

ADMIN

Network & Security

ISSUE 84

Non-Relational
Databases

NoSQL, vector, and key-value stores

Sizing Monster VMs

Rules for performance-hungry VMs

Ansible Automation-as-Code

SQL Server Replication

Migrating from SQL Server to Azure SQL

Reducing the Windows Attack Surface

AIDE

File integrity checks
discover attackers' tracks

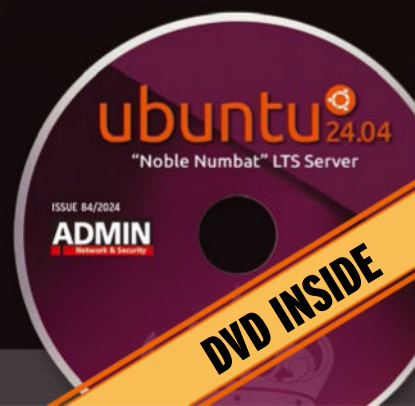
Gitea

Migrate your Git
repositories to Gitea

Raspberry Pi Storage

MicroSD, USB, and PCIe
speed tests

NVMe over Fabrics

High-redundancy
storage gears up to
replace iSCSILINUX NEW MEDIA
The Pulse of Open Source



Notebooks for every need

From ultra-portable to desktop replacement



Whether you need an ultra-portable for on-the-go use or a powerful desktop replacement, TUXEDO has the perfect notebook for every need. Our devices are specifically optimized for seamless Linux performance and can be customized to your specifications.

Whether for professional demands, personal use, or gaming, TUXEDO provides top-notch hardware tailored to your needs. Discover our wide range in our online shop and experience Linux at the highest level!



Dealing with Job Burnout

Job burnout might be part of a larger mental or physical health problem.

I first wrote about job burnout five years ago in 2019 [1]. I offered some valid advice, and the bottom line was, “Work to live rather than live to work.” This advice still stands. Compartmentalizing your life preserves your mental and physical health, but it’s not enough, and five years later, I’ve realized the shortcomings of that simplistic analysis. Burnout is far more complex than I ever imagined and dealing with it alone has given me new insights that I hope will also help you.

Burnout has many origins: stress, repetitive tasks, the feeling of no end in sight, the need to keep your technical skills sharp, negative feedback, no feedback, low or no salary incentives, no job ladder or promotions, and the feeling that people see you as a one-dimensional nerd who lives to solve problems.

First, you must realize and affirm that what you do is not who you are. Your job is a job and not a personality trait or disorder, as many believe. You are not your job. The job is a thing you do. When you get up from your desk, leave your job and the problems there. Your quality of life will improve if you separate your work from your non-work existence. Identifying yourself as your job will lead to burnout, depression, and other mental and physical ailments.

Second, you’re never done, no matter how well you do your job. Tomorrow brings more outages, failures, mistakes, errors, and problems to solve. Solve today’s issues and realize that even if you were to work 24x7x365, you’d never solve every problem. Do your best and be content with accomplishing everything you can on your shift. Working too many hours leads to mental and physical disorders that may manifest themselves as job burnout.

Third, you’re only one person – a single cog in the great corporate wheel. You’re not superhuman, and you have your role to play. You can’t solve every problem for everyone. Give help when asked, but remember that you have your own goals and tasks to accomplish. Don’t be the person who does everyone else’s job except yours. Your reviews will be poor, although you feel you’ve achieved a great deal and contributed tremendously. Poor job reviews add to burnout and harm your self-esteem and overall mental health.

Fourth, if you’re not the company owner, you are not the owner of the problems. You are hired to do a job. The company doesn’t own you or 100 percent of your time. You’re paid to perform tasks while you’re at work. As someone once told me, “Don’t trade your life for this job; it’s not worth it.” No truer words were ever spoken to me. Of course, being young and “motivated,” I ignored them. I gained nothing by being obsessed with that job. Ultimately, the owners sold the business to a larger corporation, and I became less significant and less relevant than I already was. My only accomplishment was to alienate the person who gave me that advice. Being obsessed with work is exhausting and will cause relationship problems and exacerbate your job burnout.

Fifth, and finally, take care of yourself. Take your breaks, especially your lunch break. Do *not* eat at your desk. Leave your desk and your work setting, if possible. Staying in one place too long leads to burnout. Get up from your desk during the day to take a short walk, collaborate with a colleague in person rather than by instant message or headset, or go to the gym and spend some time on the treadmill. Yes, I know that’s boring, but it’s a change of pace (no pun intended). Protect your physical health as well as your mental health. Plus, doing something physical will help your mental health. Be sure to get some sun because vitamin D deficiency is linked to an increased risk of mental health conditions such as depression, anxiety, ADHD, and suicide [2]. Studies have shown that vitamin D supplementation can improve depression and other mental health symptoms. Burnout and depression are often symptoms of a vitamin D deficiency.

You can’t always prevent job burnout. We’re all susceptible to it. Change is often the great healer, and please realize that job burnout might be part of a larger mental or physical health problem. If you’re experiencing burnout and exhaustion, visit your doctor or a counselor. Just like the computers you support, you need regular maintenance.

Ken Hess • Senior ADMIN Editor

Info

[1] “Dealing with IT Burnout” by Jeff Layton, *ADMIN*, issue 50, 2019

[<https://www.admin-magazine.com/Archive/2019/50/Dealing-with-IT-Burnout>]

[2] “Mental Health In The Sun” by James Greenblatt, MD, *Psychiatry Redefined*, October 2024:

[<https://www.psychiatryredefined.org/mental-health-in-the-sun-the-role-of-vitamin-d-deficiency-in-mental-illness/>]

ADMIN

Network & Security

Features

- 10 Vector Databases in AI**
Artificial intelligence requires special databases that rely on similarity searches for more effective processing of diverse data and embeddings to improve scalability, flexibility, and interpretability.
- 14 MongoDB**
The basic concepts, features, and applications of MongoDB emphasize the advantages offered by this established NoSQL database.
- 20 Qdrant**
Vector databases are one of the critical components for AI-based applications such as ChatGPT that rely on vector search queries to extract knowledge context from vast amounts of data.
- 26 Why NoSQL**
Tables are an established format for storing information in databases, but they don't always fit the bill - which is just one argument of many that speaks for NoSQL.

Service

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

Tools

- 30 NVMe over Fabrics**
NVMe-oF promises high-redundancy centralized storage with simple maintenance and management and almost twice the speed of its antiquated iSCSI predecessor.
- 36 From Confluence to BlueSpice**
With Atlassian's announcement that they are moving to a cloud-only solution, many organizations will be looking for an alternative that lets them keep their collaboration data local. BlueSpice is a worthy open source alternative with easy-to-use migration tools
- 42 iSCSI with Synology DSM**
The iSCSI protocol lets you access block storage across a network connection. We show you how to connect a Debian 12 system with a Synology storage device over iSCSI.
- 50 Automated Deployments**
OpenTofu and cloud-init help you deploy virtual machines in a Proxmox hypervisor and populate them automatically with services.

Containers and Virtualization

- 58 SQL Server Replication**
Wherever Microsoft SQL Server runs on local networks, individual or all databases can be migrated to Azure SQL by transactional replication. Various opportunities unfold, including analytics in the Azure cloud and global access routes for mobile users and home and branch offices.
- 64 Sizing Monster VMs**
Massive, performance-hungry VMs require proper handling to meet their dynamic requirements. We give you some rules to help size these monsters properly.

Security

- 70 AIDE**
If an attacker gains access to systems by working around your defenses, you need to discover their tracks in good time, at least to mitigate the further risk of damage, by monitoring changes to files.
- 72 Reduce Windows Attack Surface**
The sum total of all possible points of attack can be defined as the attack surface, and you need to take every opportunity to minimize it to the extent possible. Windows has built-in rules that minimize the attack surface; they simply need to be enabled.

10, 14, 20, 26, 85 | Non-Relational Databases

NoSQL, vector, and key-value stores

NoSQL databases manage data unsuitable for table-based relational databases and deliver efficiencies in scaling and high availability.

Highlights

10 Vector Databases in AI

Vector databases mark a paradigm shift in data technology for natural language processing, computer vision, recommendation systems, and other areas that require semantic understanding and data matching.

58 SQL Server Replication

Replication between on-premises SQL Server instances and Azure SQL offers a number of strategic benefits for organizations, including improved data analytics capabilities in the cloud, global access for users, and increased availability through cloud-based databases.

70 AIDE

Changes to binary data and configurations are artifacts of cyberattacks. If you notice them in good time, you might still be able to stop an attacker and prevent the damage spreading across your network.

Management

76 Ansible Automation

The Ansible automation tool makes it really easy to implement IT scenarios as code. We use structured YAML code to roll out Ansible in the form of AWX.

Nuts and Bolts

82 Gitea

Nothing is forever, not even a Git server. After the purchase of GitHub by Microsoft, I found a new home in Gitea for version control.

85 Key-Value Stores

Relational databases have been long-lived, but a completely different type of database, the key-value store, has established itself in the cloud market.

91 Perf Dojo

We test microSD, USB, and (especially) PCIe storage to push Raspberry Pi storage to the max.

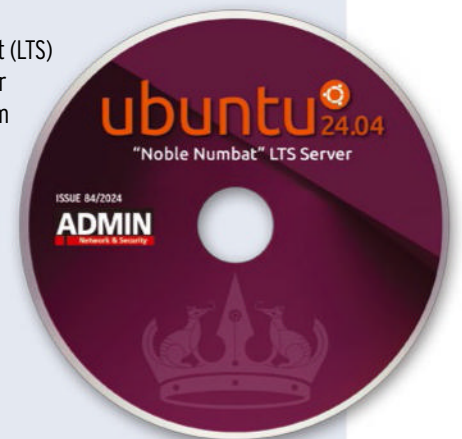
On the DVD

Ubuntu 24.04 LTS Server "Noble Numbat"

Noble Numbat is the 10th Ubuntu Long Term Support (LTS) release. The version included here is a Live server install image. In addition to Linux kernel v6.8 from March 2024, Noble Numbat comes with a number of updates to essential packages, including:

- Apache2 (v2.4.58)
- ClamAV anti-virus toolkit (v1.0.0)
- cloud-init (v24.1.3)
- containerd (v1.7.12)
- docker.io (v24.0.7)
- HAProxy (v2.8.5)
- libvirt (v10.0.0)
- and others

The new release strives to set "a new standard in performance engineering, enterprise security, and developer experience."



@admin-magazine.bsky.social



@adminmagazine



@adminmag



ADMIN magazine



@adminmagazine

News for Admins

Tech News

Open Source AI Definition Now Available

Version 1.0 of the Open Source AI Definition (OSAID – <https://opensource.org/ai>) is now available.

The definition, which was developed through a year-long, global, community process led by the Open Source Initiative (OSI – <https://opensource.org/>), defines an Open Source AI as “an AI system made available under terms and in a way that grant the freedoms to”:

- **Use** the system for any purpose and without having to ask for permission.
- **Study** how the system works and inspect its components.
- **Modify** the system for any purpose, including to change its output.
- **Share** the system for others to use with or without modifications, for any purpose.

The OSAID “offers a standard by which community-led, open and public evaluations will be conducted to validate whether or not an AI system can be deemed Open Source AI,” the announcement states (<https://opensource.org/blog/the-open-source-initiative-announces-the-release-of-the-industrys-first-open-source-ai-definition>).

“The new definition requires Open Source models to provide enough information about their training data so that a ‘skilled person can recreate a substantially equivalent system using the same or similar data’, which goes further than what many proprietary or ostensibly Open Source models do today,” said Ayah Bdeir, who leads AI strategy at Mozilla.

Read the full text of the definition at OSI: <https://opensource.org/ai>.

Sysdig Report Highlights LLMjacking and Other Security Threats

Sysdig has released its “2024 Global Threat Report,” outlining the changing nature of cyberattacks in 2024 (<https://sysdig.com/blog/sysdig-2024-global-threat-report/>).

For example, Sysdig describes the growing practice of LLMjacking, which involves stealing access to cloud accounts that host large language models (LLMs). “Mirroring earlier cryptojacking and proxyjacking techniques, LLMjacking presents an even more formidable financial threat.” In the six months since Sysdig first identified LLMjacking, the daily costs to victims have roughly doubled from \$46,000 per day to more than \$100,000 daily, the report notes.

Sysdig notes that whereas cryptomining attacks are fairly easy to identify based on CPU resource consumption, LLM usage cannot be detected this way because “there is only one behavior — a call to the LLM.” Additionally, “LLM resource consumption will vary greatly across individual users and, therefore, it is difficult to differentiate between legitimate and malicious use.” Thus, the report emphasizes the importance of establishing baselines for your enterprise cloud account LLM usage, so teams can readily identify anomalies.

This report follows the company’s “2024 Cloud-Native Security and Usage Report” (<https://sysdig.com/s-2024-cloud-native-security-and-usage-report/>) from earlier this year, which highlighted other security threats and outlined best practices. For example, the report says “less than 50% of environments have alerts set to trigger on CPU and memory use. Furthermore, a majority of users do not have maximum limits set on their CPU or memory use.” Without such limits, organizations risk having to pay for resources used by attackers in their environment.

Sysdig notes that, “with the ease of scalability in cloud environments, some campaigns can rack up \$80,000 in victim costs in just a few hours.”

Learn more at Sysdig: <https://sysdig.com/>.



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

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

Microsoft Releases OpenHCL, an Open Source Paravisor

Microsoft recently released OpenHCL, a new virtualization layer that provides guest virtual machines (VMs) with accelerated I/O and added security features.

This lightweight paravisor was written in Rust and “designed with strong memory safety principles,” says Hari Pulapaka in an introductory blog post (<https://techcommunity.microsoft.com/t5/windows-os-platform-blog/openhcl-evolving-azure-s-virtualization-model/ba-p/4248345>).

A paravisor, explains Caroline Perez-Vargas (<https://techcommunity.microsoft.com/t5/windows-os-platform-blog/openhcl-the-new-open-source-paravisor/ba-p/4273172>), “executes within the confidential trust boundary and provides the virtualization and device services needed by a general-purpose operating system (OS), enabling existing VM workloads to execute securely.” In other words, it’s “essentially an execution environment that runs within the guest VM — at a higher privilege level than the guest OS — and provides various services to the guest.”

OpenHCL runs on both x86-64 and Arm64 platforms and offers services to both confidential and non-confidential VMs, including:

- Device emulation via standard device interfaces
- Device translation via standard device interfaces, such as NVMe to para-virtualized SCSI, allowing assignment of hardware devices directly to VMs (accelerated I/O) without requiring guest OS changes
- Diagnostics support, which facilitates debugging confidential VMs
- Support for guests that are not fully enlightened — such as Windows and older versions of Linux — to run on confidential computing platforms via standard architectural interfaces (for confidential VMs specifically)

The company has also released a VM monitor component of OpenHCL, called OpenVMM. This modular, cross-platform tool, written in Rust, is available on GitHub (<https://github.com/microsoft/openvmm>) under the MIT license and supports a variety of host operating systems, architectures, and virtualization back ends.

Read more about OpenHCL and OpenVMM: <https://techcommunity.microsoft.com/t5/windows-os-platform-blog/openhcl-the-new-open-source-paravisor/ba-p/4273172>.

NASA Moves Forward with Lunar Time Zone

NASA plans to establish a Coordinated Lunar Time (LTC), which will “enable a future lunar ecosystem that could be scalable to other locations in our solar system.”

The agency will coordinate with government stakeholders, partners, and international standards organizations to “establish a time standard at the Moon to ensure the critical time difference does not affect the safety of future explorers. The approach to time systems will also be scalable for Mars and other celestial bodies throughout our solar system, enabling long-duration exploration,” the announcement says.

“As the commercial space industry grows and more nations are active at the Moon, there is a greater need for time standardization. A shared definition of time is an important part of safe, resilient, and sustainable operations,” said Dr. Ben Ashman, navigation lead for lunar relay development, part of NASA’s Space Communications and Navigation (SCaN) program (<https://www.nasa.gov/directorates/space-operations/space-communications-and-navigation-scan-program/>).

This development follows a similar announcement last year (<https://www.fosslife.org/european-space-agency-considering-lunar-time-zone>) by the European Space Agency (<https://www.esa.int/>).

Learn more at NASA: <https://www.nasa.gov/solar-system/moon/nasa-to-develop-lunar-time-standard-for-exploration-initiatives/>.

Open Source Malware on the Rise, According to Sonatype Report

Malware has infiltrated open source ecosystems at an alarming rate, according to Sonatype’s 10th annual State of the Software Supply Chain (<https://www.sonatype.com/state-of-the-software-supply-chain/introduction>).

According to the survey report, more than “512,847 malicious packages have been logged just in the past year, a 156% increase year-over-year,” highlighting the need for organizations to better understand and adapt their use of open source software.

Organizations continue to struggle with persistent vulnerabilities, such as Log4j (<https://www.sonatype.com/resources/log4j-vulnerability-resource-center>), with 13 percent of downloads remaining vulnerable three years after the issue was first exposed, the report says. Specifically:

- 80 percent of application dependencies remain un-upgraded for more than a year, even though 95 percent of the vulnerable versions have safer alternatives available.
- 3.6 percent of dependencies are still vulnerable because they were updated to another insecure version.
- Reliance on end-of-life (EOL) components, which no longer receive updates, leads to the gradual breakdown of software integrity.

The report also states that “reducing persistent risk is possible by focusing on tools that help manage dependencies and apply real-time vulnerability detection.” For example, “projects using a Software Bill of Materials (SBOM – <https://www.fossliife.org/how-sboms-strengthen-software-supply-chain>) to manage OSS dependencies showed a 264-day reduction in mean time to remediate (MTTR) compared to those that did not.”

Read more at Sonatype: <https://www.sonatype.com/state-of-the-software-supply-chain/introduction>.

Six Principles of Operational Technology Cybersecurity Released

The National Security Agency (NSA), along with the Australian Signals Directorate’s Australian Cyber Security Centre, Cybersecurity and Infrastructure Security Agency (CISA), and other government organizations, have released six general Principles of Operational Technology Cyber Security (https://media.defense.gov/2024/Oct/01/2003556960/-1/-1/0/PRINCIPLES_OF_OPERATIONAL_TECHNOLOGY_CYBER_SECURITY.PDF).

The principles, which are aimed at securing operational technology (OT) environments in critical infrastructure, are as follows:

1. Safety is paramount.
2. Knowledge of the business is crucial.
3. OT data is extremely valuable and needs to be protected.
4. Segment and segregate OT from all other networks.
5. The supply chain must be secure.
6. People are essential for OT cybersecurity.

The document includes examples and explores implications related to each of these principles to help OT professionals effectively develop processes and prioritize actions. For example, in regard to supply chain security, the guidelines note that:

“Some control systems protocols communicate via multicast or broadcast messages, which are sent to all devices on the network. As such, almost any device on the network may be able to view critical control messages and could create and inject messages to cause an undesirable action, making the supply chain of all devices critical.”

Read more at NSA: <https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/3923574/nsa-joins-australian-signals-directorate-and-others-in-promoting-six-principles/>.

New Password Rules Recommended by NIST

NIST has updated its Digital Identity Guidelines (<https://pages.nist.gov/800-63-4/sp800-63b.html#introduction>), which provides technical guidance for organizations to implement digital identity services and outlines requirements for credential service providers (CSPs) for remote user authentication at three different authentication assurance levels.

For example, the document includes updated guidelines regarding the complexity of passwords. These requirements state that verifiers and CSPs:

1. SHALL require passwords to be a minimum of eight characters in length and SHOULD require passwords to be a minimum of 15 characters in length.
2. SHOULD permit a maximum password length of at least 64 characters.
3. SHOULD accept all printing ASCII [RFC20 – <https://pages.nist.gov/800-63-4/sp800-63b.html#ref-RFC20>] characters and the space character in passwords.
4. SHOULD accept Unicode [ISO/ISC 10646 – <https://pages.nist.gov/800-63-4/sp800-63b.html#ref-ISOIEC10646>] characters in passwords.
5. SHALL NOT impose other composition rules (e.g., requiring mixtures of different character types) for passwords.

6. SHALL NOT require users to change passwords periodically. However, verifiers SHALL force a change if there is evidence of compromise.
7. SHALL NOT permit the subscriber to store a hint that is accessible to an unauthenticated claimant.
8. SHALL NOT prompt subscribers to use knowledge-based authentication (e.g., “What was the name of your first pet?”) or security questions when choosing passwords.
9. SHALL verify the entire submitted password (i.e., not truncate it).

The document notes that “length and complexity requirements beyond those recommended here significantly increase the difficulty of using passwords and increase user frustration.”

Other approaches, such as “blocklists, secure hashed storage, machine-generated random passwords, and rate limiting are more effective at preventing modern brute-force attacks, so no additional complexity requirements are imposed,” it states.

The comprehensive guidelines address many other authentication factors and detail both “process and technical requirements for meeting digital identity management assurance levels.”

Learn more at NIST: <https://www.nist.gov/>.

OpenSSH 9.9 Released

The OpenSSH team has released version 9.9 of the secure remote login tool with many updates and new features: <https://www.openssh.com/releasenotes.html>.

A significant feature of this release is support for a new post-quantum hybrid key exchange method. These key exchange methods are defined for use in the SSH Transport Layer Protocol in this IETF document (<https://datatracker.ietf.org/doc/html/draft-kampanakis-curdle-ssh-pq-ke-03#section-1-2>), which notes that “the cryptographic security of key exchanges relies on certain instances of the discrete logarithm problem being computationally infeasible to solve.” However, the capabilities of large quantum computers could render “the current key exchange and authentication methods in SSH insecure.”

To address this future problem, OpenSSH 9.9 “adds support for a new hybrid post-quantum key exchange based on the FIPS 203 Module-Lattice Key Encapsulation Mechanism (ML-KEM) combined with X25519 ECDH,” as described in the IETF document.

This release also disables the DSA signature algorithm by default at compile time. This move is a precursor to the team’s plans to remove support for the algorithm (<https://lwn.net/Articles/958048/>) altogether in early 2025. “DSA, as specified in the SSHv2 protocol, is inherently weak, being limited to a 160-bit private key and use of the SHA1 digest,” the announcement says.

Read about other features and bug fixes at OpenSSH: <https://www.openssh.com/>.

Docker Updates Usage Plans

Docker recently announced changes to its usage plans to include all-in-one subscription access to popular Docker tools.

These changes allow “development teams to access everything they need under one subscription with included consumption for each new product and the ability to add more as they need it,” says Giri Sreenivas in the announcement.

The new Docker Pro and Docker Team plans include updated pricing to reflect the value of newly added features. Specifically:

- Docker Pro will increase from \$5/month to \$9/month.
- Docker Team prices will increase from \$9/user/month to \$15/user/month (annual discounts).
- Docker Personal remains free.
- Docker Business pricing stays the same but includes new features.

Additionally, image pull and storage limits have been introduced for Docker Hub. “The new higher image pull limits will eliminate previously incurred fees for many Docker Team and Docker Business customers with Service Accounts,” the announcement says.

See the FAQ from Docker for more details: <https://www.docker.com/blog/november-2024-updated-plans-announcement/>.



Vector databases for data-driven artificial intelligence

New Dimensions

Artificial intelligence requires special databases that rely on similarity searches for more effective processing of diverse data and embeddings to improve scalability, flexibility, and interpretability. By Otto Geissler

To remain competitive, companies in all sectors need to incorporate the principle of data-driven decisions into their business logic, bearing in mind that data growth, especially the growth of unstructured data, is omnipresent. Market research institutes assume significant growth rates in the global market for data-driven artificial intelligence (AI).

AI helps structure and manage the flood of data in a meaningful way – not just in large corporations, but also in small and medium-sized enterprises (SMEs). Well-designed AI-based applications rapidly search through even the largest of datasets to gain new insights and ultimately open new business models capable of generating real added value for companies.

The World of Complex Data

Against the background of these challenges, vector databases open a new category of database management and a paradigm shift in the utilization of the exponentially growing volumes of unstructured data that remain unused in object stores.

Vector databases offer an amazing new range of opportunities, especially in searches for unstructured data; however, they can also process semi-structured and even structured data. Unstructured data (e.g., images, videos, audio, or user behavior) generally does not lend itself to approaches that use legacy database models, where “legacy” means databases that use B-tree and hash indexes, which are not ideal for storing and retrieving high-dimensional and complex data. These databases seemingly suffer from the “curse of dimensionality,” which makes them inefficient as the number of data dimensions increases.

Embeddings

Embeddings, generated by machine learning (ML) models to describe the essential features of data, overcome this curse of dimensionality, making it possible to represent complex data in a low-dimensional space and handle databases and AI algorithms. As soon as the data is represented by embeddings, it can be stored in databases capable of processing these

low-dimensional vectors in an efficient way. These databases are generally known as vector databases. Vector databases differ fundamentally from conventional databases because they support operations that are suitable for vector spaces, including, for example, vector similarity searches, nearest neighbor classification, and clustering (see the “Examples of Vector Databases” box). Queries in a vector database therefore differ from

Examples of Vector Databases

Weaviate [1] is an open source database that lets you store vector data objects and embeddings of favorite ML models and that seamlessly scales to billions of data objects. **Pinecone [2]** is specially developed for ML applications. It is fast and scalable and supports a variety of ML algorithms. Pinecone is based on Faiss, a library designed to find dense vector similarities efficiently. **Vespa [3]** is a fully featured search engine and vector database that supports approximate nearest neighbor (ANN) vector searches, lexical searches, and structured data searches, all in the same query. Through integrated model inference for ML, AI can be used to capture data in real time.

Lead Image © sdecoret, 123RF.com

conventional relational databases in that they rely on similarity searches instead of exact matches.

If you are looking for a document on a specific topic in a huge library with millions of text items and articles, conventional databases with exact match retrieval can quickly prove to be quite inefficient. However, a vector database equipped with embeddings would very easily be able to identify semantically similar documents, even if they do not contain the exact search term used for the topic. Moreover, vector databases enable faster, scalable, and more flexible data operations, which is crucial for the success of AI applications.

Vector databases store data as high-dimensional vectors (i.e., mathematical representations of features or attributes). Each vector has a certain number of dimensions, ranging from tens to thousands, depending on the complexity and granularity of the data. Vectors are usually generated by applying a transformation or embedding function to raw data (e.g., text, images, audio, video, etc.). The embedding function can be based on various methods, such as ML models, word embedding, and feature extraction algorithms.

Vector Database Benefits

Vector databases use a vector index to enable fast retrieval and insertion with a vector. Various advantages thus arise. In contrast to conventional relational databases (e.g., PostgreSQL), which store tabular data in rows and columns, or NoSQL databases, which store data in JSON documents, vector databases are designed to process data in the form of vector embeddings and nothing else. They can handle both structured and unstructured data and are therefore useful for a variety of applications, including text and image searches and recommendation systems.

As I mentioned earlier, vector databases manage high-dimensional data by using dimension reduction methods to compress high-dimensional

vectors into low-dimensional spaces while retaining important information, which makes them efficient in terms of memory and compute power requirements.

High-Dimensional Searches

The result is efficient similarity searches for high-dimensional vectors that are often used in ML and AI applications, such as image recognition, natural language processing, and recommendation systems. Vector databases use advanced indexing algorithms for high-speed search performance to enable fast retrieval of related vectors in vector space, even in very large datasets.

Vector databases are also designed to work seamlessly with ML algorithms that can be used to analyze the data stored in the database and that enable more sophisticated and accurate forecasts and real-time data processing. The databases scale horizontally, allowing large volumes of vector data to be stored and retrieved efficiently. Scalability is crucial for applications that need to search for and retrieve high volumes of data in real time. These databases are also ideal for large-scale ML applications because they can store and analyze billions of high-dimensional vectors.

Lack of Contextual Understanding

Vector databases and embeddings are very effective for certain tasks, such as finding similar documents or predicting the next word in a sentence; however, they are less suitable for answering insightful questions because they are not genuinely capable of understanding the context of text. For example, a vector database could recognize that certain words are semantically similar; however, it would not be able to understand general issues.

For several reasons, vector databases and embeddings are not sufficient to create a personalized AI assistant capable of answering questions of this

type, including contextual references. Although embeddings can handle semantic relationships, they have difficulty understanding the context of an issue, applying in particular to word embeddings, where each word has the same representation regardless of its context. Even more advanced technologies, such as embedding entire sentences or documents, have limitations when it comes to interpreting complex, nuanced meanings that depend on a larger context. Text embeddings are also not well suited to assessing long-term references or linking context across large blocks of text. As a result, they are unable to sift through decades of personal data and draw meaningful conclusions about life lessons, personal growth, or recurring themes. Embeddings are also typically trained on large, generic text bodies, which means that vector databases will not necessarily be able to interpret the unique language or experiences contained in personal or subjective text in a meaningful way. Understanding idioms, references, or unusual uses of language, which occur frequently, especially in personal diaries or email, is also problematic.

Specific Training

Various approaches can be used to create a personalized AI assistant with a better understanding of context and subjective dispositions. Instead of using pretrained embeddings, the AI model needs to be trained from scratch with a specific dataset, which would make it possible to learn your unique language usage, idioms, and personal experiences in a superior way. However, this approach requires more compute resources and, above all, a larger time budget.

For fine-tuning, a large language model (LLM) is trained by a text dataset that is relevant for the respective user, which allows the LLM to learn the vocabulary and writing style. When developing AI models, it also makes sense to analyze the text data with hierarchical models

with different levels of granularity (i.e., words, phrases, sentences, paragraphs, and documents) to help the AI assistant better understand the relationships between different parts of the text and derive more meaningful insights.

Graph-based techniques help connect and analyze entities, concepts, and relationships within your personal data so that AI assistants have a better chance to identify patterns, trends, and recurring themes in your experience. Another approach is the integration of other data types into the AI assistant, such as images, audio, or video. The use of different data formats could enable a more comprehensive understanding of your experience and allow the AI assistant to provide more accurate and meaningful insights.

The implementation of feedback loops is a further approach intended to support feedback on the AI assistant's answers so that it can learn from its mistakes and improve its understanding of your personal context over time.

When it comes to transfer learning, it makes sense to start with a basic model that has been trained on generic text data. Optimization can then take place on the basis of your specific dataset. This approach combines the advantages of a pretrained model with the benefits of customization. In this way, the AI assistant learns general language comprehension from the basic model and adapts it to your personal context. With these approaches, AI developers can work toward creating personalized AI assistants that provide deeper insights and a more meaningful understanding of your life experiences.

LLM Use Cases

The most important use cases for vector databases include LLMs, which are proving to be a disruptive force in AI, enabling machines to interpret and create human-sounding written works. These models are trained with huge volumes of

data and can estimate how likely it is that a word will be used according to the position in a phrase, which helps with tasks such as completing, translating, and summarizing texts.

The huge scope and complexity of these models translates to tough requirements in terms of handling and retrieving the high-dimensional data the models generate. Vector databases ideally support the operation of LLMs, thanks to their ability to manage high-dimensional data and perform fast similarity searches that provide a structured way to store and retrieve the vector embeddings created by these models, which allows them to search quickly for similarities in space with many dimensions.

The digital world is currently experiencing a boom in chatbots and virtual assistants. These AI-controlled units are highly dependent on understanding human language. Vector databases are crucial for natural language processing in terms of document similarity, sentiment analysis, and semantic search. They enable efficient indexing and retrieval of text material that is coded as word embeddings or sentence vectors.

Anomaly Detection

Detecting anomalies is just as important as identifying similarities. Users can compare various data points in network traffic with normal behavior patterns to identify anomalies by the distance to the typical vectors. Particularly in business sectors such as finance and security, detecting anomalies helps prevent fraud or potential security breaches.

Vector databases are also revolutionizing the performance of search engines. Conventional search engines rely on keyword matching that often fails to interpret the semantic meaning of the search query correctly. In contrast, vector databases support semantic searching by converting text into high-dimensional vectors that capture the semantic meaning of the text.

What this means is that the search engines deliver results that are semantically similar to the search query, even if they do not contain the exact keywords. For example, Google uses vector databases to improve its search results and provide more relevant and contextually accurate information.

Recommendation Systems for Customers

Vector databases also have a significant effect on recommendation systems. These systems often have to deal with high-dimensional data and find similar elements in a large dataset. Vector databases are ideal for this task because they quickly identify similar elements by the ANN search method. This function is particularly useful for systems that recommend products similar to those you have liked in the past. Companies such as Netflix and Amazon, for example, use vector databases to power their recommendation systems, resulting in more personalized and accurate recommendations.

Vector databases support clustering and classification by enabling fast, similarity-based groupings of data points. They can also be used for diagram analysis tasks such as community detection, link prediction, and diagram similarity matching, and they enable efficient storage and retrieval of diagram embeddings for better results. Last but not least, they improve the efficiency, scalability, and accuracy of these applications. They are particularly useful wherever analysis and comparison of high-dimensional data vectors is needed.

Vector databases are changing the way consumers shop in online stores. They make it possible to create advanced recommendation systems and curate personalized shopping experiences. For example, an online shopper would see product recommendations according not only to previous purchases, but also from the analysis of similarities in product attributes, as well as your behavior and preferences.

Complex Patterns in the Financial Sector

Today, the financial sector is determined by many complicated patterns and trends. Vector databases support precise analysis of this dense mass of data and help financial analysts identify even highly granular patterns in the markets with a good degree of accuracy, which is ultimately decisive for defining investment strategies. The ability to find subtle similarities or differences makes it possible to predict market movements and develop more informed investment plans.

Comparatively, personalization is of utmost importance in the health-care sector. By analyzing genomic sequences, vector databases can support more highly personalized treatments and ensure that medical solutions are better tailored to an individual's genetic makeup. From medical scans to surveillance images, the ability to compare and understand images accurately is crucial and is optimized by vector databases that focus on the essential features of images and filter out distracting elements such as noise and distortion. In traffic management, for example, images from video feeds can be quickly analyzed to optimize traffic flow and improve public safety.

Criteria for Selecting a Database

The following key factors need to be considered when choosing a vector database. For example, to investigate scalability in terms of the volume of data and the number of dimensions that a database can successfully handle, you must take into account performance metrics such as query response time and throughput to ensure that workload requirements are met.

The underlying data models and indexing methods can be equally crucial, as may support for flexible schema designs or a review of the indexing mechanisms that ensure efficient similarity search and retrieval processes. The most popular indexing strategies include tree-based structures, location-sensitive hashing (LSH), and ANN algorithms.

Simple setup, configuration, and maintenance of the vector database are decisive features in terms of usability. A user-friendly design and good documentation can curtail the learning curve. Checking the integration of the vector database with the existing systems, tools, and programming languages is also important and is essential to find out whether the vector database has APIs, connectors,

or SDKs that support integration. Good compatibility with popular frameworks and data processing tools guarantees a good user experience. Vibrant communities with various discussion forums often serve as a useful source of information and open up access to professional advice, and don't forget cost-benefit comparisons wherever the use of the vector database involves licensing or subscription fees.

Conclusions


Instead of traditional query methods that use exact matches or predefined criteria, a vector database finds the most similar or relevant data on the basis of the semantic or contextual meaning of the data. Vector databases are currently used in various areas and applications and mark a paradigm shift in data technology for natural language processing, computer vision, recommendation systems, and other areas that require semantic understanding and data matching. ■

Info

[1] Weaviate: [\[https://weaviate.io\]](https://weaviate.io)

[2] Pinecone: [\[https://www.pinecone.io\]](https://www.pinecone.io)

[3] Vespa: [\[https://vespa.ai\]](https://vespa.ai)



Introducing the NoSQL MongoDB database

Flexibility

This overview of the basic concepts, features, and applications of MongoDB emphasizes the advantages offered by this established NoSQL database. By Michael Höller

The MongoDB document database stores structured or unstructured data in a JSON-style format that can be mapped directly to native objects in most modern programming languages. It offers developers a modicum of convenience because they don't need to worry about normalizing data. MongoDB is also a good choice for large volumes of data and scales both vertically and horizontally. The company behind the database, MongoDB Inc., was founded in 2007 with the aim to develop a modern database that meets the requirements of agile, scalable applications. The first version was launched in 2009, and the database has seen continuous improvements and extensions ever since. Milestones include functions such as sharding for horizontal scaling, replica sets for high availability, and the on-going advancement of query functions and proprietary tools.

Variants

MongoDB comes in two primary variants:

- MongoDB Community Server or Enterprise Server
- MongoDB Atlas

The MongoDB Community Server is the open source version of the database that you can install and manage on your own servers. It provides all the basic database functionality, such as replication, sharding, and query processing. However, you are responsible for provisioning, scaling, safeguarding, and maintaining the database infrastructure.

In contrast, the extended feature set of MongoDB Enterprise Server appeals to those with a need for features that are primarily aimed at optimizing security, administration, and performance, including Kerberos and LDAP authentication, auditing and encryption at rest, management and monitoring tools, automated backups, the use of in-memory storage engines, and advanced compression algorithms.

MongoDB Atlas is the fully managed, cloud-based database-as-a-service (DBaaS) platform. Atlas automates management tasks (i.e., provisioning, scaling, updating, backup, and security configuration) and runs on the major cloud platforms (e.g., AWS, Azure, Google Cloud). In addition to the basic database functions, MongoDB Atlas includes advanced

features such as integrated data analytics tools, Atlas Search, Global Clusters for geographically distributed applications, and Vector Search for AI applications.

Data Model Benefits

MongoDB lets you store, manage, and retrieve data in your applications. The database currently supports 14 programming languages – C, C++, C#, Go, Java, Kotlin, Node.js, PHP, Python, Ruby, Rust, Scala, Swift, and TypeScript [1] – and uses a flexible document-oriented data model. MongoDB groups documents in collections. Each document in a collection can be different, such as having different fields or even different data types for the same field. The following example illustrates this: Assume you want a collection in MongoDB to store information about different types of vehicles. Each document in this collection has different fields, depending on the type of vehicle (Listing 1).

A vehicle of one type has fields that do not exist in the other types. For example, the Car dataset has fields such as `mileage` and `features` (lines 6

and 7), whereas `Motorcycle` has an `engineCapacity` field (line 14), and `Bicycle` has fields such as `frameMaterial` and `gears` (lines 21 and 22). The data types also differ. The `features` field in the car document is an array of strings, `engineCapacity` in the motorcycle document is a string, and `mileage` in the car document is an integer value.

Documents can also contain other documents. If you want to save the service history directly in the vehicle document, you can add an embedded document ([Listing 2](#)).

The flexible data model has a number of advantages. For one thing, it is highly adaptable. MongoDB lets you store different documents in the same collection, even if they have different fields and structures. This is particularly useful where data requirements change over time. Support for embedded documents means that related data can be stored together, which in turn simplifies data retrieval and management.

Schema changes are also easy: You don't need to define the schema in advance, which ensures fast and agile development. New fields can be added easily to existing documents without having to migrate the entire collection. Migration can also take place without downtime: With a combination of triggers and the flexible schema (schema versioning pattern), you can easily implement a schema migration without any downtime, even if the application is running on multiple instances. MongoDB also supports complex queries and indexes, even on embedded fields, which means you can carry out precise and efficient queries, even with flexible data structures.

Efficient Formats

In MongoDB, data records are stored as compressed binary JSON (BSON) documents [\[2\]](#). The documents themselves can be retrieved directly in the JSON format, which offers a number of advantages. BSON, as a binary format, for example, is specially designed to store and transfer

data efficiently. It contains length information and type specifications that ensure fast data processing.

In contrast to JSON, which is text-based and therefore requires more storage space, BSON reduces the size of the stored data through binary encoding.

BSON also supports a variety of data types that are not directly available in JSON, such as datetime, binary data, regular expressions, and 64-bit integers. The extended data types directly save memory space and speed up processing. Additionally, data can be converted to and from binary format more quickly. Data serialization and deserialization are also accelerated, which has a particularly positive effect in case of large data volumes or frequent data access.

BSON documents contain length specifications for fields and arrays, which makes parsing easier and enables quick access to the data, because the size of each element is known in advance. Furthermore, specific fields within a document can be found and processed quickly. Query performance also benefits because BSON supports indexes on embedded fields. An additional advantage is that BSON can be expanded to include new data types without affecting existing systems, which allows BSON to meet future requirements without compromising compatibility with older versions.

The following JSON example clearly shows the major advantages of the data format:

```
{ "name": "John Doe",
  "age": ` `55,
  "created_at": "2024-05-17T10:00:00Z" }
```

The corresponding BSON document would contain the following elements in binary code:

- A 32-bit integer for the length of the document
- A byte for the field types (e.g., string, integer, date)
- The names of the fields as strings,
- The values of the fields in binary form (e.g., UTF-8-encoded strings, 32-bit integers, binary date values)

Most developers find it easy to work with JSON because it is a simple and powerful method for describing data. By using BSON to store the data, you can leverage the benefits of the JSON format without sacrificing performance.

Another important point is the flexibility gains during development. Programmers have control over the database schema; they can customize the database schema without

Listing 1: Collection of Vehicles

```
01 {
02   "type": "Car",
03   "brand": "BMW",
04   "model": "X5",
05   "year": 2021,
06   "mileage": 15000,
07   "features": ["Sunroof", "Leather seats"]
08 }
09 {
10   "type": "Motorcycle",
11   "brand": "Harley-Davidson",
12   "model": "Iron 883",
13   "year": 2019,
14   "engineCapacity": "883cc"
15 }
16 {
17   "type": "Bicycle",
18   "brand": "Canyon",
19   "model": "Ultimate CF SLX",
20   "year": 2022,
21   "frameMaterial": "Carbon",
22   "gears": 22
23 }
```

Listing 2: Embedded Document

```
01 {
02   "type": "Car",
03   "brand": "BMW",
04   "model": "X5",
05   "year": 2021,
06   "mileage": 15000,
07   "features": ["Sunroof", "Leather seats"],
08   "serviceHistory": [
09     {
10       "date": "2023-01-15",
11       "serviceType": "Oil change",
12       "dealer": "BMW Service Center"
13     },
14     {
15       "date": "2023-06-20",
16       "serviceType": "Tire replacement",
17       "dealer": "Quick Tires"
18     }
19   ]
20 }
```


the help of a database administrator and reformat it as the application evolves. If needed, MongoDB can use schema validation to coordinate and control changes to the structure of documents. As always, these many opportunities dump a great deal of responsibility on the developer's shoulders. Developers need to be familiar and have experience with MongoDB schema design or seek external help.

Sharding

MongoDB implements horizontal scaling through sharding, wherein large volumes of data are distributed across several servers. In MongoDB, sharding is a central feature that improves database performance and capacity [3].

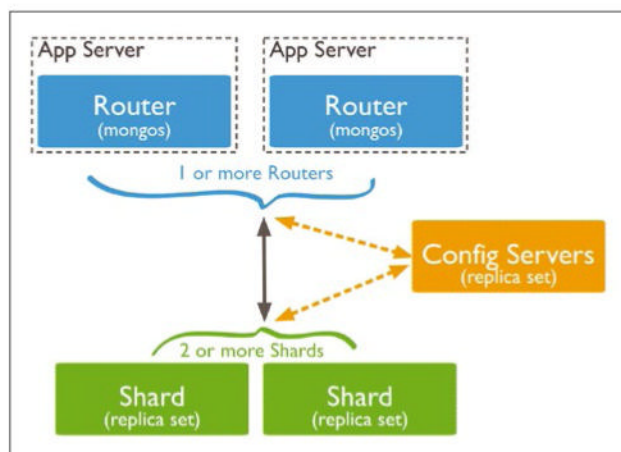


Figure 1: The dataset is distributed across shards; processes known as mongos route queries to the correct shard. The config server's metadata makes sure this happens. © MongoDB [3]

Listing 3: Sharding Configuration

```

### Set up shard cluster:
### - Launch multiple MongoDB instances as shards
### - Launch three Config Servers as a replica set
### - Launch one or more Mongos instances
### Example for one shard:
$ mongod --shardsvr --replSet shard1 --dbpath /data/shard1 --port 27018
$ mongod --configsvr --replSet configReplSet --dbpath /data/config1 --port 27019
$ mongos --configdb configReplSet/localhost:27019
### Activate sharding:
### - Connect to the Mongos instance
### - Add shards to the cluster
sh.addShard("shard1/localhost:27018")
sh.addShard("shard2/localhost:27018")
### Activate sharding for a database:
### - Activate sharding for the desired database
### - Define the sharding key for the collection
sh.enableSharding("mydatabase")
sh.shardCollection("mydatabase.mycollection", { "shardKeyField": 1 })

```

MongoDB refers to a single MongoDB server or a replication set that holds a subset of the data as a shard. The entirety of the shards over which the database is distributed is the sharded cluster, which is backed up by a special MongoDB server, the config server, which manages metadata through data distribution (Figure 1). Version 8.0 of MongoDB introduced embedded config servers, which means that the config server becomes part of a special shard known as the config shard. The MongoDB mongos processes act as query routers and forward queries to the relevant shards. They act as intermediaries that separate the application logic from the physical data distribution. The shard key is the name of one or

multiple fields in the documents that control the data distribution. A careful choice of the shard key is decisive in ensuring even load distribution. The data distributed by the shard key is broken down into logical units (chunks). Each chunk represents a range of shard key values. MongoDB monitors the distribution of the chunks across the storage media and moves the chunks, if necessary, to ensure balanced load distribution (Listing 3). Tools such as mongostat, mongotop, and the MongoDB Ops Manager application monitor the status and performance of the cluster. You can control the

distribution of chunks with MongoDB commands such as `sh.status()`, `sh.moveChunk()`, and `sh.splitAt()`. The balancer process also moves chunks automatically to ensure that they are distributed evenly. The balancer status can be checked with the `sh.getBalancerState()` command. Backups must take place at the shard and config server levels. Like any distributed database system, this task is not trivial on a system you manage yourself, but you can find some useful documentation [4] online and draw on training and external support for the initial setup of MongoDB.

High Availability

Replication is one of the core functions in MongoDB; it is used for high availability (HA), data integrity, and disaster recovery. If a user's data is stored on a single server, a server crash or network failure could prevent access [5]. The idea behind replication is to spread the data across several servers, significantly increasing data availability, reliability, and fault tolerance, which also means that the read load can be distributed across the members of the replica set [6]. MongoDB ensures high availability through replica sets (Listing 4) comprising a primary node and several secondary nodes (Figure 2). The primary node accepts write operations and handles replication to the secondary nodes. If the primary node fails, the replica set automatically selects a new primary node from the remaining nodes to ensure continuity of service. When the former primary node goes back online, it acts as a secondary server for the new primary node. The secondary nodes contain a copy of the primary node's data, accept read requests, and can be promoted to the primary node if required. An arbiter is a node that is involved in the election of the primary node without storing data itself. It helps keep an odd number of nodes so that a majority decision can be made. Arbiters help reduce costs because they require significantly fewer resources. However, you need to keep

in mind that the replication set in a primary-secondary-arbiter (PSA) architecture (Figure 3) has fewer servers storing data, which in turn affects reliability. You need to decide on a case-by-case basis whether a PSA architecture meets your needs for reliability or whether you would be better off replacing the arbiter with a full-fledged server.

Tools like `rs.status()` and the MongoDB Ops Manager dashboard help you monitor the health and performance of the replica set. MongoDB automatically detects the failure of the primary node and initiates a failover process (Figure 4) to appoint one of the secondary nodes as the new primary node. Replication in MongoDB is a robust feature that ensures high availability and data integrity. Given compliant configuration, management, and monitoring, MongoDB replica sets can provide reliable underpinnings for business-critical applications.

