytes
[  5]   5.00-6.00   sec   570 MBytes  4.78 Gbits/sec    0   1.18 MBytes
[  5]   6.00-7.00   sec   570 MBytes  4.78 Gbits/sec    0   1.18 MBytes
[  5]   7.00-8.00   sec   570 MBytes  4.78 Gbits/sec    0   1.18 MBytes
[  5]   8.00-9.00   sec   570 MBytes  4.78 Gbits/sec    0   1.18 MBytes
[  5]   9.00-10.00  sec   571 MBytes  4.79 Gbits/sec    0   1.18 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  5.57 GBytes  4.79 Gbits/sec    0             sender
[  5]   0.00-10.04  sec  5.57 GBytes  4.76 Gbits/sec                  receiver

iperf Done.
ubuntu@ip-172-31-22-150:~$ █
```

Figure 5: The first benchmark shows 4.7Gbps, but the advertised speed is 40Gbps.



```
ubuntu@ip-172-31-22-150:~$ tracepath 3.84.214.132
 1?: [LOCALHOST]                      pmtu 9001
 1:  ip-172-31-16-1.ec2.internal                          0.062ms pmtu 1500
 1:  no reply
 2:  ec2-3-84-214-132.compute-1.amazonaws.com             0.322ms reached
     Resume: pmtu 1500 hops 2 back 2
```

Figure 6: Path MTU shows an intermediate node is choking traffic.



```
ubuntu@ip-172-31-22-150:~$ iperf3 -c 172.31.20.174 -i 2
Connecting to host 172.31.20.174, port 5201
[  5] local 172.31.22.150 port 40034 connected to 172.31.20.174 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-2.00   sec  2.22 GBytes  9.55 Gbits/sec    0   1.61 MBytes
[  5]   2.00-4.00   sec  2.22 GBytes  9.53 Gbits/sec    0   1.61 MBytes
[  5]   4.00-6.00   sec  2.22 GBytes  9.53 Gbits/sec    0   1.61 MBytes
[  5]   6.00-8.00   sec  2.22 GBytes  9.53 Gbits/sec    0   1.61 MBytes
[  5]   8.00-10.00  sec  2.22 GBytes  9.53 Gbits/sec    0   1.61 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  11.1 GBytes  9.53 Gbits/sec    0             sender
[  5]   0.00-10.04  sec  11.1 GBytes  9.49 Gbits/sec                  receiver

iperf Done.
ubuntu@ip-172-31-22-150:~$ █
```

Figure 7: Avoiding the intervening gateway results in double the performance.

(Figure 8). The final piece of the puzzle is unexpected: iperf3 is a single-threaded process; thus, despite your instances sporting eight cores each, the network performance of the test is becoming CPU-bound. The older iperf [11] is multithreaded, and it delivers the best result yet, at 39.7Gbps (Figure 9). Amazon was not lying after all! And this is merely the beginning, because some instance types sport eight network adapters. ■

Info

[1] "Baselines are more important than the benchmark" by Federico Lucifredi, *ADMIN*, issue 71, 2022, pg. 94, [https://www.admin-magazine.com/Archive/2022/71/Baselines-are-more-important-than-the-benchmark]

[2] Amazon AWS, EC2 instance types: [https://aws.amazon.com/ec2/instance-types/]

[3] Amazon AWS network stress test policy: [https://aws.amazon.com/ec2/testing/]

[4] Configuring enhanced networking on EC2: [https://aws.amazon.com/premiumsupport/knowledge-center/enable-configure-enhanced-networking/]

[5] Overview of SR-IOV: [https://learn.microsoft.com/en-us/windows-hardware/drivers/network/overview-of-single-root-i-o-virtualization--sr-iov-]

[6] Instance types by network performance: [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/general-purpose-vinstances.html#general-purpose-network-performance]

[7] Enhanced networking with the ENA: [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking-vena.html]

[8] iperf3(1) man page: [https://manpages.ubuntu.com/manpages/focal/en/man1/iperf3.1.html]

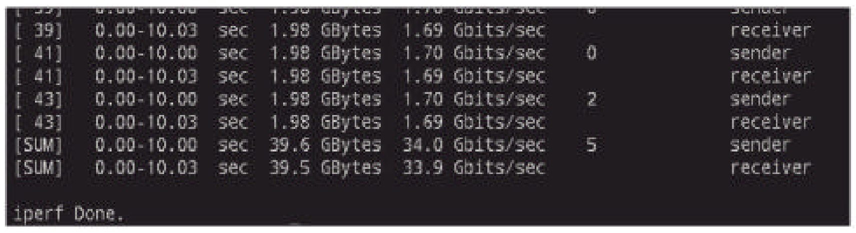
[9] tracepath(8) man page: [https://manpages.ubuntu.com/manpages/jammy/en/man8/tracepath.8.html]

[10] Network MTU: [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/network_mtu.html]

[11] iperf(1) man page: [https://manpages.ubuntu.com/manpages/focal/en/man1/iperf.1.html]

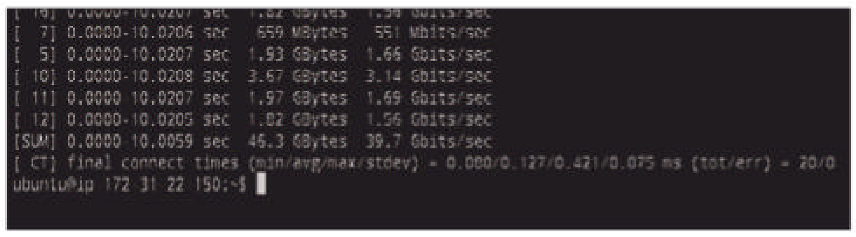
The Author

Federico Lucifredi (@0xf2) is the Product Management Director for Ceph Storage at IBM and Red Hat, formerly the Ubuntu Server Product Manager at Canonical, and the Linux "Systems Management Czar" at SUSE. He enjoys arcane hardware issues and shell-scripting mysteries and takes his McFlurry shaken, not stirred. You can read more from him in the O'Reilly title *AWS System Administration*.