Replica Set Architecture

One special feature of a MongoDB cluster is that it does not require a cluster server or a virtual IP address. In simple terms, the cluster is the entirety of the nodes. Taking a look under the hood will give you a better understanding.

When a MongoDB client connects to a replica set through a single URL, the process involves multiple steps. To begin, the DNS resolver resolves the URL into several IP addresses (DNS seed list) that correspond to the servers in the replica set. The MongoDB driver then typically establishes a connection to the first IP address determined in this way. If the connection fails or times out, it tries the next IP address and so on until it finally succeeds. As soon as the connection is established, the MongoDB

client initiates a handshake with the server, by which the client discovers the entire topology of the replica set, including the primary and secondary nodes. The primary node processes write operations. If the primary node is not identified during the first connection, the client queries the other nodes until it finds the primary node. As soon as the primary node is identified, the client sends write operations directly to the primary node, which then replicates the write operations to the secondary nodes. The client never sends its commands to all nodes; instead, it forwards each write command specifically to the primary node, which then handles the task of replicating to the secondary nodes. This method ensures

Listing 4: Replication Example

```
### Set up replica set:
### - Configure nodes as part of the same replica set
### - Start MongoDB instances for each host
$ mongod --auth --replSet "rs0" --bind_ip localhost,<host name>|<IP address>
### Configure the replica set:
### - Connect to a node and initiate replica set
### - Add further nodes
rs.initiate({
  _id: "rs0",
  members: [
    { _id: 0, host: "mdb0.example.net:27017" },
    { _id: 1, host: "mdb1.example.net:27017" },
    { _id: 2, host: "mdb2.example.net:27017" }
  ]
})
```

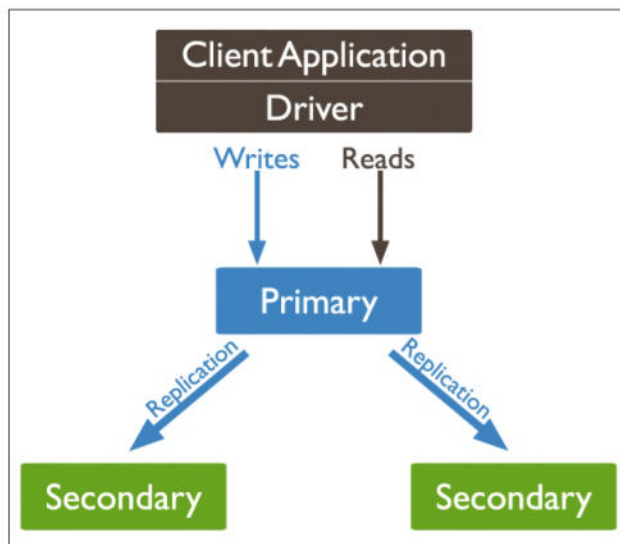


Figure 2: Architecture of a replica set with two secondary nodes (primary-secondary-secondary, PSS). © MongoDB [5]

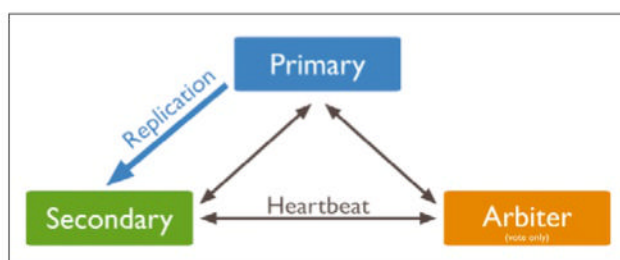


Figure 3: Architecture of a replica set with an arbiter (PSA). © MongoDB [5]

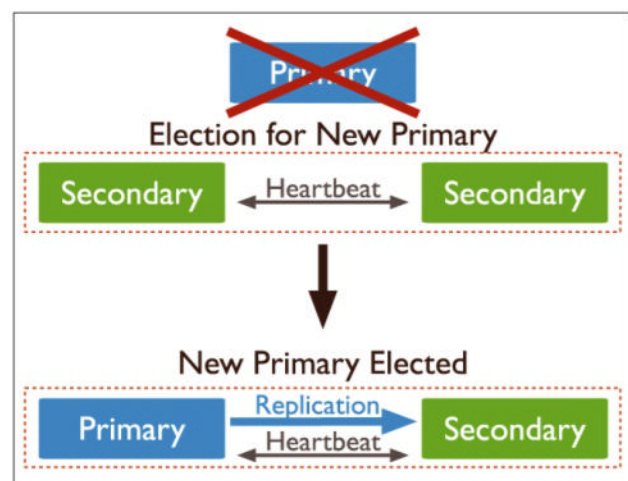


Figure 4: If a primary node fails, the automatic failover process starts. © MongoDB [7]

consistent and persistent write processes. Each member of the replica set, both the primary and the secondary nodes, participates in the topology recognition process. Depending on how you configured the read settings, secondary nodes can play a role in read operations.

If a primary node fails to communicate with the other members of the replica set for longer than the configured `electionTimeoutMillis` period (10 seconds by default), an authorized secondary node requests an election to nominate itself as a new primary element.

The replica set cannot process any write operations until the election has successfully completed. However, it can continue to process read queries during this wait – at least if the queries are configured such that secondary servers can execute them while the primary server is offline. It might therefore be advisable to set the read preference to `primary-preferred`. Ultimately, however, this is a decision to be made on a case-by-case basis.

MongoDB Atlas

So far I have looked at the core components of MongoDB server in the self-hosted version. In the case of MongoDB Atlas, the developers offer the database server as a cloud-based DBaaS platform [8].

Atlas automates management tasks such as provisioning HA clusters (georedundant, multiple providers), scaling, updates, backups (including distributed backups), automatic patching without downtime, and security configurations.

Atlas runs transparently on today's popular cloud platforms (AWS, Azure, and Google Cloud), which means you do not need a separate contract with a provider. In addition to the basic database functions, Atlas offers advanced functions,

such as data extraction and optimization (columnar storage) to isolate analytical workloads and perform business intelligence (BI) analyses; synchronization services between mobile devices and Atlas; authentication and authorization tasks; and conflict resolution. The offering also includes a single-node instance of MongoDB that runs locally in an Internet of Things scenario and handles synchronization between local devices, as well as bidirectional synchronization between the edge server and Atlas. A data API is also available that integrates data with REST endpoints over HTTPS. Atlas lets you export data in the Parquet, CSV, BSON, or Extended JSON formats, and you can merge data from a variety of sources with MongoDB (data federation).

The cloud version also offers full-text and semantic searches, stream processing, and vector searches for AI applications. Older data can be migrated to a more cost-effective storage level to optimize the cost-benefit ratio. A Performance Advisor is also available in this version for tuning to provide automated recommendations for performance optimization on the basis of query logs, index usage statistics, and database metadata.

Conclusions

MongoDB is a document-oriented NoSQL database that stores data in BSON and promises a flexible, schemaless data structure, which means developers can use dynamic and changing data models without having to modify the entire database structure.

From a technical point of view, MongoDB impresses with horizontal scalability by sharding, high availability, and reliability thanks to replication; support for various indexes, including geospatial and text indexes; support

for atomicity, consistency, isolation, and durability (ACID) transactions at the document and multidocument level; and powerful tools for data aggregation and analysis. Compared with other NoSQL databases, MongoDB offers a balanced mix of flexibility, ease of use, and performance for a variety of use cases. It is particularly suitable for fast-growing web applications, real-time analysis, and applications that require flexible data models. Although other NoSQL databases might be more efficient in specific areas, MongoDB appeals with a wide range of features that make it one of the most popular and versatile of its kind. ■

Info

- [1] Supported languages: [\[https://www.mongodb.com/docs/drivers/\]](https://www.mongodb.com/docs/drivers/)
 - [2] BSON and JSON: [\[https://www.mongodb.com/resources/basics/json-and-bson\]](https://www.mongodb.com/resources/basics/json-and-bson)
 - [3] Sharding: [\[https://www.mongodb.com/docs/manual/sharding\]](https://www.mongodb.com/docs/manual/sharding)
 - [4] MongoDB docs: [\[https://www.mongodb.com/docs/\]](https://www.mongodb.com/docs/)
 - [5] Replication: [\[https://www.mongodb.com/docs/manual/replication\]](https://www.mongodb.com/docs/manual/replication)
 - [6] Automatic failover process: [\[https://www.mongodb.com/docs/manual/tutorial/deploy-replica-set/\]](https://www.mongodb.com/docs/manual/tutorial/deploy-replica-set/)
 - [7] Replica set elections: <https://www.mongodb.com/docs/manual/core/replica-set-elections/>
 - [8] MongoDB Atlas: <https://www.mongodb.com/products/platform/atlas-database>
-

The Author

Michael Höller has worked in IT for more than 25 years. As an independent consultant and Scrum Master, he has primarily supported MongoDB-centric projects for more than 10 years. His focuses are schema design, performance optimization, security, site reliability engineering, support for paradigm shifts to flexible schemas, and project coordination. As a MongoDB Subject Matter Expert and MongoDB Champion, he helps the MongoDB University team create new certifications. ■

Hone Your Skills – with – Special Issues!

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

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

Available in print or digital format



background image © roystudio, 123RF.com



Check out the full library!
shop.linuxnewmedia.com

How vector databases work and when they're used

Vectors

Vector databases are one of the critical components for AI-based applications such as ChatGPT that rely on vector search queries to extract knowledge context from vast amounts of data. By David Myriel

Vector databases store data in a numerical format as vectors. If you look at an arbitrary group of words, such as “A blue striped cotton shirt,” the vector mapping would look something like:

```
[0.1, -0.2, 0.91, 0.7, -0.21, 0.1 7
-0.7, 0.4]
```

This numerical expression contains the coordinates of the words in a multidimensional space. When a sentence, an image, or a video is converted into a vector, the numerical

values stand for the meaning of the statement in question. In the example, the vectors of the word groups “A blue striped cotton shirt” and “A nautical cotton shirt” would point to locations close to each other in multidimensional space. The interesting thing here is that the vector database’s search algorithm processes the query and computes the distance between the two vectors that are connected in this way by measuring the similarity between vectors, which in turn generates meaning in many modern AI systems.

As a critical software infrastructure, a vector database is designed to store high-dimensional vectors in an efficient way. A traditional online transaction processing (OLTP) or online analytical processing (OLAP) database ([Figure 1](#)) organizes data in rows and columns. Queries relate to the values stored in these rows and columns. Some AI applications are based on such databases and attempt to store and retrieve data quickly. However, in some applications, such as those for image recognition or natural language processing and

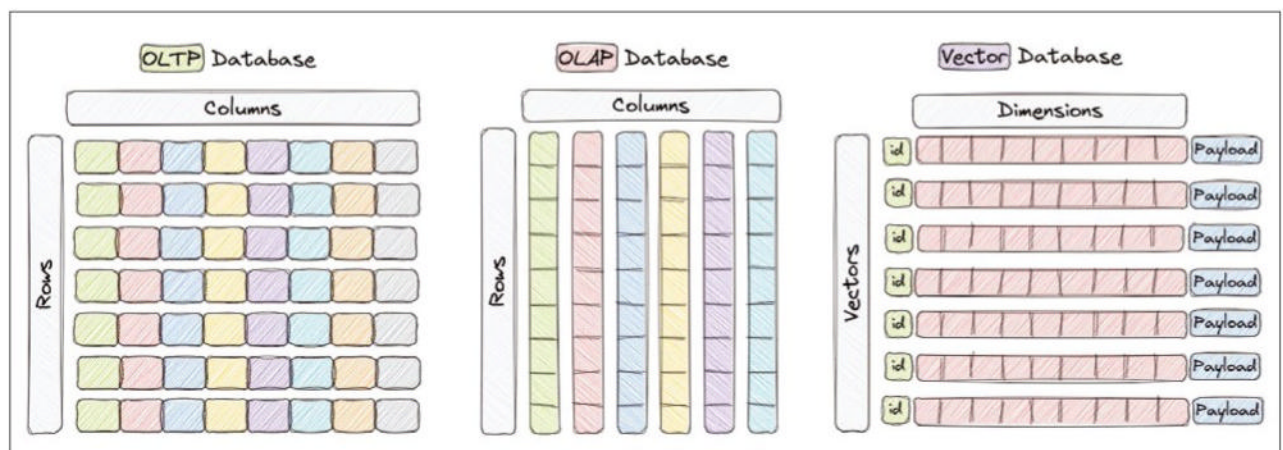


Figure 1: Vector databases do not use rows and columns but store their values in vectors, along with an ID and an attached payload. © Qdrant

Lead Image © neyro2008, 123rf.com

recommendation systems, the data is represented by vectors in a multidimensional space. A vector database stores collections of these vectors that are noted as spatial coordinates with an ID and an optional payload, which usually contains metadata that can be used to further filter the vector search for a more precise query.

Qdrant [1] is one example of an AI-compatible vector database. It can store and effectively search through huge amounts of data and deliver very accurate results at high speed. The results are typically semantically relevant to the question that is asked. As long as the data has been converted into vectors, it makes no difference whether you are looking at text, images, or audio material.

Why Vector Databases?

Vector databases play a particularly important role when it comes to searching for similarities, recommendation systems, content-based image searches, or personalized searches, for example. Thanks to their efficient indexing and search techniques, vector databases enable more accurate and faster searching in unstructured data that is already available as vectors. By doing so, they help determine the most relevant answers to a user's search queries.

If you add this capability in the context of an existing AI application, a vector database is used to retrieve context. In other words, it improves the quality of the information that AI models use to generate answers. For example, if you ask an AI model to generate a cool idea or piece of code,

a vector database can help it find inspiration faster and more accurately. The possibilities of a vector search are often the reason for saving the data as vectors in the first place, which has led to an increasing demand for vector databases in global tech companies that rely on retrieval systems to advance their AI projects. Databases are ideal in this case because of their focus on real-time storage.

How Vector Searches Work

Vector databases do not rely on conventional search methods, but use new, specialized algorithms for the search. For example, a stored three-dimensional vector could be assigned the values [1.4, 3.8, -0.8]. Each query to the database is also translated into a vector (e.g., [1.3, 3.5, -0.7]). The algorithm then tries to find out whether the stored vector is similar to the query – that is, whether the points in space that the vectors reference are close to each other. If not, the search continues until a related vector is identified. The process is known as a “similarity search” and takes just milliseconds, even with 2,000 and more dimensions and a billion entries.

In a vector search process, for example, a machine learning model uses an extensive text corpus as the input and generates a mathematical representation of all the words in the form of vectors. The vectors

are selected in such a way that words with a similar meaning and in a similar context are grouped together and lead to similar vectors. You can also compute the mean value of all the words in a query, sentence, or paragraph (Figure 2).

The process of vectorization cannot be reversed: You can no longer infer the wording of a paragraph from its vector, because its dimensions no longer indicate the presence of certain words. The stored vectors simply represent the meaning, but not the words contained in a text. On this basis, large language models (LLMs) can automatically handle synonyms. What's more, if the underlying neural networks have been trained with multilingual collections of training data (corpora), they can also translate, because the same sentence in different languages generates similar vectors (embeddings). Similar text passages can then be found across the language boundary by having the database compute the distance between the vectors (Figure 3).

Even if the entered queries contain words different from the answers, they are converted into similar vector representations, because the neural encoder can capture the meaning

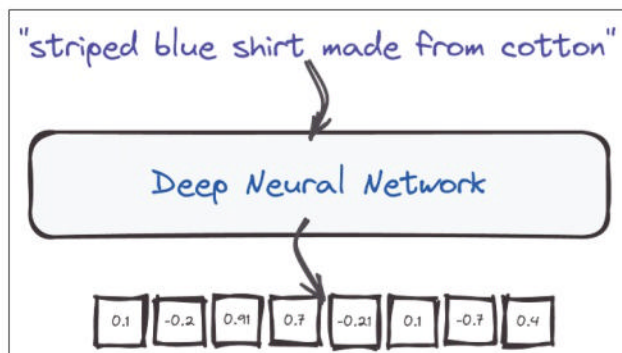


Figure 2: A neural network transforms words into vectors. © Qdrant [2]

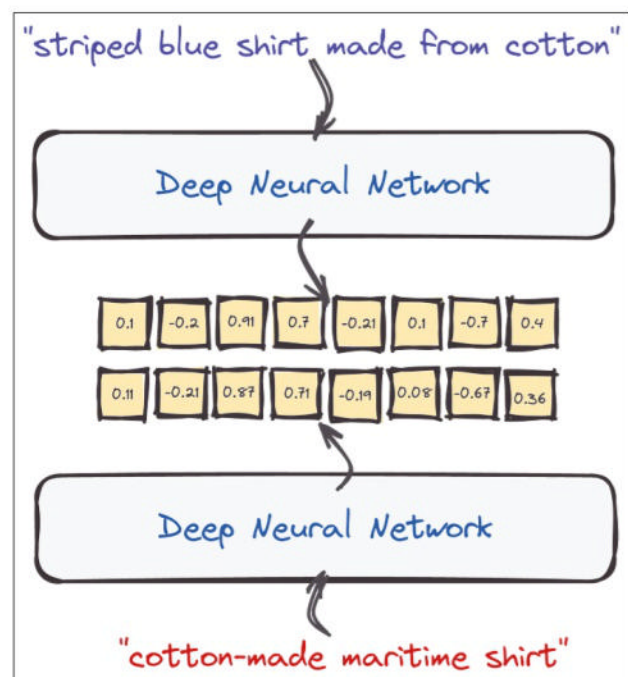


Figure 3: Phrases with a similar meaning result in similar vectors.

© Qdrant [2]

of the sentences. This function is compatible with synonyms and different languages. The entire process is known as a vector search and is based on finding similar objects according to the similarity of their embeddings.

The good thing is that you don't have to design your neural network yourself, nor do you have to train it. Many pre-trained models are available, either on Hugging Face [3] or with the help of libraries such as Sentence Transformers [4]. If you don't want to get your hands dirty with neural models, you can alternatively use a software as a service (SaaS) tool, such as the co.embed API [5], to create the embeddings.

Using Vector Databases

The challenge with vector searches begins with identifying similar documents in a very large set of objects. A naive approach would be to calculate the distances between every object to identify nearest neighbors. Although this calculation might be possible with dozens, and perhaps even hundreds, of examples, it would become a bottleneck with larger numbers. In the world of relational databases, you use indexes to speed things up – and vector databases do something similar. A full vector database shortens the search process with a graph-like structure used to find the nearest objects in sublinear time. In this case, it does not calculate the distance to

every object in the database, but only to some candidates.

The example in Figure 4 demonstrates how the vectors reference points in multidimensional space – although only three dimensions can be represented graphically. The vector search uses the approximate nearest neighbor (ANN) algorithm to find related entries. Figure 5

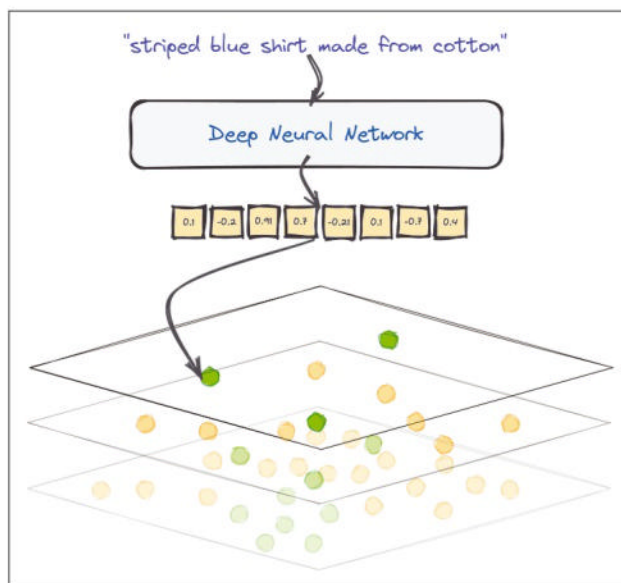


Figure 4: Each vector references a point in multidimensional space.

© Qdrant [6]

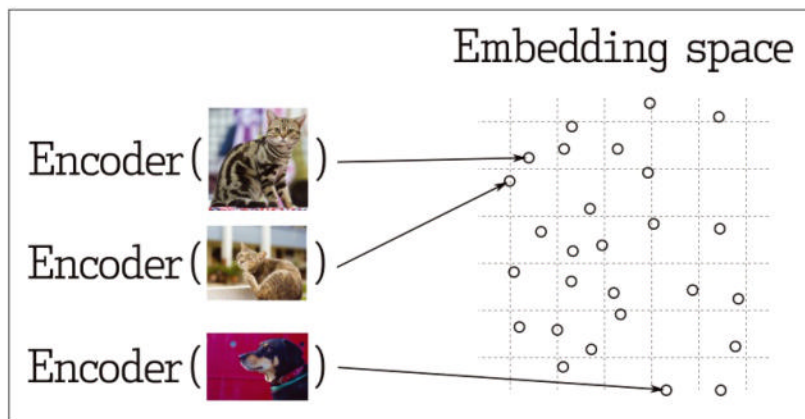


Figure 5: The encoder calculates vectors that refer to points closer together if the designated objects are more similar to each other. © Qdrant [7]

illustrates how the semantic relevance between objects (animals) can be expressed in a simple two-dimensional diagram. Objects that are similar by nature are closer together, whereas less closely related objects are further apart. You need a special tool for a large-scale semantic search. Open source vector databases are available free of charge. They are very popular with data scientists, AI developers, and machine learning engineers. Their great advantage is that they can be installed in any hosting environment, even on the most heavily secured servers, playing a very important role in projects with confidential data that must not be disclosed to authorities (e.g., in the healthcare sector or in research).

Working with a Vector Database

Once you have loaded your data into the database, you can search in various ways for semantically relevant results or configure the search so that it recommends something of possible interest, including something undesirable. An exciting search would be to discover data you never thought you would need.

The search is preceded by the three-step process of vectorization. In the first step, words, sentences, audio, or even images are translated into vectors by a transformer model. These models generally can be accessed via APIs. Some transformers are open source, and others (e.g., Jina AI [8]) charge a small fee to transform data into vectors. The result is always a number of vectors. The second step is to store the vectors in a vector database. Vectors are basically comma-separated numbers that can be represented in a number of formats (e.g., binary blobs). During storage, the vectors are indexed according to various algorithms.

The last step is the semantic search, which in turn uses various algorithms, including the following:

- **Cosine similarity** measures the angle between two objects. The smaller the angle, the more similar.

- **Euclidean distance** measures the distance between two objects.
- **Manhattan distance** computes the distance between two objects by adding up the differences in their coordinates.
- **Dot product distance** computes the sum of the products of the corresponding components of two vectors, which results in a scalar value.

Next, I briefly look into how to use the Qdrant Python client to create a database collection, load data into it, and run a simple search.

Practical Example

To begin, download the current Qdrant version from Docker Hub, and then start the service:

```
$ docker pull qdrant/qdrant
$ docker run \
  -p 6333:6333 -p 6334:6334 \
  -v $(pwd)/qdrant_storage:/qdrant/storage:z qdrant/qdrant
```

The default configuration sends all the data to the `./qdrant_storage` directory, which is also the only directory that can be seen from both the container and the container host. Qdrant can now be accessed through the API on `http://localhost:6333/` or through the dashboard on `http://localhost:6333/dashboard`.

The first step is to initialize the client; next, you need to create a database (e.g., `test_collection`) and import a handful of test data (Listing 1). The database should now respond to the `print(operation_info)` statement with the following code:

```
operation_id=0 status=2
<UpdateStatus.COMPLETED: 'completed'>
```

You can now formulate a simple query, such as *Which of the stored vectors is most similar to [0.2, 0.1, 0.9, 0.7]?* The code and the database response are shown in Listing 2. Additionally, you can apply a filter, for example, to get the most similar result that includes *London* (Listing 3). You have now created a

Listing 1: Create and Fill the Database

```
### Initialize the client
from qdrant_client import QdrantClient
client = QdrantClient(url="http://localhost:6333")

### Create a database
from qdrant_client.models import Distance, VectorParams
client.create_collection(collection_name="test_collection", vectors_config=VectorParams(
    size=4, distance=Distance.DOT),
)

### Import test data
from qdrant_client.models import PointStruct
operation_info = client.upsert(collection_name="test_collection", wait=True, points=[
    PointStruct(id=1, vector=[0.05, 0.61, 0.76, 0.74], payload={"city": "Berlin"}),
    PointStruct(id=2, vector=[0.19, 0.81, 0.75, 0.11], payload={"city": "London"}),
    PointStruct(id=3, vector=[0.36, 0.55, 0.47, 0.94], payload={"city": "Moscow"}),
    PointStruct(id=4, vector=[0.18, 0.01, 0.85, 0.80], payload={"city": "New York"}),
    PointStruct(id=5, vector=[0.24, 0.18, 0.22, 0.44], payload={"city": "Beijing"}),
    PointStruct(id=6, vector=[0.35, 0.08, 0.11, 0.44], payload={"city": "Mumbai"}),
])
```

Listing 2: Sample Query

```
search_result = client.search(
    collection_name="test_collection", query_vector=[0.2, 0.1, 0.9, 0.7], limit=3)
print(search_result)
### Output:
ScoredPoint(id=4, version=0, score=1.362, payload={"city": "New York"}, vector=None),
ScoredPoint(id=1, version=0, score=1.273, payload={"city": "Berlin"}, vector=None),
ScoredPoint(id=3, version=0, score=1.208, payload={"city": "Moscow"}, vector=None)
```

database, filled it with vectors, and queried it. Qdrant finds the closest answers and outputs them with a similarity score.

Retrieval-Augmented Generation

Since the release of ChatGPT, the hype over language models has exploded on the Internet. Language models can write essays and program code, or create memes, although I'm not sure that's a good thing. As brilliant as these chatbots seem, they still struggle with tasks that require external knowledge and factual information. Sure, they can describe the honeybee's waggle dance down to the smallest detail, but they would be far more valuable if they could gain insights from

all the data provided, not just their original training data. Because horrendous amounts of money and months of training are needed to establish these large language models from scratch, a better way is needed for existing LLMs to access custom data. As a short-term solution, you can be creative with how you word your prompts. LLMs can only include a limited amount of text from the

Listing 3: Apply a Filter

```
from qdrant_client.models import Filter, FieldCondition, MatchValue
search_result=client.search(
    collection_name="test_collection",
    query_vector=[0.2, 0.1, 0.9, 0.7],
    query_filter=Filter(
        must=[FieldCondition(
            key="city",
            match=MatchValue(value="London"))]),
    with_payload=True, limit=3,)
print(search_result)
### Output:
ScoredPoint(id=2, version=0, score=0.871,
    payload={"city": "London"}, vector=None)
```

prompt in their responses, which is known as the context window. Some models (e.g., GPT-3) can handle up to a 4,096-token context, which is not enough for most knowledge bases.

Figure 6 shows how a retrieval-augmented generation (RAG) system works: Before a question reaches the LLM, a layer intervenes that first consults a knowledge base and reads the relevant information, which in this case is the expenditure data for the last month. Your LLM can now

generate a relevant, honest response about your budget.

As the dataset grows, more efficient methods are needed to find the most relevant information for the LLM's limited memory. In this case, you need a suitable method to store and retrieve the specific data required for your query without the LLM having to remember it.

Vector databases store information as vector embeddings. It is precisely this format that supports efficient similarity searches to find relevant data for

your query. Qdrant, for example, is specially designed to produce results quickly, even in scenarios with billions of vectors.

At its core, a RAG system consists of two components: a retriever and a generator. When the retriever receives a question, it uses the similarity search to extract the most relevant vectors for answering the question from a huge knowledge database made up of vector embeddings. The system uses various techniques to determine what is important. The first step is to vectorize the request by using the same pre-processing and vectorization methods used to fill the database, so that the results are automatically compatible. Then it is the turn of a vector similarity procedure to determine the most relevant documents or passages for answering a query. Vector similarity is a fundamental concept in machine learning and natural language processing (NLP) that quantifies the similarity between vectors mathematically represented by data points. The generator then comes into play, formulating and assembling the final answer in natural language.

The LLM is typically a model (e.g., GPT, BART, T5) that has been trained with a massive dataset to understand and produce natural language. It not only fields the question as input, but also the additional passages or documents identified by the retriever that can help with the answer. The interaction between retriever and generator is shown in **Figure 7**.

Where RAG Systems Are Used

Because of their more informed and contextualized answers, RAG models are now appearing in many fields, especially fields where factual accuracy and depth of knowledge are important, such as advanced answering systems that draw on knowledge databases and generate answers in natural language. Text generators also benefit

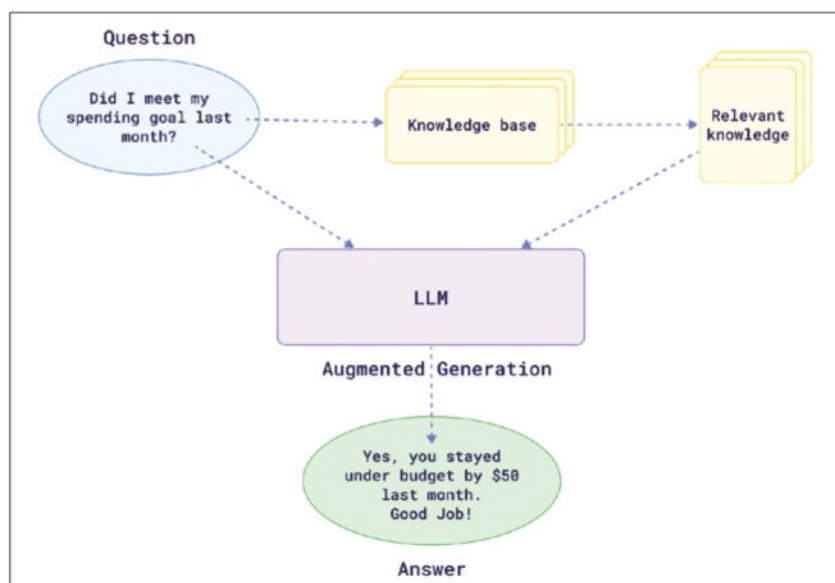


Figure 6: Relevant knowledge from a knowledge base is also made available to the LLM.

© Qdrant [9]

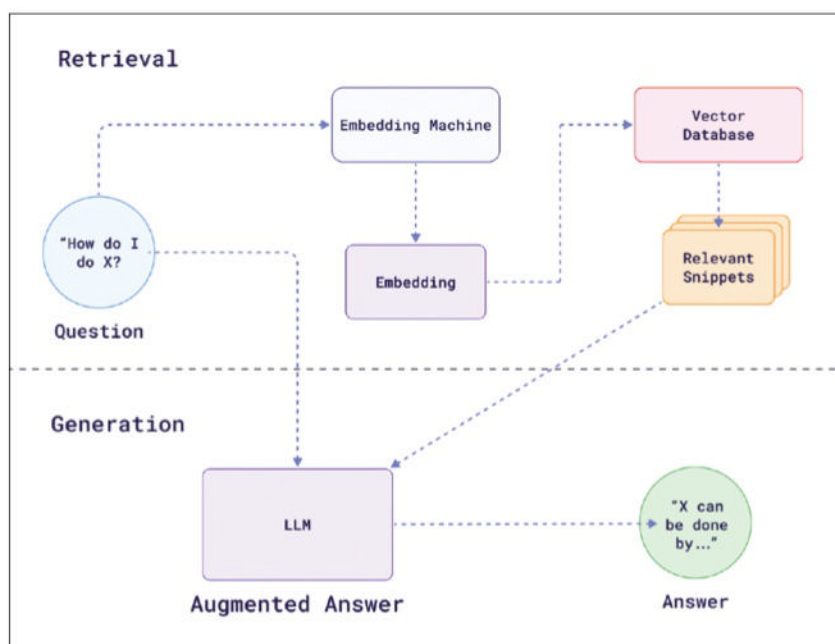


Figure 7: The interaction of retriever and generator in a RAG system. © Qdrant [9]

from RAG (e.g., for contextualized summaries of longer text). Another field of application is the transformation of data into text for business intelligence reports or to describe insights gained from data visualization. At the end of the day, RAG is not restricted to text, but can process multimedia information and even provide textual responses to questions about images or videos.

Conclusions

The combination of LLM and a vector database creates a powerful system that provides accurate answers. The vector database aims to produce precise results with minimal use of resources, which is helpful given that

one of AI applications' biggest disadvantages is their resource requirements (CPU, RAM, GPU), all of which translate to high costs. Operating a chatbot can consume considerable sums of money every day. Vector databases play their trump card by providing better and more accurate results at a lower cost in advanced AI applications. ■

Info

- [1] Qdrant: [\[https://qdrant.tech\]](https://qdrant.tech)
- [2] Vector search: [\[https://qdrant.tech/documentation/overview/vector-search/\]](https://qdrant.tech/documentation/overview/vector-search/)
- [3] Hugging Face: [\[https://huggingface.co\]](https://huggingface.co)
- [4] Sentence Transformers: [\[https://huggingface.co/sentence-transformers\]](https://huggingface.co/sentence-transformers)
- [5] co.embed API: [\[https://huggingface.co/Cohere/Cohere-embed-multilingual-v3.0\]](https://huggingface.co/Cohere/Cohere-embed-multilingual-v3.0)

- [6] "What is Vector Quantization?" by Sabrina Aquino, September 2024: [\[https://qdrant.tech/articles/what-is-vector-quantization/\]](https://qdrant.tech/articles/what-is-vector-quantization/)
- [7] Similarity search: [\[https://qdrant.tech/documentation/concepts/search/\]](https://qdrant.tech/documentation/concepts/search/)
- [8] Jina AI: [\[https://jina.ai/\]](https://jina.ai/)
- [9] "What is RAG: Understanding Retrieval-Augmented Generation" by Sabrina Aquino, March 2024: [\[https://qdrant.tech/articles/what-is-rag-in-ai/\]](https://qdrant.tech/articles/what-is-rag-in-ai/)

The Author

David Myriel is a professional author and developer representative in the field of vector search and machine learning. He works as Director of Developer Relations at Qdrant and has more than 10 years of professional experience in the training of developers. He leads a team of eight that supports Qdrant's global open source community.

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

Want to subscribe?

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

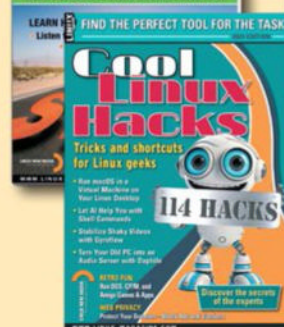
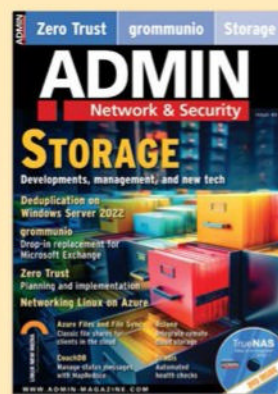
➤ shop.linuxnewmedia.com



➔ shop.linuxnewmedia.com

DIGITAL & PRINT
SUBSCRIPTIONS

SPECIAL EDITIONS





When you should and should not use NoSQL databases

Different Store

Tables are an established format for storing information in databases, but they don't always fit the bill - which is just one argument of many that speaks for NoSQL. By Jens-Christoph Brendel

Viewed at an abstract level, very much of what is known revolves around the relationship of two or more things to each other: events to dates, nuclear charges to chemical properties, items to prices, authors to works, and so on. Rows and columns in a table are an obvious way of mapping these relationships. The ancient Greeks were literally already aware of this advantage - Ptolemy started recording astronomical observations in tabular form in the second century. Since then, the table has triumphed in all areas of life, from religion, where the description of the same events in different gospels was recorded in tabular form centuries ago (canon tables), to music, where musical notation can be understood as a table in which the bar lines mark the columns and the staves the rows.

When the search for efficient ways to manage large amounts of data began in computer science in the 1970s, spreadsheets were used. Edgar F. Codd developed the relational database model and the forerunner of the SQL query language. Both still exist today, with the relational database management system (RDBMS) still

dominating the market as the top four most popular databases (DBs) in November 2024: Oracle, MySQL, Microsoft SQL Server, and PostgreSQL [1]. So why look for alternatives when traditional database technology uses such a universal basic form of information storage?

Specialist Data

Not everything can be recorded in a table, including unstructured data (e.g., text), for which the elements of a statement are not broken down into individual attributes and their values. In turn, this means they cannot simply be assigned to rows and columns, which makes tables unsuitable in this case. Similarly, tables are not recommended where the type and number of attributes to be stored vary greatly. Because the relational model works with a database schema that creates fixed and immutable tables at the outset, it fails to integrate attributes that occur spontaneously at a later time. Alternatively, if the schema included all conceivable possibilities up front, you would have to make do with

inefficient tables because they would be sparsely populated.

Even fairly standardized and structured data is not necessarily a good match for tables. One problem that vector databases solve, for example, is data dimensions. They handle data defined as vectors, which you can imagine as arrows in three-dimensional space. The arrows point from the origin of the coordinate system in a specific direction and to a point in space for a specific length. Vectors are used, for example, in embeddings, such as those used to represent words in large language models.

Because human imagination is only capable of dealing with three dimensions, you can imagine, in a very simplified way, that the vectors of the terms "Dog," "Cat," and "Mouse" would point to an area in the lower left rear corner of a room and that the vectors of "Engine," "Gearbox," and "Tire" would point to the upper right front corner (Figure 1). In reality, words do not have three dimensions, but hundreds or thousands. Nevertheless, the distances between them can be computed - just as in three-dimensional space.

In this way, you then determine from the distances between the words that "Cat," "Dog," and "Mouse" are meaningful in a common context that can be described with the term "Animal."

Photo by Artem Gavrysh on Unsplash

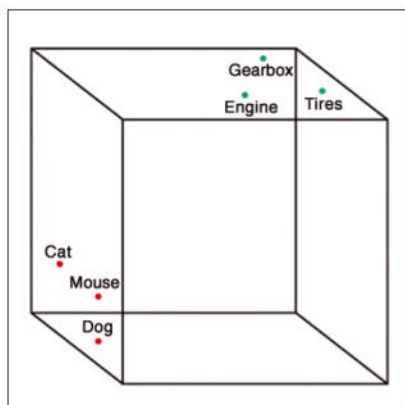


Figure 1: Vector databases can be used to store words as vectors that, in reality, possess many more dimensions than can be represented by an image.

Similarly, the word “Gearbox” belongs to the “Car” context. However, a word with hundreds of vectors is not just difficult to fit in a table, it would also be tricky to formulate the pertinent question in SQL: Which words occur in the same context (i.e.,

they’re close to the original term in the multidimensional space)? These nearest neighbor queries are a specialty of vector databases.

In addition to data type and dimensionality, other properties can drive a relational database model to its limits, such as scalability and reliability. Relational databases tend to scale vertically by upgrading the database server (more CPUs, more RAM, faster mass storage). Technology availability sets limits, of course. Comparatively, horizontal scaling, which numerous NoSQL databases support, is basically limitless. Instead of more powerful hardware, more of the same off-the-shelf hardware is added. To work, it must be possible to parallelize the workload and distribute the data across many nodes. If the distribution is also redundant, it contributes to reliability, because the surviving nodes with the same database can replace the failed nodes.

CAP Theorem

All of the desirable characteristics will never come together, as the CAP theorem states, according to which only two of the following three requirements can ever be met: consistency (C), availability (A), and partition tolerance (P) (Figure 2). Consistency means the most recently updated

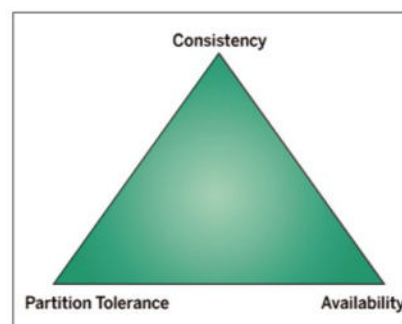


Figure 2: The CAP theorem asserts that only two of the three properties of a distributed system can be implemented simultaneously.

IT Highlights at a Glance



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

Linux Update • **ADMIN Update** • **ADMIN HPC**

Keep your finger on the pulse of the IT industry.

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

Linux Update: bit.ly/Linux-Update

data is always received for each read operation. If this is not possible, an error message must be displayed. Availability means every query has an answer – but possibly not one with the most up-to-date data. Partition tolerance means the system continues to work even though messages between the cluster nodes have been lost or delayed because the database has been partitioned.

Because all three requirements cannot be met at the same time, relational databases are usually CA systems; that is, they maintain consistency and availability but are not partition-tolerant because they cannot be distributed across several nodes such that a breakdown in communication between the nodes could be tolerated. CP systems, such as the single-master MongoDB database with exactly one primary node for all write operations, maintain consistency and are partition-tolerant; they can cope with the loss of communication between two nodes. However, they are then no longer available until all secondary nodes have reconnected to the newly deployed primary node. It therefore suffers a lack of availability (A) during this time. AP systems (e.g., Cassandra or CouchDB) are always available and also partition-tolerant, but not always consistent.

It's up to you to decide. If consistency plays the main role for you (e.g., for financial software), you will probably prefer a relational database. On the other hand, if you attach more importance to availability or partition tolerance, you will opt for a NoSQL variant.

Transactions

The CAP system also has an effect on the transaction model. The model that focuses on consistency is ACID (atomicity, consistency, isolation, durability), which is the predominant transaction model for relational databases and guarantees consistency, even at the cost of potentially reduced availability.

Another transaction model is BASE (basically available, soft state, eventual consistency), which values availability more highly than consistency and always guarantees an answer, even if it might not always use the most up-to-date data. For example, if the number of contacts on a social network is temporarily not up to date, it might be more acceptable than the system throwing an error message because of its inability to access the latest values. The BASE model is more common among NoSQL candidates.

Replication

Replication problems are closely linked to the transaction model; they are especially likely to occur where the system needs to adhere strictly to the ACID model. If a transaction is replicated to different nodes under these circumstances, not only is it important which actions are replicated and committed, but also in which order; otherwise, inconsistencies can arise between the replicas.

Therefore, distributed relational databases need special protocols, such as the two-phase commit, which comprises two roles: the coordinator and the agent. In the initial, preparatory phase, the coordinator familiarizes the agents with the transaction and asks them to vote on it. If everyone agrees, the transaction is committed in the second phase. If the vote is not unanimously in favor, or if the question elicits no responses, the coordinator prompts all agents to roll back the transaction. The procedure is complicated and, in the case of many short transactions in particular, the resulting locks can take orders of magnitude longer to complete than the actual transaction because of the network communication required. Far easier is if you can tolerate temporary consistency violations, which many NoSQL databases allow. You can mitigate ACID guarantees in two ways: (1) The guarantee is only given for a part of the database (e.g., for a

single tuple in a key-value store), a database partition, or shard. A complicated distributed commit protocol is then no longer needed, but application developers must now ensure that transactions that span several of these sub-areas are skillfully divided. (2) You can completely do without strong consistency, although it only seems acceptable for some web applications. In any case, NoSQL databases can achieve considerable performance gains in this way.

Conclusions For and Against

Relational databases (RDBs) are older than non-RDBs and are the most widespread model; typically, they strictly observe consistency, often store their data in standardized tables without redundancy, and use a sophisticated query language (SQL) that allows data to be linked in a very efficient way (joins). You pay for these advantages with difficulties in terms of scaling and high availability and with poorer performance.

On the other hand, various types of NoSQL databases can manage data for which tables are not a viable option. Moreover, they often scale better, are more efficient, and offer high availability on the side. However, this flexibility comes at a price: Consistency cannot be guaranteed in all circumstances, storage might be redundant and therefore take up more space, the query language is less powerful than SQL, and application developers need to give some cross-domain queries careful thought up front. These characteristics and limitations together lead to the conclusion that, in general, neither of the two models is inherently superior. Instead, you need to assess which model offers the greater advantages for your current use case, while meeting all requirements. ■

Info

[1] DB-Engines Ranking:
[\[https://db-engines.com/en/ranking\]](https://db-engines.com/en/ranking)

Looking for your place in open source?



Set up job alerts and get started today!

OpenSource JOB HUB



opensourcejobhub.com/jobs

NVMe-oF gears up to replace iSCSI

Changing of the Guard

NVMe over Fabrics promises high-redundancy centralized storage with simple maintenance and management and almost twice the speed of its antiquated iSCSI predecessor. By Martin Loschwitz

Centralized storage in the form of network-attached storage (NAS) or a storage area network (SAN) are implicitly or explicitly set in stone in the data center. Wherever you look, you will likely find a device whose sole task it is to provide persistent storage to the other systems.

From the administrator's point of view, it makes sense to handle central tasks such as redundancy and backup with a single point of administration. The alternative would be uncontrolled local growth with complicated backup and redundancy mechanisms. Of course, local storage is occasionally inevitable (e.g., in cloud setups where virtual machines require extremely fast storage). However, cloud providers tend to assume that the customer's application takes care of data redundancy and regular backups itself. For almost all other scenarios, centralized storage is the ideal solution for system administration. Like any technology, centralized storage is subject to the whims of time.

For what felt like an eternity, the Fibre Channel (FC) protocol was the gold standard, and every virtualization server in the setup needed its own host bus adapter (HBA) to connect it to the FC network. The manufacturers

of centralized storage have probably made a fortune over the years simply by selling Fibre Channel hardware (Figure 1), without even taking into account that storage appliances are not exactly kind on the customer's wallet.



Photo by Mathew Browne on Unsplash

Figure 1: Fibre Channel hardware such as this switch from QLogic has long been standard in the data center. Today, FC is increasingly giving way to Ethernet. © Wikipedia/Melee

Slow and Old Fashioned

If you want systems to use Ethernet to access storage, they need a protocol that is based on the IP stack. Because TCP/IP is practically as widespread as Ethernet, iSCSI – which is quite simple in itself – was created. The protocol uses SCSI commands but wraps them in TCP/IP packets, enabling access by IP addresses and port numbers. However, iSCSI was never particularly popular with admins. On the one hand, the protocol has a reputation for being relatively sensitive with regard to the network infrastructure and storage used. If you have ever tried to connect VMware ESXi to a RADOS cluster's iSCSI target, you can probably tell a tale or two about your experience. The iSCSI initiator by VMware is a proprietary product that reacts in a particularly sensitive way if, say, the storage cannot cope with the applied load, forcing retransmits, which in the worst case can cause the entire backing device of an instance to go offline, taking down the instance itself in the process. iSCSI also reacts in an extremely sensitive way to unstable networks.

Moreover, iSCSI is not exactly famous for its performance. The solution has a reputation for significantly adding to latency. Ethernet already has significantly higher latency than Fibre Channel, for example, and the iSCSI protocol aggravates the situation, which helps even less. If you then combine iSCSI with implicitly slow storage such as RADOS, in which the CRUSH algorithm (controlled, scalable, decentralized placement of replicated data) causes high internal latency, you run the risk of having such high latency for single small writes that the installation becomes largely unusable.

Worse still, you can't do much tuning in a setup of this kind. The iSCSI target is a fixed part of RADOS and cannot simply be replaced, and RADOS latency can hardly be improved. In short, most admins view iSCSI as old fashioned and slow; where used, it is typically out of necessity rather than by choice.

In the meantime, however, Fibre Channel is no longer considered state of the art. Time has not stood still, of course, and FC HBAs with 128Gbps bandwidth are available. Today, companies shy away from the additional expense of operating a separate network architecture just for central storage. Instead, Ethernet, with the established outdated iSCSI protocol (see the “Slow and Old Fashioned” box), is asked to cover all eventualities because, after all, network technology has been consistently geared for greater bandwidth over the past 15 years: NVIDIA and others have long been offering 400Gbps switches (Figure 2), with research targeting 800Gbps and more.

Most customers grudgingly accept that improvements in Ethernet latency have been neglected, and latency is a key factor in the storage context. What they get in return is a centrally maintained, ubiquitous network installation that is also extremely scalable.

The NVMe Alternative

Vendors of large storage systems are well aware of this dilemma. NetApp, for example, has now practically scrapped its entire Fibre Channel portfolio and now only offers storage systems that use Ethernet. Other vendors are taking a similar approach. Thus far, they have all used iSCSI as an interim solution. Behind the scenes, several vendors in the Storage Networking Industry Association (SNIA) have been working together for several years on an alternative to the unpopular iSCSI protocol that has much to offer.

The underlying idea is relatively simple and very similar to the idea behind iSCSI: a modern, well designed, and fast protocol for accessing storage drives called NVMe. Redesigning NVMe so that it no longer uses the PCI bus directly for communication with the storage devices, but uses a network stack (fabric) instead, offers a powerful iSCSI replacement. iSCSI itself began in a similar way. However, NVMe is expected to deliver significantly improved performance and greater reliability. The manufacturers refer to their new project as NVMe over Fabrics (NVMe-oF).

Quick Storage

To understand the background of NVMe-oF in more detail, you need to look at the development of the NVMe standard, which has been established for years. NVMe primarily differs from the far more widely used SATA interface, in that it eliminates the need for protocol conversion between the system's PCI bus and the storage medium. Each storage device must be able to communicate with the other system components in some way. PCI is practically always used for this purpose. The problem is, SATA is a separate protocol, and to connect SATA drives over PCI, you need a converter that translates SATA commands into PCI commands, which causes latency and limits the achievable performance to the performance of the PCI-SATA converter.

NVMe does not need to detour through a storage controller because it communicates directly with PCI, and therefore also directly with the CPU and the other devices in the system, ensuring that it is primarily



Figure 2: Ethernet switches such as the SN4700 offer 400Gbps ports and easily outperform even the latest Fibre Channel switches. © NVIDIA

the PCI bus bandwidth that limits performance and that latency mainly arises in the PCI bus and storage device controllers. On top of that, NVMe has been massively tweaked in recent years. Today's NVMe devices are by far the fastest storage devices. In combination with fast flash chips, they achieve impressive bandwidths and impressively low latency. As far as Intel, IBM, NetApp, and others could see, it was an obvious choice to use the protocol as the basis for an iSCSI replacement.

How NVMe-oF Works

Don't get too excited at this point. NVMe derives a significant part of its performance from its direct interaction with the CPU and RAM and no requirement for any intermediary interfaces. However, if you remove NVMe from its original context and put it on a network, things look quite different again. Like all other packets, the NVMe packets wrapped in IP need to navigate the entire network stack of the sending node's operating system, then the physical network, and finally the kernel at the receiving end. Therefore, NVMe inevitably picks up more latency in a number of places on the network than in its traditional habitat, which clearly indicates that the manufacturers were not primarily concerned with getting NVMe speed through the network. Instead, the main aim of NVMe-oF is to replace the outdated iSCSI protocol with a more modern, more secure, easier-to-configure protocol with more options.

For example, as mentioned, an iSCSI target exists for RADOS, the Ceph object store. It now belongs to IBM after detouring by way of Red Hat, who stated that this target was "feature-frozen" a year and a half ago. The provider is therefore no longer developing anything new, and bugfixes are likely to be limited to serious errors or security issues. Instead, the hard work is being allocated to the NVMe-oF target for RADOS – I will get back to that later.

Whether you like it or not, Red Hat Ceph Storage 6 no longer contains the

iSCSI target. Other providers are likely to take a similar approach with their iSCSI implementations, so iSCSI quite possibly will have largely disappeared from the world's data centers in a few years' time.

Terminology

If you initiate a discussion on NVMe-oF, you often only see questioning faces, but NVMe-oF is not nearly as new as it feels; vendors have been marketing the function since 2019. The reason it has so far gone unnoticed by many admins is quite interesting. What's more, the solution has already been on a long journey.

When the big players in the storage industry began to think about iSCSI alternatives, they failed to reach a consensus. Some developers argued in favor of implementing NVMe on the basis of the Fibre Channel protocol, an approach now known as NVMe over Fibre Channel (NVMe-oFC). Other vendors, however, strictly refused this approach – more specifically, vendors who either never had their own FC portfolio or had gotten rid of it because FC was no longer very popular.

The idea of transplanting NVMe into the Ethernet TCP/IP stack (NVMe over TCP, or NVMe/TCP) quickly found more support. Its inventors were well aware of the implicit disadvantages of the solution, such as the significantly higher latency than with Fibre Channel; however, precisely because Ethernet is so widespread, they were prepared to accept these disadvantages. Because NVMe-oFC practically never made it past the experimental stage, it is now practically synonymous with NVMe/TCP.

However, newcomers didn't necessarily find the subject any easier to understand. Quite a few admins instinctively think of shared buses and remote direct memory access (RDMA) when they consider NVMe-oF for enabling access to devices over the network. Both a special network configuration and suitable hardware are required. None of these factors apply to NVMe/TCP. You don't even need

NVMe drives. SATA or SAS hard disks can be made available on the local network over NVMe/TCP.

Obviously, this method is not very fast, and the practical benefits of such a solution are limited, but the example makes one thing clear: NVMe/TCP can be used as a replacement practically anywhere that iSCSI was previously used, which was one central concern of the NVMe-oF inventors. The only requirement is an appropriate target. However, the otherwise totally diverse Linux community still has some catching up to do in this area. I'll look at two examples to illustrate.

Example 1: RADOS

The NVMe-oF solution for the RADOS object store is the core component of the Ceph storage solution. The Ceph developers are now keen to avoid detours through the block device layer of the Linux kernel to the extent possible. Instead, most access to a RADOS cluster takes place in userland. RADOS offers a wide range of integration at the programming level. The *librados* library enables direct access to RADOS objects, whereas *librbd* (where "rbd" is the RADOS block device) accesses RBD images in RADOS as if they were local storage drives. No one is surprised that IBM's primary objective is to implement an NVMe-oF solution for RADOS such that it uses *librbd* in the background and passes the data back and forth between the client and RADOS. However, this implementation makes the NVMe-oF target, which the RADOS developers have already provided, unusable for all solutions other than RADOS.

To make matters worse, the quality of the current 1.0 version of NVMe-oF for Ceph is quite poor to date. IBM explicitly identifies the function as a tech preview and therefore refuses all requests for support. If you want to take at least a sneaky look at the implementation and possibly compare it with a running iSCSI setup, you won't find nearly all the information you need in the Red Hat

documentation. The current documentation of the NVMe-oF target for Ceph is simply completely inadequate; instead, you need to google the commands you need.

The setup is not very different from iSCSI. First, you launch an instance of the NVMe-oF target with the `cephadm` utility, Ceph's internal orchestration solution, which means passing the RBD image in RADOS to `cephadm`; the image must be accessible over NVMe-oF. Five commands, not two, are involved, but this situation is unlikely to faze experienced admins. Finding the commands you need (i.e., extracting them from the small number of results returned by a *RADOS NVMeoF* search on Google) is still a challenge at the moment.

Moreover, interested admins need to be aware that a rolled-out NVMe-oF tech preview version 1.0 target cannot be used in a production setup. For example, no high-availability (HA) mode exists that would be required to implement the kind of multipathing NVMe-oF (like iSCSI) provides. Clients can take several network paths to the same storage device (e.g., because two servers publish the device on the network in parallel). High-availability mode is definitely intended and has already been implemented in the development versions of the 1.1.x series. However,

this means having a brand new version of the monitoring server in RADOS, which is not yet available as a stable Ceph version. The patch had not even made it into the Ceph Git directory at the beginning of July 2024. Instead, a discussion relating to the corresponding merge request has been raging for months. When and how the HA mode for NVMe-oF will be available in Ceph is still completely up in the air.

Even if the required functions do become available in Ceph in the foreseeable future, you need to be prepared for a bumpy road to a working configuration. Like practically all Ceph components, the NVMe-oF target comes exclusively in container form. The same applies to the associated command-line tool, `nvmeof-cli`. You need to use various `exec` commands for Podman or Docker to store the configuration in the first place. You can only hope that by then the documentation for the overall solution will be in a state that justifies its name.

Example 2: DRBD

Linbit in Vienna does things a little better. Of course, the in-house distributed replicated block device (DRBD) replication solution is far less complex under the hood than Ceph.

DRBD resides in the block device layer of the Linux kernel and shovels packets back and forth between the data carrier and the network. The obvious docking point for an NVMe-oF target is therefore the block device layer itself.

Linbit has implemented appropriate solutions both for use with the in-house storage manager Linstor and with classic tools such as Pacemaker. The NVMe-oF target for Linux is used to publish the desired drives as an alternative to iSCSI (Figure 3). The provider even provides its own tool [1] to build a Linstor cluster quickly with a published NVMe-oF target from three generic servers. High availability for the target itself is an implicit part of the solution. This construct can also be connected to other virtualizers, such as Proxmox (Figure 4).

Performance Questions

The Linbit solution seems interesting in that Linstor allows NVMe-oF, as well as its quasi-predecessor iSCSI, to be rolled out in a few simple steps. Linstor therefore implicitly offers the option of directly comparing the two protocols on the same hardware, which allows one to make statements about the improvements that NVMe-oF offers. You can also benefit from DRBD's simple architecture, which

```
# linstor-gateway iscsi list
```

IQN	Service IP	Service state	LUN	LINSTOR state
iqn.2001-09.com.linbit:my-target	192.168.122.222/24	Started	1	OK

Figure 3: Linstor supports both iSCSI and NVMe-oF, so the tool can be used to make excellent comparisons on the same hardware.

```
# nvme discover -t tcp -a 192.168.10.200 -s 4420
# nvme connect -t tcp -n linbit-nqn0 -a 192.168.10.200 -s 4420
# nvme list
```

Node	Generic	SN	[...]
/dev/nvme0n1	/dev/ng0n1	ae9769ab8eac	[...]

Figure 4: Proxmox offers support for NVMe-oF with integration of remote storage drives for use locally.

prevents the storage software itself from falsifying the measured values. A strictly documented comparison under lab conditions was not possible before this issue went to print. In this respect, the values determined in the quick test do not allow any kind of final judgment on the question of whether NVMe-oF is really vastly superior to its predecessors. However, the collected data at least suggests this possibility. On the same hardware, NVMe-oF, in combination with DRBD, delivered around 20 percent more throughput with nearly half the latency. In other words, a freshly created iSCSI logical unit number (LUN) achieved a good 36,000 I/O operations per second (IOPS) with a 4KB request size, an I/O depth of 32, and four jobs running at the same time; this number went up to 68,000 IOPS with NVMe-oF – including the latency caused by DRBD protocol C replication. If the developers of NVMe-oF targets for Linux and the manufacturers of corresponding devices succeed in achieving comparable improvements in NVMe-oF compared with iSCSI, iSCSI will probably disappear more quickly than if it were just the outdated protocol being replaced. Any admin looking for an iSCSI alternative is strongly recommended to carry out more extensive tests.

More Bang with Custom Hardware

NVMe-oF performance is also a happening thing. NVIDIA, for example, offers NVMe-oF offloading on its ConnectX network cards (**Figure 5**) if you use the vendor's OpenFabrics Enterprise Distribution (OFED) driver. The network card then takes care of NVMe-oF packets and can also make use of technologies such as the aforementioned RDMA, which could even make it possible to reduce the latency of NVMe-oF to a level comparable to that of Fibre Channel – potentially a milestone in terms of performance tuning for remote storage under Linux.

You might want to avoid getting overly excited. The storage performance (SPDK) and data plane development kit (DPDK) setup required for this type of operation is complicated, currently not very well documented, and not supported by every Linux distribution. Installations of this type are still either carefully crafted by hand, or their use is limited to the laboratory and test environments. However, in view of the considerable improvements that can be achieved with NVMe-oF compared with iSCSI, this situation is

unlikely to remain the case. What's more, the trend toward outsourcing network tasks to powerful network chips is by no means new. In this respect, major developments are still around the corner.

Conclusions

NVMe-oF is far more than just the changing of the guard for iSCSI. Manufacturers familiar with the subject have given much thought to how to design and implement a storage protocol for remote access without repeating the mistakes of iSCSI. The result is a standard with a misleading name, because virtually all NVMe-oF setups today are actually NVMe/TCP implementations.

In terms of performance, NVMe-oF leaves you wanting more with performance that is almost twice as fast as its antiquated predecessor in comparable situations. Vendors have long since begun to divert resources away from iSCSI and invest in the development of NVMe-oF solutions, which is a good thing, because if NVMe-oF turns out to be as vastly superior as it currently appears, you no longer have a good reason to hold on to the antiquated iSCSI.

The hope is that NVMe-oF will gain more acceptance on the part of Linux manufacturers and distributors and that the integration in Red Hat, Ubuntu, and SUSE Enterprise Linux Server will improve. If this happens, it won't be long before the sun sets on iSCSI. ■

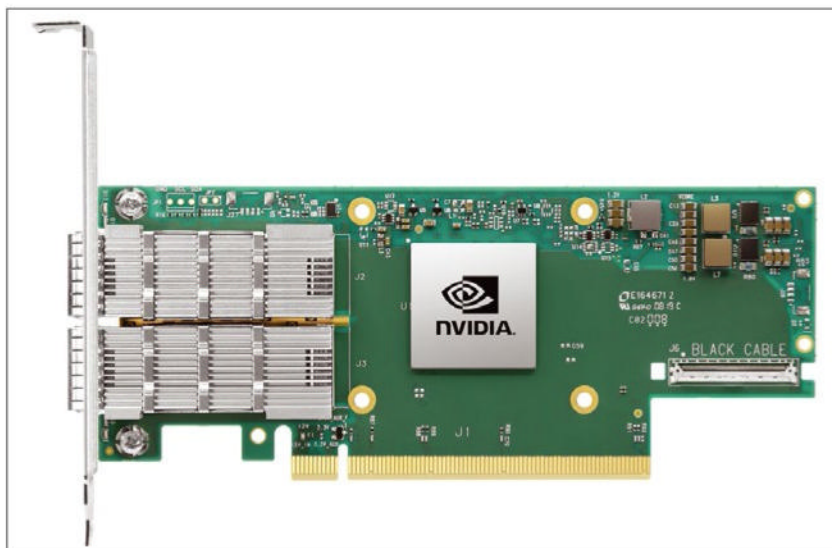


Figure 5: Smart network interface cards (SmartNICs) such as the ConnectX-6 from NVIDIA process protocols such as NVMe-oF ("offloading"), achieving considerable performance gains by doing so. © NVIDIA

Info

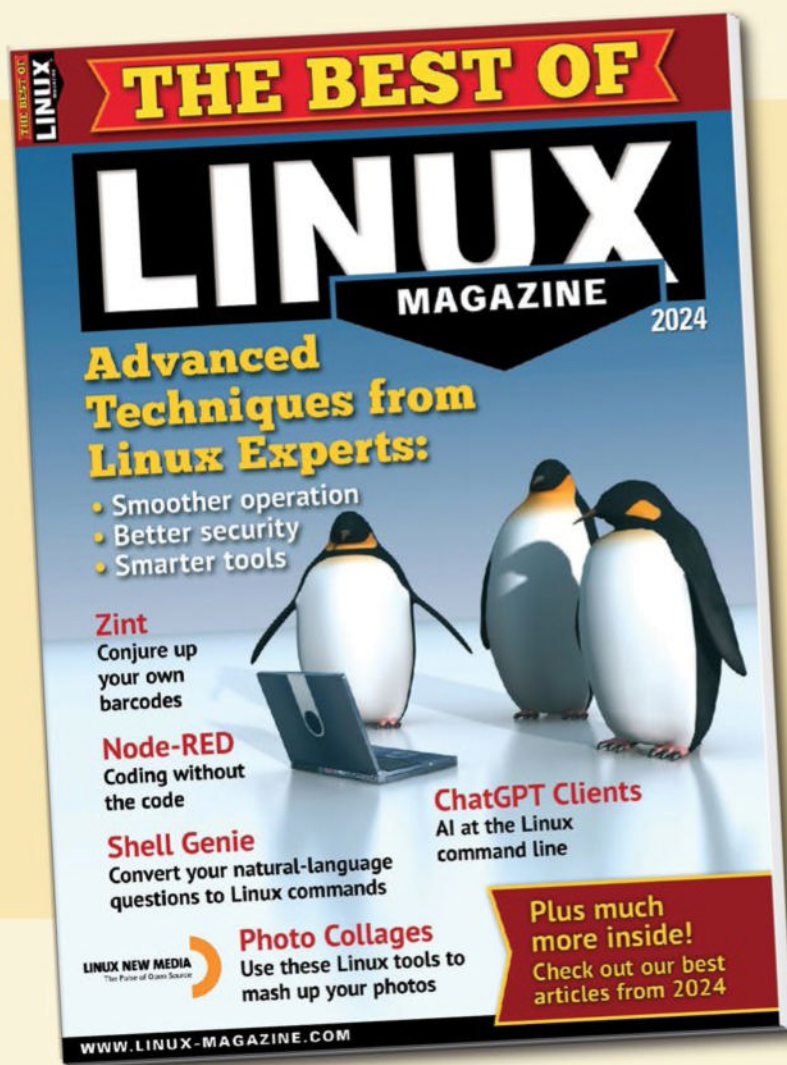
- [1] Linstor how-to for DRBD and NVMe-oF:
[\[https://github.com/LINBIT/linstor-gateway\]](https://github.com/LINBIT/linstor-gateway)

The Author

Martin Loschwitz is the founder and managing director of True West IT Services GmbH, which offers scalable IT infrastructure based on OpenStack and Kubernetes.



THE BEST OF LINUX MAGAZINE 2024



Advanced Techniques from Linux Experts

This new special edition brings you the best and most practical articles from the 2024 editions of *Linux Magazine*.

Whether you're new to Linux or a seasoned veteran, you'll find something useful in the tips, tools, and technologies inside.

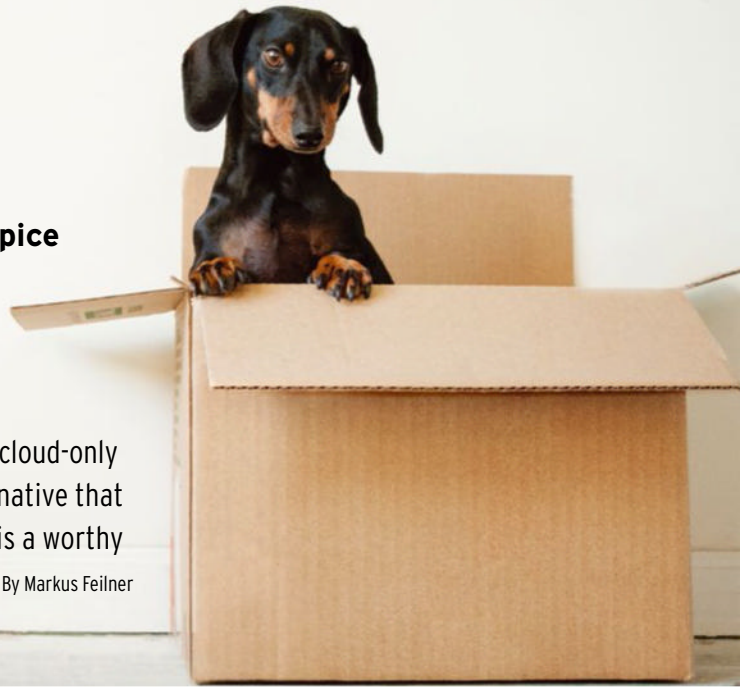


ORDER ONLINE:
shop.linuxnewmedia.com

Moving from Atlassian Confluence to BlueSpice

Big Move

With Atlassian's announcement that they are moving to a cloud-only solution, many organizations will be looking for an alternative that lets them keep their collaboration data local. BlueSpice is a worthy open source alternative with easy-to-use migration tools. By Markus Feilner



Atlassian Confluence is a popular collaboration platform used by many organizations around the world. Users are accustomed to relying on Confluence for meetings, memos, wikis, and project management; however, Atlassian recently announced that they are discontinuing the server edition in favor of a cloud-only configuration. Many customers with sensitive information, such as research companies in

the high-tech industry, lawyers, financial service providers, and journalists, are concerned about moving all their collaboration data to the web. Others, even if they aren't in particularly sensitive industries, might not want to give their data to the cloud for any number of policy or practical reasons. Even as some Confluence admins grapple with the challenge of moving to the Atlassian cloud, others are

looking for alternatives that will let them keep their data local and avoid the vendor lock-in of proprietary tools like Atlassian Confluence, but migrating data is often time-consuming, although it is possible. The transfer of Atlassian data to a new system is typically the most complex task during migration. Users who want to leave will usually not receive any help from

XWiki

French company XWiki produces the Java-based open source software of the same name and has been successful in many setups, including as a part of the digitally sovereign openDesk [1] web desktop from the German and French governments. Although anything else in XWiki is free, the company has not made the supported version of its migration tool available as open source software. For EUR1,900, the customer receives the Confluence Migration Toolkit [2], which interested parties are allowed to test for 30 days free of charge after entering a test key [3]. A free license (trial) for the Migration Toolkit is available on request at any time. The XWiki customer must send their instance ID to the manufacturer and enter it in the *Global Administration / Licenses* section under *Add License* (Figure 1).

Global Administration: Licenses

View the list of installed extensions that require a license. Set the license owner. Buy licenses for paid extensions or get trial licenses to test them. Check for license updates. Configure automatic upgrades for licensed extensions.

- Users & Rights
- Extensions**
- Extensions
- History
- Updater
- Licenses
- Look & Feel
- Content
- Editing
- Mail
- Search
- Social
- Wikis
- Other

License Ownership

Fill in the following fields in order to buy a license or to get a trial one.

FIRST NAME

LAST NAME

EMAIL

You will use this email address to contact the support, so make sure it is valid.

INSTANCE ID

Unique identifier for your XWiki instance. Used to associate your licenses with your XWiki instance.

Licensed Extensions

Here's a list of extensions installed in your wiki that require a license to be activated. You can buy a license or get a 10-day trial license that can be extended only once with 10 days. A license covers all versions of the licensed extension and is shared by the main wiki and all its subwikis. A licensed extension is listed multiple times if different versions are installed on different subwikis, but all those versions will share the same license.

Results 1 - 1 out of 1 per page of 15

Extension Name	Version	Expiration Date	Support	User Limit	Wikis	Actions
Diagram Application (Pro)	1.17.3	No license available	-	-	Home	<input type="button" value="Buy"/> <input type="button" value="Get Trial"/>

Results 1 - 1 out of 1

Add License

If you already have a license, copy it here to activate the extension(s) it's associated with.

1. Add your details

2. Purchase or request a trial license

Figure 1: The XWiki migration wizard is available for a fee. Users need a registration key for the 10-day test and a number of other XWiki extensions. © XWiki [2]

Photo by Erda Estremera on Unsplash

Atlassian, and Confluence support service providers expect good money for their help. To make matters worse, Confluence is different from most wikis. Unlike its open source competitors, Confluence manages individual pages in “workspaces,” has its own access rights, and handles subordinate pages, as well as numerous macros. Some of these features might not correspond directly to equivalent features in the alternative tools.

Confluence administrators are happy to see that export functions are available, at least export the content to, for example, XML, so those who

know how to use scripts will achieve quick and good results manually.

A pair of leading open source wiki tools advertise migration assistants to help you migrate your Confluence data. One of those tools, XWiki, is fully open source once everything is moved over, but their supported migration assistant costs EUR1,900 after 30 days (see the “XWiki” box). Another enterprise alternative that offers a less restrictive migration path is BlueSpice.

XWiki offers a number of resources, including a blog with numerous articles about migration and 60 supported Confluence macros [4]. The

XWiki Migration Toolkit also has a guided migration process that the manufacturer has integrated directly into the XWiki web interface.

BlueSpice

BlueSpice, which is developed by the company Hallo Welt! [5], is available as open source on GitHub, including the scripts tested in the example in this article. The company advises its customers to seek support during the migration process, perhaps because of the business model or the complexity of the tasks. In many cases, migrating everything on your own might seem

Live Macro Check – Test your compatibility now

Find out immediately how compatible your system is with BlueSpice and which content and functions can be directly adopted by our script. Use the macro check as follows:

- Step 1** Open your Confluence and click the gear icon in the top right corner.
- Step 2** The page reloads. In the “Data Management” section on the left, select “Macro Usage”.
- Step 3** Scroll down to “All macros”. Select all the information displayed. Copy them with Ctrl+C.
- Step 4** Paste the copied text into the macro checker with Ctrl+V.
- Step 5** Find out how compatible your system is with BlueSpice.

Advanced Macros
 blog-posts (4)
 recently-updated (3)
 excerpt (1)
 tip (12)
 code (8)
 info (8)
 36
 Confluence Live Search Macros Plugin
 livesearch (1)
 1
 Confluence Roadmap Planner
 roadmap (1)
 1

Check

84% compatible

Advanced Macros

- ☒ **blog-posts (4)**
Unfortunately, this macro is currently not supported by the automatic migration script.
- ☒ **recently-updated (3)**
This macro is supported.
- ☒ **excerpt (1)**
This macro is partially supported, reworking may be necessary.
- ☒ **tip (12)**
This macro is supported (uses Extension:ContentDroplets).
- ☒ **code (8)**
This macro is supported (uses Extension:SyntaxHighlight_GeSH).
- ☒ **info (8)**
This macro is supported (uses Extension:ContentDroplets).

Confluence Live Search Macros Plugin

- ☒ **livesearch (1)**
Unfortunately, this macro is currently not supported by the automatic migration script.

Confluence Roadmap Planner

- ☒ **roadmap (1)**
Unfortunately, this macro is currently not supported by the automatic migration script.

One of your important macros is flagged as unsupported? It is very likely that the content can still be transferred. We will be happy to check this in more detail and advise you. Just get in touch with our sales team.

Figure 2: The BlueSpice macro wizard has found an unknown macro – manual work is necessary.

unrealistic, but with BlueSpice, the attempt is possible, even without the manufacturer's help, thanks to open source migration scripts. Some American universities and institutions have done so without support from the vendor, even from Atlassian's products to BlueSpice's upstream project, MediaWiki, with the use of Hallo Welt! tools.

Migration with BlueSpice takes five steps: provide infrastructure (server, storage, network, installation), extract content, check data homogeneity, correct language and media, and integrate [6]. The initial macro check is followed by the XML export from

Confluence and then migration with data import and checking. Many Confluence installations use macros (built-in or programmed by third parties) to prepare, link, and display content. So many of these macros exist that not all of them can be supported by BlueSpice or XWiki.

Hallo Welt! offers a test on its website that you can use to find out which macros are supported during migration: Macro management is also found in the data management settings in Confluence, where you can select all macros and copy them to the clipboard (Ctrl + C). On the BlueSpice website [6] you will then

find the live macro check, where you can paste the macros found (Ctrl + V) and click *Check* (Figure 2).

Exporting a Space to XML

Once all macro problems have been resolved, the next step is to transfer an entire space from Confluence to BlueSpice (Figure 3). Confluence can export data to CSV, HTML, XML, and PDF (Figure 4). XML is the best choice when it comes to structured data migration (and as in the following example, neatly packaged in an archive). During migration, the data is extracted from Confluence and the

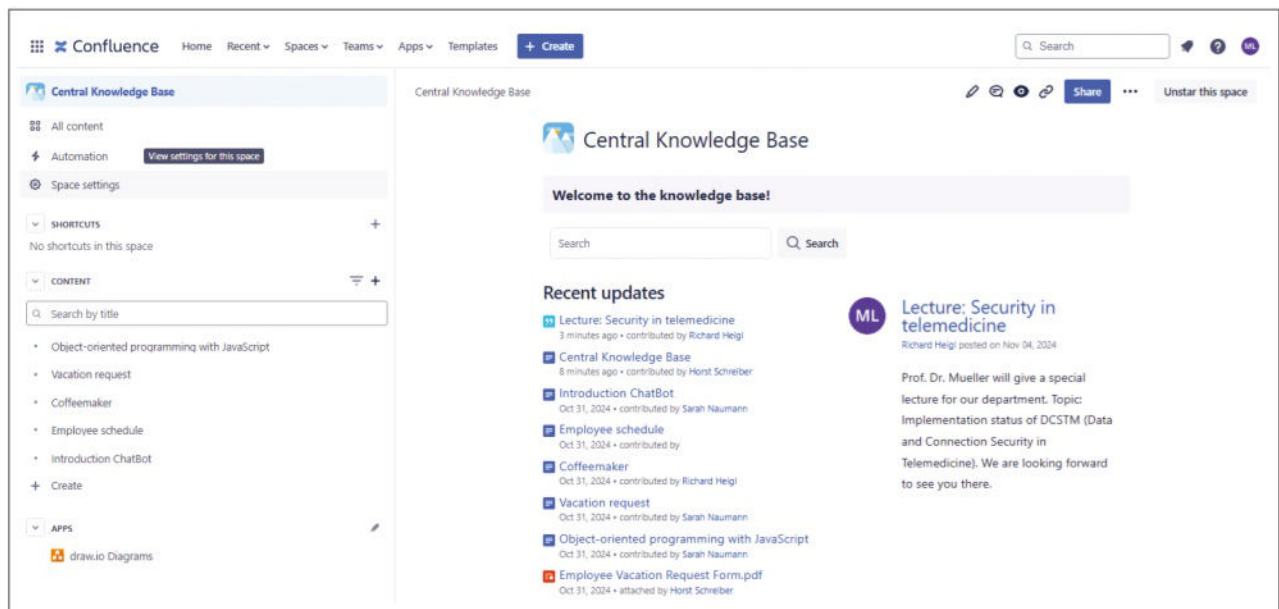


Figure 3: In the *Space settings* of Confluence ...

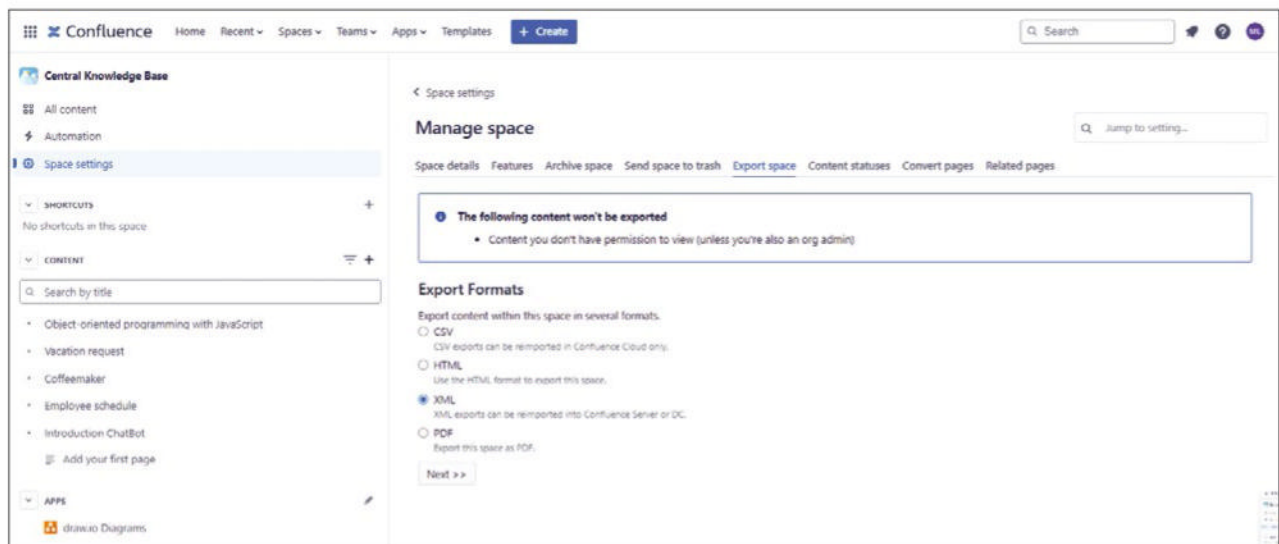


Figure 4: ... you will also find the settings for data export.

archive is stored on the BlueSpice server or the administrator's working machine. The subdirectories `input` and `workspace` must be created, and the Confluence archive is unpacked to `input`. Now it is time to get the migration tools from GitHub [7] and test their functionality; you will need the well-known universal documentation tool `pandoc` and more (all documented in the `README.md` file).

The scripts prepare the data for import, which ends up in the `workspace/result` subdirectory. Now the `migrate-confluence` migration script (from GitHub) can be started.

Starting the Migration

The `migrate-confluence.phar` file is called four times in succession: once to analyze the data (`analyze`, Figure 5), then to extract (`extract`), convert (`convert`), and

finally compose in the new format (`compose`).

The following illustrations show the syntax in examples and the feedback from the system for a simple Confluence space, as well as the result in the directory tree. The parameter `--config=config.yaml` in the analysis is only necessary if you have defined special configuration options in a YAML file. If you use it, you must specify it in the first three steps. The command

```
php migrate-confluence.phar 2
analyze --src=input --dest=workspace
```

analyzes the data and stores files with this information in the workspace directory. The `analyze` parameter prepares the export. The `--config=` parameter (as used in Figure 5) is only necessary if the optional configuration mentioned earlier is used; otherwise, it can be

omitted. The output provides information about the exported content. In the second step, the command,

```
php migrate-confluence.phar 2
extract --src=input --dest=workspace 2
--config=config.yaml
```

takes the exported and analyzed data from Confluence and stores it in the destination – in this case the `workspace` directory (Figure 6). Now, the command

```
php migrate-confluence.phar 2
convert --src=workspace 2
--dest=workspace 2
--config=config.yaml
```

takes the data from workspace and converts it into a format that can be imported by BlueSpice (Figure 7). The `workspace` directory now usually contains numerous raw files with the

```

/data/hallowelt# php /opt/migrate-confluence.phar analyze --src=input --dest=workspace --config=config.yaml
Source: /data/hallowelt/input
Destination: /data/hallowelt/workspace

Fetching file list ...done.

Finding users
- 'Margit.Link-rodrique' (ID:624c4eb01da0e100713d309c)
- '6035864ce2020c0070b5285b' (ID:8a7f808a7a045426017a047e745f0023)
- '5b70c8b80fd0ac05d389f5e9' (ID:8a7f8089759cb58301759d3871b3004e)

Finding namespaces
- CKB (ID:698843143)

Finding pages
- 'CKB:Coffeemaker' (ID:699105303)
- 'CKB:Main_Page' (ID:698843436)
- 'CKB:Vacation_request' (ID:699105281)
- 'CKB:Object-oriented_programming_with_JavaScript' (ID:698581004)
- 'CKB:Introduction_ChatBot' (ID:698417166)
- 'CKB:Employee_schedule' (ID:698417154)

Finding attachments
- 'CKB_CKB_unknown'

Finding users
- 'Margit.Link-rodrique' (ID:624c4eb01da0e100713d309c)
- '6035864ce2020c0070b5285b' (ID:8a7f808a7a045426017a047e745f0023)
- '5b70c8b80fd0ac05d389f5e9' (ID:8a7f8089759cb58301759d3871b3004e)

Finding namespaces
- MS (ID:699662340)

Finding pages
- 'ISM:Management-system/Work_instructions/Preparing an offer' (ID:699105475)
- 'ISM:Management-system/Risk_management/Risk "Mobile devices"' (ID:699105513)
- 'ISM:Management-system/Risk_management' (ID:699105498)
- 'ISM:Main_Page' (ID:699662633)
- 'ISM:Management-system/Risk_management/Incident - Loss of notebook' (ID:699105530)
- 'ISM:Management-system/Introduction' (ID:699105331)
- 'ISM:Management-system' (ID:699105314)
- 'ISM:Management-system/Introduction/Process_map' (ID:699105361)
- File '699662710' (ISM:Management-system/Introduction/Process_map/Prozesslandkarte.drawio.png) not found
- File '699170838' (ISM:Management-system/Introduction/Process_map/Prozesslandkarte.drawio) not found
- File '699629576' (ISM:Management-system/Introduction/Process_map-Prozesslandkarte.drawio.tmp) not found
- File '698810393' (ISM:Management-system/Introduction/Process_map-Prozesslandkarte.drawio.tmp) not found
- File '699662699' (ISM:Management-system/Introduction/Process_map/Prozesslandkarte.drawio) not found
- 'ISM:Management-system/Introduction/ISO_9001_Certification' (ID:699105346)
- 'ISM:Management-system/Organizational structure' (ID:699105393)
- 'ISM:Vorlage - Fehlerbehebungsartikel' (ID:699662669)
- 'ISM:Management-system/Process_organization (Process_descriptions)/Sales_process' (ID:699105425)
- File '698613812' (ISM:Management-system/Process_organization (Process_descriptions)_Sales_processUnbenanntes_Diagramm-1723543301910.drawio.png) not found
- File '699432980' (ISM:Management-system/Process_organization (Process_descriptions)_Sales_processUnbenanntes_Diagramm-1723543301910.drawio) not found
- File '699400240' (ISM:Management-system/Process_organization (Process_descriptions)_Sales_processUnbenanntes_Diagramm-1723543301910.drawio) not found
- File '699531318' (ISM:Management-system/Process_organization (Process_descriptions)_Sales_processUnbenanntes_Diagramm-1723543301910.drawio.png) not found
- 'ISM:Management-system/Process_organization (Process_descriptions)' (ID:699105410)
- 'ISM:Management-system/Work_instructions' (ID:699105460)

Finding attachments
- 'ISM_MS_unknown'
Done.
/data/hallowelt#

```

Figure 5: The BlueSpice migration assistant analyzes the data.


```

/data/hallowelt# php /opt/migrate-confluence.phar convert --src=workspace --dest=workspace --config=config.yaml
Source: /data/hallowelt/workspace
Destination: /data/hallowelt/workspace

Fetching file list ...done.
/data/hallowelt/workspace/content/raw/698417155.mraw
/data/hallowelt/workspace/content/raw/698417164.mraw
/data/hallowelt/workspace/content/raw/698417167.mraw
/data/hallowelt/workspace/content/raw/698417186.mraw
/data/hallowelt/workspace/content/raw/698449925.mraw
/data/hallowelt/workspace/content/raw/698449931.mraw
/data/hallowelt/workspace/content/raw/698449948.mraw
/data/hallowelt/workspace/content/raw/698449951.mraw
/data/hallowelt/workspace/content/raw/698482695.mraw
/data/hallowelt/workspace/content/raw/698482709.mraw
/data/hallowelt/workspace/content/raw/698482712.mraw
/data/hallowelt/workspace/content/raw/698482716.mraw
/data/hallowelt/workspace/content/raw/698515464.mraw
/data/hallowelt/workspace/content/raw/698515469.mraw
/data/hallowelt/workspace/content/raw/698548238.mraw

```

Figure 6: The content is being converted.

```

/data/hallowelt# php /opt/migrate-confluence.phar compose --src=workspace --dest=workspace
Source: /data/hallowelt/workspace
Destination: /data/hallowelt/workspace

Processing: CKB:Coffemaker

Getting '699105304' body content...

Processing: CKB:Main_Page

Getting '698843437' body content...

Processing: CKB:Vacation_request

```

Figure 7: The compose command compiles the raw data extracted from Confluence and stores it as XML files.

prepared Confluence export. The last step (if everything went well) is to use the compose option,

```

php migrate-confluence.phar 2
  compose --src=workspace 2
    --dest=workspace

```

to assemble the various elements. The successfully converted export can now be found under `workspace/result`.

Preparing Namespaces

If everything has worked up to this

point, you can create the necessary namespaces in BlueSpice, check the support for file types (by their file extensions), and import images. The required namespaces can be seen either during the analyze step (in the output of the script) or directly in the file `workspace/result/output.xml`. Under Linux, this information is discovered by a simple grep command:

```

grep -in "<title>" 2
workspace/result/2
output.xml

```

The resulting list shows all titles and namespaces (e.g., *Management:Work instructions*). The syntax here is the same as used by Wikipedia: *Management* is the namespace, and the colon separates it from the *Work instructions* page. All namespaces found must now be created in the page *Special:NamespaceManager* (aka Namespace manager; Figure 8).

File Formats and Endings

The next step is to ensure in the BlueSpice configuration manager (*Special:BlueSpiceConfigManager*) that all required file extensions (i.e., all media and file formats that were previously embedded in Confluence) are permitted (Figure 9). Mind that the BlueSpice importer is selective. Only files from the `workspace/result/images` directory are imported that have file extensions specified here; others are ignored.

Import and Check Success

Now it's time to copy the Confluence data to the BlueSpice server (e.g., to its `/tmp` folder). Figure 10 shows the output of the command

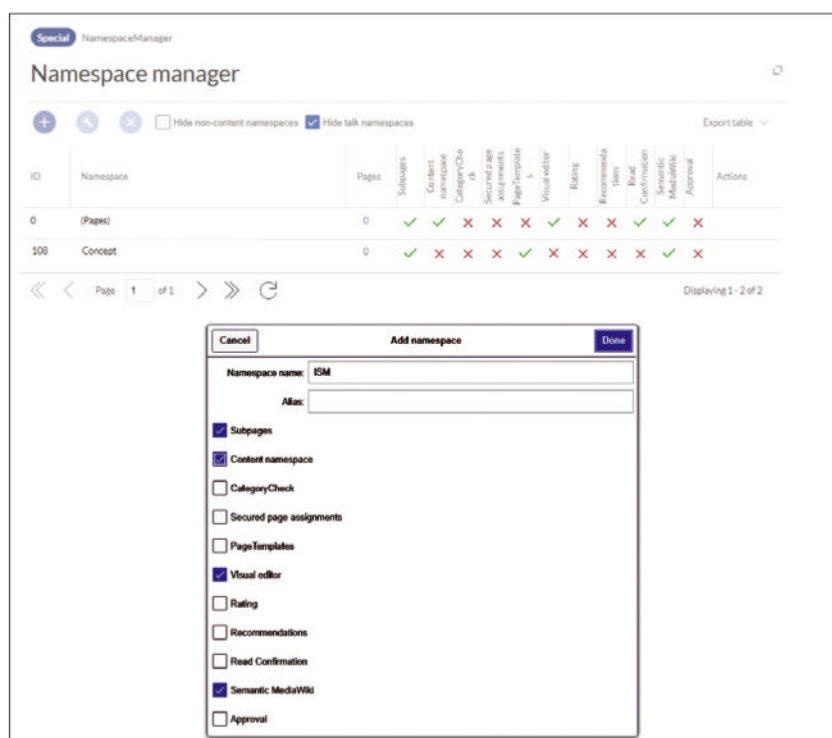


Figure 8: The required namespaces must be created in the Namespace manager (on the wiki page, *Special:NamespaceManager*) of the BlueSpice Wiki before the import.

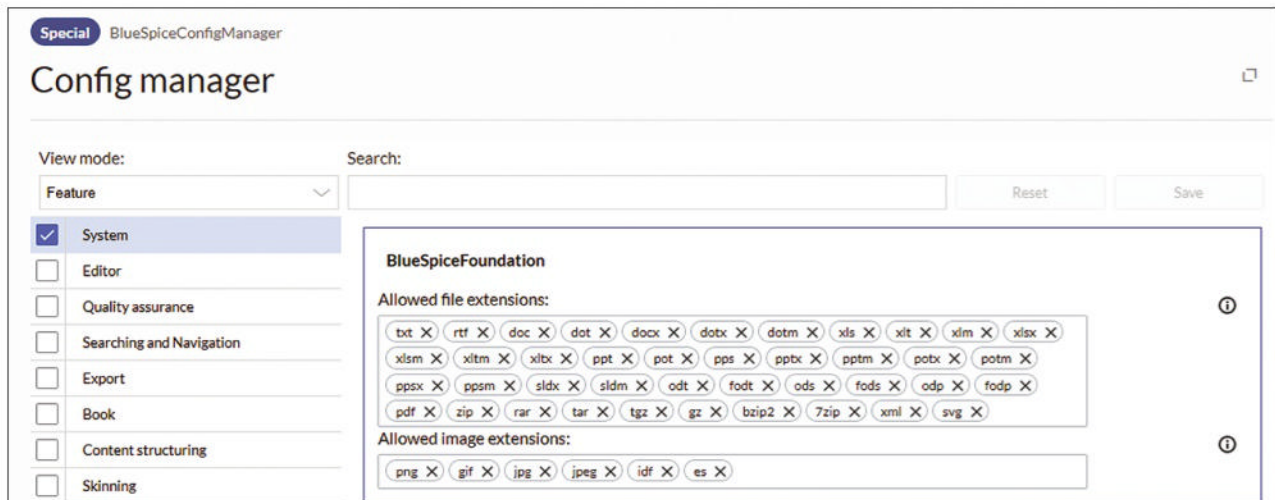


Figure 9: In the configuration of BlueSpice, you have to ensure that all media file formats that were used in Confluence are listed. Only those will be imported.

```
php maintenance/2
importImages.php /tmp/2
result/images/
```

```
./app/c/migrate-confluence-test# php maintenance/importImages.php cache/result/images/
Importing Files
```

Figure 10: Data import of media files on the wiki server.

for the data import in the installation path of the wiki. Each line represents a media file that it has imported successfully.

In the last step, `importDump`, the migration script pulls the content from the migration result XML file into its database with the command:

```
php maintenance/importDump.php 2
/tmp/result/output.xml
```

Now all the data has been imported from Confluence to BlueSpice. Finally, call `php maintenance/runJobs.php` to processes any tasks that have arisen. To refresh various tables that contain links for images and to update the search index, enter:

```
php maintenance/rebuildall.php
php extensions/BlueSpiceExtendedSearch/2
maintenance/rebuildIndex.php
```

Finally, call

```
php maintenance/runJobs.php
```

to processes any tasks that have arisen. It might be necessary to run this script more than one time.

Conclusion

If your organization uses the Atlassian Confluence collaboration platform and you aren't interested in following Atlassian into the cloud, consider BlueSpice as an alternative. BlueSpice is powered by the same technology behind Wikipedia and provides free and easy-to-use migration tools for moving your data. Keep in mind that this kind of migration almost always requires manually reworking and creating data, especially when it comes to user and metadata.

With BlueSpice, thanks to open source, you can even adapt the code

Author

Markus Feilner is a consultant for open source strategies in Regensburg, Germany, with experience working with Linux since 1994. Markus is now grommunio's Open Source Ambassador and previously was deputy editor-in-chief for the German-language *Linux-Magazin*. His company, Feilner IT, focuses on solving problems on OSI Layers 8 to 11.



of the migration wizards, as institutions such as NASA [8] and the University of Illinois [9] have done. ■

Info

- [1] openDesk: <https://interoperable-europe.ec.europa.eu/collection/open-source-observatory-osor/opendesk/>
- [2] XWiki Confluence Migration Toolkit: <https://store.xwiki.com/xwiki/bin/view/Extension/Confluence%20Migration%20Toolkit/>
- [3] Confluence to XWiki migration: <https://xwiki.com/en/confluence-to-xwiki-migration/>
- [4] Confluence Migrator (Pro): <https://xwiki.com/en/Blog/Easiest-migration-from-Confluence-to-XWiki/>
- [5] Hallo Welt!: <https://hallowelt.com/en/>
- [6] Confluence to BlueSpice migration: <https://bluespice.com/confluence-migration-process/>
- [7] migrate-confluence: <https://github.com/hallowelt/migrate-confluence>
- [8] NASA and MediaWiki: <https://diff.wikimedia.org/2016/05/05/mediawiki-nasa/>
- [9] University of Illinois wiki: <https://itaccessibility.illinois.edu/ewh/meeting/2022/01/11/>



Storage across the network with iSCSI and Synology DiskStation Manager

Across the Block

The iSCSI protocol lets you access block storage across a network connection. We show you how to connect a Debian 12 system with a Synology storage device over iSCSI. By Daniel LaSalle

The Internet small computer systems interface (iSCSI) [1] is a block-level [2] interfacing protocol used in conjunction with any network-attached storage (NAS) unit. As the name implies, iSCSI uses the locally focused SCSI block storage protocol and adds the ability to reference resources across the network. By issuing SCSI commands over the TCP/IP suite, iSCSI makes it possible to integrate a remote mount-point located anywhere on the Internet, including in another time zone. The iSCSI protocol essentially lets you treat remote storage as if it were local. In other words, you can set up a storage area network (SAN) with existing resources – and without a lot of expensive equipment.