```
[ 39]   0.00-10.03  sec  1.98 GBytes  1.69 Gbits/sec                  receiver
[ 41]   0.00-10.00  sec  1.98 GBytes  1.70 Gbits/sec    0             sender
[ 41]   0.00-10.03  sec  1.98 GBytes  1.69 Gbits/sec                  receiver
[ 43]   0.00-10.00  sec  1.98 GBytes  1.70 Gbits/sec    2             sender
[ 43]   0.00-10.03  sec  1.98 GBytes  1.69 Gbits/sec                  receiver
[SUM]   0.00-10.00  sec  39.6 GBytes  34.0 Gbits/sec    5             sender
[SUM]   0.00-10.03  sec  39.5 GBytes  33.9 Gbits/sec                  receiver

iperf Done.
```

Figure 8: Using 20 parallel connections gets closer to the network link's capacity.



```
[  7] 0.0000-10.0206 sec   659 MBytes   551 Mbits/sec
[  5] 0.0000-10.0207 sec  1.93 GBytes  1.66 Gbits/sec
[ 10] 0.0000-10.0208 sec  3.67 GBytes  3.14 Gbits/sec
[ 11] 0.0000 10.0207 sec  1.97 GBytes  1.69 Gbits/sec
[ 12] 0.0000-10.0205 sec  1.82 GBytes  1.56 Gbits/sec
[SUM] 0.0000 10.0059 sec  46.3 GBytes  39.7 Gbits/sec
[ CT] final connect times (min/avg/max/stdev) - 0.000/0.127/0.421/0.075 ms (tot/err) - 20/0
ubuntu@ip 172 31 22 150:~$
```

Figure 9: A multithreaded version of the benchmark overcomes the CPU-bound network performance.

■

ADMIN Network & Security NEWSSTAND

Order online:
bit.ly/ADMIN-Newsstand

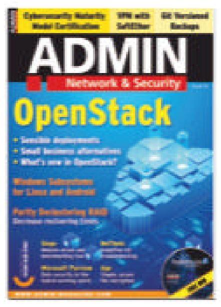
ADMIN is your source for technical solutions to real-world problems. Every issue is packed with practical articles on the topics you need, such as: security, cloud computing, DevOps, HPC, storage, and more! Explore our full catalog of back issues for specific topics or to complete your collection.

#72- November/December 2022

OpenStack

Find out whether the much evolved OpenStack is right for your private cloud.

On the DVD: Fedora 36 Server Edition.



#71 - September/October 2022

Kubernetes

We show you how to get started with Kubernetes, and users share their insights into the container manager.

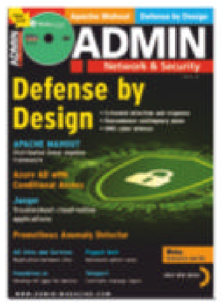
On the DVD: SystemRescue 9.04

#70 - July/August 2022

Defense by Design

Nothing is so true in IT as "Prevention is better than the cure." We look at three ways to prepare for battle.

On the DVD: Rocky Linux 9 (x86_64)



#69 - May/June 2022

Terraform

After nearly 10 years of work on Terraform, the HashiCorp team delivers the 1.0 version of the cloud automation tool.

On the DVD: Ubuntu 22.04 "Jammy Jellyfish" LTS server Edition

#68 - March/April 2022

Automation in the Enterprise

Automation in the enterprise extends to remote maintenance, cloud orchestration, and network hardware

On the DVD: AlmaLinux 8.5 (minimal)

#67 - January/February 2022

systemd Security

This issue, we look at how to secure systemd services and its associated components.

On the DVD: Fedora 35 Server (Install)

WRITE FOR US

Admin: Network and Security is looking for good, practical articles on system administration topics. We love to hear from IT professionals who have discovered innovative tools or techniques for solving real-world problems.

Tell us about your favorite:
- interoperability solutions