The iSCSI bus is a good fit for heavy read and write scenarios (e.g., high-availability (HA) environments). Even though iSCSI naturally inherits the flaws of TCP/IP, you can enhance the transmission capacity by adding additional network interface cards (NICs).

My Favorite Recipe

For an example of iSCSI in the real world, I will show you how to integrate a Synology DiskStation storage device with a Debian v12 “bookworm” environment. I assume you have already purchased drives and a NAS enclosure, and you already have a working Debian client system. Although this article is focused on Synology, the steps for configuring iSCSI in other environments are similar. My newly assembled test environment will be, on the server side,

- 1x Synology DiskStation DS414j
- 4x WDC WD201KRYZ

and on the client side:

- 1x Debian bookworm on a custom built system

Synology uses the proprietary DiskStation Manager (DSM) interface, which I will refer to in this article; other vendors offer similar GUI configuration tools.

Getting Started

Regardless of whether your server is proprietary or self-built, you will

want to tackle the following tasks before proceeding:

- Always run the latest and greatest of everything available on both the client and the server side
- Communicate only on the iSCSI port from the server side
- Enable firewall and possibly set a specific allow/block list on both the client and the server side
- Do not allow drives to fall into sleep mode on the server side
- Make the server as quiet as possible

Therefore, the first thing you’ll want to do is harden your NAS device by applying the latest system update with the *Update & Restore* option group in the Control Panel. After the device spends some time patching and restarts, you should begin to strip all unwanted features.

So that only the iSCSI sockets reach the data of your device, you need to deactivate everything that has to do with file sharing under *File Services* located under the Control panel (**Figure 1**). From there, uncheck all entries located under any of the available

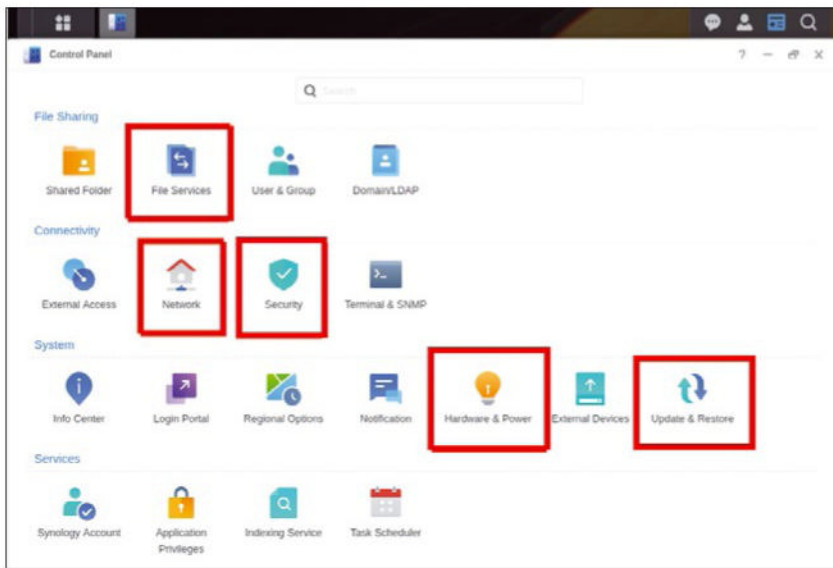


Figure 1: The Synology Control Panel bundled with these DiskStation devices are packed with relevant and modern features, but for the sake of this article, I only use the five groups marked in red.

tabs at the top (in my case, these are *SMB*, *AFP*, *NFS*, *FTP*, *rsync*, and *Advanced*).

The next milestone is located under *Security* (Figure 1), where you can enable the device's firewall while defining allow and block lists on the basis of specific IPs or IP ranges. Moving forward, the last portion of the process will be to ensure NAS availability. As already specified, if you are in an HA environment, you will want to make sure your drives never fall into inactivity. To do so you will need to disable the power save mode feature located under the *Hardware & Power* group (Figure 1). The last configurations to apply are located under the *General* tab in the same group and concerns fan noise. Synology devices allow you to select from four performance profiles: *Full-Speed Mode*, *Cool Mode*, *Quiet Mode*, and *Low-Power Mode*. In this case you will select the *Quiet Mode* Option. The final touch will be to make sure you

get alerts when something fails at 3:00am, so you will be notified (or not). The Beep Control option is especially useful if your server device is hosted in the middle of your living room and you are not living alone.

Extra! Extra!

At this point, some of you will be ready to rock and roll, whereas others will feel the need to crank up the robustness of their setup one notch by improving link speeds while pairing their NAS with a battery backup in case of a power failure. In my case, such technical improvements will come to fruition in the form of

creating an adaptive load-balanced link that will be aggregated across all four NICs embedded in my device. In the *Network* group options (Figure 1), simply click the *Create* button, select all of the available NICs, and aggregate them into a single instance. When completed, stamp a static IPv4 address on this bad boy while making sure all IPv6 is disabled.

My local electricity provider is not always reliable, and to mitigate their service degradations I have coupled my NAS unit with an uninterruptible power supply (UPS) that will communicate with it over a regular USB-A cable. Afterward, I visited *Hardware & Power* | *UPS* to enable UPS support.

Storage Manager and Pool

Regardless of your hardware setup, you are now at the point where the system portion of your server is in place, and you could freeze it in this state. The next step will be to configure all of the storage on which you've spent your hard-earned dollars. In that respect, the last steps server-side are to configure the storage pool and volume and map it to a newly created iSCSI qualified name (IQN), which, simply put, will be a unique hardware identifier. Following this mapping, the NAS will start accepting incoming connections on the default iSCSI port, which is TCP/3260.

The combination of that IQN and TCP/3260 is the socket that allows client-server communication. It is

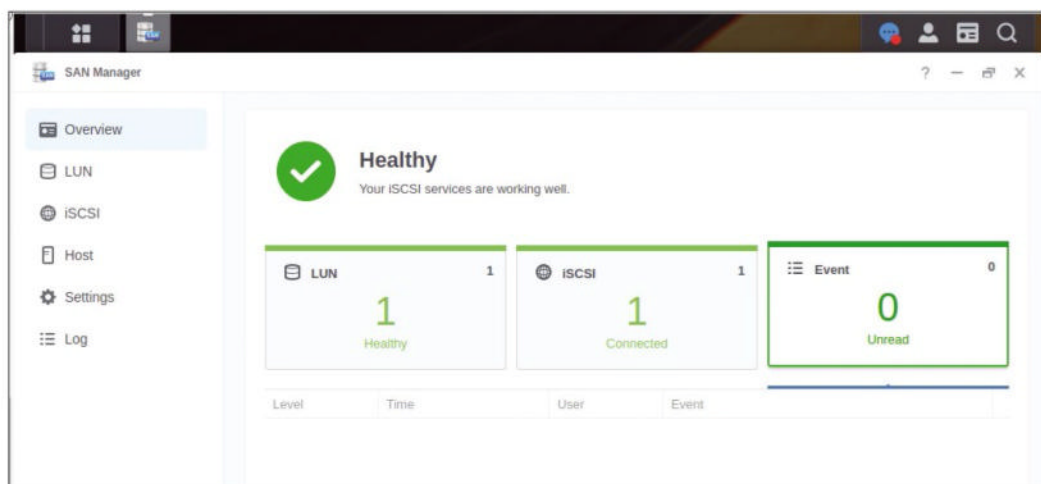


Figure 2: Victory! The SAN Manager now shows that all configurations are correctly in place on the NAS.

```

root@synology:~# df -H
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        2.6G  1.2G  1.4G   46% /
devtmpfs        249M    0  249M    0% /dev
tmpfs           254M   17k  254M    1% /dev/shm
tmpfs           254M  41M  213M   16% /run
tmpfs           254M    0  254M    0% /sys/fs/cgroup
tmpfs           254M 623k  253M    1% /tmp
/dev/vg1/volume_1 8.0T  8.0T  258M 100% /volume1
root@synology:~# cd /volume1/@iSCSI/LUN/FILE/cc6445f1-e2db-4e83-a68c-458350342000/
root@synology:/volume1/@iSCSI/LUN/FILE/cc6445f1-e2db-4e83-a68c-458350342000# ll
total 772905/540
drwx----- 2 root root      4096 Jun 23 16:53 .
drwx----- 3 root root      4096 Jun 23 16:50 ..
-rw----- 1 root root 1099511627776 Jul 11 10:47 BU_00000
-rw----- 1 root root 1099511627776 Jul 10 22:12 BU_00001
-rw----- 1 root root 1099511627776 Jul 10 22:11 BU_00002
-rw----- 1 root root 1099511627776 Jul 11 10:47 BU_00003
-rw----- 1 root root 1099511627776 Jul 10 22:12 BU_00004
-rw----- 1 root root 1099511627776 Jun 27 11:40 BU_00005
-rw----- 1 root root 1099511627776 Jun 27 12:17 BU_00006
-rw----- 1 root root 217969590272 Jun 27 12:25 BU_00007
root@synology:/volume1/@iSCSI/LUN/FILE/cc6445f1-e2db-4e83-a68c-458350342000# uname -a
Linux synology 3.2.101 #42962 SMP Mon May 29 14:34:16 CST 2023 armv7l GNU/Linux synology_concerto2k_ds414j
root@synology:/volume1/@iSCSI/LUN/FILE/cc6445f1-e2db-4e83-a68c-458350342000# cat /volume1/@iSCSI/LUN/FILE/iscsi_lun.conf
root@synology:/volume1/@iSCSI/LUN/FILE/cc6445f1-e2db-4e83-a68c-458350342000# cat /volume1/@iSCSI/LUN/iscsi_lun.conf
[LUN_cc6445f1-e2db-4e83-a68c-458350342000]
lun=1
rootpath=/volume1
name=BU
uid=cc6445f1-e2db-4e83-a68c-458350342000
vpd_unit_serial=cc6445f1-e2db-4e83-a68c-458350342000
bytes=7914550984704
restored_time=0
devtype=1
dev_attrs=emulate_3pc:0,emulate_tps:0,emulate_caw:1,emulate_tpu:0,emulate_fua_write:0,emulate_sync_cach
e:0,can_snapshot:0
pre_alloc=yes
bkb_obj=0
vaai_support=no
direct_io_pattern=0
description=Backup
dev_config=
inquiry_prod=Storage
dev_qos=iops_enable:0,dev_limit:0,dev_reservation:0,dev_weight:0
create_from=
root@synology:/volume1/@iSCSI/LUN/FILE/cc6445f1-e2db-4e83-a68c-458350342000#

```

Figure 3: A working configuration from the server side of the iSCSI channel.

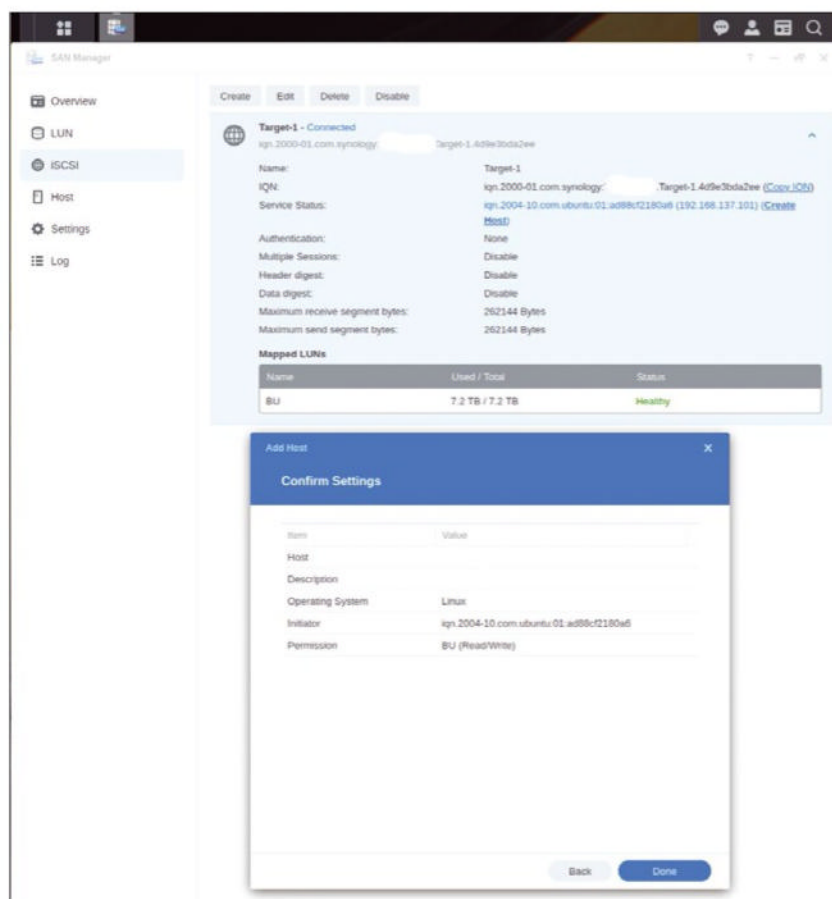


Figure 4: One final step on the server side is the creation of the IQN. Don't forget to bind!

important to understand that every party member of the iSCSI dance needs to have one of these addresses. Both server and client require a unique IQN properly configured on both ends to be able to attach (and manage) remote devices from the Debian system.

To do so, you first need to access your server's storage manager, which, in my case, is located at the top left-hand side of the DSM Main Menu (Figure 2). Next, create both a storage pool and a volume using the built-in guidance wizard, which prompts you to select the preferred RAID type. In my case, JBOD, RAID 0, RAID 1, RAID 5, RAID 6, and even a proprietary solution called SHR were available.

For data flow and business continuity, I opted to use RAID 6, selecting all four of my drives to create one big happy pool and then clicked *Next*. I was then asked to specify the maximum size, for which, easily enough, a button is provided to hit the max, after which the desired storage pool was then created successfully. Depending on the size and speed of your drives, this step will take from a few minutes up to several hours to complete its creation.

Once finished, the volume shows up under the storage pool of your device. You then have to bind this newly created storage pool with an existing iSCSI logical unit number (LUN) from the SAN Manager icon located under the top left-hand menu.

From the SAN Manager page, you will also be able to confirm in the *Overview* section (Figure 2) the status of any currently existing (or not) LUNs. This dashboard also helps end-users get a quick assessment of the number of connected LUNs, alongside any possible errors that might be plaguing their NAS. Clicking on the *iSCSI* entry in the left sidebar of the SAN Manager will allow you to launch a creation wizard that lets you create a new LUN. The reason behind this step is to initiate a new LUN, to which you can later attach your newly created storage volume comprising your (in my case, four) drives.

Because Synology has launched itself into the storage business, it is possible your drives will not be on their compatibility list (as mine weren't).

However, that will not be a concern; rest assured that you can safely move forward ignoring anything having to do with drive compatibility.

```
[/usr/sbin]$ cd /etc/iscsi
[/etc/iscsi]$ ll
Permissions Size User Date Modified Name
-rw-r----- 349 root 24 Jun 17:16 initiatorname.iscsi
-rw-r--r-- 14k root 8 Apr 10:46 iscsid.conf
drwx----- - root 24 Jun 21:37 nodes
drwx----- - root 24 Jun 17:49 send_targets
[/etc/iscsi]$ cat iscsid.conf
#
# Open-iSCSI default configuration.
#
# Note: To set any of these values for a specific node/session run
# the iscsiadm --mode node --op command for the value. See the README
# and man page for iscsiadm for details on the --op command.
#
#####
# iscsid daemon config
#####
#
# If you want iscsid to start the first time an iscsi tool
# needs to access it, instead of starting it when the init
# scripts run, set the iscsid startup command here. This
# should normally only need to be done by distro package
# maintainers. If you leave the iscsid daemon running all
# the time then leave this attribute commented out.
#
# Default for Fedora and RHEL. Uncomment to activate.
# iscsid.startup = /bin/systemctl start iscsid.socket iscsiui.socket
#
# Default for Debian and Ubuntu. Uncomment to activate.
# iscsid.startup = /bin/systemctl start iscsid.socket
#
# Default if you are not using systemd. Uncomment to activate.
# iscsid.startup = /usr/bin/service start iscsid
#
# Check for active mounts on devices reachable through a session
# and refuse to logout if there are any. Defaults to "No".
# iscsid.safe_logout = Yes
#
# Only require UID auth for MGMT IPCs, and not username.
# Checking username is a legacy security practice, and is on the path
# to deprecation.
# Set to "No" for legacy compatibility.
# Defaults to "Yes".
# iscsid.ipc_auth_uid = No
#
#####
# NIC/HBA and driver settings
#####
# open-iscsi can create a session and bind it to a NIC/HBA.
# To set this up see the example iface config file.
#
#####
# Startup settings
#####
#
# To request that the iscsi service scripts startup a session, use "automatic":
# node.startup = automatic
#
# To manually startup the session, use "manual". The default is manual.
# node.startup = manual
#
# For "automatic" startup nodes, setting this to "Yes" will try logins on each
# available iface until one succeeds, and then stop. The default "No" will try
# logins on all available ifaces simultaneously.
# node.leading_login = No
#
#####
# CHAP Settings
#####
#
# To enable CHAP authentication set node.session.auth.authmethod
# to CHAP. The default is None.
# node.session.auth.authmethod = CHAP
#
# To configure which CHAP algorithms to enable, set
# node.session.auth.chap_algs to a comma separated list.
# The algorithms should be listed in order of decreasing
# preference - in particular, with the most preferred algorithm first.
# Valid values are MD5, SHA1, SHA256, and SHA3-256.
# The default is MD5.
# node.session.auth.chap_algs = MD5 SHA1 SHA256 SHA3-256
```

Figure 5: A look at the iSCSI protocol configuration from the client system perspective.

Once this step is complete, you should see the newly created iSCSI target displayed on the SAN Manager dashboard. Rejoice! Again, as quickly as this can happen with state-of-the-art hardware, older systems might require much more time before your LUN is created and made accessible.

When the build process finally completes, the NAS displays the much-anticipated LUN. The last thing you'll need to do in the *Edit | Mapping* tab in DSM is bind the iSCSI target by selecting (and applying) the IQN target to the recently built LUN. In my case, none existed, so I had to create one by following the user interface's creation wizard (Figures 3 and 4).

Client

Back on the 'bookworm' system, I want to deploy the open-iscsi package [3] (the RFC 3720 iSCSI implementation for Linux). To create a secure infrastructure, couple it by installing the cryptsetup package, as well. After the iSCSI protocol has successfully attached the remote drive to the client (Listings 2 and 3), I format it with LUKS [4], followed by the ext4 filesystem. Only then can I really start relying on my entrusted mountpoint. To install these packages, issue the command:

```
$ sudo apt install cryptsetup open-iscsi -y
```

Once both packages are installed and you are back at the prompt, you can truly begin to profit from your iSCSI journey (Figure 5). Issuing the `lsblk`

Listing 1: Client System Before Attaching an iSCSI Device

```
$ uname -a
Linux DANSBOX 6.8.0-35-generic #35-Ubuntu SMP PREEMPT_
DYNAMIC Mon May 20 15:51:52 UTC 2024 x86_64 x86_64
x86_64 GNU/Linux
$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sda 8:0 0 25G 0 disk
|-sda1 8:1 0 1M 0 part
|-sda2 8:2 0 25G 0 part /var/snap/firefox/
common/host-hunspell
```



```
[ 18.488547] kernel: iscsi: registered transport (tcp)
[ 18.489885] kernel: scsi host9: iSCSI Initiator over TCP/IP
[ 18.512287] kernel: scsi 9:0:0:1: Direct-Access      SYNOLOGY Storage      3.1 PQ: 0 ANSI: 5
[ 18.513922] kernel: sd 9:0:0:1: Attached scsi generic sg1 type 0
[ 18.514348] kernel: sd 9:0:0:1: [sdb] 15458107392 512-byte logical blocks: (7.91 TB/7.20 TiB)
[ 18.514530] kernel: sd 9:0:0:1: [sdb] Write Protect is off
[ 18.514534] kernel: sd 9:0:0:1: [sdb] Mode Sense: 3b 00 00 00
[ 18.514914] kernel: sd 9:0:0:1: [sdb] Write cache: disabled, read cache: enabled, doesn't support DPO o
r FUA
[ 18.525071] kernel: sd 9:0:0:1: [sdb] Attached SCSI disk
```

Figure 6: After all has been put into place successfully, the client system will automatically recognize the iSCSI drive at the next client boot. Moving forward, the `dmesg` command will show the remote attachment of a drive being automatically made by the client. Listing 3 confirms the new availability of a 7.2TB storage unit.

command from the client reveals the kind of direct storage units available on this node. As you can see from [Listing 1](#) no remote units are currently attached.

In my case, I also want to configure my client's firewall to ensure that traffic goes through port TCP/3260:

```
$ sudo ufw allow 3260/tcp
```

Because I set a static IP on my device earlier, I can also whitelist the entire host:

```
$ sudo ufw allow from 192.168.1.10
```

Perhaps the most secure method is to confine it to the static IP and port, instead of including the entire host:

```
$ sudo ufw allow from 192.167.1.10/24 to any port 3260
```

Implementing all of these steps and restarting the service with

```
$ sudo systemctl status iscsi.service
```

should give you the green light and show that the service is up and running, as you can see in [Listing 2](#), which also confirms the IQN served by the NAS.

The time has come to establish the first iSCSI session and attach the LUN. To do so, send a discovery command with the `iscsiadm` binary:

```
$ sudo /sbin/iscsiadm --mode discovery --op update --type sendtargets --portal 192.168.1.10
```

You can unpack this command by reading the `iscsiadm` man page [\[5\]](#). Briefly, it allows you to discover the IQN of your server and, therefore, allows you to attach the unit to the client. You can confirm that you do have a new storage unit by replaying the `lsblk` command ([Figure 6](#)).

Because I am working in a secure environment, I will go ahead and create a LUKS container:

```
$ sudo cryptsetup -y -v luksFormat /dev/sdb
```

Once (or while) this process completes, you need to make sure

Listing 2: Forcing an iSCSI Connection

```
$ sudo /sbin/iscsiadm -m node -T iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee -p 192.168.1.10 -l
Logging in to [iface: default, target: iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee, portal: 192.168.1.10,3260]
Login to [iface: default, target: iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee, portal: 192.168.1.10,3260] successful.
$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sda 8:0 0 25G 0 disk
|-sda1 8:1 0 1M 0 part
|-sda2 8:2 0 25G 0 part /var/snap/firefox/common/host-hunspell
sdb 8:16 0 7.2T 0 disk
```

Listing 3: Confirming iSCSI Status

```
START<<<
$ sudo systemctl status iscsi.service
● open-iscsi.service - Login to default iSCSI targets
   Loaded: loaded (/usr/lib/systemd/system/open-iscsi.service; enabled; preset: enabled)
   Active: active (exited) since Thu 2024-07-04 20:55:15 EDT; 5 days ago
     Docs: man:iscsiadm(8)
           man:iscsid(8)
   Main PID: 1696 (code=exited, status=0/SUCCESS)
    CPU: 10ms

Jul 04 20:55:15 DANSBOX systemd[1]: Starting open-iscsi.service - Login to default iSCSI targets...
Jul 04 20:55:15 DANSBOX iscsiadm[1646]: Logging in to [iface: default, target: iqn.2000-01.com.synology]
Jul 04 20:55:15 DANSBOX iscsiadm[1646]: Login to [iface: default, target: iqn.2000-01.com.synology:MYDEVICE]
Jul 04 20:55:15 DANSBOX systemd[1]: Finished open-iscsi.service - Login to default iSCSI targets.
>>>END
```

Listing 4: Discover Running Sessions

```
$ sudo /sbin/iscsiadm -m node -T iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee -p 192.168.1.10 -l
Logging in to [iface: default, target: iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee, portal: 192.168.1.10,3260]
Login to [iface: default, target: iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee, portal: 192.168.1.10,3260] successful.

$ sudo iscsiadm -m session
[~]
tcp: [15] 192.168.1.10:3260,1 iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee (non-flash)
```

everything is in place to mount this new drive (e.g., creating a new path; changing its ownership, possibly to your unprivileged user; and setting permissions to 777). When LUKS formatting is complete and your system has everything needed to host the mount, you can mount the LUKS partition by issuing (in my case),

```
$ sudo cryptsetup luksOpen /dev/sdb BU
```

which associates the device `/dev/sdb` with the label `BU`; moving forward, you can reference the device by the more simplistic label name instead of having to work with its longer name. Next, format the LUKS container with the filesystem of your choice. In my case, I opted for the venerable `ext4`:

```
$ sudo mkfs -t ext4 /dev/sdb
```

Of course, LUKS is agnostic to whichever filesystem layer you end up choosing. Only at this point will everything be configured properly on the client system. You can also become better acquainted with your cryptographic footprint with,

```
$ sudo cryptsetup -v status BU
```

```
[~]$ sudo cryptsetup luksDump /dev/sdb
LUKS header information
Version:      2
Epoch:       3
Metadata area: 16384 [bytes]
Keyslots area: 16744448 [bytes]
UUID:         eb4d4d7b-b0e0-452c-9339-4e340cc0ffb8
Label:        (no label)
Subsystem:    (no subsystem)
Flags:        (no flags)

Data segments:
0: crypt
   offset: 16777216 [bytes]
   length: (whole device)
   cipher: aes-xts-plain64
   sector: 512 [bytes]

Keyslots:
0: luks2
   Key:        512 bits
   Priority:    normal
   Cipher:     aes-xts-plain64
   Cipher key: 512 bits
   PBKDF:      argon2id ←
   Time cost:  7
   Memory:    1048576
   Threads:   4
   Salt:       14 e6 14 a1 58 22 0a 08 4c 05 eb 50 69 04 48 a7
               7c b4 4e 83 34 9c 09 6a ab 0b f9 ac 85 47 b3 d3
   AF stripes: 4000
   AF hash:    sha256
   Area offset: 32768 [bytes]
   Area length: 258048 [bytes]
   Digest ID:  0

Tokens:
Digests:
0: pbkdf2
   Hash:       sha256
   Iterations: 185129
   Salt:        02 f2 24 bd 71 2e f9 3b e2 40 e1 be f9 5f bd 20
               5c 09 b3 54 a0 ab 78 11 f0 5e ed 3c 85 51 e8 89
   Digest:      9e 84 92 74 3c 68 d3 97 ed 0c e8 4c 88 60 3e fa
               1f f1 51 b4 20 58 c5 a6 3e 9a 00 63 0e 05 1d 28

[~]$
```

Figure 7: When dealing with any LUKS container, be sure the password-based key derivation function (PBKDF) value is set to `argon2id`; otherwise, it will be more vulnerable to brute force attacks.

```
iface.dhcp_vendor_id_state = <empty>
iface.dhcp_vendor_id = <empty>
iface.dhcp_slp_da = <empty>
iface.fragmentation = <empty>
iface.gratuitous_arp = <empty>
iface.incoming_forwarding = <empty>
iface.tos_state = <empty>
iface.tos = 0
iface.ttl = 0
iface.delayed_ack = <empty>
iface.tcp_nagle = <empty>
iface.tcp_wsf_state = <empty>
iface.tcp_wsf = 0
iface.tcp_timer_scale = 0
iface.tcp_timestamp = <empty>
iface.redirect = <empty>
iface.def_task_mgmt_timeout = 0
iface.header_digest = <empty>
iface.data_digest = <empty>
iface.immediate_data = <empty>
iface.initial_r2t = <empty>
iface.data_seq_inorder = <empty>
iface.data_pdu_inorder = <empty>
iface.erl = 0
iface.max_receive_data_len = 0
iface.first_burst_len = 0
iface.max_outstanding_r2t = 0
iface.max_burst_len = 0
iface.chap_auth = <empty>
iface.bidl_chap = <empty>
iface.strict_login_compliance = <empty>
iface.discovery_auth = <empty>
iface.discovery_logout = <empty>
node.discovery_address = 192.168.137.69
node.discovery_port = 3260
node.discovery_type = send_targets
node.session.initial_cmds_n = 0
node.session.initial_login_retry_max = 8
node.session.xmit_thread_priority = -20
node.session.cmds_max = 128
node.session.queue_depth = 32
node.session.nr_sessions = 1
node.session.auth.authmethod = None
node.session.auth.username = <empty>
node.session.auth.password = <empty>
node.session.auth.username_in = <empty>
node.session.auth.password_in = <empty>
node.session.auth.chap_algs = MD5
node.session.timeo.replace_timeout = 120
node.session.err_timeo.abort_timeout = 15
node.session.err_timeo.lu_reset_timeout = 30
node.session.err_timeo.tgt_reset_timeout = 30
node.session.err_timeo.host_reset_timeout = 60
node.session.iscsi.FastAbort = Yes
node.session.iscsi.InitialR2T = No
node.session.iscsi.ImmediateData = Yes
node.session.iscsi.FirstBurstLength = 262144
node.session.iscsi.MaxBurstLength = 16776192
node.session.iscsi.DefaultTime2Retain = 0
node.session.iscsi.DefaultTime2Wait = 2
node.session.iscsi.MaxConnections = 1
node.session.iscsi.MaxOutstandingR2T = 1
node.session.iscsi.ERL = 0
node.session.scan = auto
node.session.reopen_max = 0
node.conn[0].address = 192.168.137.69
node.conn[0].port = 3260
node.conn[0].startup = manual
node.conn[0].tcp.window_size = 524288
node.conn[0].tcp.type_of_service = 0
node.conn[0].timeo.logout_timeout = 15
node.conn[0].timeo.login_timeout = 15
node.conn[0].timeo.auth_timeout = 45
node.conn[0].timeo.noop_out_interval = 5
node.conn[0].timeo.noop_out_timeout = 5
node.conn[0].iscsi.MaxXmitDataSegmentLength = 0
node.conn[0].iscsi.MaxRecvDataSegmentLength = 262144
node.conn[0].iscsi.HeaderDigest = None
node.conn[0].iscsi.DataDigest = None
node.conn[0].iscsi.IFMarker = No
node.conn[0].iscsi.OFMarker = No
# END RECORD
```

Figure 8: The `iscsiadm` command issued as the root user can show more results than are contained in the default configuration file located under the `/etc/iscsi/nodes/` path.

Listing 5: Unmounting Encrypted iSCSI

```
$ sudo umount /BU
$ sudo cryptsetup luksClose BU
$ sudo /usr/sbin/iscsiadm -m node -T iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee -p 192.168.1.10:3260 -u
Logging out of session [sid: 14, target: iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee, portal: 192.168.1.10,3260]
Logout of [sid: 14, target: iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee, portal: 192.168.1.10,3260] successful.
$ sudo iscsiadm -m session
iscsiadm: No active sessions.
$ df -H
tmpfs          411M  1.4M  410M   1% /run
/dev/sda2       27G   13G   13G  50% /
tmpfs           2.1G    0  2.1G   0% /dev/shm
tmpfs           5.3M    0  5.3M   0% /run/lock
tmpfs           411M  193k  411M   1% /run/user/1000
```

block-level-mounted device, first use the `umount` command to detach the ext4 partition from your system, as you would generally do, and then close the LUKS container with:

```
$ sudo /usr/sbin/cryptsetup
luksClose BU
```

or create more verbose output with

```
$ sudo cryptsetup luksDump /dev/sdb
```

as shown in [Figure 7](#).

Client - Daily Interaction

All iSCSI session connection specifics are displayed by the command:

```
$ sudo /sbin/iscsiadm -m node -o show
```

[Figure 8](#) shows a lot more information

from the iSCSI sessions' properties located under the server folder:

```
/etc/iscsi/nodes/iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee/
192.168.1.10,3260,1/default
```

If you do not want to use a terminal to access the remote drive, you can use the MATE Caja file manager. Given it does not display drives in the left tree panel, you should issue commands from [Listing 4](#).

If you need to detach from a

session:

```
$ sudo /usr/sbin/iscsiadm
-m node
-T iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee
-p 192.168.1.10:3260
-u
```

The entire process is shown in [Listing 5](#).

Troubleshooting

At the end of the day, if you've executed the deployment, step by step, as suggested, you should have been able to integrate your remote drives flawlessly into your client system. During the course of writing this article, one of the problems I faced was attempting to establish an iSCSI session ([Listing 6](#)).

This particular error, alongside any other session-establishing error, will either be caused by a firewall blocking TCP/3260, a mismatch in any of the client-server IQNs, or both. Therefore, one solution will be to check the firewall rules, and another will be to look at the iSCSI target profile located on either the server or client side. The last situation arises mainly when a profile somewhere has been deleted and a re-initiating connection that uses the invalid values is still cached by `iscsi.service`. From a client perspective, the file to examine on the 'bookworm' system

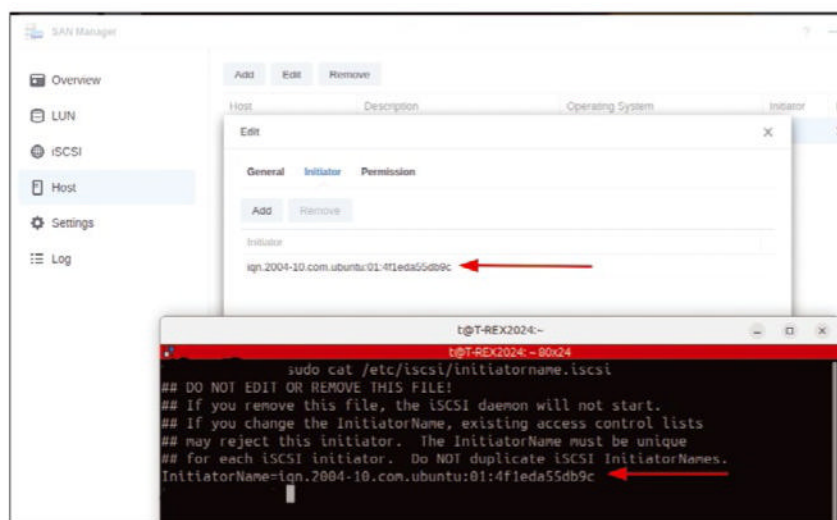


Figure 9: When encountering an iSCSI login failure message while attempting to establish a session on your client, the most common issue will be that `/etc/iscsi/initiatorname.iscsi` from the client side does not match the Host entry value under the SAN Manager on the server side.

Listing 6: Mismatching IQNs

```
$ sudo /sbin/iscsiadm -m node -T iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee -p 192.168.1.10 -l
iscsiadm: initiator reported error (19 - encountered non-retryable iSCSI login failure)
iscsiadm: Could not log into all portals
```


is /etc/iscsi/initiatorname.scsi, whereas from the server's perspective (and in this case), the file would be located in the SAN Manager (Figure 9). Once any change has been made, it will be necessary to restart the iscsi.service on the client side by invoking either one of the two commands:

```
$ sudo systemctl restart iscsid.service
$ sudo service iscsid restart
```

Another fairly common problem you could face is hitting a lock on the mount itself, which makes it impossible to close your newly attached device properly. In the event that you are facing a *umount: target is busy* error message and cannot find anything (with the `lsof` command) to kill that will free this lock, you should invoke the `/usr/bin/fusermount` command paired with the `-u` switch and followed by the mount path in question. If you have no time to waste, you can unmount ungracefully by adding the `-z` switch to send a SIGKILL (Listing 7), which will then cause the lock to be released immediately.

Conclusion

Whether you have decided to opt for a “miracle” solution provided by a vendor or have built your system from the ground up with equipment of your choosing, the challenges are the same: ensuring data confidentiality, integrity, and accessibility. However, whichever path you chose when sailing the sea of digital sovereignty [6], the same considerations of total capacity, access latency, and ease of integration with the current infrastructure will be of concern. The best thing you can do for the future of your organization is to make sure you build a stack that can be adapted, while still allowing a realistic degree of vendor or community support.

In this article, I paired the highly mature and community-driven Open-iSCSI and LUKS projects with one moderately closed solution from a vendor with some level of adaptability.

The conclusion of this setup turned out to be that I was able to pair the best of both worlds: harmonizing a hardware vendor solution with community-driven software. With the deployment of such a device, I am fully confident that all of my stack, from the device's firmware down to any software needed, will be supported for the next several years. ■

Info

- [1] Internet Small Computer Systems Interface: [\[https://en.wikipedia.org/wiki/iSCSI\]](https://en.wikipedia.org/wiki/iSCSI)
- [2] Block-level storage: [\[https://en.wikipedia.org/wiki/Block-level_storage\]](https://en.wikipedia.org/wiki/Block-level_storage)

- [3] Linux Open-iSCSI initiator: [\[https://github.com/open-iscsi/open-iscsi\]](https://github.com/open-iscsi/open-iscsi)
- [4] cryptsetup: [\[https://gitlab.com/cryptsetup/cryptsetup/blob/master/README.md\]](https://gitlab.com/cryptsetup/cryptsetup/blob/master/README.md)
- [5] iscsiadm man page: [\[https://linux.die.net/man/8/iscsiadm\]](https://linux.die.net/man/8/iscsiadm)
- [6] “Digital Sovereignty – a Primer” by Henri Attan, Harvard, July 2021: [\[https://www.harvard.co.uk/digital-sovereignty/\]](https://www.harvard.co.uk/digital-sovereignty/)

Author

Daniel LaSalle was introduced to the command prompt while in 5th grade, but his addiction to technology spans over 30 years. In the last decade, he's been using Linux every day and freelancing as an infrastructure specialist [\[https://www.linkedin.com/in/daniellasalle/\]](https://www.linkedin.com/in/daniellasalle/).

Listing 7: Unmounting the LUKS Mountpoint

```
$ df -H
Filesystem            Size  Used Avail Use% Mounted on
tmpfs                  411M  1.4M  410M   1% /run
/dev/sda2              27G   13G   13G  50% /
tmpfs                  2.1G   0  2.1G   0% /dev/shm
tmpfs                  5.3M   0  5.3M   0% /run/lock
tmpfs                  411M  193k  411M   1% /run/user/1000
/dev/mapper/BU        7.9T   29k  7.5T   1% /BU
$ sudo umount /BU
umount: /BU: target is busy.
$ sudo fusermount -zu /BU
$ df -H
Filesystem            Size  Used Avail Use% Mounted on
tmpfs                  411M  1.4M  410M   1% /run
/dev/sda2              27G   13G   13G  50% /
tmpfs                  2.1G   0  2.1G   0% /dev/shm
tmpfs                  5.3M   0  5.3M   0% /run/lock
tmpfs                  411M  193k  411M   1% /run/user/1000
$ lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda   8:0    0   25G  0 disk
|-sda1 8:1    0    1M  0 part
|-sda2 8:2    0   25G  0 part /var/snap/firefox/common/host-hunspell
sdb    8:16   0   7.2T  0 disk
|-BU    252:3   0   7.2T  0 crypt
$ sudo cryptsetup luksClose BU
$ sudo /usr/sbin/iscsiadm -m node -T iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee -p 192.168.1.10:3260 -u
Logging out of session [sid: 15, target: iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee, portal: 192.168.1.10,3260]
Logout of [sid: 15, target: iqn.2000-01.com.synology:MYDEVICE.Target-1.4d9e3bda2ee, portal: 192.168.1.10,3260] successful.
$ sudo iscsiadm -m session
iscsiadm: No active sessions.
$ df -H
Filesystem            Size  Used Avail Use% Mounted on
tmpfs                  411M  1.4M  410M   1% /run
/dev/sda2              27G   13G   13G  50% /
tmpfs                  2.1G   0  2.1G   0% /dev/shm
tmpfs                  5.3M   0  5.3M   0% /run/lock
tmpfs                  411M  193k  411M   1% /run/user/1000
```



Automating deployments on Proxmox with OpenTofu and cloud-init

Go-Between

Use OpenTofu and cloud-init to deploy virtual machines in a Proxmox hypervisor and populate them automatically with services. By Rubén Llorente

In the old days, before virtualization was practical, if you wanted to run a service, you installed an operating system (OS) on a physical machine and then manually installed the service. If more than one instance of the OS was required, you needed to purchase additional computers and repeat the process manually. As you might imagine, administering a large fleet of computers this way is unworkable in the long run, so automation tools were created. Fully Automatic Installation (FAI) [1] is a good example of an early, generic tool for deploying Linux distributions over a sizable number of machines.

Tools such as FAI created for physical machines are of use when deploying virtual machines, as well, as are distro-specific tools (e.g., *autoinstall* for OpenBSD [2], *presecd* for Debian [3], *Kickstart* for Enterprise Linux-compatible distributions [4], etc.). However, these tools might take a bit too long to build the environment you want to your specifications, because installing every OS from the ground up is a slow process.

In this article, I demonstrate the capabilities of modern automation tools

through example. I deploy a set of web servers and a reverse proxy in a virtual environment hosted by a Proxmox VE hypervisor. (See the “Why Proxmox?” box if you are not familiar with it.)

OG Virtualization

When virtualization became good enough for massive use, the number of OS instances system administrators had to manage multiplied. You no longer had to purchase a computer for each system you wanted to run, because you could just build a virtualization server and run 20 virtual computers. Instances became disposable: You could create a virtual machine for testing some service and discard it 10 minutes later. You could deploy a chat service and a game server for a weekend LAN party and delete everything afterward, virtually free of cost.

The traditional way of creating a virtual machine instance was to run the usual install process, which meant you would boot the virtual machine from a regular install medium and go through every setup step. Again,

this process only works if you create and destroy virtual machines every so often; however, for environments in which virtual environments are counted with more fingers than you have on your hands, that method no longer works.

Enter OpenTofu

Terraform used to be the tool sys admins used when they wanted to deploy services in the cloud or in a virtual environment, especially if they didn’t want a full orchestrator with all the associated complexity and overhead. Terraform is programed in a declarative language, which defines the final desired state. In other words, if you declare that you want your infrastructure to reside on an OpenBSD virtual machine called *rproxy* with three OpenBSD instances called *webserver-1*, *webserver-2*, and *webserver-3*, Terraform creates that for you. If you ever want to add a new machine, you just add the characteristics of the new instance to your declaration and run Terraform again. If you need to remove one of the machines, you remove it from the declaration

Why Proxmox?

Proxmox VE is a hypervisor intended to be installed on your hardware on which virtual machines and containers are deployed. Proxmox is a commercial distribution, and therefore a paid subscription is required for accessing the production-ready repositories. Non-production repositories are available free of charge and without restrictions and are quite sufficient for use in testing and for home labs.

Proxmox VE is based on Debian and can act as a host for LXC containers and Qemu/KVM virtual machines. It offers a nice web interface for management (Figure 1) and supports a self-hostable backup solution (Proxmox Backup Server) that makes it quite convenient to take backups from your guest systems and store them in an orderly manner. Proxmox VE has been covered before in other articles [5][6], so I suggest you check them out if you want an in-depth review.

The main reason to use Proxmox VE is because it is inexpensive while offering lots of features. VMware's strategy has pivoted into turning ESXi into a product for high-budget customers [7], so Proxmox VE might end up being one of the last good affordable hypervisors. Finally, OpenTofu supports Proxmox VE through providers that are far from featureful but get the job done more often than not.

and execute Terraform one more time. Terraform has been covered in other articles [8] and, because of its popularity, is one of the options most people know. Unfortunately, the company developing it switched its licensing model away from the Mozilla Public License (MPL 2.0) into the vague, commercially non-free Business Source License (BSL). Because Terraform was a critical component for so many organizations, the OpenTofu fork appeared shortly thereafter. As a fork of Terraform, OpenTofu supports every Terraform provider that was available for Terraform 1.6 – where *provider* refers to a module supporting a cloud or hypervisor system (e.g., AWS, ESXi, or Proxmox). OpenTofu does not have feature parity with Terraform as a goal, so its feature set is expected to differ in the future.

OpenTofu is intended to run on the administrator's computer and issue instructions to the cloud environment or hypervisor from there. OpenTofu offers packages for a number of Linux distributions; has a generic installer for Windows, macOS, Linux, and FreeBSD; and is available in OpenBSD's port tree.

Introducing cloud-init

OpenTofu is a provisioner, which means it will deploy your virtual machines and infrastructure, but it won't configure them. You could say OpenTofu is a bit like a technician contracted to set your machines in place, run the wiring, and install an operating system, and then let the next specialist take over.

A brand new install in its default configuration is not of much use. For it to do any good, somebody has to create users, configure credentials, install packages, and so on. OpenTofu is not smart enough to accomplish all of this on its own, so it is usually paired with a complementary tool. Ansible is a common choice: OpenTofu creates the infrastructure, and Ansible configures it. Another widely available industry standard is cloud-init, Ubuntu's proposal for configuring new installs automatically. It is supported in one form or another by most relevant cloud providers and is typically used as follows:

1. Install a cloud-init-capable image of the OS you need on your virtual machines. You could either build your own or download a

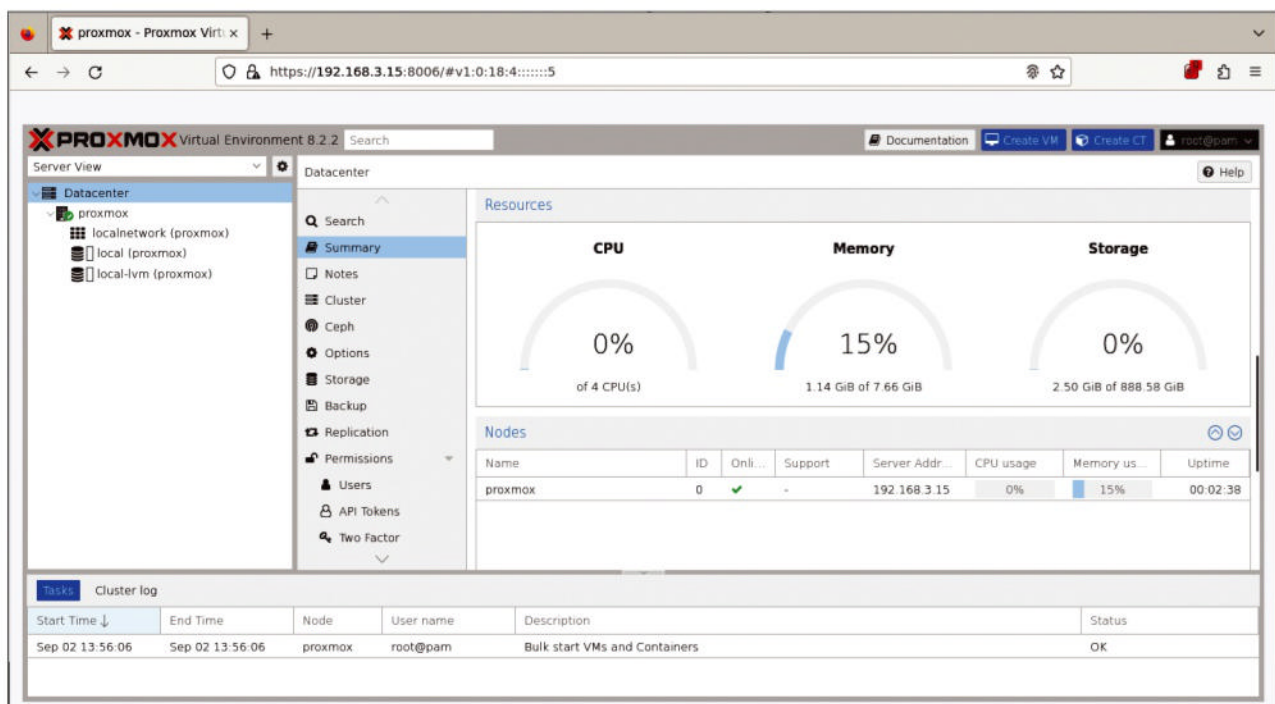


Figure 1: A recently created Proxmox 8.2 install. Proxmox offers a web interface that acts as a centralized control panel for a myriad of virtualization-related functions.

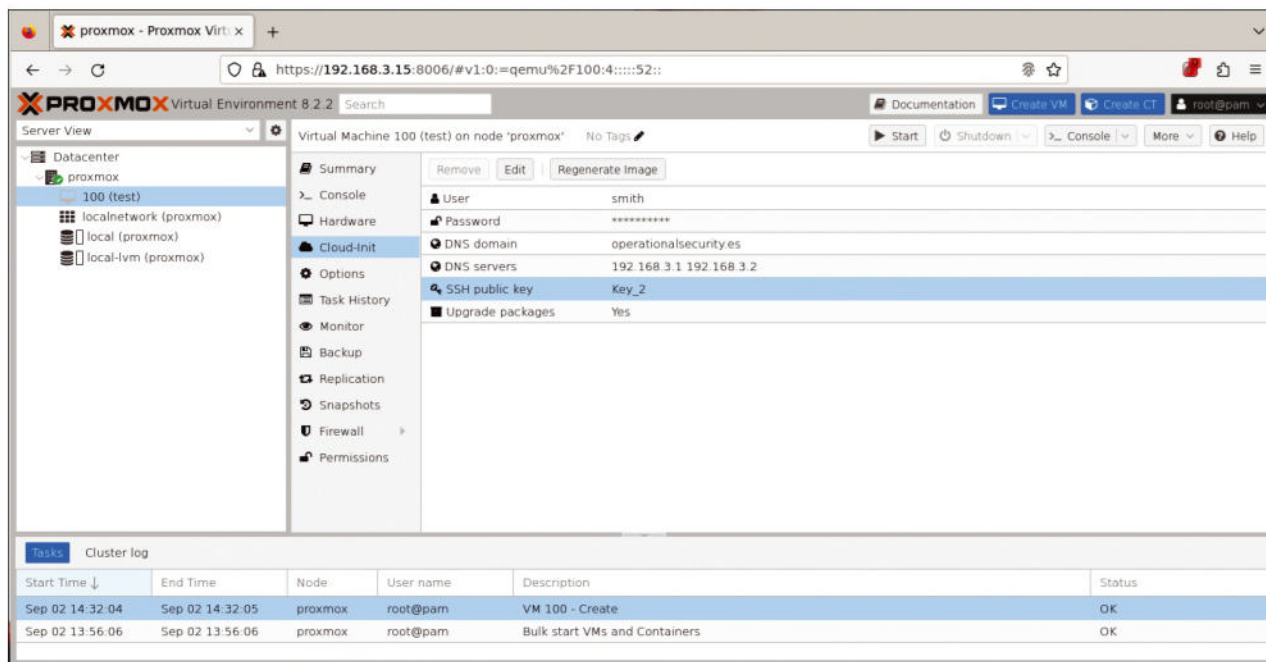


Figure 2: Proxmox VE offers limited support for cloud-init in its web interface, but full cloud-init capabilities can be leveraged through different mechanisms.

prepackaged image; many distributions offer either official or unofficial versions of cloud-init. For this article, I use an unofficial OpenBSD image from a reputable source [9], because both OpenBSD's HTTP daemon and reverse proxy are extremely simple to set up.

2. Load your cloud-init configuration into your host provider, which should have a facility to introduce the parameters you need cloud-init to set for each of your virtual machines. For example, Proxmox allows you to introduce a limited number of parameters from its web interface (Figure 2). In the case of Proxmox, your configuration options would be built into a special virtual CD drive that would be attached to the virtual machine you want to configure.
3. Boot each of the machines; cloud-init, which is already included in their images, would then fetch its configuration from the provider and apply it at first boot. The cloud-init tool can set users and passwords, import SSH public keys, install and upgrade packages, and run arbitrary scripts.

This setup is all you need for a simple deployment.

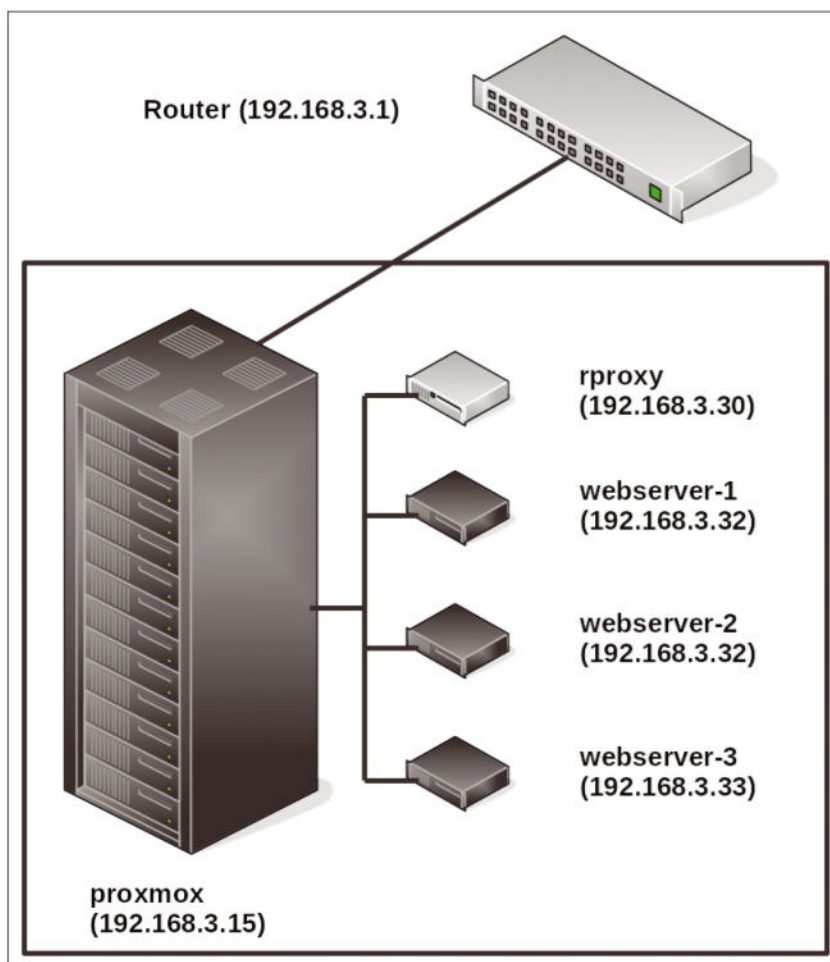


Figure 3: The desired end state: The Proxmox host has address 192.168.3.15. The router hosts a DNS resolver set beforehand.

Putting Everything Together

My plan is to use OpenTofu to create my virtual machines and then cloud-init to configure them. As stated, the goal of this article is to deploy a reverse proxy called *rproxy* and a number of web servers in the configuration shown in [Figure 3](#). I usually let the virtual machines get their IPs from the router over DHCP, because OpenTofu is capable of interfacing with certain routers to set static DNS entries and DHCP reservations; however, for the sake of simplicity, I will skip the router configuration and have cloud-init set static IP addresses instead.

The first step toward this goal is to set up a Proxmox VE host. Proxmox can be installed according to the official documentation [\[10\]](#), and it isn't more complex than installing any other Linux distribution. For this example, I use a humble desktop computer made from cannibalized parts with 8GB of RAM and an i3 CPU, which is fine for toying around. For long-term usage, though, it is much better to use server hardware. At the office, we use

PowerEdge T40 servers in tower form as an inexpensive option.

Once your Proxmox is up and running, you need to configure its storage to accept *snippets*, which is a fancy way of saying you are allowing Proxmox to store extraneous files that are needed so OpenTofu can upload cloud-init configuration files to the host ([Figure 4](#)).

Connect to your Proxmox server over SSH and issue the commands in

Listing 1: Create Template for VMs

```
# Download the image. https://bsd-cloud-image.org/ is used as a source

mkdir tmpimages
cd tmpimages
wget https://github.com/hcartiaux/openbsd-cloud-image/releases/download/v7.5_2024-05-13-15-25/
  openbsd-min.qcow2

# Create a virtual machine and assign the downloaded image to it

qm create 100 --name openbsd-master --memory 1024 --agent 1,type=isa --scsihw virtio-scsi-single --boot
  order='scsi0' --net0 virtio,bridge=vbr0 --serial0 socket --vga serial0

qm set 100 --scsi0 local-lvm:0,import-from=/root/tmpimages/openbsd-min.qcow2

# Turn the VM into a template

qm template 100
```

[Listing 1](#) to download a cloud-init-capable OpenBSD image and create a template from it. OpenTofu will use this template as a base for every virtual machine you want to create. The steps in [Listing 1](#) are mostly self-explanatory, but some of the arguments passed to `qm create` are worth a comment:

- `--agent 1,type=isa` tells the *Proxmox* host to communicate with the guest with a *Qemu Agent*, which

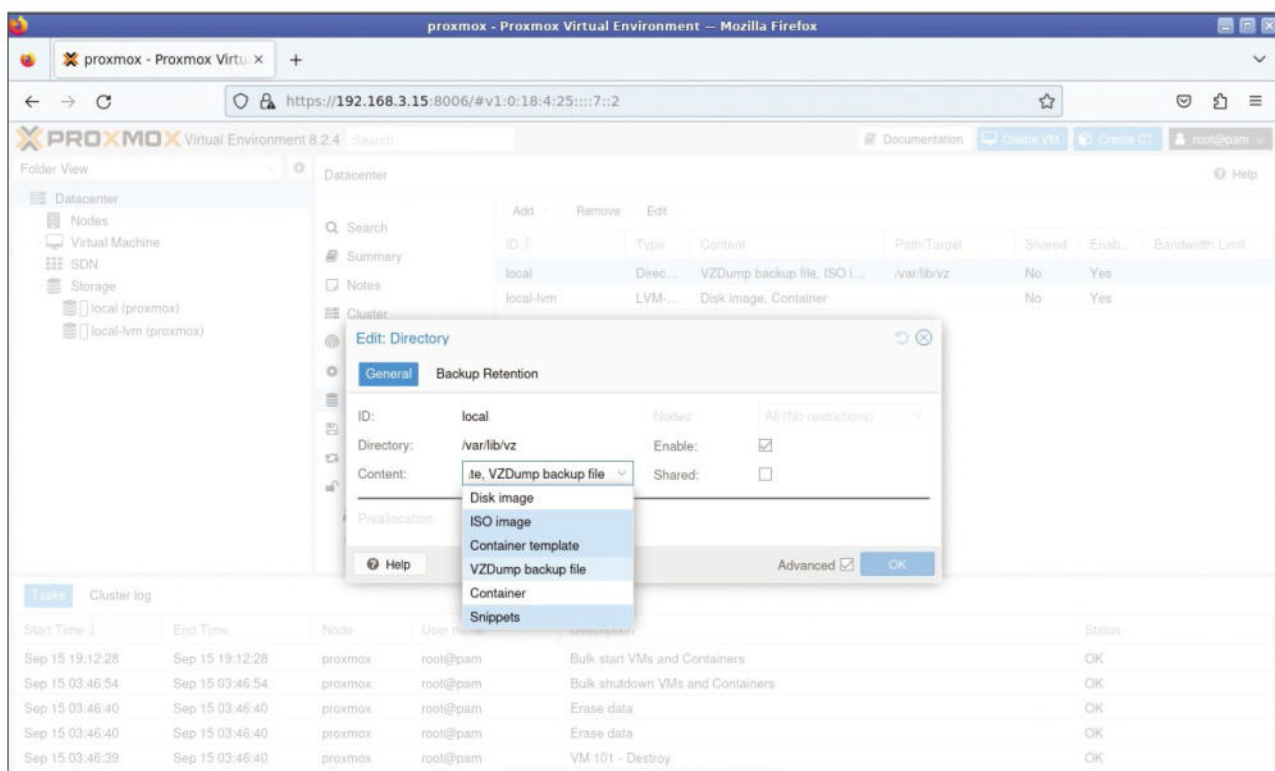


Figure 4: Enable snippets for your Proxmox storage. This step is easy to forget when in a hurry, but it is absolutely necessary to leverage the full capabilities of *cloud-init*. From the Datacenter pane, select *Storage* and edit the properties of the *Local* storage as needed.

Listing 2: Installing OpenTofu on Devuan

```
# Download the installer and mark it as executable

wget https://get.opentofu.org/install-opentofu.sh
chmod +x install-opentofu

# Execute the installer. OpenTofu will be downloaded and
the repositories from the OpenTofu project will be
added to your sources.list

./install-opentofu.sh --install-method deb
```

Listing 3: tofu_project/main.tf

```
01 terraform {
02   required_providers {
03     proxmox = {
04       source = "bpg/proxmox"
05       version = "0.64.0"
06     }
07   }
08 }
09
10 provider "proxmox" {
11   endpoint = "https://192.168.3.15:8006/"
12   username = "root@pam"
13   password = "proxmox"
14   insecure = true
15   tmp_dir = "/var/tmp"
16
17   ssh {
18     agent = true
19   }
20 }
```

Listing 4: tofu_project/machines.tf

```
01 resource "proxmox_virtual_environment_vm" "rproxy" {
02   name = "rproxy"
03   description = "Reverse proxy"
04   node_name = "proxmox"
05
06   clone {
07     vm_id = 100
08   }
09
10   network_device {
11     model = "virtio"
12     bridge = "vmbbr0"
13   }
14
15   depends_on = [
16     proxmox_virtual_environment_file.rproxy_cloud_config,
17     proxmox_virtual_environment_file.network_cloud_config,
18   ]
19
20   initialization {
21     user_data_file_id = proxmox_virtual_environment_file.rproxy_cloud_
22       config.id
23     network_data_file_id = proxmox_virtual_environment_file.network_
24       cloud_config[0].id
25   }
26 resource "proxmox_virtual_environment_vm" "webserver" {
27   count = "3"
28   name = "webserver-${count.index+1}"
29   description = "Generic web server"
30   node_name = "proxmox"
31
32   clone {
33     vm_id = 100
34   }
35
36   network_device {
37     model = "virtio"
38     bridge = "vmbbr0"
39   }
40
41   depends_on = [
42     proxmox_virtual_environment_file.webserver_cloud_config,
43     proxmox_virtual_environment_file.network_cloud_config,
44     proxmox_virtual_environment_vm.rproxy,
45   ]
46   initialization {
47     user_data_file_id = proxmox_virtual_environment_file.webserver_
48       cloud_config.id
49     network_data_file_id = proxmox_virtual_environment_file.network_
50       cloud_config[count.index+1].id
51   }
52 }
```

is a daemon that runs inside the guest and accepts instructions from Proxmox when Proxmox needs to perform tasks such as turning off the guest. Communication will occur over a virtual ISA serial port.

- `--scsihw virtio-scsi-single` defines the virtual controller for the virtual hard drive of the guest.
- `--boot order='scsi0'` ensures the virtual machine boots from its associated virtual drive.
- `--serial0 socket` and `--vga serial0` create a serial interface that acts pretty much like a serial terminal.

The next step is to install OpenTofu on your workstation. The generic installer from the OpenTofu website can be used as shown in [Listing 2](#) on a Devuan system. Keep in mind, your user needs to be granted sudo privileges before running the installer (e.g., with `visudo`).

A project folder must be created and populated. It is common practice to put the project folder under version control (with Git, CVS, or a similar tool), but this is not necessary. For this example, I create a folder called `tofu_project` and populate it with the files `main.tf`

([Listing 3](#)), `machines.tf` ([Listing 4](#)), and `cloud-init.tf` ([Listing 5](#)).

The `main.tf` file defines which providers and which versions (if need be) are used for the project. The provider I selected, `bpg/proxmox`, connects to the host over SSH to create automatically the environment defined by the project. OpenTofu can also use the regular Proxmox API with this provider. The configuration parameters passed to the Proxmox provider in lines 10-19 in [Listing 3](#) are self-explanatory. Keep in mind that hardcoding credentials in the project file is fine for testing, but for a production environment you should consider passing the username and password with the environment variables `PROXMOX_VE_USERNAME` and `PROXMOX_VE_PASSWORD` [11].

The `machines.tf` file contains information about the virtual machines you want to create. OpenTofu creates new virtual machines by cloning from the template defined earlier. In the example, a `proxmox_virtual_environment_vm` resource named `rproxy` is created ([Listing 4](#), lines 1-24), whose task will be to act as a reverse proxy for the web servers. The next block in the file defines a resource named `webserver`

(lines 26-50) with a `count = "3"` parameter, so three instances are created with names `webserver-1`, `webserver-2`, and `webserver-3`. The `rproxy` resource is defined as a dependency for the web servers in line 44 to ensure it is created before the rest of the machines.

You will notice some `proxmox_virtual_environment_file` resources are declared as hard dependencies for the virtual machines to ensure they are not created without a proper `cloud-init` configuration in place. I have already mentioned that Proxmox's capabilities for setting `cloud-init` parameters from the GUI are very limited; thankfully, you can leverage the full power of `cloud-init` with `cicustom` files that contain all the `cloud-init` parameters you want to pass to each of your virtual machines, and they are uploaded to the Proxmox VE host before each instance is created. The `cloud-init.tf` file lists the `cicustom` files that will be uploaded to Proxmox. A `cloud-init` configuration file for `rproxy` is defined in lines 1-10 in [Listing 5](#) and another in lines 12-21 that is applied to all the web servers. These files are stored in the project folder and are uploaded to the server by OpenTofu just before the virtual machines are created. You can see both files in [Listings 6 and 7](#). (See the "Different cloud-init Files" box.) The `users` directive in each file commands the creation of user `openbsd`

with password `openbsd` (see the "How User Creation Works" box). The `write_files` directive instructs `cloud-init` to place some configuration files that the services will need. Finally, the `runcmd` directive lists all the commands `cloud-init` will run on first boot to

Different cloud-init Files

You will notice I use two separate types of `cloud-init` files to supply configuration data to the virtual machines: *user data files* and *network data files*. `cloud-init` supports three configuration levels and you may use a `cicustom` file for each:

- The *vendor* file is used to provide the virtual machine with the configuration chosen by the vendor. It is used by cloud providers to set a baseline for all the instances that run on the platform. If you rent a virtual private server (VPS) from a cloud operator, your image could come preconfigured at the vendor level.
- The *user* file contains settings specified by the person who requested the machine. When you rent a VPS, you get to choose things like your password, for example. User-specific data is intended to be supplied at the user level.
- The *network* file, as you might have guessed, is used to configure the virtual machines' networking.

The `cloud-init` files in this article are YAML files. Note that it is imperative to avoid the use of tabs for indentation, because tabs, when used instead of spaces, will break the files.

Listing 5: `tofu_project/cloud-init.tf`

```
01 resource "proxmox_virtual_environment_file" "rproxy_
    cloud_config" {
02   content_type = "snippets"
03   datastore_id = "local"
04   node_name   = "proxmox"
05
06   source_file {
07     path = "rproxy.yml"
08     file_name = "user_data_vm-rproxy.yml"
09   }
10 }
11
12 resource "proxmox_virtual_environment_file"
    "webserver_cloud_config" {
13   content_type = "snippets"
14   datastore_id = "local"
15   node_name   = "proxmox"
16
17   source_file {
18     path = "webserver.yml"
19     file_name = "user_data_vm-webserver.yml"
20   }
21 }
22
23 resource "proxmox_virtual_environment_file" "network_
    cloud_config" {
24   count = 4
25   content_type = "snippets"
26   datastore_id = "local"
27   node_name   = "proxmox"
28
29   source_raw {
30     data = templatefile("network.tftpl", {myip =
        "192.168.3.${count.index+30}"})
31     file_name = "network_data_vm-${count.index}.yml"
32   }
33 }
```

Listing 6: `tofu_project/rproxy.yml`

```
01 #cloud-config
02 users:
03   - name: openbsd
04     gecos: openbsd
05     groups: wheel
06     plain_text_passwd: openbsd
07     lock_passwd: false
08     doas: [permit nopass openbsd]
09
10 write_files:
11   - path: /etc/relayd.conf
12     content: |
13       table <webserver> { 192.168.3.31 192.168.3.32 192.92.168.33 }
14       http protocol "http" {
15         match request header set "X-Forwarded-For" value "$REMOTE_ADDR"
16         match request header set "X-Forwarded-Port" value "$SERVER_
            PORT"
17         match request header set "X-Forwarded-By" value "$SERVER_
            ADDR:$SERVER_PORT"
18       }
19
20       relay "webservice" {
21         listen on 192.168.3.30 port 80
22         protocol "http"
23         forward to <webserver> port 80 mode loadbalance \
24           check http "/index.html" code 200
25       }
26       owner: 'root:wheel'
27       permissions: '0644'
28       defer: true
29
30 runcmd:
31   - pkg_add qemu-ga
32   - rcctl enable qemu_ga
33   - echo 'qemu_ga_flags="-t /var/run/qemu-ga -m isa-serial -p /dev/
        cua01 -f /var/run/qemu-ga/qemu-ga.pid"' >> /etc/rc.conf.local
34   - rcctl start qemu_ga
35   - rcctl enable relayd
36   - rcctl start relayd
```

install, enable, configure, and start `qemu-agent` and then enable and start the main service for the virtual machine with `rcctl` (the OpenBSD command for managing services).

The configuration of the network (Listing 5, lines 23-33) is a bit tricky and is

Listing 7: `tofu_project/webserver.yml`

```
01 #cloud-config
02 users:
03   - name: openbsd
04     gecos: openbsd
05     groups: wheel
06     plain_text_passwd: openbsd
07     lock_passwd: false
08     doas: [permit nopass openbsd]
09
10 write_files:
11   - path: /etc/httpd.conf
12     content: |
13       server "default" {
14         listen on * port 80
15       }
16   - path: /var/www/htdocs/index.html
17     content: |
18       <!DOCTYPE html>
19       <html lang="en">
20         <head>
21           <meta charset="utf-8">
22           <title>Hello, World</title>
23         </head>
24         <body>
25           <h1>Hello, world!</h1>
26           <p>This is an example file</p>
27         </body>
28       </html>
29
30 runcmd:
31   - pkg_add qemu-ga
32   - rcctl enable qemu_ga
33   - echo 'qemu_ga_flags="-t /var/run/qemu-ga -m
      isa-serial -p /dev/cua01 -f /var/run/qemu-ga/
      qemu-ga.pid"' >> /etc/rc.conf.local
34   - rcctl start qemu_ga
35   - rcctl enable httpd
36   - rcctl start httpd
```

Listing 8: `tofu_project/network.tftpl`

```
network:
  version: 2
  ethernet:
    vif0:
      addresses:
        - ${myip}/24
      gateway4: 192.168.3.1
      nameservers:
        addresses:
          - 192.168.3.1
```

accomplished by templates (line 30). A `network_cloud_config` resource is created with `count = 4`, so OpenTofu creates a network cloud-init file for each of the virtual machines. These files are named `network_data_vm-0.yml`, `network_data_vm-1.yml`, and so on. The files resulting from the `network.tftpl` template (Listing 8) are uploaded to Proxmox and assigned the virtual machine IP addresses from 192.168.3.30 onward. The `user_data_file_id` and `network_data_file_id` parameters for each machine in Listing 4 ensure that Proxmox loads the corresponding cloud-init files when creating each virtual machine. The `rproxy` resource loads `network_data_vm-0.yml`, and the web servers each load a file from `network_data_vm-1.yml` onward.

Time for Deployment

With all of the code in place, it is time to initialize OpenTofu and command

How User Creation Works

The `users` directive in `cloud-init` is not very intuitive, so it deserves some explanation. To begin, `name` defines the username of the new user, `gecos` defines the “real name,” and `group` adds the `openbsd` user to the `wheel` group, which is a group with certain administrative rights (e.g., the ability to become the superuser with the use of `su`, if they have a valid password and root happens to be unlocked).

The `cloud-init` utility creates users with locked passwords by default to force users to adopt SSH key authentication instead of regular passwords. You are supposed to use the `ssh_authorized_keys` parameter to provide your SSH public key. Because this article is didactic in nature, I have chosen to simplify, set a password with `plain_text_passwd`, and unlock it with `lock_passwd: false`. Keeping credentials in plain text in your project folder is considered a bad idea for production, so keep this in mind while you play around with OpenTofu.

The `doas` configuration in the examples might seem alien to Linux users; it is the OpenBSD equivalent of `sudo`. The `doas` parameter in my `cloud-init` files grants the `openbsd` user some `doas` rights. Think of it as granting a regular Linux user the ability to use `sudo`. Although `sudo` is available in OpenBSD, OpenBSD users tend to favor `doas`.

it to perform the deployment. First, the initialization command installs all the modules required by the project (Figure 5):

```
# tofu init
```

Second, ensure OpenTofu will do the right thing when you request your systems be deployed by running the following command from the project folder (Figure 6):

```
# tofu plan
```

Final deployment could take some time, but at this point it will be fully automated. Just issue

```
# tofu apply --parallelism=1
```

and watch OpenTofu work its magic. In this example, I use `--parallelism=1` because my testing hardware is weak, and I want to limit OpenTofu to a single concurrent operation. For regular server hardware, it can be omitted. When you get bored of your testing environment, you can trash it with `tofu destroy`.

Conclusion

Automating deployments is an involved process, but once the process is completed, it will save you a lot of time. OpenTofu, in combination with `cloud-init`, is a good option for automating deployments on Proxmox. The Proxmox provider used in this article comes from *bpg*, but it is worth noticing that *Telmate* [12] also has a provider. Both have limitations that are hit when you attempt complex tasks (e.g., leveraging Proxmox’s native high-availability mechanisms or using its native firewall systems). Still, what you can accomplish with currently existing code is impressive. Combining providers can be a very powerful proposition. For example, if you have a MikroTik router, you can use OpenTofu to add and remove DHCP leases and DNS entries at the same time you deploy your virtual machines. You can also use it to set firewall rules in the router. Although

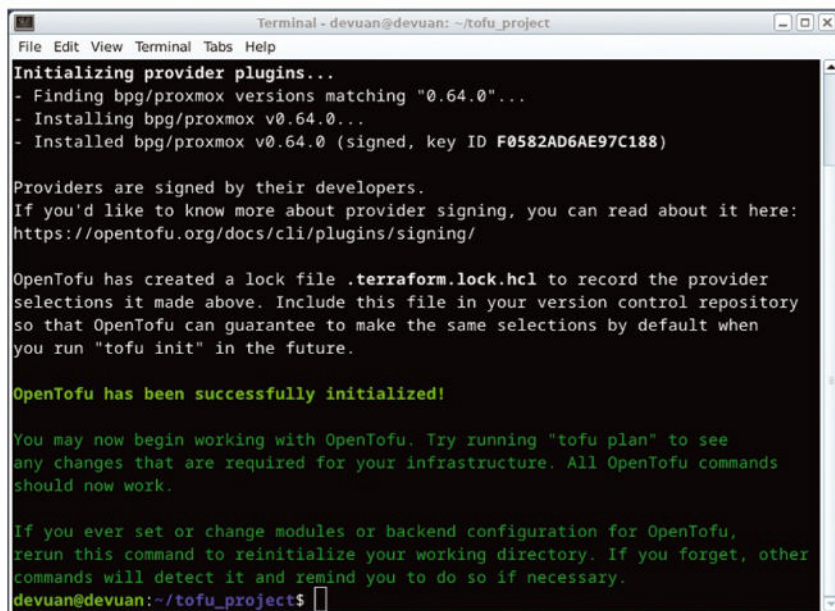
the example shown in this article is quite basic, once you become proficient with deployment automation, the sky is the limit.

The code in this article has been kept simple for didactic reasons and has much room for improvement. For example, the number of web servers deployed is hardcoded, instead of taken from a user-defined variable. The `cloud-init` user file for `rproxy` also has the web servers' IPs hardcoded, instead of being automatically defined. The way the

network configuration is assigned to each virtual machine is a bit fragile. If you feel like solving these issues, the official documentation will be useful [13].

Info

- [1] "Automatically Install and Configure Systems" by Martin Loschwitz, *ADMIN*, issue 52, 2019, pg. 62, [\[https://www.admin-magazine.com/Archive/2019/52/Automatically-install-and-configure-systems\]](https://www.admin-magazine.com/Archive/2019/52/Automatically-install-and-configure-systems)



```

Terminal - devuan@devuan: ~/tofu_project
File Edit View Terminal Tabs Help

Initializing provider plugins...
- Finding bpg/proxmox versions matching "0.64.0"...
- Installing bpg/proxmox v0.64.0...
- Installed bpg/proxmox v0.64.0 (signed, key ID F0582AD6AE97C188)

Providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://opentofu.org/docs/cli/plugins/signing/

OpenTofu has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that OpenTofu can guarantee to make the same selections by default when
you run "tofu init" in the future.

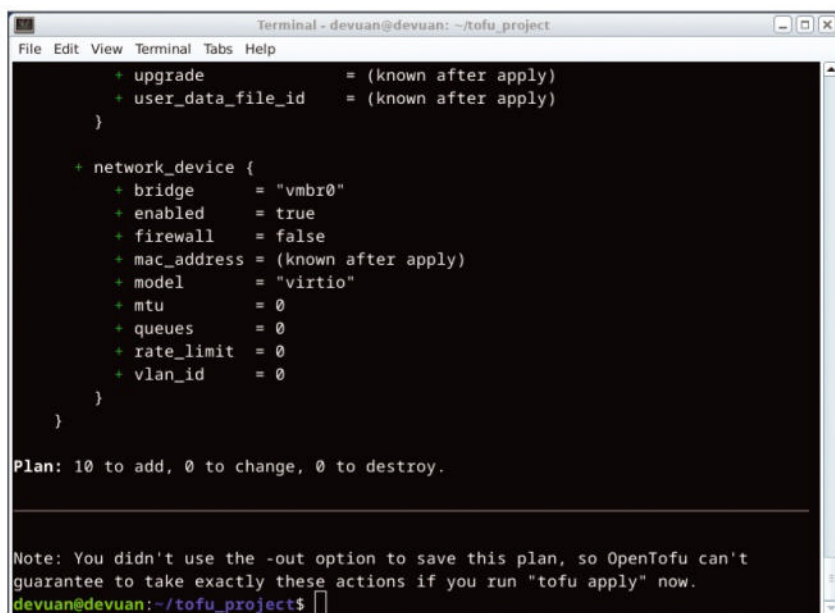
OpenTofu has been successfully initialized!

You may now begin working with OpenTofu. Try running "tofu plan" to see
any changes that are required for your infrastructure. All OpenTofu commands
should now work.

If you ever set or change modules or backend configuration for OpenTofu,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
devuan@devuan:~/tofu_project$

```

Figure 5: Successful OpenTofu initialization.



```

Terminal - devuan@devuan: ~/tofu_project
File Edit View Terminal Tabs Help

+ upgrade = (known after apply)
+ user_data_file_id = (known after apply)
}

+ network_device {
+ bridge = "vmb0"
+ enabled = true
+ firewall = false
+ mac_address = (known after apply)
+ model = "virtio"
+ mtu = 0
+ queues = 0
+ rate_limit = 0
+ vlan_id = 0
}

Plan: 10 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so OpenTofu can't
guarantee to take exactly these actions if you run "tofu apply" now.
devuan@devuan:~/tofu_project$

```

Figure 6: You can check which actions OpenTofu will take in advance to ensure your project is configured as intended before execution.

- [2] Unattended OpenBSD installation and upgrade: [\[https://man.openbsd.org/autoinstall.8\]](https://man.openbsd.org/autoinstall.8)
- [3] preseed: [\[https://wiki.debian.org/DebianInstaller/Preseed\]](https://wiki.debian.org/DebianInstaller/Preseed)
- [4] Kickstart installation at AlmaLinux wiki: [\[https://wiki.almalinux.org/documentation/kickstart-guide.html\]](https://wiki.almalinux.org/documentation/kickstart-guide.html)
- [5] "Virtualization with the Proxmox Virtual Environment 2.2" by Martin Loschwitz, *Linux Magazine*, issue 150, May 2013, pg. 22, [\[https://www.linux-magazine.com/Issues/2013/150/Proxmox-VE\]](https://www.linux-magazine.com/Issues/2013/150/Proxmox-VE)
- [6] "Proxmox Virtualization Manager" by Martin Loschwitz, *ADMIN*, issue 42, 2017, pg. 58, [\[https://www.admin-magazine.com/Archive/2017/42/Proxmox-virtualization-manager\]](https://www.admin-magazine.com/Archive/2017/42/Proxmox-virtualization-manager)
- [7] "Broadcom's Stated Strategy Ignores Most VMware Customers" by Simon Sharwood, *The Register*, May 2022, [\[https://www.theregister.com/2022/05/30/broadcom_strategy_vmware_customer_impact/\]](https://www.theregister.com/2022/05/30/broadcom_strategy_vmware_customer_impact/)
- [8] "Infrastructure as Code with Terraform Blueprint" by Christian Rost, *ADMIN*, issue 43, 2018, pg. 42, [\[https://www.admin-magazine.com/Archive/2018/43/Infrastructure-as-Code-with-Terraform\]](https://www.admin-magazine.com/Archive/2018/43/Infrastructure-as-Code-with-Terraform)
- [9] A collection of prebuilt BSD cloud images: [\[https://bsd-cloud-image.org/\]](https://bsd-cloud-image.org/)
- [10] Installing Proxmox VE: [\[https://pve.proxmox.com/pve-docs/chapter-pve-installation.html\]](https://pve.proxmox.com/pve-docs/chapter-pve-installation.html)
- [11] Environment variables with bpg/proxmox: [\[https://github.com/bpg/terraform-provider-proxmox/blob/main/docs/index.md#environment-variables\]](https://github.com/bpg/terraform-provider-proxmox/blob/main/docs/index.md#environment-variables)
- [12] Telmate Proxmox provider: [\[https://github.com/Telmate/terraform-provider-proxmox\]](https://github.com/Telmate/terraform-provider-proxmox)
- [13] OpenTofu documentation: [\[https://opentofu.org/docs/\]](https://opentofu.org/docs/)

The Author

Rubén Llorente is a mechanical engineer, whose job is to ensure that the security measures of the IT infrastructure of a small clinic are both legally compliant and safe. He is also an OpenBSD enthusiast and a weapons collector.



Replication between SQL Server and Azure SQL

Migration

Wherever Microsoft SQL Server runs on local networks, individual or all databases can be migrated to Azure SQL by transactional replication. Various opportunities unfold, including analytics in the Azure cloud and global access routes for mobile users and home and branch offices. By Thomas Joos

Replication of local databases to the cloud offers many benefits, including improved availability: Databases in the cloud keep working even if the local SQL Server fails. Many companies also rely on this function to ensure smooth migration of SQL databases to the cloud, which means local SQL servers can ultimately be retired. Organizations that operate their own SQL servers need to choose the best possible hardware to keep the database servers running efficiently. With requirements for database servers constantly growing, licenses becoming increasingly expensive, and numerous services requiring data from your servers for their own operations, you have reason enough to outsource databases or entire database servers to the cloud. Replication with Azure SQL can be a simple approach, because as soon as all the data has been transferred, the local database can go offline, and you can work with the Azure-based database from then on. For many reasons, outsourcing

local databases to the cloud is sensible. Microsoft provides support with functions that are directly integrated into Microsoft SQL Server Management Studio (SSMS). In this article, I show you the steps you need to take to connect your local SQL Server to Azure and configure the setup.

SQL Server in the Cloud

Although you could start with a conventional SQL database in Azure, Azure offers other options that rely on the replication of local database servers. The setup is basically identical; the only differences are in how the databases are managed in the cloud.

Anyone with an Azure SQL Managed Instance, for example, can integrate a complete Microsoft SQL database from the cloud into their own network without having to worry about the server settings. Microsoft is responsible for maintaining, backing up, and updating the database server.

If you opt for a Managed Instance of SQL Server, you do not have to buy, rent, or license hardware or software. The usage costs are included in the subscription model; the organization is only billed for actual database use. Another way to make databases available in the cloud is to set up a virtual server that runs the required version of SQL Server, which allows organizations to retain full control over all server settings, because the databases are only one component of the virtual server, just as in a local installation. No hardware or software of your own is required; licensing is covered by your Azure subscription. In this use case, a complete virtual server is automatically installed, and administrators on the network have to take care of it.

If you create an Azure SQL database, you also need a SQL server in the cloud, but you don't need a virtual machine (VM) on which to set up and manage the operating system and SQL server. Admins can manage the

Lead Image © Charles Gibson, 123RF.com

virtual SQL server themselves, and licensing is again covered by the Azure subscription.

Creating Virtual SQL Servers and Databases

Before you can use transaction log replication to migrate local databases to the cloud, you should choose *Databases | SQL databases* in the Azure portal, where you create a new SQL database and matching virtual SQL server. You do not need to manage this server; it just needs to be created to host your databases. The SQL server can be created along with the first database, which is not a VM, but an Azure object.

When you create the new server, you need to specify its name and storage location (Figure 1). The name is important later on because you will be using it to set up the connection between your on-premises database and the Azure SQL database in the local SSMS. When you create the SQL server, you also need to define the administrator and the authentication method. The recommendation here is to work with the *Microsoft Entra authentication only* option. SQL authentication is also available at this point if you really need it. Setting up replication is often easier, but also far less secure, if you opt for SQL logins. Once the server has been created, the next step is to create the database to which you will later be transferring the transaction logs from the local database. You have the same options as when using Azure SQL without local replication. Once you have configured all the settings for the cloud database, Azure creates it in the appropriate resource group. The database is then ready for you to set up for replication.

Establishing a Connection to Azure SQL

If you have created a database in Azure SQL, are working with an Azure SQL Managed Instance, or are using an Azure VM on which Microsoft SQL Server is installed, it is a good idea to test whether you

Home > SQL databases > Create SQL Database >

Create SQL Database Server

Microsoft

Server details

Enter required settings for this server, including providing a name and location. This server will be created in the same subscription and resource group as your database.

Server name * sqlj00s ✓ .database.windows.net

Location * ((Europe) West Europe ✓

Authentication

Enter required settings for this server, including providing a name and location. This server will be created in the same subscription and resource group as your database.

Azure Active Directory (Azure AD) is now Microsoft Entra ID. [Learn more](#)

Select your preferred authentication methods for accessing this server. Create a Server admin login and password to access your server with SQL authentication, select only Microsoft Entra authentication. [Learn more](#) using Microsoft Entra user, group, or application as a Microsoft Entra admin [Learn more](#), or select both SQL and Microsoft Entra authentication.

Authentication method

- ☒ Use Microsoft Entra-only authentication
- ☐ Use both SQL and Microsoft Entra authentication
- ☐ Use SQL authentication

Set Microsoft Entra admin

tho
Admin object/app ID: 3
[Set Admin](#)

Figure 1: Creating a new SQL database server, including server name and storage location, in Azure.

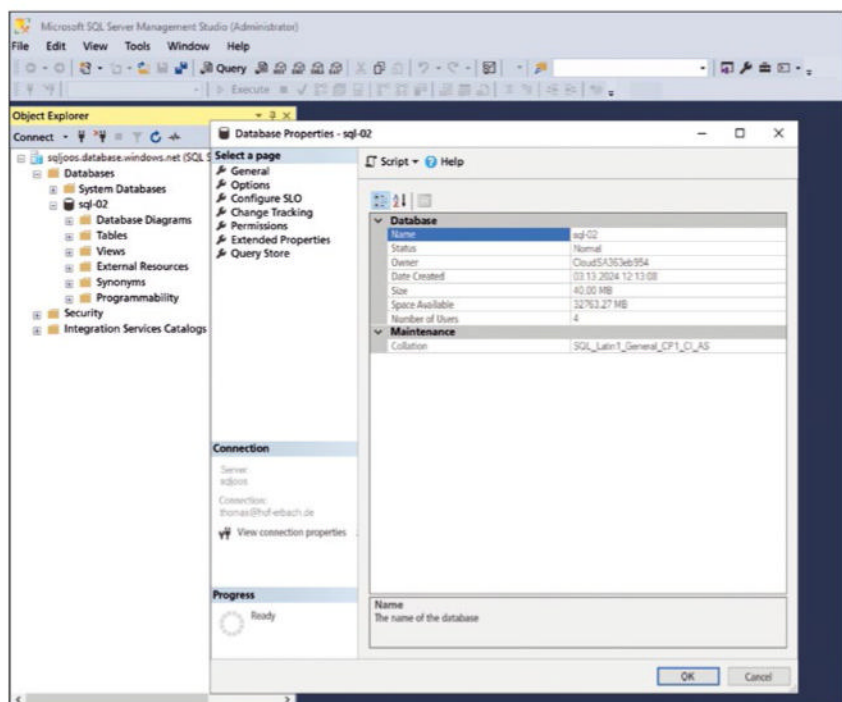


Figure 2: A successful connection between a locally installed SSMS and a database server in Azure SQL.

can open a connection to the Azure SQL database in SSMS on the local network before setting up replication (Figure 2). To do so, enter the name of your virtual database server in Azure as the server name when you launch the SSMS.

Next, go to the Authentication section, click *Options*, and select the *Microsoft Entra MFA* option for Login, assuming you are using multifactor analysis (MFA) in Entra ID. Alternatively, you can log in with a SQL user account. Authenticate with the admin account you defined when creating the database, and then click *Connect* to log in to Azure. The connection should be opened and you will be able to view it in the SSMS. The database you created is also available here, which means later you will be able to configure replication successfully.

Before you move on to configuring the replication setup, it is a good idea to call up the SQL Server Configuration Manager and go to *SQL Server Services* to make sure the SQL Server Agent service is configured for automatic startup, which is required for replication actions. You can use the wizard to start the agent when you set up replication, but this method often does not work reliably.

Setting Up Replication

To set up transactional replication between a local database and Azure SQL, you first need to launch SSMS and connect to your local database server. On SQL Server 2022 (some of the options in older versions have slightly different names, but the setup will be otherwise similar), open the *Replication* node, right-click *Local Publications*, and select *New publication*. A wizard launches to help with the setup.

In the next window, choose whether or not the current SQL server will also handle replication. Alternatively, you can use a different database server with the setup in place that handles the replication of several databases. Next, choose to launch the SQL server agent automatically. This

action makes sense in the context of replication and is also generally advisable. Now select the directory for storing the snapshots created during migration.

Things start to get interesting in the next window where you choose the database you want to replicate. At this point, the wizard displays all the databases available on the local server; choose one and specify the type of publication. The window offers the choices *Snapshot publication*, *Transactional publication*, *Peer-to-peer publication*, and *Merge publication*. The differences are described in the dialog (Figure 3).

Publication Variants for Replication

In Microsoft SQL Server 2022 (and in some cases, also in older versions) the four main types of publications shown in Figure 3 are available for setting up replication.

Snapshot publication creates and distributes copies in the form of snapshots of data at a specific point in time. This publication type is really useful for information that is rarely changed or wherever you can operate with the copies not always being up to date. The advantage of this method

lies in its simplicity and uncomplicated setup. The disadvantage is that the entire dataset is transferred with each replication, which means you need more bandwidth and can see inefficiencies with large volumes of data.

Transactional publication enables a virtually simultaneous transfer of data changes. As soon as changes are made to the source database, the system replicates these changes to the target servers. In this case, the server transfers the data to Azure SQL. This method is a good choice for applications in which the data must be up to date; additionally, continuous data streams can be managed efficiently. That said, transactional publication involves more complexity in terms of the setup and maintenance.

Peer-to-peer publication in Microsoft SQL Server enables scalable and flexible data distribution between multiple servers that communicate with each other as peers. In this replication approach, each node transmits changes to all the other nodes on the network, ensuring continuous data synchronization across all peers. This is a particularly good choice for systems that require high availability and reliability, because you do not have a central node as a potential single point of failure.

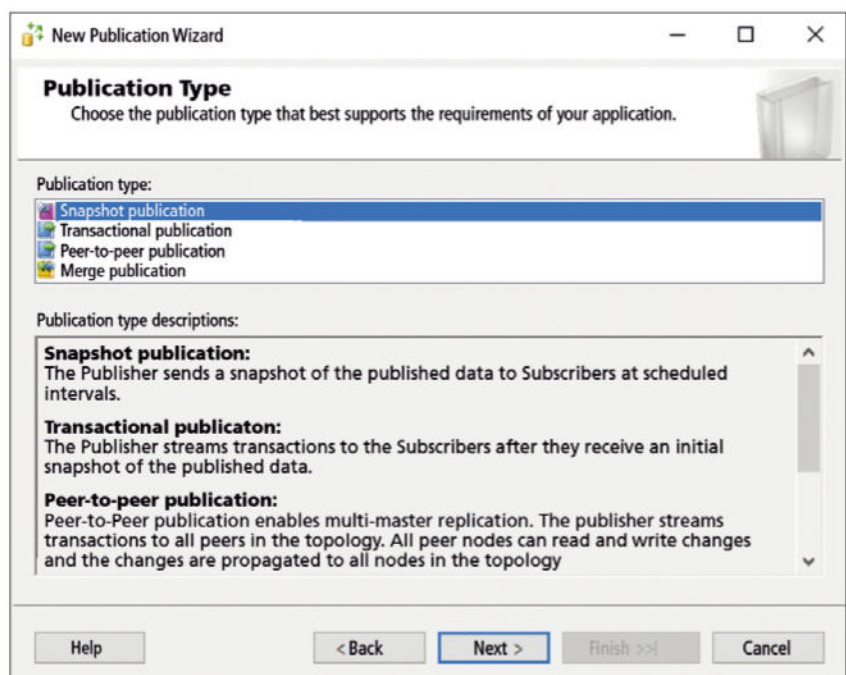


Figure 3: Choosing the publication type for replication between local databases and Azure SQL.

One other major advantage of peer-to-peer replication is the ability to distribute read and write operations across several nodes, which improves load distribution and boosts the performance of the overall system. That said, peer-to-peer replication requires careful planning and configuration to avoid conflicts when updating data. Changes can take place on any node and consistent synchronization is essential. The complexity of conflict resolution and the need for robust monitoring and management therefore pose challenges.

Merge publication allows for changes made in different locations to be merged later on. This method is ideal for distributed server environments where a constant connection is not guaranteed. Merge publication enables flexible data management and synchronization across multiple locations. The advantage lies in the high degree of flexibility and the ability to handle intermittent connections. On the other hand, conflict resolution complexities can occur when merging changes to the same data from different sources.

Replication by Transactional Publication

In this example, I use transactional publication, a popular method that covers most use cases. The local database server streams its data to the cloud, effectively in real time. Transactional publication is an advanced replication strategy that specifically targets seamless integration and synchronization between on-premises SQL Server instances and Azure SQL databases.

Another advantage of this method is its ability to ensure data integrity and consistency across geographically distributed systems. When implementing transactional publication in Azure SQL, particular attention must be paid to network latency and bandwidth limitations, because these factors can have a significant effect on replication performance. Organizations will want to evaluate their network infrastructures thoroughly and

optimize where needed to ensure an efficient data flow.