- practical tools for cloud environments
- security problems and how you solved them
- ingenious custom scripts
- unheralded open source utilities
- Windows networking techniques that aren't explained (or aren't explained well) in the standard documentation.

We need concrete, fully developed solutions: installation steps, configuration files, examples – we are looking for a complete discussion, not just a "hot tip" that leaves the details to the reader.

If you have an idea for an article, send a 1-2 paragraph proposal describing your topic to: *edit@admin-magazine.com*.

Authors

Amber Ankerholz	6
Gregor Bauer	10
Mark Heitbrink	70
Ken Hess	3
Thomas Joos	56, 66
Ankur Kumar	36
Martin Kuppinger	62
Federico Lucifredi	93
Martin Gerhard Loschwitz	24
Franck Pachot	30
Thorsten Scherf	88
Christian Schulenburg	82
Evgenij Smirnov	46
Guido Söldner	14
Andreas Stolzenberger	18
Nico Thiemer	52
Matthias Wübbeling	79, 90

Contact Info

Editor in Chief
Joe Casad, jcasad@linuxnewmedia.com

Managing Editors
Rita L Sooby, rsooby@linuxnewmedia.com
Lori White, lwhite@linuxnewmedia.com

Senior Editor
Ken Hess

Localization & Translation
Ian Travis

News Editor
Amber Ankerholz

Copy Editors
Amy Pettle, Aubrey Vaughn

Layout
Dena Friesen, Lori White

Cover Design
Lori White, Illustration based on graphics by Daniil Peshkov, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Publisher
Brian Osborn

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

Printed in Nuremberg, Germany by Zeitfracht GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN is published bimonthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA (Print ISSN: 2045-0702, Online ISSN: 2831-9583). January/February 2023. Periodicals Postage paid at Lawrence, KS. Ride-Along Enclosed. POSTMASTER: Please send address changes to ADMIN, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

CLOUDFEST

March 21–23, 2023
Europa–Park, Germany

6,000+ Participants
250+ Speakers

150+ Partners
65 Countries

The world's largest cloud industry event is ready to once again take over a spectacular European amusement park to facilitate new partnerships, deep knowledge sharing, and the best parties the industry has ever seen.

Start your CloudFest Journey
AND SAVE €399!
With your FREE Code: CF23ADMIN

scan me!

reg.cloudfest.com

```
// TUXEDO PULSE 15 - Gen2
//////////////////////////
document.addEventListener("pulse15-gen2_market-launc
    myPurchaseDecision();
});

var highres_screen = true;
var highly_efficient_AMD_Ryzen = true;
var huge_battery = true;
var quiet_cooling = true;
                    = true;

                    () {
                    ighly_efficient_AMD_Ryzen
                    -Gen2.RIGHT_NOW!

                    ers! GET IT NOW!"
```

Qui(e)te powerful!

TUXEDO Pulse 15 - Gen2

AMD Ryzen 7 5700U-35W
8 cores | 16 threads

WQHD display
2560 x 1440 | 165 Hz

Up to 18 h of runtime
91 Wh battery

Rigid magnesium chassis
1,7 cm thin | 1,5 kg light

100% Linux

5 Year Warranty

Lifetime Support

Built in Germany

German Privacy

Local Support

TUXEDO

tuxedocomputers.com