For transactional publication between local SQL servers and Azure SQL databases to work correctly, it makes sense to plan the connection settings and security configurations carefully. A dedicated virtual private network (VPN) connection or ExpressRoute can increase the security and reliability of data transmission. However, replication can also be carried out directly across the Internet without concern. Moreover, you also might need to adapt the replication frequency and time windows to suit your organization's business hours and network utilization to avoid performance hits in peak periods. Monitoring the replication process plays a crucial role in identifying and remediating potential problems at an early stage.

Setting Up Transactional Publication

After selecting *Transactional publication*, you need to specify the data you want to replicate to Azure SQL. As a rule, you will choose *Tables*. At this point, you can replicate all the tables in the database or specifically exclude individual tables. If a database cannot be replicated, the wizard will tell you so.

In this example, I do not replicate the *Stored Procedures*, *Views*, *Indexed Views*, and *User-defined Functions*. I am only interested in the data; however, if you want to duplicate these objects in the cloud, select an option and check whether it works for all of your data. The next window offers even more precise filters, and you can choose to exclude certain rows in the tables if required (Figure 4).

The Snapshot Agent plays a central role in initializing the replication process in the scope of a transactional publication. The agent creates a snapshot of the current data and schema objects of the database to be replicated. This snapshot serves as the basic version of the data on which subsequent transactional sequences are based. As soon as the initial snapshot has been created and applied to the subscriber, the sequential transfer of transactions based on the base version begins, enabling continuous synchronization of the databases.

The size and complexity of the database can influence how long it takes to create the snapshot, which in turn affects the availability of the data for replication. An effective strategy might be to create the snapshot at a time of low system utilization to minimize the effect on

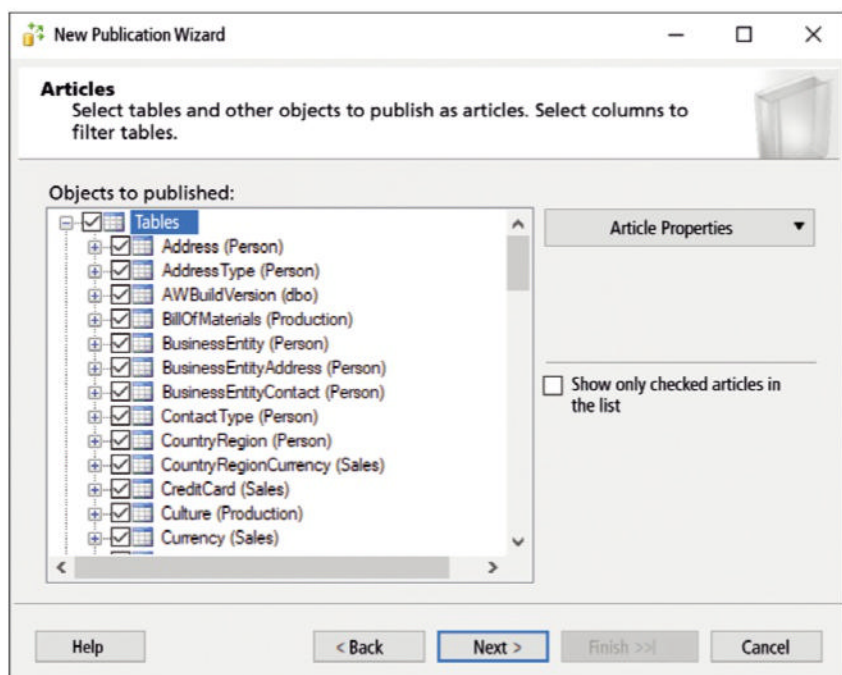


Figure 4: Selecting the tables to be replicated for Azure SQL is easy.

your production systems. The *Create Snapshot immediately and keep the snapshot available to initialize subscriptions* option is useful in this context. However, you can also specify a different time at this point if you do not want replication to take place immediately because the high volume of data would expose your server to excessive load.

In the next window, specify the account for running the snapshot agent. Click *Security Settings* and select *Run under the SQL Server Agent service account* (Figure 5). You can use a different account at this point if required for your environment. In the section at the bottom, go to *Connect to the Publisher* and specify how the connection to the local SQL Server will be established. You can use the process' account here, but also a SQL login or a login to Microsoft Entra if you are working with SQL Server 2022. In this case, the agent uses the Windows account with which the snapshot agent is executed. Microsoft describes the various options in more detail in the documentation [1].

To create the publication, give it a name (e.g., *DBReplication*) and complete the process. The wizard sets everything up in a short time. The individual replication jobs can be found under *Replication | Local Publications*. Unfortunately, the SSMS often only displays them after a restart or a connection update.

Managing Replication Publication

With a right-click on *Replication*, choose the *Launch Replication Monitor* context menu option to manage the publication and see the individual actions that the server carries out in the scope of this replication publication. To do so, click on the server name on the left to see the individual publications in the window to the right with their statuses in the *Publications* tab. The status should be *OK* in each case.

Once the publication is working, you can set up a connection to Azure

SQL for replication with the *New Subscriptions* option in the publication's context menu. This process is all wizard based. It is important that you have created and set up a database in Azure SQL previously. The wizard first displays the available publications. Select the one you created, and then select *Run all agents at the Distributor, ... (push subscriptions)*.

In the next window, select *Add Subscriber | Add SQL Server Subscriber*, and then log in to the SQL Server in Azure SQL, as you did at the beginning when testing the connection to the SSMS. After successfully logging in, select the virtual SQL server in Azure at the top of the window and go to the Subscription Database column to select the database you created as the target for the replication data. On the next page, you need to configure the connection between Azure and the local SQL server. To do this, click on the button with the ellipsis under *Connection to Subscriber* on the right-hand side. The recommended approach is to select *Run under the SQL Server Agent service account* again.

Of course, you can optimize security at this point with special accounts, depending on your environment. Microsoft discusses further options in the documentation [2]. For *Connect to the publisher*, either use *By impersonating the process account* or log in with a Microsoft Entra ID or a SQL user account in Azure SQL. Next, complete the setup wizard. SSMS now shows the new subscription below the publication you set up. When you call up the subscription with the *Launch Replication Monitor* context menu, you should not see any errors, and the replication process should be running. You can check the replication status at any time in SSMS.

Transferring the Database Directly to Azure

Besides the options for replicating databases, you can launch a wizard in SSMS that migrates a complete database to the cloud by selecting *Tasks | Deploy Database to Microsoft Azure SQL Database*. This action does not

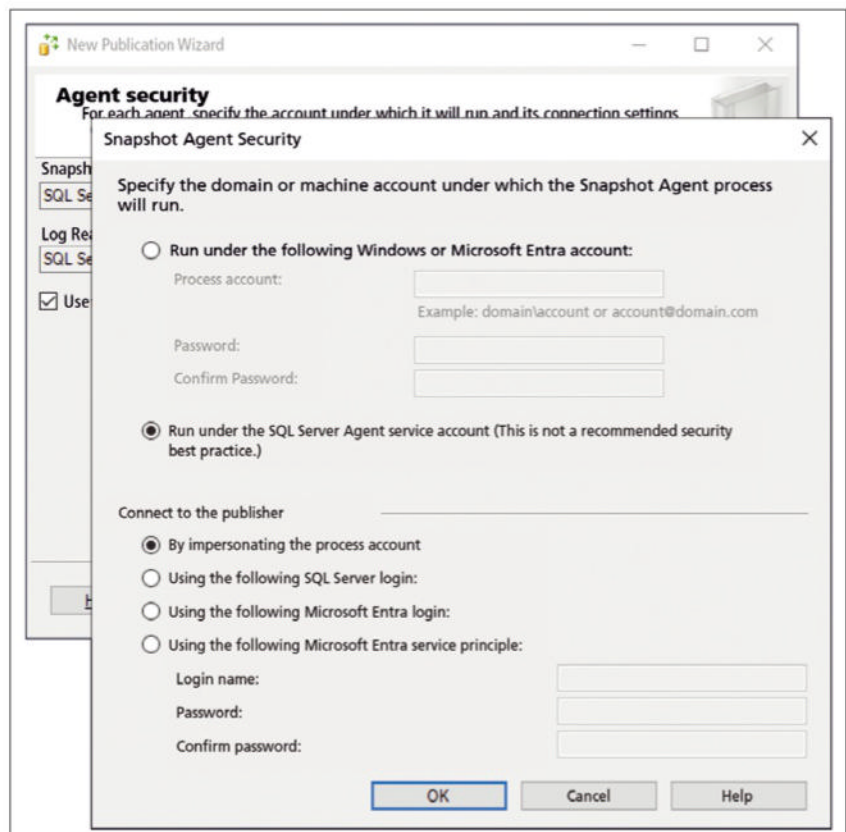


Figure 5: You can log in to the Snapshot Agent with different types of accounts.

involve replication; instead, the wizard transfers the information directly to the cloud without any detours by way of logs or snapshots. The setup is wizard based. First enter the name of the database server in Azure and log in to the server; then, assign a name to the database for use in Azure SQL (Figure 6).

A summary then appears and you can trigger the process by pressing *Finish*. The wizard exports the database from the local SQL server and imports it to Azure SQL. You can view the status in the window, including the transfer progress for

all tables and other objects. As soon as the transfer is complete, you can close the window. After you complete this action, the database is still available locally and in parallel in the cloud. However, there is no replication between the two.

Conclusions

Replication between on-premises SQL Server instances and Azure SQL offers a number of strategic benefits for organizations, including improved data analytics capabilities in the cloud, global access for users, and increased

availability through cloud-based databases. You can also transfer and deploy entire databases to Azure SQL in one fell swoop with Microsoft SQL Server Management Studio.

In particular, transactional replication enables near real-time synchronization, which is essential for disaster recovery and high availability. Of course, when implementing this replication method, you do need to think about various factors (e.g., network latency, bandwidth restrictions, and the configuration of security settings) to ensure an efficient and secure data flow. Choosing the right replication options and precise configuration is key to reaping the full benefits of cloud migration and ongoing data integration between on-premises and cloud environments.

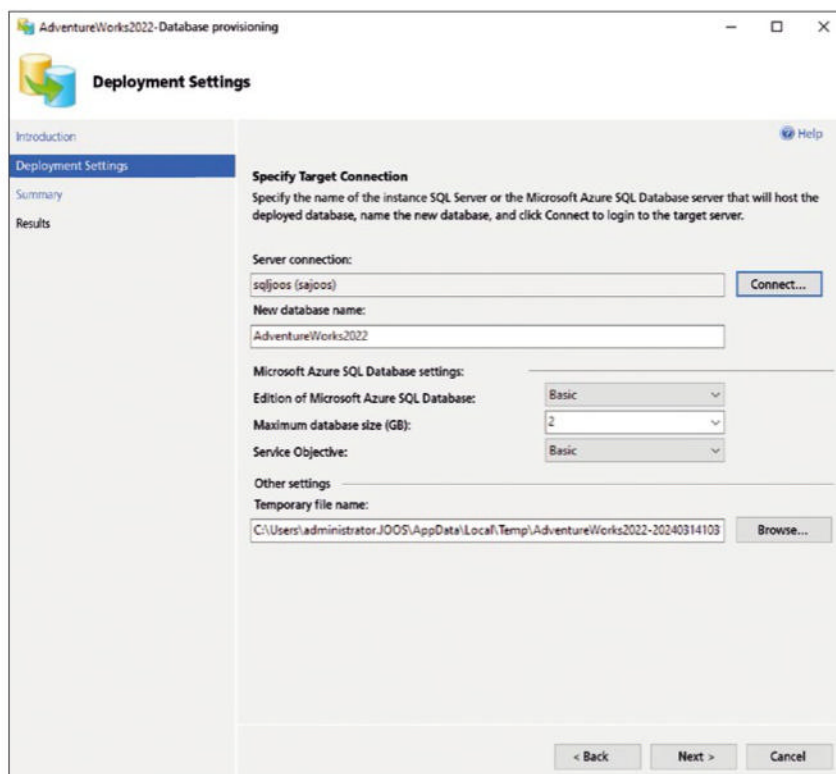


Figure 6: Transferring a local SQL database to Azure SQL involves just a few steps.

Info

- [1] Snapshot agent security:
[<https://learn.microsoft.com/en-us/sql/relational-databases/replication/snapshot-agent-security?view=sql-server-ver16>]
- [2] Distribution agent security:
[<https://learn.microsoft.com/en-us/sql/relational-databases/replication/distribution-agent-security?view=sql-server-ver16>]

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>].



Getting your virtual machine dimensions right

Tailor Made

Massive, performance-hungry VMs require proper handling to meet their dynamic requirements. We give you some rules to help size these monsters properly. By Wolfgang Weith

Admins often face the challenge of meeting dynamic requirements.

Although the trend is toward cloud-native applications and microservices – as seen primarily in the form of large numbers of comparatively small virtual machines (aka container hosts) – some applications still require massive, performance-hungry virtual machines (VMs, e.g., SAP HANA, an in-memory, column-oriented relational database management system). These monster VMs require proper handling.

Running large, resource-hungry, and monolithic applications such as Oracle, Microsoft SQL, and especially SAP HANA successfully on VMware vSphere is challenging, because they have many more factors to consider than do smaller VMs. Many of the configuration best practices mentioned in this article originate from the SAP HANA environment but are just as useful for other monster VMs and their applications. An in-depth report would go beyond the scope of this article, which is why the focus is

on the most important rules. These rules come into play in all life cycle phases of a monster VM, which are difficult to create, manage, and move. The challenges include:

- clean configuration of the physical server (BIOS, CPU, memory, network cards),
- correct initial sizing (vCPUs, NUMA alignment, reserves for the hypervisor), and
- vMotion (timing, duration, performance, negative effects).

Correct CPU Usage

The virtual CPU configuration plays a central role in the VM's performance. If you get the CPU configuration right, each VM will have the compute power it needs to perform its tasks efficiently without unnecessarily overloading the host's resources. vSphere distinguishes between physical CPUs (pCPUs) and virtual CPUs (vCPUs). pCPUs sit in the physical server and are the hardware components that perform the compute

operations. A pCPU socket can contain several CPU cores; each core can process two threads simultaneously by hyperthreading, leading to the first performance pitfall: Hyperthreading gives you two logical CPUs (the hyperthreads), which means that when hyperthreading is enabled, there are twice as many options for executing a thread as there are physical cores installed.

The hypervisor thus has the opportunity to schedule vCPUs better and run more smoothly. However, hyperthreading does not double performance, which is why the sizing of a CPU-hungry application should always be based on the physical core count and not on the number of hyperthreads, which usually correlate to the number of vCPUs available for the VM. In other words, the effect of hyperthreading on performance and performance enhancement needs some critical evaluation. In some cases, especially with applications that are heavily dependent on the CPU cache, hyperthreading can even cause performance losses, because several threads share the same cache. Thorough testing before commissioning and continuous monitoring in production are essential to ensure

Photo by ROCCO STOPPOLONI on Unsplash

that hyperthreading offers the benefits you expect.

In contrast, vCPUs are an abstraction that enables a VM to use computing resources as if it were accessing the physical hardware directly. The number of vCPUs assigned to a VM should not be based on the maximum possible, but on the performance or scaling required (e.g., the number of parallel database users). Application vendors usually also have good sizing tools for this process.

Non-Uniform Memory Access

Non-uniform memory access (NUMA) optimizes efficient access to RAM and is a decisive aspect in the architecture of modern multicore and multisocket systems. In NUMA systems (multiprocessor systems, to put this simply), the RAM is divided into different areas, each of which is assigned to a specific CPU socket.

The combination of the processor (socket) and the directly connected main memory is the NUMA node. If a VM with its vCPUs resides in a socket, it can access the directly connected RAM with the highest bandwidth and the lowest latency, also known as NUMA locality. When a CPU accesses the RAM that resides behind the memory controller of another CPU, the RAM access latency increases. **Figure 1** illustrates this for a two-socket system.

Depending on the server type (two-socket, four-socket, or eight-socket system), different memory bus architectures are used, which causes different path lengths from the RAM of one CPU to another. In other words, the choice of server can directly affect the expected performance of the virtualized monster application.

vSphere supports NUMA in virtual environments by ensuring that the memory and CPU resources are used efficiently to maximize performance. A correct NUMA configuration can offer significant performance benefits, especially for large VMs with high memory and compute requirements. Regardless of the server type and size, various BIOS/UEFI settings are fundamental to high-performance monster VMs:

- Enabling Turbo Boost (if configurable), which gives you a balanced workload on unused cores.
- Hyperthreading for better CPU scheduling of the VMs.
- Enabling NUMA by deactivating node interleaving in systems such as HPE servers.
- Enabling extended CPU functions such as Intel and AMD virtualization (VT-x and AMD-V), extended page tables (EPT), no execute and execute disable (NX/XD), and rapid virtualization indexing (RVI).
- Disabling all unused devices and BIOS functions (e.g., video RAM

cacheable, onboard audio, serial ports, CD-ROM, or USB).

- Switching off the “C1E Halt State.”
- Disabling “Enhanced C-States.”
- Configuring power management for high performance, which prevents power saving and the subsequent performance hits.

These host configurations lay the foundations for creating monster VMs.

Correct VM Size

The “bigger is better” principle is only partially true in the context of virtual machines. Of course, the aim is to make maximum use of the host and its CPU and RAM resources. Any SAP HANA or Oracle admin will also want to use as many CPUs, cores, and hyperthreads as possible for the VM – which is where the second big mistake happens.

The hypervisor also needs resources. The ESXi server’s VMkernel has to take care of many elements because it provides the basic hypervisor functionality, such as CPU scheduling, RAM management, the virtual network (i.e., the standard vSwitch (VSS) or distributed virtual switch (VDS)), the storage stack (NFS, clustered filesystem (VMFS), virtual volumes (vVOLs), and virtual storage area network (vSAN)), including multipathing for block storage, to name just a few of the central components.

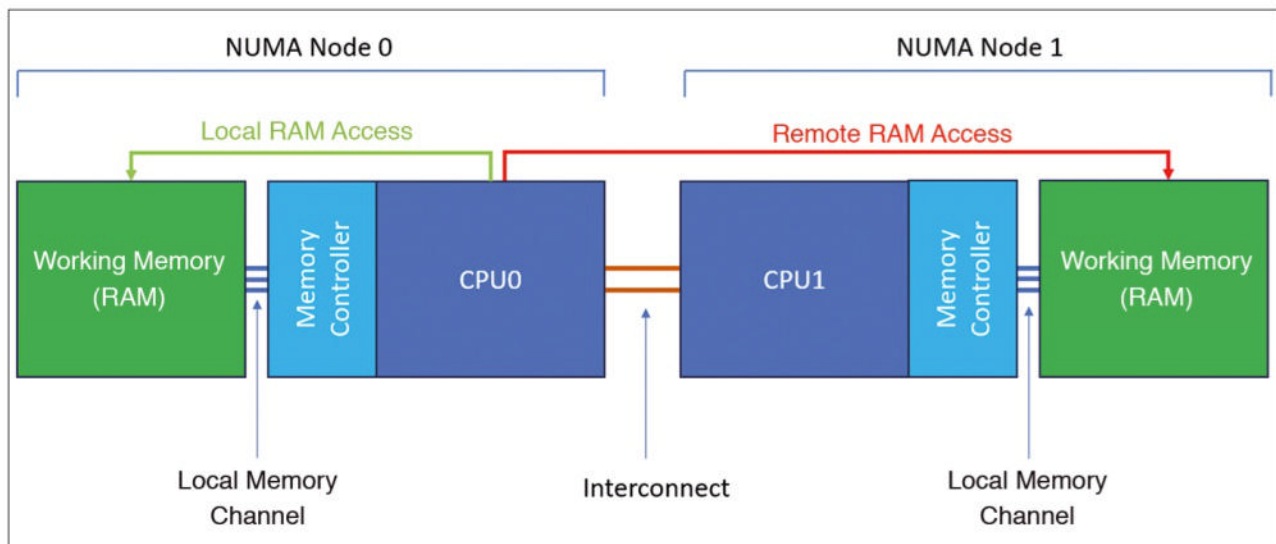


Figure 1: A two-socket system (two NUMA nodes) showing local and remote RAM access from CPU0.

Another need might be for integration, such as with VMware network and security virtualization (NSX) and distributed firewalling (DFW). Finally, functions such as the distributed resource scheduler (DRS), high-availability (HA) clustering, and vMotion also need to be usable.

On top of this, each VM also requires VMkernel processes at runtime to monitor the CPU and RAM and map virtual network cards, virtual SCSI adapters, virtual graphics cards, and the like. If you are looking for

a generic term, you can call this the virtualization overhead. CPU and RAM resources need to be reserved for the kernel processes and the virtualization overhead of the VMs (large or small); in other words, these resources must not be occupied by the VMs themselves.

Big Enough Buffer

Availability and performance are crucial, especially for monster VMs. By choosing VM dimensions that are

slightly smaller than the available NUMA node or host resources, you can prevent competition between the VM and the hypervisor. The kernel functions listed earlier are hypervisor functions, which means they take priority over the VM.

If you were to size VMs such that all the RAM and CPU were used up and the VMkernel now had to perform another task (e.g., a vMotion migration), the VMs would be deprioritized in terms of their access to CPU scheduling; on the guest operating system and in the application this choice would at best lead to performance hits or, in the worst case, a freeze. It is essential to prevent this through correct VM sizing.

What does correct sizing look like; that is, how much CPU and RAM does the hypervisor actually need? The answer is: It depends on which features you are using on the ESXi. If you use the hypervisor out of the box, including clustering (DRS/HA), you can expect to use between five and seven percent of your RAM and CPU resources. If you add vSAN, you will naturally require more RAM and more CPU time, which can quickly take the share up to 10 percent or more.

An example shows how to determine the CPU overhead:

- The server has four 28-core CPUs (i.e., 56 hyperthreads per CPU).
- Five percent of 28 cores is 1.4 cores.
- Seven percent of 28 cores is 1.96 cores.

If you reserve two cores per socket for the kernel and virtualization overhead, you are on the safe side. Therefore, you have 26 cores or 52 hyperthreads per NUMA node for the VM. **Figure 2** shows a four-socket server with 28 cores per socket and 6TB of RAM.

Sizing VMs Correctly

The question is how to size monster VMs correctly given the above rules? In **Figure 3** you can see the server illustrated in **Figure 2** with 6TB of

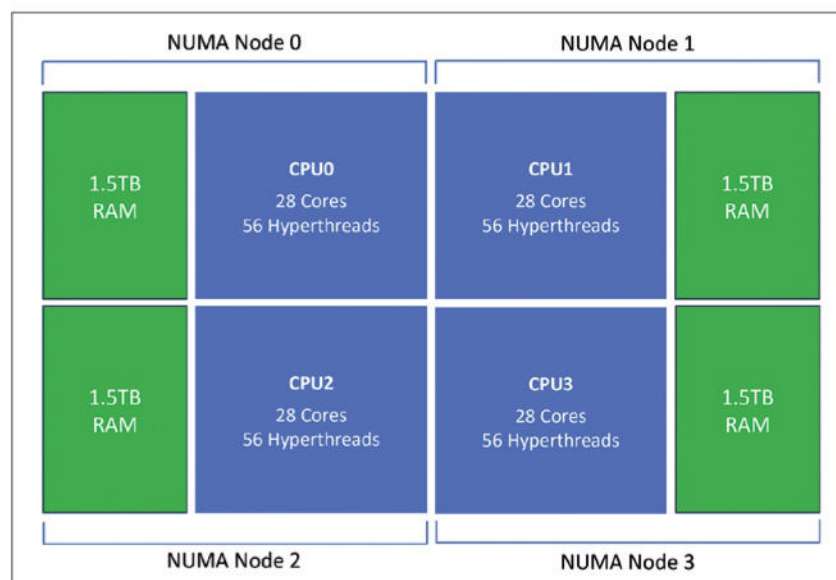


Figure 2: A four-socket system (four NUMA nodes) with 6TB of RAM is equivalent to four CPU sockets with 28 cores each and 1.5TB of RAM per NUMA node.

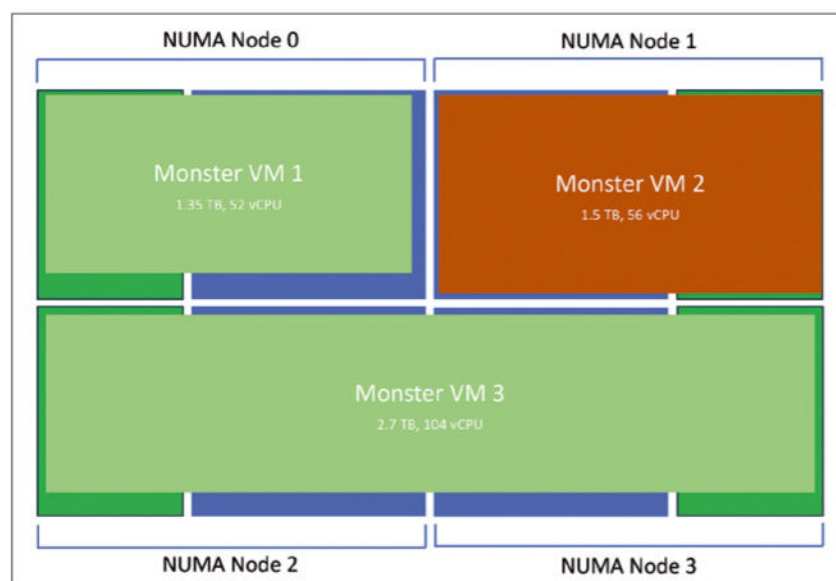


Figure 3: Four-socket system (four NUMA nodes) with 6TB of RAM: four CPU sockets with 28 cores each and 1.5TB of RAM per NUMA node. Two single-NUMA-node VMs and one dual-NUMA-node VM.

RAM and four sockets, each with 28 cores or 56 hyperthreads. If you now create a VM, you could get the dimensions completely wrong and go for 56 vCPUs and 1.5TB of vRAM (shown in the figure as the red monster VM2). On the other hand, you could do it the right way: 26 cores or 52 hyperthreads per NUMA node. I've been generous here and allotted 10 percent for the RAM overhead. The resulting clean VM size is 1.35TB of RAM and 52 vCPUs (i.e., the number of hyperthreads that correlates to 26 cores). This VM is the Monster VM1 in green.

When talking about monster VMs, you will often see significantly larger VMs being built: VMs that are larger than a NUMA node. **Figure 3** also shows a dual-NUMA-node VM that spans nodes 2 and 3. The same rule applies here: two cores per NUMA node as a reserve for the ESXi host and 10 percent RAM. The resulting size is 104 vCPUs and 2.7TB of RAM. If you inflate the VM to the max (call it monster VM4), it hogs all four NUMA nodes and is sized at 5.4TB with 208 vCPUs:

4(28 – 2) cores
 x 2 hyperthreads
 = 208 hyperthreads

So far, so good. As you will remember, the goal for SAP HANA, Oracle, and other business-critical monster VMs is worry-free operation, maximum availability, and maximum performance. A few more arrangements are in order by reserving all the RAM and the entire configured CPU performance of the VM for vMotion advantages later on. Note that this reservation is not made as a CPU count, but as the correlating megahertz/gigahertz number. Going back to the NUMA alignment, if the VM is larger than a NUMA node, it is advisable not just to give the VM vCPUs, but also to adapt the virtual CPU topology accordingly (i.e., to work with virtual cores and sockets). If you create a new VM, the default for the CPU topology is *Assigned at power on*. You could also say that ESXi selects the appropriate topology.

However, for multi-NUMA-node VMs, explicitly determining the topology makes sense in the *VM Options* tab, where you enter the appropriate number in the *Cores per socket* field. For example, if 16 vCPUs are assigned to the VM and it is a dual-NUMA-node VM, you would enter eight cores per socket here. You now automatically get to see *Sockets: 2* and a performance warning; however, the settings are correct, and you can ignore the warning with a clear conscience.

Controlling NUMA Alignment

If you now assign VM options to the monster VM4 from the previous example it would be configured with 208 CPUs and 52 cores per socket, resulting in four virtual sockets. The VM therefore has a NUMA layout that perfectly matches that of the physical server.

Back at the smaller monster VM (single or dual NUMA node), you have already configured the perfect size, but how do you ensure that the VM is really restricted to the physical NUMA node and not unfavorably distributed across several nodes? If a VM fits perfectly on a NUMA node, it is referred to as “NUMA aligned;” if not, it is “NUMA unaligned.” You will want to avoid the second case, because it

causes issues with remote memory access and unwanted latency, which I referred to earlier.

Monster VMA in **Figure 4** is therefore perfectly aligned, whereas VMB is unaligned. How can you remedy this situation? To begin, you want the vCPUs to use the hyperthreads on the local socket and not switch to other cores on other sockets, which you can achieve with a VM advanced setting in the VM's *vmx* file:

```
numa.vcpu.preferHT=TRUE
```

An additional parameter will ensure that VMA in the example is really restricted to NUMA node 0:

```
numa.nodeAffinity = 0
```

If you also want to clean up the monster VMB NUMA alignment and assign it to NUMA node 1, you would need the following configuration for VMA:

```
numa.vcpu.preferHT=TRUE
numa.nodeAffinity = 0
```

and this configuration for VMB:

```
numa.vcpu.preferHT=TRUE
numa.nodeAffinity = 1
```

In **Figure 5**, monster VMC is assigned to the remaining two NUMA nodes

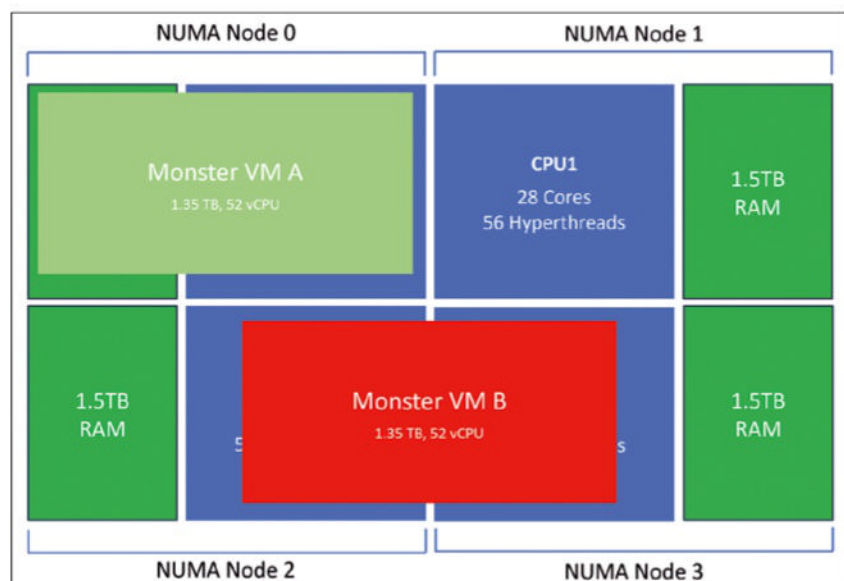


Figure 4: For comparison's sake, NUMA-aligned vs. NUMA-unaligned.

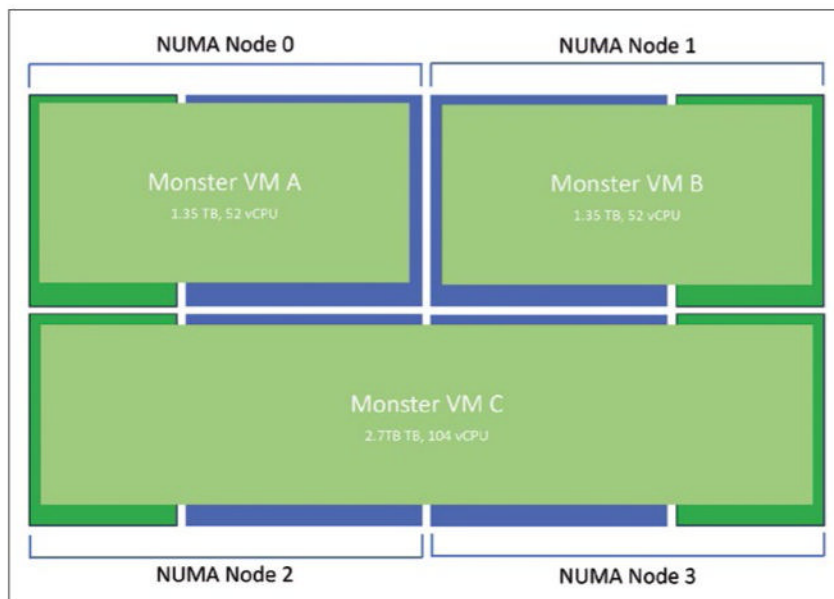


Figure 5: NUMA-aligned VMs with monster VMC.

and now spans two nodes (NUMA nodes 2 and 3). This VM needs the

```
numa.vcpu.preferHT=TRUE
numa.nodeAffinity = 2,3
```

configuration parameters.

Moving VMs Quickly

You now have a VM that is cleanly sized and NUMA-aligned on a host that is perfectly configured on the UEFI side. Oracle or HANA are installed and running without any problems. Sooner or later, though, the server firmware will certainly need to be updated or the ESXi server upgraded to a new version or patch level, and that triggers a restart. In this case, one of the best and most popular features of VMware virtualization comes into play: vMotion (i.e., live migration from one ESXi host to another).

The good news is that vMotion has repeatedly been optimized over the various vSphere versions, with the last major improvements being introduced in vSphere 7.0U2. vMotion can move most VMs without restriction, regardless of size.

If you want to migrate a VM, you will usually also want it to happen as quickly as possible, especially if the ESXi host has state-of-the-art

25, 40, or even 100 gigabit (Gb) network adapters, the VM is extremely large, and, say, 5.4TB of RAM needs to change locations. Now for the bad news: Even if the server does have a 100Gb adapter, you will still need to do a bit of vMotion tuning to be able to use the bandwidth. I'll skip the basics of how vMotion works here and get right to the tuning side. For the sake of simplicity, assume you have a 100Gb adapter and a single vMotion VMkernel port, which is the reality – or close to it – in many environments.

Setting out to move a VM triggers a vMotion process and various helper processes. The latter includes the Completion helper, the Crypto helper, and the Stream helper. However, focus on the Stream helper here, because this process is the one responsible for transferring the RAM content from the source ESXi server to the target; these processes also exist on the target server.

Now for the actual problem: A single vMotion stream (and therefore Stream helper) is limited to an average bandwidth of 15Gb, which means that despite the 100Gb network interface controller (NIC), you would only achieve a maximum of 15Gb vMotion bandwidth, and the vMotion process would be relatively slow. However, if you create multiple streams, the

bandwidth increases and vMotion is accelerated:

- one stream ~ 15Gb
- two streams ~ 30Gb
- three streams ~ 45Gb
- six streams ~ 90Gb
- seven streams ~ 105Gb

In other words, if you want to make full use of your 100Gb interface, you have to configure seven streams, which is not usually automatic, and you will want to be on the safe side with your monster VM. To do this, you need to set the *Migrate.VMotionStreamHelpers* parameters in the *Advanced Systems Settings* on all your ESXi servers to the appropriate value: 7 in this case. The default value is 0, which means that streams are created automatically, and you will not usually see more than four Stream helpers launched. Therefore, you set the values manually.

However, you also have a hardware queue, or multiple hardware queues, between the VMkernel port and the physical NIC. To achieve maximum vMotion performance, you also need to adjust the number of hardware queues to the number of software queues (the same as the number of Stream helpers). Again this is 7 in this example. You can do this by editing the *Net.TcpipRXDispatchQueue* parameter in the ESXi host's advanced system settings. The default is 2. Because this change is hardware related, the server then needs to reboot.

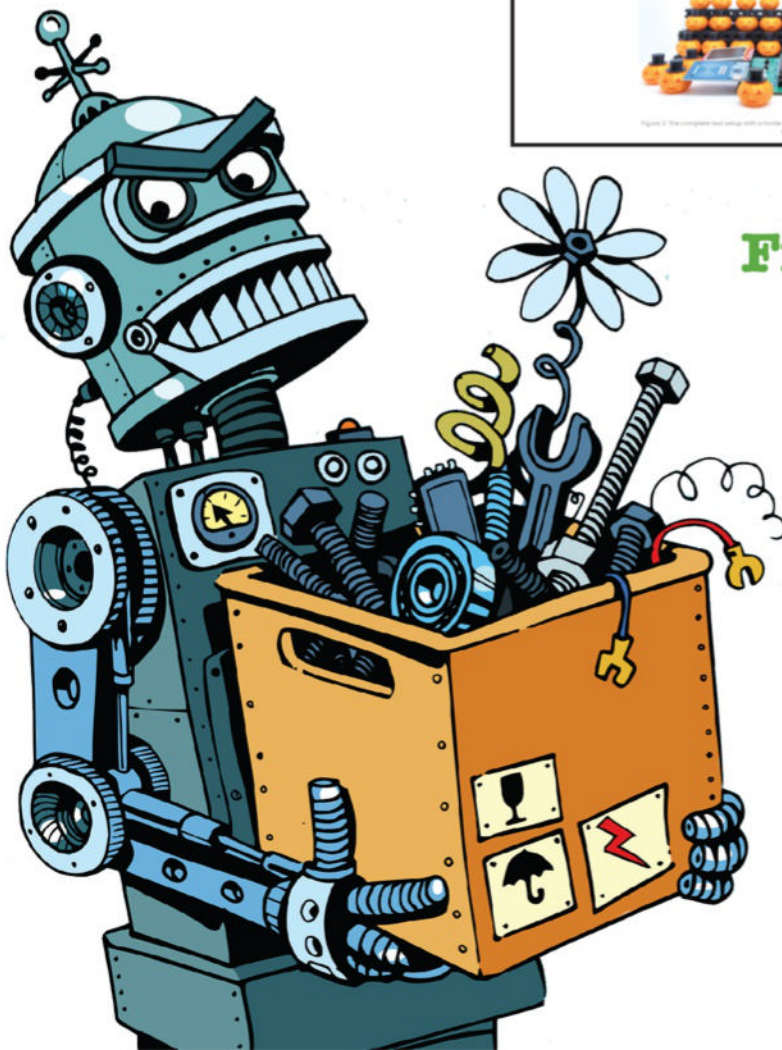
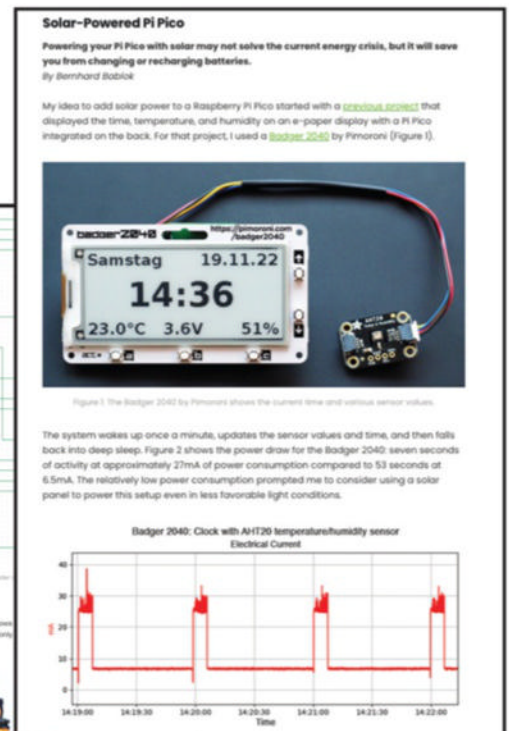
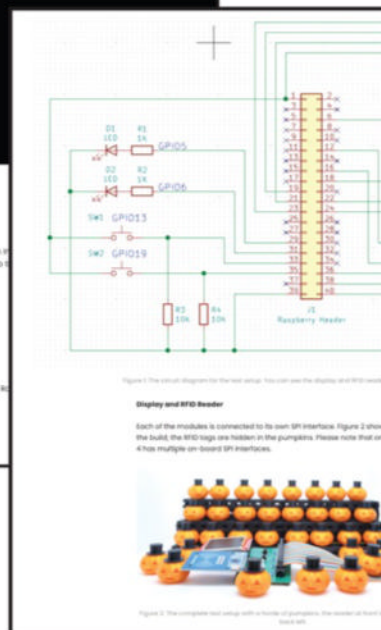
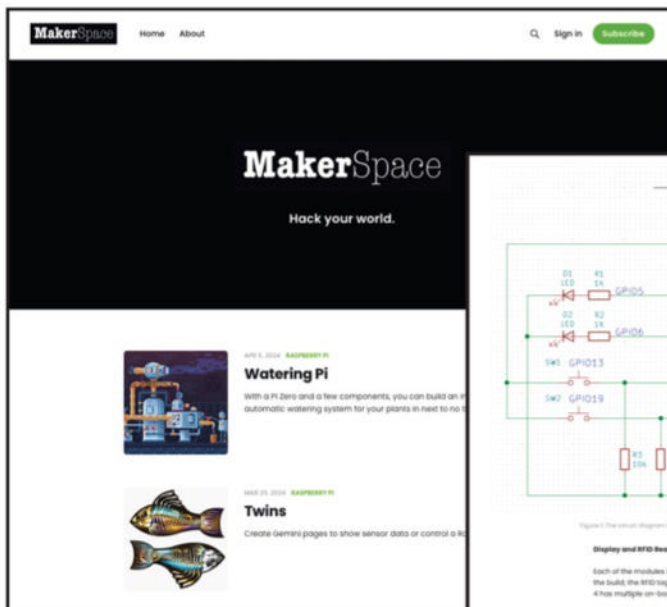
Conclusions

You have now completed the most important steps to size your monster VMs properly, distribute them to NUMA nodes, and create the conditions for worry-free vMotion. If you follow these rules, 85 to 90 percent of all monster VMs, whether HANA, Oracle, or Microsoft SQL, should be available, offer good performance, and support moving. The remaining 10 to 15 percent will probably present pitfalls, such as high-latency sensitivity in the memory area and requirements for the lowest possible network latency because of system replication or real-time applications. ■

MakerSpace-Online

New Maker Content Every Week

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



Fresh Content, Delivered

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

Join Now!



<https://makerspace-online.com>



File Integrity Checks with AIDE

Detection

If an attacker gains access to systems by working around your defenses, you need to discover the attacker's tracks in good time, at least to mitigate the further risk of damage. We show you how to monitor changes to files with the Linux AIDE tool. By Matthias Wübbeling

Advanced Intrusion Detection Environment (AIDE) uses various techniques to detect the manipulation of files, starting with regular expressions for selecting the files to be included in the integrity checks. The files are then processed with hashing tools to generate checksums. Additionally, the associated filesystem properties, such as access rights, inodes, SELinux, Amazon Elastic File System (EFS), and other extended attributes, are also taken into account.

Setup and Use

To use AIDE for integrity checks, you first need to install the tool with your distribution's package manager. AIDE is included in all the popular distributions; if your environment is not supported, you can easily download the release from the GitHub project [1]. After the install, launch AIDE directly; you will need to be root, use `sudo`, or launch a root shell. Launching AIDE without passing in command-line arguments starts a check directly or complains that no database for the check exists below the path specified in the configuration. To prepare the AIDE database with the current status, trigger the database `init` with the command:

```
aide --init
```

In our lab, this took north of one and a half minutes for around 318,000 files. AIDE consumed virtually no resources on a CPU with 12 cores. The file `/var/lib/aide/aide.db.new` was created and had to be renamed `aide.db` for use in checks. The output will contain the checksums of the various hash procedures that are supported. Of course, you also need to keep an eye on these checksums to detect potential manipulation. Because several hashes are generated, an attacker cannot simply leverage potential vulnerabilities in individual hash functions to manipulate data without you noticing. If you now run a check directly after creating the database, you would naturally expect AIDE not to find any changes to the files. However, even after a short wait, changes to folders or files can occur on the basis of the default configuration, which AIDE will then show. Of course, you can also trigger output as shown in **Figure 1** by simply changing one of the files to be checked yourself. Try out various changes and get a feel for the output and how you can use it. Checking the hashes of the uncompressed database during each run is also important.

In this example, the `/etc/security/access.conf` file is modified. Of course, this prompts changes to

the `/etc` and `/etc/security` folders, which AIDE reports under the *Changed entries* section. The letter *d* at the head of the lines stands for "directory," and *f* stands for "file." The `=` sign means no files have been added or removed from the folder, which would be shown by the `>` and `<` symbols (size increased, size decreased), respectively.

The individual dots are placeholders for the various checks performed by AIDE. The letters *m* and *c* show changes to the timestamps in the filesystem for the "modification time" (for changes to the file/directory contents) and the "change time" (for changes to the filesystem status information). The indicators *i* and *h* for the file itself show a change in the inode and a discrepancy in at least one of the hashes that was checked. Further details for each change follow this summary.

If you automate the use of AIDE, you will definitely need to evaluate the program's return values. Like (virtually) any application, a return value of 0 means that everything worked correctly with no errors during execution. If you check or update the database, values 1 through 7 are returned if successful. AIDE uses these three bits to inform you whether files were created, deleted,

or modified. In AIDE, error codes start with return values greater than or equal to 14.

Configuration and Updates

The configuration file for AIDE resides in `/etc/aide/` and contains the documentation for the available check procedures and predefined combinations for different file types, such as configuration files, binary files, or logfiles. The files you want AIDE to check are then defined by regular expressions. An exclamation mark `!` to the left of the expression means that

files that were included with previous rules are ignored later.

In production, you will need some time to get set up and will repeatedly need to exclude files on your systems from the check if they are subject to regular changes during normal operation. Both binary files and configuration files change, particularly when the software you use is updated. In these cases, you need to update the AIDE database after the software update. Of course, you could also simply initialize a new database, but then you would not get an overview of modified files. Obviously, you need

to make sure that only the updated files have changed at the time of the update. The command

```
aide --update
```

lets you update the database and outputs the modified files.

Manipulation Protection

Unfortunately, AIDE does not scan remote files by default, which means an attacker could change the AIDE binary and, for example, manipulate the output in such a way that malicious changes are not displayed.

To rule this possibility out, you can secure the binary with checksums and compare the values before starting. As is always the case with host-based attack detection, this strategy does not make things 100 percent secure, but it significantly limits the attacker's options.

Conclusions

Changes to binary data and configurations are artifacts of cyberattacks. If you notice them in good time, you might still be able to stop an attacker and prevent the damage spreading across your network. In this article, I looked into how to detect these changes with AIDE.

Info

[1] AIDE: <https://github.com/aide/aide/releases/>

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.

```
it-administrator.de ~ * aide
Start timestamp: 2024-04-05 13:33:07 +0200 (AIDE 0.18.4)
AIDE found differences between database and filesystem!!

Summary:
  Total number of entries: 318193
  Added entries: 0
  Removed entries: 0
  Changed entries: 3

-----
Changed entries:
-----

d =.... mc.. : /etc
d =.... mc.. : /etc/security
F >.... mcl.H : /etc/security/access.conf

-----
Detailed information about changes:
-----

Directory: /etc
Mtime : 2024-04-05 09:34:46 +0200 | 2024-04-05 13:32:27 +0200
Ctime : 2024-04-05 09:34:46 +0200 | 2024-04-05 13:32:27 +0200

Directory: /etc/security
Mtime : 2024-04-03 17:47:10 +0200 | 2024-04-05 13:33:05 +0200
Ctime : 2024-04-03 17:47:10 +0200 | 2024-04-05 13:33:05 +0200

File: /etc/security/access.conf
Size : 4564 | 4583
Mtime : 2024-03-11 02:11:44 +0100 | 2024-04-05 13:33:05 +0200
Ctime : 2024-03-11 02:11:57 +0100 | 2024-04-05 13:33:05 +0200
Inode : 15266310 | 55574654
MD5 : 3C1079ad2VexH0hMc0KGr== | Cw04uyVHCjipBLuSuFYwdg==
SHA256 : 9okVx0tjegy/oBzybcRppz9urLBJXvxL | tK3mKd8dSQ+Nhwa3L1YKqV+s8EaF9+R
B07Lx1a8s0U= | ry3KIg3mp40=
RMD160 : hf4KQHB2TmDV6S3FYFZAtukVnEY= | BN1i1k65iMSoU6zhgVbt9TYgm7o=

-----
The attributes of the (uncompressed) database(s):
-----

/var/lib/aide/aide.db
MD5 : zm1Yx2hZLqEz3/ZSS/nHuQ==
SHA1 : n65MEBvH2ZBU863am7/H29rh780=
SHA256 : Ak4KvSyctshHfU0zqeV8sVrGcMeLnED
T71u30Qc3dH=
SHA512 : N+paq3H4Cn4x8RZVBepC27tMTb2BxMKT
waRm88k0LAETgTlKwEnBR27xmN6HJNad
yNa0ubvAtJTtVqYtCfPolog==
RMD160 : bmRoCX1oRbWbRFUe9P6eC12C52k=
TIGER : NMLxNlH43ndrFyuhp623MgwaXSP15gpS
CRC32 : TSgHgw==
WHIRLPOOL : vuH4GNGNv2Dv3mNbG7McqKeATTJapaqM
50sS9RT6KpHZ82R471iTdJzZzXxU08LH
/uxpe2uCKuxFUMMcUUpKg==
GOST : M0X8/4PJI07Axyfv/uV3+ESx2IEytLuD
DTPC1Vad3+U=
STRIBOG256: y71062bU01dfeVfnznSosiRpP+bpQt9
0fcH1p7Kd0E=
STRIBOG512: twa8R0IGj/Fa8M+jwFz/KK1La6JSTLgt
TPvfrntR0JbJru7MvYhcg+VF3iH17x
7Q3ABsxHMY4b2zyH1s5ug==

End timestamp: 2024-04-05 13:36:10 +0200 (run time: 3m 3s)
it-administrator.de ~ * echo $?
4
```

Figure 1: Test how AIDE reacts when you make changes to a file.

Reducing the Attack Surface in Windows

Strong Defense

The sum total of all possible points of attack can be defined as the attack surface, and you need to take every opportunity to minimize it to the extent possible. Windows has built-in rules that minimize the attack surface; they simply need to be enabled. By Matthias W#bbling

The classic protection mechanisms for corporate IT infrastructure have always included regular software updates, up-to-date virus and spam protection, one or multiple firewalls (think network segmentation), and intrusion detection and prevention systems. However, even admins that can tick each of these boxes are not automatically safe and can see their companies fall victim to hackers. If you conceptualize an organization's IT infrastructure, you can imagine a figurative surface that might include web services offered to the outside world over a network, although it by no means comprises all the elements of the interface. The "attack surface" on which Microsoft documentation [1] focuses is the sum total of potential attack points on the computer systems of an IT network that unauthorized users could exploit. Other terms for these points of attack include security gaps or vulnerabilities, which basically also include physical access to protected hardware.

Besides all the obvious network components, including every type of hardware and the firmware installed and running on it, you also have potential points of attack for hackers on the software side. These vulnerabilities do not necessarily have to be errors in the development of the server software itself. Internet Information Services (IIS) for Windows Server, Apache or N, mail servers, and many other standard services usually come with a secure basic configuration, but the software running on or behind the server often offers direct access to further infrastructure or data in the form of APIs or comparable interfaces. Even human interfaces can be a relevant part of the attack surface. Cybercriminals often focus on access to the employee or customer user accounts and the infrastructure resources that can be accessed from those accounts. Of course, weak, easily guessed, or compromised passwords used for multiple services pose a risk that is as serious as

user-managed devices in bring-your-own-device (BYOD) scenarios or shadow infrastructures set up without the knowledge of the IT department.

Attack Vector and Surface

The attack vector is the method or steps a hacker uses to penetrate a network or system, bypass existing security measures, and gain access to resources – or simply cause irreparable damage. Attack vectors can be diverse and comprise several steps, which, at times, makes them difficult to detect in the mass of legitimate user activities. Technically speaking, then, attack vectors consist of a mix of phishing attacks, malware in email attachments, zero-day exploits of software vulnerabilities, man-in-the-middle attacks, and physical access to critical resources.

This distinction forms the basis for the development of effective security strategies. Whereas the attack vectors represent concrete methods that criminals leverage to gain access step

Photo by Sander Sammy on Unsplash

by step to a resource, the attack surface offers a holistic view of all possible attack vectors and the relevant devices or services. Three classes can be distinguished: unknown, known, and malicious assets.

Unknown assets are one of the biggest security problems and are a kind of shadow infrastructure that is managed individually, and usually without extensive specialist knowledge, by employees outside the control of the IT department. Known assets are the opposite: devices and services managed by the IT department, such as all client computers and servers, including the services running on them and the associated dependencies. Malicious assets are sometimes known but outside the direct control of the IT department, such as malware domains from domain generators, (web) servers hijacked by criminals, or mail servers used to send spam.

Microsoft Defender to Start

The core of attack surface mitigation on corporate assets consists of Microsoft Defender for Endpoint or Windows E5 licenses in an environment with Entra ID and Intune. In fact, you can also install Defender on Linux devices in your infrastructure, but rulesets other than those recommended in the standard documents are likely to be more useful.

The rules for reducing the attack surface can be enabled on Windows Server from version 2012 and Windows 10 and work without the Defender portal or group policies, even on home computers (more on this later). Armed with Defender's built-in tools, you can use the checklists provided by Microsoft to create policies that reduce the existing attack surface, and you can enable or disable the rules in different modes in the Endpoint security management console.

Before you simply go ahead and wildly enable rules in your organization, Microsoft recommends careful planning and testing to avoid enabling policies that can later

cause problems in your infrastructure. In the planning phase, you first want to establish an overview of the business areas and identify the guidelines that make sense for those areas. Departments with software developers probably need different authorizations for Windows functions than for accounts.

You also need to keep an eye on the applications in use and the effects planned changes will have on them. During the changeover phase, you should put together a team that will act as a point of contact for this business area, evaluate any issues reported, and resolve them promptly. In large organizations, the team will comprise one or more admins and at least one analyst from your security operations center. In small companies, the admins will often have to analyze the reported problems and possible threats by themselves.

Rolling Out Rules

If you want to deploy rulesets for a very large number of devices in your organization, you need to define rings like those you probably already use for Windows updates for step-by-step deployment. It makes sense to match the rings for Windows deployment itself. Start with the inner ring and enable the selected rules in audit mode initially. After some time, you will start seeing reports on the Defender portal. In a 30-day view, you can see the effects of your ruleset and how many incidents the individual rules caused.

Work through these reports, identifying problems and false positives, and, if necessary, define exceptions for the further process. Note that existing Defender exceptions also apply to most of the attack surface reduction rules. When folders, programs, or processes are shared, they are not monitored, although this does not apply to the Windows Management Instrumentation (WMI) persistence rule.

In the next phase, you will exit audit mode and activate the defined rules. If you identified rules that repeatedly caused problems during your audit,

you might not want to activate block mode for them but use warning mode first to collect further data. You can define exceptions, and they remain in effect on the local device for 24 hours. It is no problem to enable only the simple and clearly problem-free rules at first. Again, you will want to monitor the messages on the Defender portal during this phase and talk to the employees concerned. Also bear in mind that a reported incident is not automatically a false positive just because a user reported the problem; a legitimate attack could be behind the warning.

Check Logfiles Regularly

Once you have finished with one ring, move on to the next and carry out the same steps until all the devices have appropriate rulesets in place. Operations only start at this point. In this phase, it is important to continue to pay attention to the rules and to reports in Defender. Even though rulesets can be considered to be established after some time, you still want to check the reports regularly and analyze outliers. In fact, Microsoft recommends doing this on a daily basis. It is also important to check regularly any exceptions that were created and remove them again if needed. If you are unsure about the effect of a rule, you can switch it back to audit mode at any time, of course. If you want to check the rules that have been set, you can use the Microsoft Defender Advanced Threat Protection (ATP) portal [2], which is the place to go for in-depth information on potential attacks and to pick up scripts or code snippets that simulate attacks, and determine the effects on your systems' attack surfaces.

Three Recommended Standard Rules

Knowing the rules for reducing the attack surface and investigating them for your own requirements is important. **Table 1** is an overview of the current crop of 19 attack surface rules (ASRs). All existing rules are unlikely

to make sense in every infrastructure or situation. Microsoft only lists three rules for standard protection that you will want to enable without further analysis.

The basic rules include blocking the theft of credentials from the local Windows security authority subsystem (lsass.exe), blocking the exploitation of vulnerable signed drivers, and preventing persistence mechanisms with WMI event subscription, which is primarily an option used by malware to hide itself by using the WMI repository and to execute repeatedly on the target system. Incidentally, the first of the three rules for securing login information can result in numerous log entries, because legitimate apps sometimes also access this type of user data.

The rules can be roughly divided into five categories:

1. protection against exploits and exploitable vulnerabilities,
2. preventing the spread of malware by email and the web,
3. restrictions for Office applications that prevent the execution of malicious code,

4. measures against the theft of login information, and
5. blocking the execution of unsigned or untrusted processes.

Some of the rules use analysis results from Microsoft or other providers in the background. The rule that detects vulnerable signed drivers, for example, checks whether they have been recognized as dangerous elsewhere as soon as they are downloaded. If this is the case, the download is prevented. However, if a malicious driver is already running on a system, it can still be executed, and because these are signed drivers, no other protections are in place. If you would like to have a driver checked explicitly before using it, you can send it to Microsoft for analysis [3].

Hardening Office

Many of the rules relate to the use of Microsoft Office and the files or processes it creates. Time and time again, criminals use this software as a gateway for malware. You can deploy a rule to keep Office applications from creating executable

content that allows malware to break out of the Office context and infect the system. Another Office rule keeps all Office applications from creating child processes. In concrete terms this means that processes cannot be started by Visual Basic for Applications (VBA) macros, for example, and stops malware that acts as a dropper from downloading custom code from the Internet and trying to execute it.

The rule for extended ransomware protection also draws on findings from the Microsoft cloud and the Microsoft Antimalware Protection Service (MAPS). To enable it, cloud protection must be active in Defender. This rule does not block files that have been identified as harmless online, have a valid signature, or are so widespread that they cannot be ransomware.

Another point of attack for malware can be shut down with the rule for blocking untrusted and unsigned processes that are executed over USB. This rule stops executable files (e.g., files with the extensions EXE, DLL, or SCR) launching from USB

Table 1: Attack Surface Reduction Rules

ASR Rule	Recommended Standard Protection	GUID
Blocks abuse of exploited, vulnerable, signed drivers	Yes	56a863a9-875e-4185-98a7-b882c64b5ce5
Stops Adobe Reader from creating child processes	No	7674ba52-37eb-4a4f-a9a1-f0f9a1619a2c
Stops all Office applications from creating child processes	No	d4f940ab-401b-4efc-aadc-ad5f3c50688a
Blocks credential theft from the local security authority subsystem (lsass.exe)	Yes	9e6c4e1f-7d60-472f-ba1a-a39ef669e4b2
Blocks executable content from email clients and web email	No	be9ba2d9-53ea-4cdc-84e5-9b1eeee46550
Stops executable files executing unless they meet a distribution, age, or trusted list criterion	No	01443614-cd74-433a-b99e-2ecd07bfc25
Blocks the execution of potentially obfuscated scripts	No	9e6c4e1f-7d60-472f-ba1a-a39ef669e4b2
Stops JavaScript and VBScript starting downloaded executable content	No	d3e037e1-3eb8-44c8-a917-57927947596d
Stops Office applications from creating executable content	No	3b576869-a4ec-4529-8536-b80a7769e899
Stops Office applications from inserting code into child processes	No	75668c1f-73b5-4cf0-bb93-3ecf5cb7cc84
Stops the Office communication application from creating child processes	No	26190899-1602-49e8-8b27-eb1d0a1ce869
Blocks persistence through WMI event subscription (Defender exceptions do not apply)	Yes	e6db77e5-3df2-4cf1-b95a-636979351e5b
Blocks the creation of processes by PSEXEC and WMI commands	No	d1e49aac-8f56-4280-b9ba-993a6d77406c
Blocks rebooting of the computer in safe mode (preview)	No	33ddedf1-c6e0-47cb-833ede6133960387
Blocks untrusted and unsigned processes executed by USB	No	b2b33d-6a65-4f7b-a9c7-1c7ef74a9ba4
Blocks the use of copied or imitated system tools (preview)	No	c0033c00-d16d-4114-a5a0-dc9b3a7d2ceb
Blocks web shell creation for servers	No	a8f5898e-1dc8-49a9-9878-85004b8a61e6
Blocks Win32 API calls from Office macros	No	92e97fa1-2edf-4476-bdd6-9dd0b4dddc7b
Deploys advanced protection against ransomware	No	c1db55ab-c21a-4637-bb3fa12568109d35

data carriers on a device, including files that users have copied from a connected USB data carrier to the device's hard drive. The rule is not as easy to work around as you might think at first glance.

Configuration by PowerShell and Tools

You will normally want to use the graphical tool to configure the rule-sets on the Defender portal. The rules and evaluations are available side by side, and issues can be found quickly with an advanced search. If the portal is not available, the rules can be enabled and disabled with PowerShell, but you must use the GUIDs listed in the table. To enable a rule, open PowerShell in administrator mode. If you want to set the rule *Block executable content from email client and web-mail* to audit mode; you would need to run the cmdlet:

```
Add-MpPreference 2
-AttackSurfaceReductionRules_Ids 2
be9ba2d9-53ea-4cdc-84e5-9b1e46550 2
-AttackSurfaceReductionRules_Actions 2
AuditMode
```

Instead of `AuditMode`, you can define the other variants by stipulating `Warn` or `Block` as an argument. To disable a rule, use the `Disable` argument. If you want to use PowerShell to enable exceptions for folders, files, or processes, run the cmdlet:

```
Add-MpPreference 2
-AttackSurfaceReductionOnlyExclusions 2
<parth or resource>
```

To execute your actions for all existing rules, you can write a small script that first queries the existing GUIDs with

```
(Get-MpPreference).AttackSurface2
ReductionRules_Ids
```

and then works its way through this list. If you do not have the option of setting the ASR rules by group policies and the configuration by PowerShell is too complicated for you, two free tools can help: the zero-installation `ConfigureDefender` [4] by Andrzej Pluta and `DefenderUI` [5] by VoodooShield.

Both tools let you set all ASR rules in a clear-cut interface, and you can customize the properties of Microsoft Defender, for example, by defining the sensitivity level for the scanner in its response to suspicious files, how long Defender will wait for a response from the Microsoft cloud before waving a file through, or how much compute power malware protection can harness for its own needs.

Both tools come with various predefined security levels that you can activate with a click of the mouse, if necessary, and include, for example, recommended settings (*High* for `ConfigureDefender` and *Recommended* for `DefenderUI`), an interactive level with more frequent queries, and a maximum security level that basically enables all rules. If you misconfigure something and don't know which setting is causing problems all of a sudden, both programs also let you revert to the Windows default settings at the push of a button. All the settings configured in the GUI can be set by the PowerShell `Set-MpPreference` cmdlet, and its `Get` counterpart queries the settings.

Conclusions

Windows offers more in terms of security than meets the eye. A number of rules under the hood reduce the

attack surface. In this article, I looked into the basic aspects of attack surface reduction. The 19 rules might seem fairly simple, but they do address a number of popular attack gateways, so you should consider using them to protect your infrastructure. What's more, these built-in resources do not have any third-party code. Under normal circumstances, you will experience no, or hardly any, restrictions in your daily work, although you have slammed the door in the face of many attackers. ■

Info

- [1] Vangel, D., M. Athavale, and J. Bregman. Understand and Use Attack Surface Reduction Capabilities. Microsoft Learn, June 4, 2024, [<https://learn.microsoft.com/en-us/defender-endpoint/overview-attack-surface-reduction>]
 - [2] Microsoft ATP portal: [<https://demo.wd.microsoft.com>]
 - [3] Uploading drivers for analysis: [<https://www.microsoft.com/en-us/wdsi/driversubmission>]
 - [4] `ConfigureDefender`: [<https://github.com/AndyFul/ConfigureDefender/>]
 - [5] `DefenderUI`: [<https://defenderui.com>]
-

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.



Explore automation-as-code with Ansible

Automated

The Ansible automation tool makes it really easy to implement IT scenarios as code. We use structured YAML code to roll out Ansible in the form of AWX. By Andreas Stolzenberger

The everything-as-code trend is popular in IT today. The complete hardware and software configuration is stored in text files, and an automation tool rolls everything out. Fortunately, arcane file formats (e.g., `sendmail.cf`) are pretty much a thing of the past; instead, you can expect easily readable formats such as YAML and perhaps JSON. The information in these files can usually be converted into one of the other formats without loss.

YAML is the preferred format of the Ansible automation tool, which is used here for as-code automation. Thanks to its modular structure, Ansible can control pretty much everything in the data center and

the cloud, including rolling itself out from its command line. This scenario might sound strange, but in the automation-as-code example, it makes sense to roll out a complete AWX setup in this way on Kubernetes and configure the AWX server with all the required settings.

The code here can act as a skeleton, or a template if you prefer, for your own more extensive as-code scenarios, as well as for cases in which you roll out and configure other applications. Because a huge amount of code is involved, I can only describe fragments of it in detail here. The complete code is available on GitHub [1]. Incidentally, it works with both the free AWX and, with

a few minor changes, the commercial Ansible automation controller implementation.

Two Approaches

The first approach to automating the AWX rollout involves setting up an empty AWX server, configuring it in the web user interface (UI), and then exporting the complete configuration as a JSON or YAML file. You can then upload the results to a new, empty AWX server. The approach is mainly used if you want to clone a setup or migrate to a new platform or a new release (e.g., Ansible Tower to Ansible controller).

However, this approach does not lend itself to as-code scenarios, because the YAML files you need to export can be fairly large and confusing. With AWX as code, you will be looking to create your AWX configuration directly in YAML, which you will then be able to roll out automatically, which is why the code used here needs to be simpler, more clear-cut, and readable for the admin.

The second approach uses a playbook that fetches the configuration

data from a YAML document. You can format and expand the document to suit your own requirements, which makes the YAML document far simpler, at the price of no longer being compatible with the exported format. Of course, the as-code idea is to roll out the installation from the code and not go down the opposite route.

Code vs. Config

One basic rule when using Ansible is that you want to avoid assigning values to variables in your Ansible playbook, which would massively affect the playbook's flexibility. Therefore, variable declarations should always be kept separate from the actual Ansible code.

In this case, you will use two separate files: `vars.yaml` contains the complete configuration of the AWX server to be rolled out, and `secrets.yaml` stores the passwords and keys you need for your AWX service's credentials.

Once you have entered all the access credentials and passwords in readable form in the YAML file, you need to encrypt `secrets.yaml` with Ansible vault. When executing the code, and only then, you can unlock access with the `--ask-vault-pass` option.

Hierarchical Variables and Dictionaries

The configuration variables for the as-code scenario are grouped hierarchically:

```
scm:
  name: "Github"
  User: "user01"
  url: "https://www.github.com"
```

The playbook then accesses variables like `{{ scm.url }}`, which means you can easily see the sub-section to which a variable belongs. If you want to extend the code later, you can add sub-variables at any time.

The second entity in the variables area is the dictionary. With this un-numbered array, Ansible can execute a loop for each element:

```
user:
  kirk:
    firstname: "James T"
    rank: "Admiral"
  spock:
    rank: "Captain"
  uhura:
    firstname: "Nyota"
    rank: "lieutenant"
```

An Ansible task now uses,

```
"with_dictionary: "{{ user }}"
```

or, in line with the more modern (but not necessarily better) notation,

```
"loop: "{{ user | dict2items }}"
```

to iterate through a loop. In a loop task, Ansible then accesses parts of the dictionary with `items`, which is a hierarchical loop variable. In this example, the first execution of a task in the loop would give you the following variable content:

```
item.key = "kirk"
item.value.firstname = "James T"
item.value.rank = "Admiral"
```

When declaring a dictionary, you do not need to declare all of a variable's hierarchies, as in the case with the `spock` entry. That said, the playbook task must be able to cope with empty variables in this case. Ansible has a default function for this:

```
- ansible.builtin.debug:
  msg: "firstname : {{
    {{ item.value.firstname |
      default('Mister') }}"
```

If the `item.value.firstname` variable is empty in this case, Ansible uses `Mister` instead. This approach is used later to configure the SSH ports of the inventory entries.

Armed with these simple tips for formatting YAML dictionaries and using loops to evaluate them in Ansible, you can now basically configure arbitrary hardware and software with Ansible. The AWX rollout in this example is just one of many options.

Route vs. Ingress

OpenShift and MicroShift use name-based routes to forward HTTP and HTTPS traffic to applications in pods. You need to resolve all the subdomains in your DNS with the same IP address as your Kubernetes node (or cluster virtual IP). For example, if your single-node setup is running on `kube.demo.com` with the IP address `192.168.1.1`, you need to resolve all sub-domains (`*.kube.demo.com`) to `192.168.1.1`, too.

The router in your Kubernetes installation can then forward the packets to the service on a name basis. For example, one AWX setup could then run on `http://awx01.kube.demo.com` and another on `http://awx02.kube.demo.com`, which is why the demo code on GitHub has a route definition in the `01a_awx_kube_route.yml` playbook containing:

```
apiVersion: route.openshift.io/v1
kind: Route
```

Note that this only works with OpenShift or MicroShift. Starting in Kubernetes version 1.19, the developers adopted the OpenShift route concept and implemented it there as an Ingress route (Figure 1); therefore, a second playbook (`01a_awx_kube_ingress.yml`) in the demo code declares an Ingress route:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
```

This strategy also works with MicroK8s or K3s. You might have to adjust the Ingress definition if routers such as Traefik cannot cope directly.

You need to include the task from the main playbook. Depending on which Kubernetes distribution you are using, comment out `Route` or `Ingress`. Accordingly, you must use the `Route` or `Ingress` section in the `99_cleanup.yml` playbook.

AWX in Five Minutes

You need a Kubernetes environment for AWX. A simple single-node

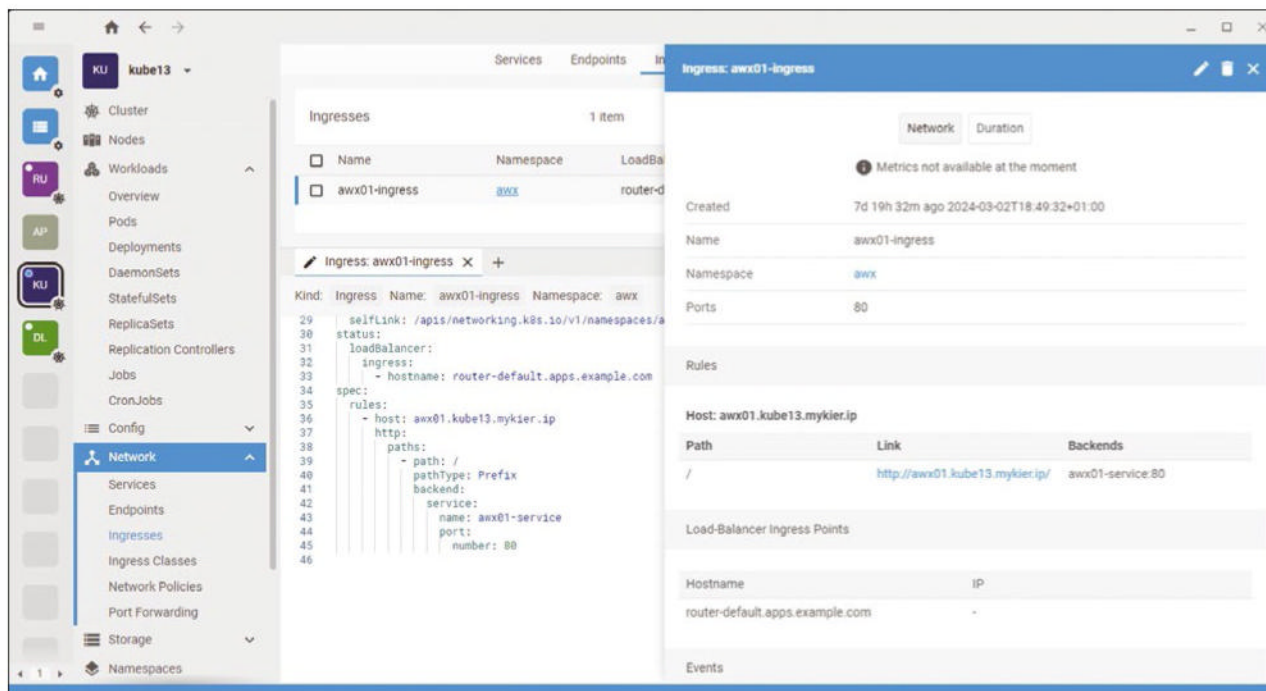


Figure 1: Depending on your Kubernetes distribution, select an OpenShift or – as shown here – an Ingress route for accessing the AWX service.

setup with distributions such as MicroShift, K3s, or MicroK8s is all you need; then, you can use Kustomize or Helm to install the AWX operator with default settings, as described in the documentation [2]. Operator version 2.12.2 was used for this example.

The main playbook for the AWX `00_awx_full.yml` rollout has the same structure as a workflow template in AWX itself and includes other playbooks in the appropriate places. In the best Ansible style, you want to avoid transferring additional playbooks as `include_tasks` and create appropriate roles instead. However, because the tasks to be included are so simple and do not use any variables, files, or templates other than the general configuration variables, it is not worth paying the price of the

overhead for roles in this case. Of course, if you are planning to use an AWX server to roll out further AWX servers, you would want to convert the `00_awx_full.yml` playbook into a suitable workflow template. The playbook fetches all the variable values from two files (`secrets.yml` and `vars.yml`) and then defines shell variables (environment) with the access credentials for the Kubernetes cluster and the AWX setup to be rolled out, which are adopted by the playbook modules further down the line. To keep the example simple, just add the `kubeconfig` file for authentication to the playbook.

In larger installations, you would define a bearer token instead with restricted access to the defined namespace. The AWX service runs without SSL in this scenario and is where you would select the HTTPS protocol in a production environment and declare a route or Ingress with SSL.

What is originally missing, though, is the `CONTROLLER_PASSWORD`. According to the AWX operator documentation, you can store your own password as a Kubernetes secret when rolling out an AWX setup. Unfortunately, this did not work

in our lab for unknown reasons. I decided to leave out the declaration, prompting the AWX operator to generate a random, secure password for the AWX setup during the rollout itself before simply parsing this out at a later date and adding it to the environment.

The first playbook, `01_awx_kube.yml`, does not require any AWX credentials as yet because it only rolls out the AWX service on Kubernetes. In this rollout, the playbook requires two persistent volumes (PVs) instead of the usual one:

```
spec:
...
  postgres_storage_requirements:
    requests:
      storage: 10Gi
  projects_persistence: true
  projects_storage_size: 10Gi
```

The `postgres_storage_requirements` variable can be used to specify the size of the PVs for the database. You definitely need this parameter because the AWX operator only allocates 1GB to the database by default, which will fill up quickly during active use of your AWX environment. The controller uses the second

Listing 1: Wait for the AWX API

```
- name: Wait for the AWX API to be accessible
  ansible.builtin.uri:
    url: "http://{{ awxsetup.name }}.{{ awxsetup.baseurl }}api/v2/ping"
  register: result
  until: "result.status == 200"
  retries: 30
  delay: 10
```


volume to save the project data (i.e., the Git repositories with playbooks) in non-volatile storage. AWX basically also works without persistent project storage but then has to reload completely all the project repositories when restarting the pod or redeploying. Persistent volumes offer a faster experience.

As described in the “Route vs. Ingress” section, depending on the Kubernetes variant used, one of the playbooks that defines the OpenShift or Ingress route to your AWX setup then follows. In practice, tasks that wait for something to happen need to be included in many places in your Ansible playbooks. Ansible is often faster than the services it automates, which is why the configuration playbook, `02_aws_config.yml`, starts with a wait loop (Listing 1).

This code tells Ansible’s URI module to check the AWX REST API at predefined intervals of 10 seconds. The API has a ping function that responds with the HTTP code `200 OK` once the AWX environment is ready. If AWX fails to respond after five minutes, the playbook aborts with an error. The ping API responds without you having to log in to the AWX server, because you do not yet know the admin password at this point. If you use HTTPS for the AWX server as described earlier, you need to adjust this accordingly in the queue task.

The wait loop is followed by a second static 30-second wait loop. Experience shows that the ping API often returns an `OK` too soon and that the downstream configuration tasks cannot reach it after all. You can avoid errors here by giving the API another 30 seconds. If the API responds, the AWX rollout is complete, and the operator has also stored the admin password in a Kubernetes secret. The next task retrieves the password from the secret (Listing 2) and makes it available to the subsequent steps.

Again, you need to resort to a little trick: Ansible can set the environment variables for the complete automation run. In this case, they appear at the start of the playbook before the

tasks section. Alternatively, Ansible can assign environment variables to each individual task, which then only apply to the respective tasks. No “add key-value to existing environment” function exists, so you summarize the following configuration tasks to create a block. Because the block itself is considered to be a task, and the `CONTROLLER_PASSWORD` environment variable was added to the block, it is automatically available for all tasks within the block (Listing 3).

The playbook uses the `aws.ansible` collection to upload the AWX configuration. The collection’s modules address the AWX REST API and use it to generate the required settings, but it is important to adhere to the correct sequence. For example, you cannot create a job template until the associated project template exists and is synchronized with its source code management (SCM) system.

The project can only be created after you have created both an organization and suitable SCM credentials. The correct order is: Organizations, Credentials (SCM, Machine, Registry (optional)), Registry and Execution Environments (optional), Inventories, Hosts (if static; otherwise, dynamic inventory), Projects (must be *Update Revision on Launch*), Templates (job and workflow (optional)).

This example only creates one organization with one project and several job templates; however, this idea can be extended as desired. For example, you could bundle the configurations for several projects into separate files, outsource part of the playbook from `aws.ansible.project` to a separate file, and include it in a

```
with items: project1.yml, project2.yml
```

loop type.

Creating Organizations the Right Way

Pitfalls await as soon as the first module (`aws.ansible.organization`) is called up. When you start a task on AWX, the system first generates a Kubernetes

pod with the execution environment, which then performs the automation. The standard container template for this (usually `quay.io/ansible/awx-ee:latest`) includes only a few Ansible collections.

To use modules from collections such as `ansible.windows`, `containers.podman`, or `kubernetes.core` in your playbook, the execution environment first needs to load them. In its project folder, the `collections/requirements.yml` file lists the collections that the execution environment inserts directly after the start. As a rule, you can pick up freely available collections from the Galaxy website [3]. You do not need to log in to this service before using it.

Alternatively, you can find repositories for collections that require registration; a small bug in AWX raises its head here. You always need to assign at least one credential of the type *Ansible Galaxy/Automation Hub API Token* to an organization, even if you do not need it for the open Ansible Galaxy service. If you manually create a new organization in the AWX web UI, you will not notice anything because AWX automatically inserts Ansible Galaxy pseudo-credentials unless you select otherwise. The catch is that if you use the REST API to create an organization, AWX does not run in the background and the credentials

Listing 2: Retrieve Password

```
- name: Get Admin PW
  kubernetes.core.k8s_info:
    api_version: v1
    kind: Secret
    name: "{{ awxsetup.name }}-admin-password"
    namespace: "{{ awxsetup.namespace }}"
  register: awxadmin
```

Listing 3: Group Tasks in a Block

```
- name: Upload Config to AWX block:
  block:
    - name: Create Organization
      aws.ansible.organization:
    ... ((Many config tasks)) ...
  environment:
    CONTROLLER_PASSWORD: "{{ awxadminresources[0].data.password | b64decode }}"
```

remain empty, which means an execution environment running in this organization cannot reload any collections.

Unfortunately, no meaningful error messages alert you to this problem. To manage this omission, you need an entry in your playbook:

```
- name: Create Organization
  awx.awx.organization:
    name: "{{ org.name }}"
    description: "{{ org.desc }}"
    galaxy_credentials: - Ansible Galaxy
    state: present
```

Listing 4: Use Default Functions

```
- name: Create Inventory Hosts
  awx.awx.host:
    name: "{{ item.key }}"
    description: "{{ item.value.description }}"
    inventory: "{{ inventory.name }}"
    state: present
    enabled: true
    variables:
      ansible_host: "{{ item.value.ip }}"
      ansible_port: "{{ item.value.port | default('22') }}"
    with_dict: "{{ inventory_host }}"
```

The `galaxy_credentials` specification must be included in this task. In the following tasks, your playbook creates the most important basic credentials. Unfortunately, this has to be done in separate tasks one after the other, because it is difficult to combine different credential types in a credentials loop. The parameters to be transferred are just too different and depend heavily on the types. Of course, you can build credential dictionaries for similar types and combine them in a loop (e.g., for all credentials that only require a combination of a username and a password). The sample code copies your standard SSH private key from `~/.ssh/id_rsa` to the machine credentials. You will need to change this line if your AWX setup uses a different key.

Now things become relatively easy. The other tasks generate the required resources on the AWX setup one after the other. This example generates an inventory with static hosts, as is usually the case for standard infrastructure machines on the LAN. I also used the

previously mentioned trick with the default function, as you can see in [Listing 4](#).

The task reads the host information, such as the name and IP address, from the dictionary in `vars.yml`. If you do not enter a port parameter for some hosts, Ansible will simply fill out the field with 22.

As a final step, the AWX rollout then creates your job templates ([Figure 2](#)). Of course, this only works if the playbooks specified in the job templates actually exist in your project Git and were synchronized correctly when the project was created. The code in this example only contains the as-code rollout, so you will need to provide a project with templates yourself.

Even More Automation

In practice, the automation presented here takes less than five minutes to set up a fully functional AWX server. After you have customized the `vars.yml` and `secrets.yml` files, simply start the code by typing:

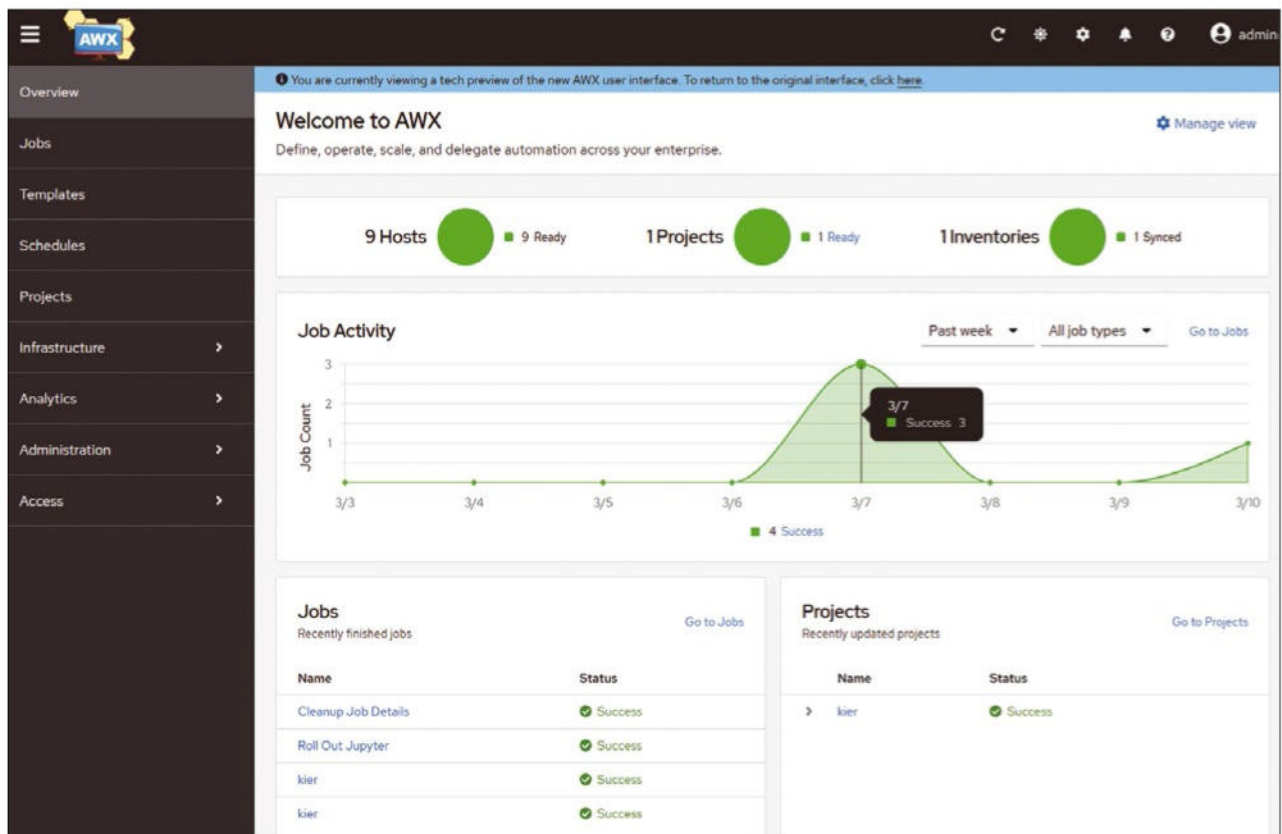


Figure 2: The playbook rolls out an AWX server on Kubernetes and uploads its complete configuration via API.

```
ansible-playbook 00_aws_full.yml 2
--ask-vault-password
```

You can use the old `ansible-playbook` command-line tool here because the more modern `ansible-navigator` tool unfortunately does not support an interactive `--ask-vault-password` scenario.

Of course, the code leaves plenty of room for expansion. For example, you could add elements to create workflow and job templates. At the beginning of the rollout, tasks can be added that include the basic AWX setup, including LDAP and directory integration.

As mentioned at the beginning, you can also model the `00_aws_full` playbook as a workflow on an existing AWX server that can then be connected to a Git repository with your as-code declaration by callback or webhook while the Ansible-via-Ansible rollout runs as a GitOps pipeline,

which can be triggered with a simple commit.

Conclusions

Structured YAML and Ansible make it easy to describe complete IT environments as code and roll them out automatically. The fully automated AWX setup is just the beginning. The as-code setups of several systems can then be linked to workflows in AWX: First the rollout of the physical or virtual systems, then the basic software, followed by the application data. The demo code definitely offers some room for improvement. The MicroShift scenario used here, including the AWX operator, can easily be rolled out under the same principle. Although the AWX tool discussed here is still available, it has not been updated since June 2024 because the project is at end of life (EOL);

however, it is presented as an “as-code skeleton,” so it can be used for other applications. ■

Info

- [1] Automated AWX rollout:
[https://github.com/astolzen/awx_auto]
- [2] Installing AWX and Helm:
[<https://ansible.readthedocs.io/projects/awx-operator-helm/helm-install-on-existing-cluster.html>]
- [3] Galaxy: [galaxy.ansible.com]

The Author

Andreas Stolzenberger worked as an IT magazine editor for 17 years. He was the deputy editor in chief of the German *Network Computing* magazine from 2000 to 2010. After that, he worked as a solution engineer at Dell and VMware. In 2012 Andreas moved to Red Hat. There, he currently works as principal solution architect in the Technical Partner Development department.



GET TO KNOW ADMIN

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

Subscribe to ADMIN
and get 6 issues
delivered every year

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



@adminmagazine



@adminmag



ADMIN magazine



@adminmagazine



Migrate your Git repositories to Gitea

Exodus

Nothing is forever, not even a Git server. After the purchase of GitHub by Microsoft, I found a new home in Gitea for version control. By Thomas Reuß

Anyone who uses Git will quickly see the true value of version management. I have set up my own Git server with all kinds of features on a Raspberry Pi with the Gogs self-hosted Git service software, but as of now, the Gogs repository has had very little activity, with releases only appearing every six months and offering very little in terms of new features. This situation led me to believe that the software will probably not survive much longer. Unfortunately, I didn't realize the unhealthy state of Gogs until I was already dealing with versions that have no easy migration path to Gitea, a fork of Gogs and the new home I chose for my code. The two projects have drifted apart since the fork. In this article, I show how to migrate your repositories to Gitea.

Git Origin

Unlike veterans such as CVS or Subversion, Git is not a client-server application, but a distributed system, which means that no node is favored or disadvantaged compared with others. Git allows each repository to act as a server, making migration easier. You just have to make sure that the working copies contain all the branches and tags of the origin.

In the Git world, the origin is a local alias of a specific remote repository. It removes the need for you to enter the full URL for every fetch, pull, or push operation. In fact, you could even rename the origin alias to, say, source by typing:

```
git remote rename origin source
```

What is far more serious than these names in the context of a server migration is that all data from the previous origin is only available locally. If the previous server is deleted, development branches could be lost.

Checking for Completeness

If you want to migrate your repositories, first make sure all of them are fully available locally. I had no choice but to make all my remote repositories available locally with the `git clone` command. Once that was done, I was able to check each repository's branches and tags. For each individual repository, all the branches and tags had to be up to date in the local version. A few Git commands and a little shell scripting made this task easier.

The following script shows how to check out all the branches of a repository:

```
for mybranch in $(git branch -a | \
  grep -o -E '(remote.*)' | \
  grep -v ' -> ' | \
  cut -f3 -d'/')
do
  git checkout "${mybranch}"
  git pull
done
```

The `git branch -a` command in the first line returns the names of all branches in the repository. For the purpose of migration, I was mainly interested in the branches that were not yet available locally, so I used `grep` to filter out the remote branches. I then removed the line that denoted my HEAD reference – that is, the currently active branch, which is marked with an arrow (`->`).

The `for` loop iterates over the branches it finds and executes the `git checkout` command, making the branch in question available locally. You just need to repeat this procedure for all tags (`git fetch --tags`).

Everything Is a File

True to my motto of “a lazy admin is a good admin” or “if you can do so, let the computer work for you,” I thought about the easiest way to handle the migration. Looking at the operating system first, for me, one of the greatest

Photo by Ray Hemessy on Unsplash

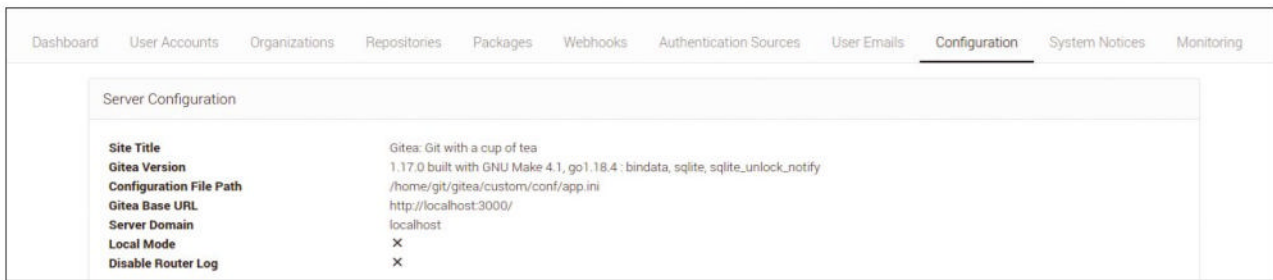


Figure 1: The most important server settings can be found under *Configuration*.

strengths of free software is that practically everything that makes up the system is available as a file. This arrangement might sound trivial at first, but it will probably become clearer when you consider what you need to do in the scope of a server move.

The migration all starts with the Git software, which I want to change; the reverse proxy – NGINX in my case, the Exim4 mail server, and various shell scripts, cron tables, and so on. With no exceptions, I store global cron tables in the `/etc/cron.d/` directory, global shell scripts in `/usr/local/bin`, and non-distribution software in `/opt/`. This structure makes it easy to create a backup of all the relevant files on this system with a one-liner:

```
$ sudo tar -vzcf /home/thomas/githorst.tar.gz \
/etc/ /opt/ /usr/local/bin/
```

The archive created in this way contains the files required to ensure the features on my old hardware exist in the same way on my future hardware. My Raspberry Pi needs an operating system release change because

a Raspbian version based on Debian 12 became available. The next step, then, is to copy the new Raspbian image to an SD card; Raspberry Pi Imager gives you a very convenient way of doing this. After booting the new system, it's time to install some software. A simple

```
sudo apt install postgres exim4 nginx
```

makes light work of this task. I then copied the required configuration files (NGINX, Exim4, and Gogs) to `/etc/`. Installing Gitea, including setting up a systemd service, was quick and painless thanks to the good documentation [1]. The service can be accessed on port 3000. Because I wanted to transfer the configuration of my existing Gogs instance to the new Gitea instance to the extent possible, I had to edit `app.ini`, the central configuration file. Once again, Vim proves to be a great all-rounder, because `vimdiff` – assuming you have some experience with Vim – makes it easy to find the differences between the former and the current configuration files and merge them as required.

Once you have set up the configuration to

suit your needs, I would recommend sending a test email to check the mail server settings. You will find the most important server settings under *Configuration | Site Administration* (Figures 1 and 2) and can send a test email by entering an email address (Figure 3).

In general, I can warmly recommend the Gitea configuration cheat sheet [2], which contains every setting and every flag that can occur in the `app.ini` file. However, I would definitely recommend adjusting the logging settings here. By default, Gitea only logs on the console, which makes very little sense in server operation. I would recommend configuring the settings for logging to files in the log section:

```
[log]
#MODE = console
MODE           = file
LEVEL          = Warn
ROOT_PATH      = /opt/gitea/log
LOG_ROTATE     = true
DAILY_ROTATE   = true
MAX_DAYS       = 7
COMPRESS       = true
COMPRESSION_LEVEL = -2
```

Merging the settings and setting up NGINX as a reverse proxy – again,

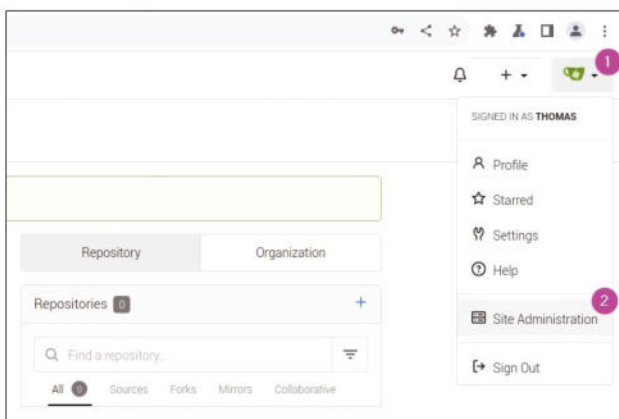


Figure 2: You can access *Site Administration* (2) from a personal menu (1).

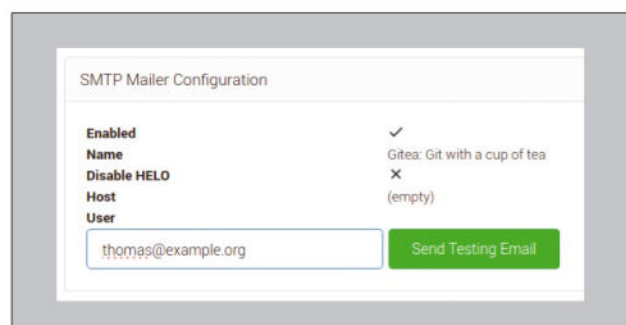


Figure 3: Test email can be sent from the SMTP Mailer Configuration dialog.

with the old configuration – gives you a running, clean, empty Gitea instance. This process was even easier than I originally thought it would be.

Now is the time for migration (i.e., physically moving the existing repositories to the new server). My Gitea server has the same fully qualified domain name (FQDN) as the former Gogs; moreover, Gitea has the same Let's Encrypt certificates, which was exactly the plan I had in mind when I backed up the original system. Later, I will look into tried and tested procedures for moving a repository to a new server. In my case, all the server settings remained the same, minus repositories. My first attempt was a naive `git push`; after all, what could go wrong? To be honest, I was not so far off the mark and received a very informative error message, as well. **Figure 4** shows the message in the first line of output, which sent me running to read the Gitea documentation, which clearly states:

In the `app.ini` file, set `ENABLE_PUSH_CREATE_USER` to `true` and `ENABLE_PUSH_CREATE_ORG` to `true` if you want to allow users to create repositories in their own user account and in organizations they are a member of respectively. Restart Gitea for the changes to take effect. You can read more about these two options in the Configuration Cheat Sheet. (Gitea push docs [3])

Voilà – exactly the information I needed.

I enabled two flags,

```
ENABLE_PUSH_CREATE_USER = true
ENABLE_PUSH_CREATE_ORG = true
```

in the `[repository]` section of the `app.ini` file. After restarting the Gitea service, I tried my luck again, and with success this time. As you can see in **Figure 5**, Gitea accepts push-and-create for both user and organization-owned repos. Given my fears that the move could turn into a major project, this outcome was really satisfying.

Classic Migration

Most moves involving Git repos are likely to differ significantly from my special case. You will usually need to move from `SERVER_1` to `SERVER_2`. You can see the specific steps in **Listing 1**. The first line clones the repository from the previous server to `<LOCAL_REPO>` and is followed by a change to the directory of the new `<LOCAL_REPO>` working copy. The `git branch -a` lists all branches and checks them out. The fetch ensures that all the tags in the repository exist in the working copy.

With the tags, all the data from the remote repository are now on the local system. The command

```
git remote rm origin
```

cuts the connection to the previous server to receive the URL of the new

server as the origin in the next `git remote` line. Finally, the content and tags are pushed to their new home in the last two lines.

Conclusions

With very little risk and manageable overhead, you can get rid of potentially outmoded software within a few hours. Although Gogs might live on and regain momentum again at some point, I don't want to rely on that possibility. I consider it important not to leave something as fundamental as version control to chance. During this move, I realized once again how sophisticated and powerful Git really is. It almost seems as if Linus Torvalds anticipated all the use cases during development. ■

Info

- [1] Gitea installation:
[\[https://docs.gitea.com/installation/install-from-binary\]](https://docs.gitea.com/installation/install-from-binary)
- [2] Gitea configuration cheat sheet:
[\[https://docs.gitea.com/administration/config-cheat-sheet\]](https://docs.gitea.com/administration/config-cheat-sheet)
- [3] Gitea push doc:
[\[https://docs.gitea.com/usage/push\]](https://docs.gitea.com/usage/push)

Author

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

Listing 1: Classic Migration

```
git clone <ORIGIN-URL> <LOCAL_REPO>
cd >LOCAL_REPO>
git branch -a
for current_branch in
$(git branch -a | grep -o -E '(remote.*)' | grep -v '
->' | cut -f3 -d'/'); do
git checkout "${current_branch}";
git pull;
done
git fetch --tags
git remote rm origin
git remote add origin <NEW_ORIGIN_URL>
git push origin --all
git push --tags
```

```
thomas@raspi02:~/git/personal/linux-magazin/2022/01_2022 $ git push
Gitea: Push to create is not enabled for organizations.
fatal: Could not read from remote repository.

Please make sure you have the correct access rights and the repository exists.
thomas@raspi02:~/git/personal/linux-magazin/2022/01_2022 $
```

Figure 4: The error message put me on the right track and pointed me toward the documentation.

```
thomas@raspi02:~/git/personal/linux-magazin/2022/06_2022 $ git push
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 8 threads
Compressing objects: 100% (27/27), done.
Writing objects: 100% (29/29), 342.10 KiB | 14.87 MiB/s, done.
Total 29 (delta 12), reused 8 (delta 2), pack-reused 0
remote: . Processing 1 references
remote: Processed 1 references in total
To githorst.reuss.ocks:Linux_Magazin/11_2022.git
 * [new branch]      main -> main
thomas@raspi02:~/git/personal/linux-magazin/2022/06_2022 $
```

Figure 5: After I adjusted the configuration, pushing – which also creates the new branch – worked as desired.

Key-value stores: an alternative to relational databases

Data Silos

Relational databases have been long-lived, but a completely different type of database, the key-value store, has established itself in the cloud market. By Martin Loschwitz

When it comes to structured data storage, admins have viewed relational databases such as MySQL or PostgreSQL to be the ideal solution for decades. No other tool can store large volumes of data in such an efficient way, and virtually no other language can access databases as well as the powerful and versatile SQL. In the meantime, relational databases have become so popular that the top players in this genre have virtually

become generic terms for databases themselves, to which the LAMP stack (e.g., Linux, Apache, PHP, MySQL) bears witness to this day. However, just as LAMP has become less important over the years, the hype surrounding relational databases has died down, at least a little, for understandable reasons: The complexity of a relational database is by no means helpful or even necessary where data needs to be stored in a

structured way, which quickly becomes clear when you take a closer look at where databases are used today.

Cloud computing environments have been very popular for many years, which explains why Kubernetes is ubiquitous: The idea of making all the functions of a state-of-the-art data center above the hardware level controllable by API, and of supporting automation by doing so, is too

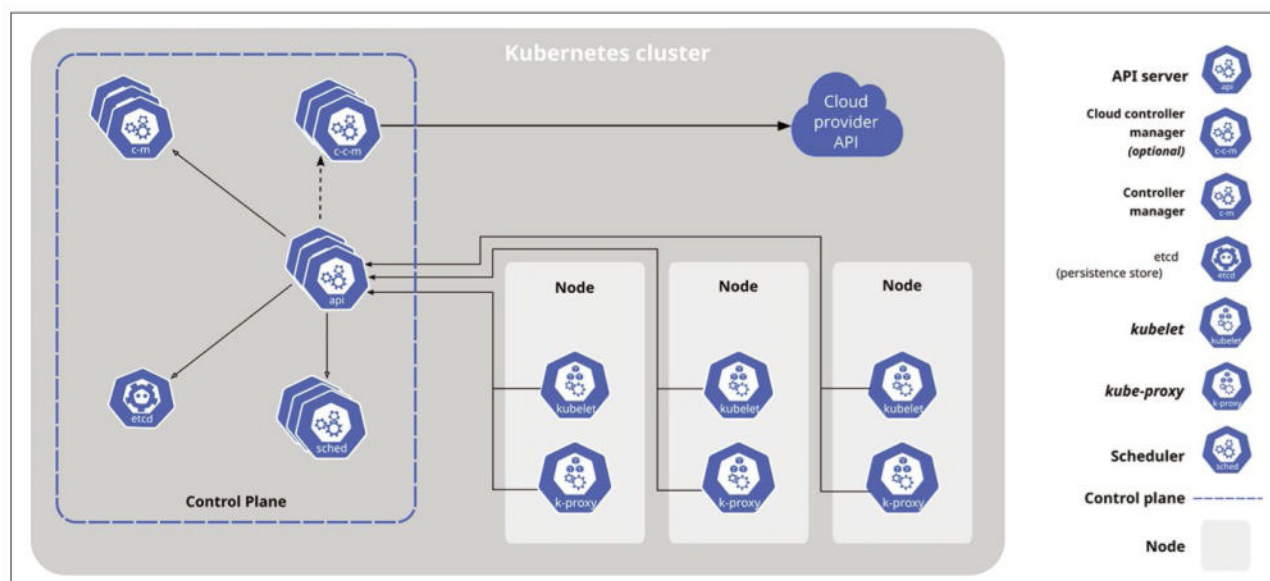


Figure 1: Unlike OpenStack, Kubernetes opted for etcd as a key-value store for its persistent configuration data. © Kubernetes/CNCF

tempting from the perspective of many corporations. On request, Kubernetes will give you persistent storage along with dynamic IP addresses in the context of a software-defined networking (SDN) environment. OpenStack [1], a project with similar goals and comparable functionality in part, opted for the legacy approach years ago, and it still relies on a relational database (usually MySQL) to store its persistent metadata. Kubernetes went down a different path from the outset. The metadata is stored in a database, as well, but it is not relational. Instead, etcd [2] is a mandatory component of state-of-the-art Kubernetes installations (Figure 1). Admins also refer to the product as a consensus algorithm. Etcd establishes a consensus for the specific values of individual configuration parameters across all nodes of a Kubernetes installation. Unlike OpenStack, Kubernetes does not build large and complex tables with innumerable entries (e.g., for storage or IP addresses it has created). Instead, Kubernetes follows the approach of simply storing a single entry in etcd for each storage entity and its ID. Kubernetes components wanting to manipulate the data stored there iterate over the list of existing entries and choose the entry they need.

Unneeded Guarantees

The Kubernetes developers impressively demonstrate what many a passionate administrator has long been aware of: Just because you have persistent data storage does not force you to install a relational database with all its guarantees. MySQL and the like, with their atomicity, consistency, isolation, and durability (ACID) guarantees, are undoubtedly a blessing when used in a setup that handles complex data structures, but if the

sole reason for being is to store and retrieve simple key-value pairs, relational databases can become stumbling blocks, because no matter what content it contains, relational databases must implement the promised ACID guarantees, even for the most trivial datasets.

If you are looking at simple data structures, you will often discover that meeting these ACID guarantees massively prolongs the database query execution time. Also, don't forget distributed databases, in which several database instances have to communicate with each other over a network. The effect is even more noticeable; after all, the consistency guarantees not only need to be fulfilled for one host, but for all instances involved in the replication setup.

Comparatively, the vast majority of today's key-value stores were practically born with network capability, which turns out to be a massive benefit in today's distributed setups, because it is an implicit part of the solution instead of a retrofitted need, as is the case with solutions such as Vitess (a MySQL-compatible cloud database). As a result, key-value stores often easily outperform relational databases when handling

trivial datasets, without being significantly less reliable in terms of data consistency. Etcd leads the way in this category, which explains why it is no longer used only for Kubernetes metadata in Kubernetes clusters.

Consul Delivers

To avoid misunderstandings arising at this point, note that key-value stores should not be seen as a universal replacement for relational databases or as a drop-in replacement to enable massive performance gains in any conceivable constellation. Before deciding on a key-value store as the backbone for storing persistent data, you therefore need to check carefully whether the simple key-value storage interface is suitable for your application or whether you need more complex data structures that cannot be implemented by a key-value store. I have experienced many strange examples of developers not thinking things through. For example, a piece of software that stores data in a key-value store initially, but then uses JSON or other constructs that need to be read and evaluated to find the data is not necessarily wrong, but the constructs developed in this way

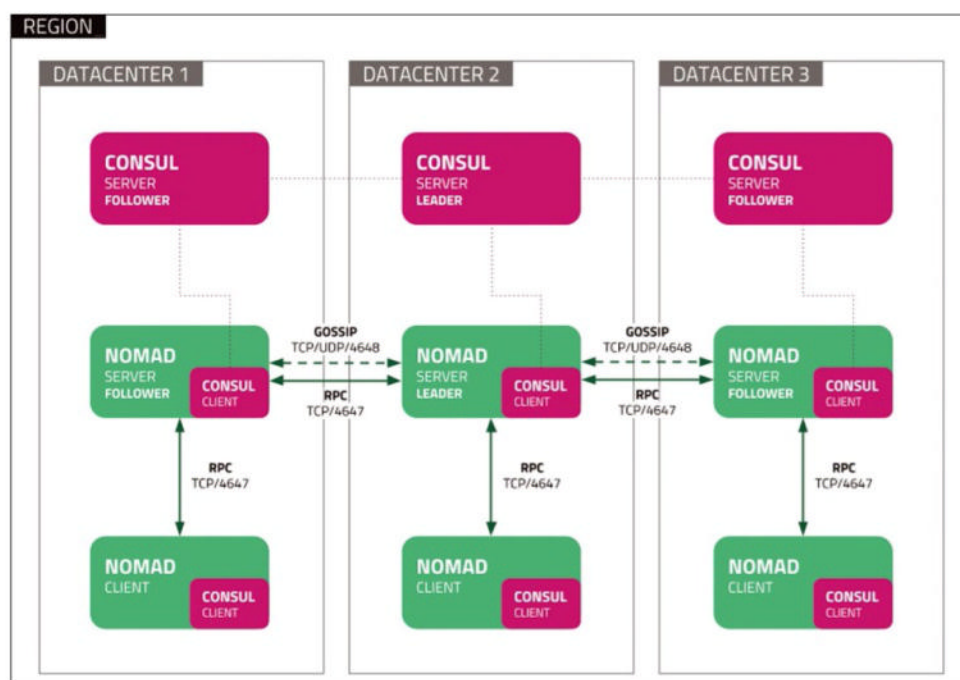


Figure 2: Consul is a key-value store by HashiCorp that integrates a consensus algorithm and acts as a data store in the Kubernetes alternative, Nomad, among other use cases. © HashiCorp/IBM

will often turn out to be slower than a finely tuned relational database, while adding far more complexity at the code level. If you go for this kind of approach, you might not have understood the purpose and benefits of key-value stores. Etcd and the like can only play to their real strengths when the data to be processed is available in the form of keys and values. Another solution, and one that has always been something like etcd's natural archenemy, demonstrates this problem vividly.

Consul [3] (Figure 2), from HashiCorp, a corporation that recently became part of IBM, was designed to complement other tools (e.g., Nomad). This container orchestration tool targets small setups in which admins want to avoid the extra work that Kubernetes involves. However, it quickly becomes clear where this journey is heading: Like etcd, which drives virtually every Kubernetes cluster, Consul is a consensus-finding algorithm for individual key-value pairs in distributed environments.

Just like etcd, Consul is distributed, running on every node of the entire installation, supporting local queries for key-value pairs, and ensuring that the local content of the database matches the content of the databases on all other nodes. Again, like etcd, you find defined ways and means of creating or changing key-value pairs throughout the cluster. At least in part, key-value stores mimic the consistency guarantees of classic databases, although fundamentally different tools are used. Etcd, for example, implements its own consensus algorithm, which has certain similarities with Paxos or Raft. If this reminds you of Pacemaker [4], you're right: Paxos is used for consensus finding there, as well, and every Pacemaker instance is an extremely rudimentary key-value store without a direct interface to the outside world.

Outside the Mainstream

Although Consul and etcd are undoubtedly the most popular players

in the key-value store field, they are by no means the only ones. Apache Zookeeper [5], for example, is also a consensus algorithm for distributed systems with a simple key-value store under the hood. Unlike etcd and Consul, Zookeeper is specifically designed to act as a configuration database in distributed environments. Although the kings of the hill, etcd and Consul, are typically used in the same way today, unlike Zookeeper, they can also be used as key-value stores in cases other than service recognition and service metadata storage.

High-tech Silicon Valley corporations have long recognized the benefits of key-value storage. Many people think of Meta (formerly the Facebook company) exclusively as the operator of the social network Facebook, but Meta has operated for some time as a tech giant that develops and provides products – and not just for use on its own social platform. In fact, Meta is behind a number of products that can be found in other projects.

RocksDB [6] is an excellent example. The company almost modestly describes RocksDB as persistent key-value storage for use in flash and RAM. In practice, though, RocksDB is now regarded as a kind of one-size-fits-all application for high-performance, persistent data storage. Besides following a few basic rules, the RocksDB developers significantly tuned their product for data store applications. RocksDB was inspired by Google's LevelDB [7], a key-value store capable of running exclusively in RAM.

RocksDB itself is a significant advancement of LevelDB. It does not come as a separate service in the sense of a daemon, but as a library that can be integrated into applications. This arrangement improves performance in itself by completely eliminating what can often be a resource-hungry detour through the TCP/IP protocol and the entire network stack when querying databases.

Moreover, RocksDB is designed to store its entire database permanently in RAM. Although modern flash drives and drives with an NVMe

interface are not nearly as slow as their rotating counterparts from the past, RAM is still significantly faster today. By storing its own data in RAM, RocksDB fully utilizes the speed benefits of RAM compared with NVMe drives and other flash devices and passes these benefits on to the user. What is quite remarkable is that RocksDB's internal structure is quite simple, because RocksDB can also be used as a plain vanilla store for key-value pairs.

RocksDB Instead of a Filesystem

RADOS uses RocksDB instead of a filesystem, which generates some unexpected performance advantages in surprising places in everyday life, as is demonstrated by the Ceph object store solution (although Ceph is typically cited as a negative example, especially in terms of latency). RocksDB plays a central role in Ceph clusters. However, when Ceph inventor Sage Weil put the first ideas about Ceph and the underlying object storage RADOS down on paper as part of his dissertation, he knew that Ceph would have to access block drives to store data persistently. Like any other service that accesses block storage devices, Ceph's object storage daemon (OSD) services rely on disks to provide organized and structured access. The classic approach to solving this problem has always been to use a POSIX-compatible filesystem. Much like relational databases, POSIX offers numerous guarantees related to concurrent file access, file consistency, and various other factors. Weil initially wanted to use Btrfs [8] for Ceph, but because Btrfs appeared to be incomplete for a long time – and is still incomplete today – he simply made do with XFS. XFS is undoubtedly a tried-and-tested, rock-solid filesystem for everyday data center use that fully complies with all POSIX rules for filesystems. From the Ceph developers' point of view, however, XFS soon turned from a blessing into a curse. At the end of

the day, the RADOS object store at the heart of Ceph does not need many of the guarantees that a POSIX-compatible filesystem has to provide. For example, if a user overwrites a binary object in RADOS, RADOS does not replace the existing object with the new data in the background. Instead, it creates a new object and changes the pointer in its internal metadata that points from the name of the object to the specific data in the filesystem. As part of its caretaking activities, RADOS deletes the orphaned object autonomously and without user intervention.

The problem is even more obvious when it comes simply to accessing binary objects that are located somewhere on an OSD on RADOS peripherals. Because XFS offers POSIX guarantees, it has to perform various tests when accessing a file (e.g., to prevent uncoordinated concurrent access to a file). Of course, a concurrent access scenario has no way to occur in RADOS because of its internal implementation.

Nevertheless, complying with POSIX costs a lot of valuable time whenever a RADOS OSD is accessed. Dissatisfaction soon grew in the developer community, ultimately prompting a radical break. Without further ado, Weil developed a kind of minimal trunk filesystem named BlueFS for RADOS OSDs. Anyone who has ever heard the term “BlueStore” in the context of RADOS can find more information online [9].

BlueFS is a simple beast under the hood: An individual OSD does not need much more information than the physical address on a data carrier, where RADOS can find a specific object to implement access. The combination of the object name and the storage address is, as you can probably guess, a typical key-value pair that can be stored perfectly in a key-value database. Weil opted for RocksDB; therefore, BlueFS’s only real tasks are to provide the functionality needed to help RocksDB store data and provide a write-ahead log for the OSD in question (Figure 3).

BlueStore still works exactly along

this principle today. It has not only superseded the conventional method, which was subsequently renamed FileStore, but it also delivers substantial speed advantages in a direct comparison (Figure 3), not least because with the new on-disk format with BlueFS and RocksDB, RADOS now offers significantly lower latency in high-performance environments than was the case just a few years ago.

Evaluating All Options

Speaking of RADOS, if you think about the function of the object store, you will quickly realize that RADOS is basically a key-value store. If you do not use RADOS through one of the three front ends (i.e., if you do without the RADOS Gateway, RADOS block device (RBD), and CephFS), you can create arbitrary information in RADOS directly at the programming level with a specific name that is unique throughout the cluster. Admittedly, this is a very special way of using a key-value store, but it’s not total nonsense: In its role as a key-value store, RADOS opens opportunities that legacy databases, and especially relational databases, cannot address.

Generations of database administrators have racked their brains over how to store typical asset data such as images or other binary data in relational databases, mainly to avoid the embarrassment of having to provide an application with an interface for accessing a POSIX-compatible filesystem, for example, in addition to the database interface. Solutions of this

kind were never far removed from hacks, and they go against the way in which the creators of MySQL and the like want to see their databases used. Today’s database diversity puts an end to these makeshift solutions (e.g., in the form of RADOS).

Blurred Dividing Lines

To my mind, it would be unfair not to mention in an article on key-value stores that the boundaries between the two approaches started to blur some while back in various projects. Yugabyte [10] is a perfect example. In essence, it is a classic key-value store at first glance. The product has all the advantages that a key-value store generally offers: fast queries for trivial data records, a consensus mechanism available out of the box for operation in distributed infrastructures, and, as a result, practically unlimited horizontal scalability. These features, in turn, make it possible to tweak performance, especially with regard to read operations – and reads still make up the majority of database queries in most setups today.

However, Yugabyte can do far more: The developers have taught it to speak the PostgreSQL dialect through a compatibility interface (Figure 4). In other words, a translator component can be used to access a Yugabyte instance in the same way as a PostgreSQL instance. The Yugabyte authors admit that their PostgreSQL interface does not implement all commands in 100 percent the same way as the original, but for the vast majority of applications, the general

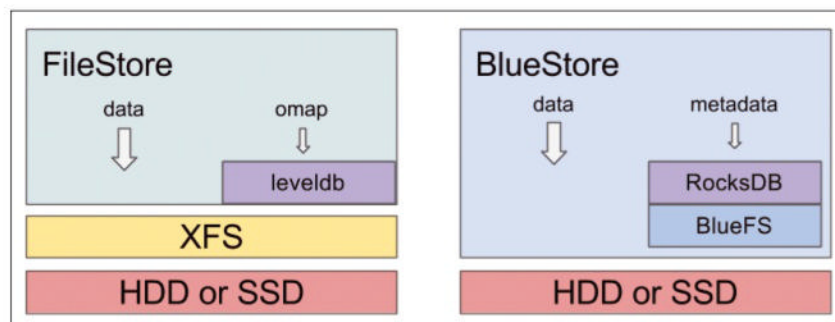


Figure 3: Because Red Hat did not need the POSIX guarantees of XFS, they replaced it in Ceph with a paired-down filesystem and the RocksDB key-value database, generating considerable speed gains. © Red Hat/IBM

message is that Yugabyte's PostgreSQL compatibility is totally fit for purpose.

Running a web application developed for PostgreSQL will not normally pose too tough a challenge for Yugabyte and its SQL interface. Unlike classic PostgreSQL, Yugabyte promises genuine scalability on top with ACID-like consistency guarantees, even for geo-redundant setups (Figure 5).

The main beneficiaries are – as you would expect – microarchitecture applications that also scale horizontally. In a complicated mesh construct in a Kubernetes environment, for example,

where you can easily call on Helm to roll out Yugabyte, the software lets you implement distributed databases with a MySQL interface. If you want to go one better, you can also deploy a solution such as Istio [11] to achieve comprehensive load balancing.

Thanks to its PostgreSQL compatibility, Yugabyte is also establishing itself as an alternative to solutions designed to help relational databases scale, such as Vitess [12] for MySQL. That said, the technical differences between the traditional scalability add-ons for MySQL or PostgreSQL and Yugabyte could hardly be more

apparent. For example, Vitess adds sharding to MySQL but keeps the database in the background, which leads to a number of compromises and significant performance effects compared with legacy MySQL. The Yugabyte developers, on the other hand, state that their goal is keeping pace with PostgreSQL. The product is said to be capable of processing up to one million write requests if you run it on hardware that is powerful enough for the job.

Conclusions

Key-value stores have long since arrived in everyday IT as a valid alternative to relational databases. In terms of scalability and speed, they are superior right from the start. Now, they are also getting close to outplaying their conventional predecessors in classic use cases for relational database management systems, thanks to the detour of storing keys and the values associated with them.

Undoubtedly, special fields of application will remain that need legacy MySQL or PostgreSQL, but it is equally certain that solutions such as Yugabyte, with its simple key-value store, will make inroads into the

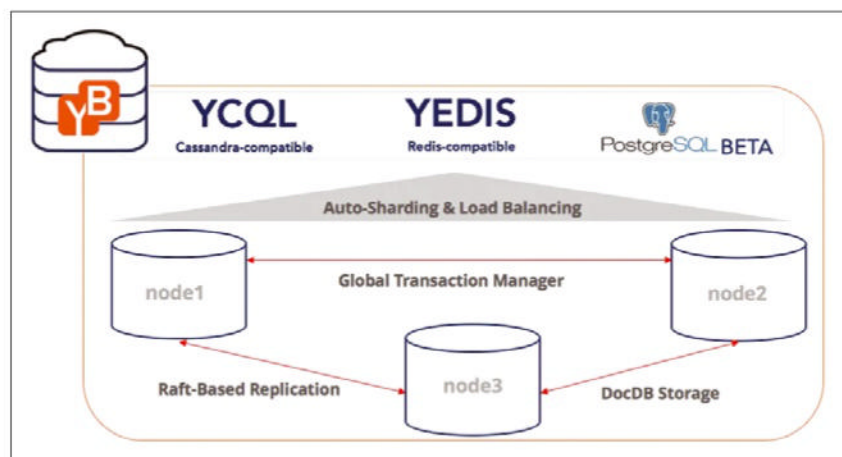


Figure 4: Under the hood, Yugabyte is a key-value store based on the Raft protocol, but it also exposes a PostgreSQL interface to the outside world. © Yugabyte

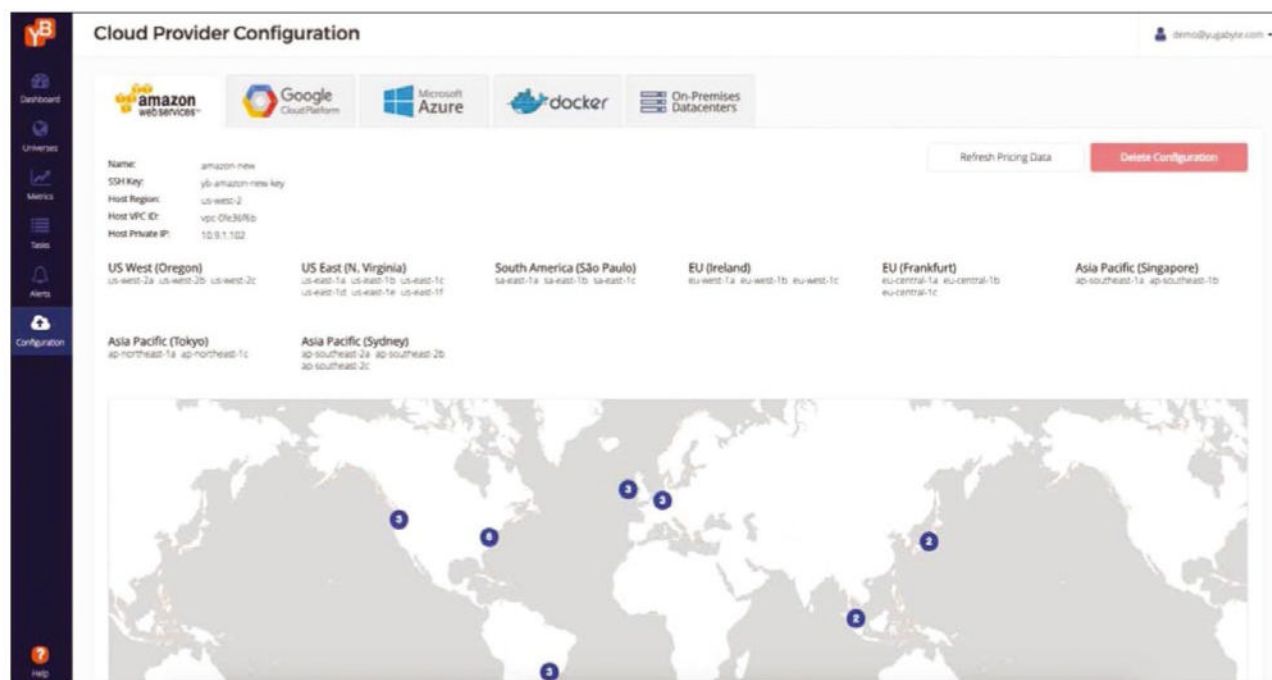


Figure 5: Because Yugabyte does without complex data structures, even cross-site replication is far easier to accomplish with the tool than with genuine PostgreSQL. © Yugabyte

established relational database market segment in the future. Ultimately, both developers and administrators can only benefit: Lower administrative overheads, improved scalability, and faster performance are just some of the compelling arguments. For application developers in particular, though, the rise of key-value memory also means more development effort. Key-value stores are not the answer for every imaginable use case. If you set your mind on a key-value store at too early a phase of development, you might end up with problems that you have to iron out with crude hacks at the programming level further down the line. If you are looking to store data persistently in your application, you first need to take a very close look at the structure of the data to be stored. If the data can be meaningfully represented in a structure

comprising a key and an assigned value, you have no reason not to use a key-value store.

However, if you need a more complex structure, you are probably better off sticking with a relational database, because MySQL and the like provide classic ACID guarantees, whereas key-value stores only implement consensus algorithms. Reduced to the level of individual transactions, key-value stores simply do not offer the same consistency guarantees as ACID-compatible databases. That said, ACID is not necessary in all key-value store use cases. As always, good planning is half the battle when it comes to data management.

Info

- [1] OpenStack: [\[https://www.openstack.org\]](https://www.openstack.org)
- [2] etcd: [\[https://etcd.io\]](https://etcd.io)
- [3] Consul: [\[https://www.consul.io\]](https://www.consul.io)

- [4] Pacemaker: [\[https://clusterlabs.org\]](https://clusterlabs.org)
- [5] Apache Zookeeper: [\[https://zookeeper.apache.org\]](https://zookeeper.apache.org)
- [6] RocksDB: [\[https://rocksdb.org\]](https://rocksdb.org)
- [7] LevelDB on GitHub: [\[https://github.com/google/leveldb\]](https://github.com/google/leveldb)
- [8] Btrfs: [\[https://docs.kernel.org/filesystems/btrfs.html\]](https://docs.kernel.org/filesystems/btrfs.html)
- [9] BlueFS: [\[https://www.ibm.com/docs/en/storage-ceph/6?topic=bluestore-ceph-bluefs\]](https://www.ibm.com/docs/en/storage-ceph/6?topic=bluestore-ceph-bluefs)
- [10] Yugabyte: [\[https://www.yugabyte.com\]](https://www.yugabyte.com)
- [11] Istio: [\[https://istio.io\]](https://istio.io)
- [12] Vitess for MySQL: [\[https://vitess.io/docs/archive/12.0/reference/compatibility/mysql-compatibility/\]](https://vitess.io/docs/archive/12.0/reference/compatibility/mysql-compatibility/)

The Author

Martin Loschwitz is the founder and managing director of True West IT Services GmbH, which offers scalable IT infrastructure based on OpenStack and Kubernetes.



Linux Magazine Subscription

Print and digital options
12 issues per year

► SUBSCRIBE
shop.linuxnewmedia.com

Expand your Linux skills:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- News on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

Go farther and do more with Linux, subscribe today and never miss another issue!



Need more Linux?
Subscribe free to Linux Update

Our free Linux Update newsletter delivers insightful articles and tech tips to your inbox every week.

<https://bit.ly/Linux-Update>





Pushing Raspberry Pi storage to its limit

Rocket Fuel

We test microSD, USB, and (especially) PCIe storage to push Raspberry Pi storage to the max. By Federico Lucifredi

The **Raspberry Pi 5** added a new port for a flexible printed cable (FPC) exposing PCI Express (PCIe) for the first time in the history of the beloved single-board computer (SBC; **Figure 1**). Eclipsed by the convenience of the equally new and faster USB 3.0 ports that now support simultaneous 5Gbps operation and a much improved high-speed microSD card interface, you can understand the Foundation's lack of urgency in delivering an expansion board to add PCIe storage to the lineup. However, the wait is finally over, and you can now assess the full set of new storage options available to Pi users.

M.2 Expansion

With new silicon, last year the Pi 5 delivered close to three times the CPU and GPU performance and twice the memory and I/O bandwidth of its predecessor **[1]**. The two MIPI (mobile industry processor interface) connectors now support both camera serial (CSI)-2 and display serial

interfaces (DSI), making the dedicated screen connector redundant. By retiring the older display port and reusing the freed space, the Pi 5 now sports a small FPC ribbon connector exposing a single-lane PCIe 2.0 bus. NVMe Express (NVMe) **[2]** drives connect to the PCIe bus via an M.2 **[3]** adapter – in this case, to the new M.2 HAT+ released by Raspberry Pi this spring **[4]** (**Figure 2**). The HAT+ standard format enables automatic detection of the board as well as the device connected through the single M Key edge interface it provides. M2 form factors are described by their dimensions, with the M.2 HAT+ supporting 2230 or 2242 modules 30mm and 42mm in length, respectively. The board can supply up to 3A of power and includes LED blinkenlights for signaling power and activity. The M.2 HAT+ cost \$10 **[5]** at the time of writing and includes

all the hardware needed for installation, but not the solid state drive (SSD) itself. I opted for a cost-effective Inland TN446 3D TLC flash unit nominally capable of read speeds up to 4,900MBps (and up to 3,700MBps write speed), with total capacity of 512GB **[6]** (about \$70). The unit is rated at a staggering 400K read and 900K write I/O operations per second (IOPS), respectively. Note how the throughput prevails in read, whereas the IOPS are twice as high in write: The drive (and the kernel) can cache writes, but cold reads have to come from the device.

Its quite a bit of storage for an SBC, but I have future plans at the lab that include testing AI models that will take plenty of space and I/O.

Installing the unit requires swapping

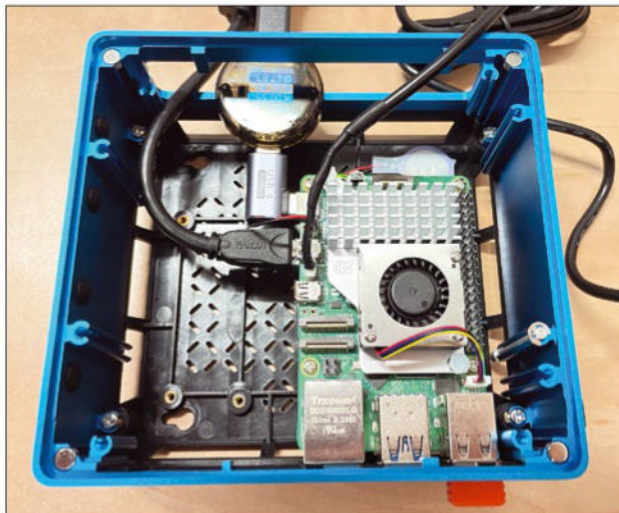


Figure 1: Raspberry Pi 5 lab setup at the Dragon Propulsion Lab – a 3W idle power baseline includes wireless keyboard and power meter.



Figure 2: The standoffs and GPIO extension included with the M.2 HAT+ permit installing clear of the CPU fan.



Figure 3: Installing the M.2 HAT+ and NVMe drive altered the idle power draw a hardly noticeable 0.2W.

standoffs because of the stacked configuration I use, but I preserved the designer's choice of plastic standoffs in case they were part of the electrical design (Figure 3). As you would expect, I am adding a heat sink [7] to the SSD; overheating it just would not do. The heat sink neatly wraps around the unit with a metal cage padded with gel. Perhaps the hardest part of installing it was the size mismatch; the sink is designed to hold longer 2242 modules (Figure 4).

Mass Storage

The HAT+ mostly performed as expected, and the Raspberry Pi OS detected it transparently. I did not experience the ultimate magic (an automatic desktop mount) because my drive vendor did not ship a valid partition (Figure 5), and I had to initialize the disk myself first. The Gnome-Disks [8] command is a simple and intuitive strategy to that end (Figure 6). Once that was taken care of, I was able to round up power observations. Running eight disk write stressors (`stress -i 4-d 4`) [9] within a present working directory located on the NVMe drive increased load to

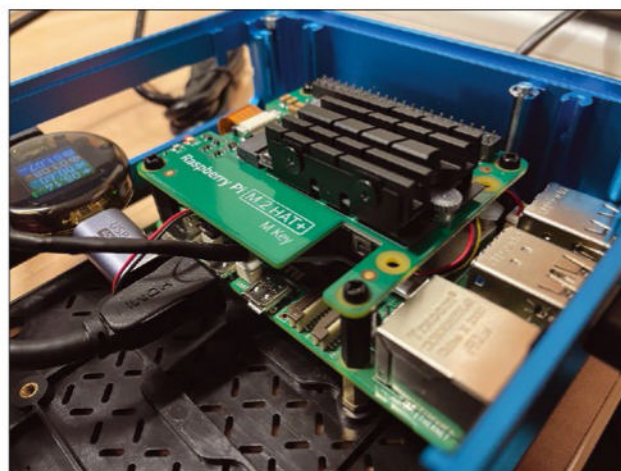


Figure 4: Adding an SSD heat sink (\$6) should really be a no-brainer.

an approximate 7W (Figure 7).

One pitfall to watch for is that the Pi's firmware should not be older than December 6, 2023.

If it is, use `raspi-config` [10] to choose the latest bootloader, followed by

```
sudo rpi-eeprom-update -a
```

to upgrade (Figure 8). The `raspi-config` tool is also the place to head if you intend to use the PCIe device as the boot device (Figure 9).

Before I delve into performance figures, it is incumbent on me to point out that the PCIe bus exposed by the M.2 adapter can support much more

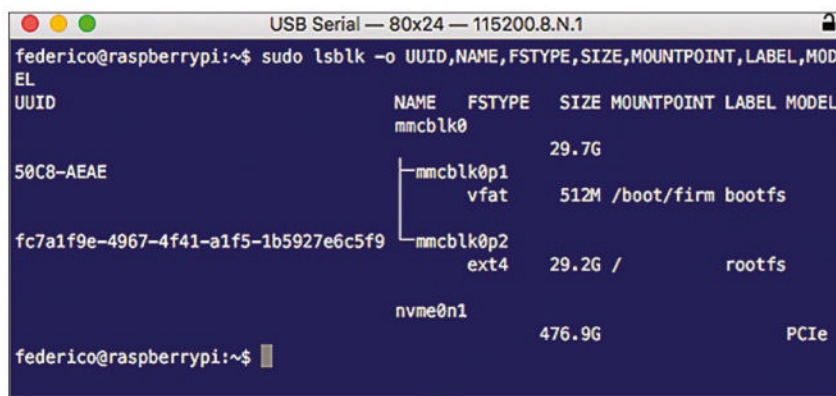


Figure 5: Detected but not configured. This problem was the drive vendor's fault in my case because the disk shipped without a partition.

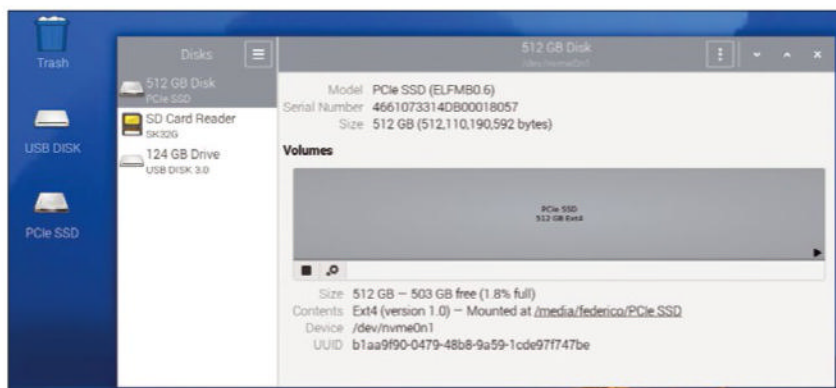


Figure 6: The absent automatic desktop mount was easily resolved by adding a partition and mountpoint in Gnome Disks.

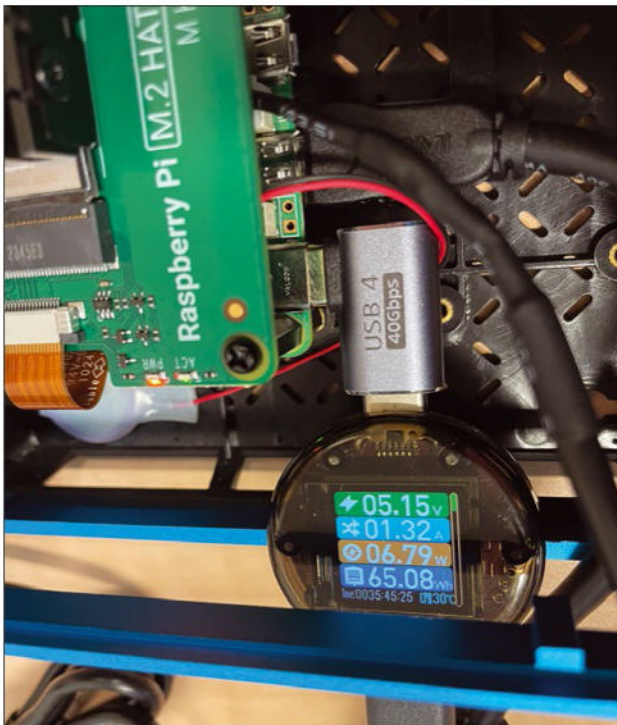


Figure 7: An average extra watt at load, with spikes at 1.5W, is the observed power load.

than just mass storage devices, as long as the kernel drivers to do so are in place. Jeff Geerling maintains what's closest to an official list on his site, ranging from GPU and AI accelerators to RAID controllers and sound cards [11].

Benchmarks

I ran the same benchmarks (see “The Individual Tests” box) I used for the Inland NVMe flash unit with

a generic Micro Center 128GB USB 3 stick, as well as the 32GB SanDisk Ultra PLUS SD card booting the system (V10 A1 Class 10 HC I). The differences could not be more pronounced. As expected, the NVMe drive easily exceeded the sequential write throughput of the SD card (20.7MBps) and USB drive (81.1MBps), posting a staggering 314MBps result (Figure 10). Read throughput figures are even higher, with the NVMe device attaining up to 438.8MBps in my tests (Figure 11).

The more illustrative random I/O test performed in IOzone was not able to achieve the full drive specification but posted a remarkable 225K write IOPS and 77K read IOPS – on a Raspberry Pi (Figure 12)! The NVMe device is posting completion of the writes as soon as the data is received but before it's written to its final NAND location, which explains the higher performance until the single-level cell (SLC) write cache is overrun, after which point performance drops significantly. NVMe access

```

USB Serial — 80x24 — 115200.8.N.1

raspberrypi login:
Last login: Sun Oct 20 20:13:21 EDT 2024 on tty1
federico@raspberrypi:~$ sudo rpi-eeprom-update
*** UPDATE AVAILABLE ***
BOOTLOADER: update available
CURRENT: Fri Feb 16 03:28:41 PM UTC 2024 (1708097321)
LATEST: Mon Sep 23 01:02:56 PM UTC 2024 (1727096576)
RELEASE: default (/lib/firmware/raspberrypi/bootloader-2712/default)
Use raspi-config to change the release.
federico@raspberrypi:~$

```

Figure 8: Checking that firmware support is current and is supporting the M.2 HAT+.

```

USB Serial — 80x24 — 115200.8.N.1

Raspberry Pi Software Configuration Tool (raspi-config)

A1 Expand Filesystem    Ensures that all of the SD card is available
A2 Network Interface Names Enable/disable predictable network i/f names
A3 Network Proxy Settings Configure network proxy settings
A4 Boot Order           Choose SD, network, USB or NVMe device boot pri
A5 Bootloader Version   Select latest or factory default bootloader sof
A6 Wayland              Switch between X and Wayland backends
A7 Audio Config         Set audio control system
A8 PCIe Speed           Set PCIe x1 port speed

<Select>                <Back>

```

Figure 9: Option 4 of `raspi-config` also allows adding the PCIe storage device to the boot sequence.

The Individual Tests

Sequential reads is a measurement of raw throughput without help from the Linux caches, which are active but not primed. Run by the `hdparm` [12] tool,

```
sudo hdparm -t /dev/mmcblk0
```

the remarkably consistent results with modern SD cards are easily exceeded by USB and NVMe drives.

Sequential writes, run by the `dd` [13] tool,

```

sudo dd if=/dev/zero of=/home/pi/test 2
bs=8k count=50k conv=fsync; 2
sudo rm -f /home/pi/test

```

is a large continuous write, forcing filesystem sync to ignore caching effects.

Random reads and random writes, run by the `iozone` [14] benchmark,

```

iozone -e -I -a -s 100M -r 4k 2
-i 0 -i 1 -i 2 [-f <path/to/file>]

```

are 4K random I/O operations at the heart of a filesystem in general compute use. The random write portion was historically problematic for low-end SD cards.


```

USB Serial — 80x24 — 115200.8.N.1
federico@hippolita:~$ sudo dd if=/dev/zero of=/home/federico/test bs=8k count=50k conv=fsync; sudo rm -f /home/federico/test
51200+0 records in
51200+0 records out
419430400 bytes (419 MB, 400 MiB) copied, 20.2586 s, 20.7 MB/s
federico@hippolita:~$ sudo dd if=/dev/zero of=/media/federico/USB\ DISK/test bs=8k count=50k conv=fsync; sudo rm -f /media/federico/USB\ DISK/test
51200+0 records in
51200+0 records out
419430400 bytes (419 MB, 400 MiB) copied, 5.17295 s, 81.1 MB/s
federico@hippolita:~$ sudo dd if=/dev/zero of=/media/federico/PCIe\ SSD/test bs=8k count=50k conv=fsync; sudo rm -f /media/federico/PCIe\ SSD/test
51200+0 records in
51200+0 records out
419430400 bytes (419 MB, 400 MiB) copied, 1.3376 s, 314 MB/s
federico@hippolita:~$

```

Figure 10: Write throughput benchmark for SD card, USB drive, and NVMe storage, respectively.

```

USB Serial — 80x24 — 115200.8.N.1
federico@hippolita:~$ sudo hdparm -t /dev/nvme0n1

/dev/nvme0n1:
Timing buffered disk reads: 1318 MB in 3.00 seconds = 438.87 MB/sec
federico@hippolita:~$ sudo hdparm -t /dev/mmcblk0p2

/dev/mmcblk0p2:
Timing buffered disk reads: 258 MB in 3.02 seconds = 85.42 MB/sec
federico@hippolita:~$ sudo hdparm -t /dev/sda1

/dev/sda1:
Timing buffered disk reads: 376 MB in 3.00 seconds = 125.17 MB/sec
federico@hippolita:~$

```

Figure 11: Read throughput benchmark for NVMe storage, SD card, and USB drive, respectively.

latency averages around 0.06ms with a significant tail around 0.15ms. (See Figure 13 for a performance comparison with USB 3 storage.)

The introduction of an SDR104 SD drive has essentially doubled the performance of SD cards meeting the ultrahigh speed (UHS)-1 spec when comparing Raspberry Pis 4 and 5; throughput rose to 85.42MBps compared with 43.53MBps [15] in my tests with SanDisk Ultra Plus cards. Swapping OS images quickly and painlessly is one of the benefits of the SD drive, but using USB for the same purpose adds performance but also reliability. USB's peak throughput has risen from 363 to 420MBps, but the bigger gains come from the new support for dual 5Gbps operations: If the two USB 3.0 ports are in simultaneous use, aggregated combined bandwidth will reach 842MBps [16]. ■

Info

- [1] "Introducing: Raspberry Pi 5!" by Eben Upton, *Raspberry Pi News*, September 28, 2023, [<https://www.raspberrypi.com/news/introducing-raspberry-pi-5/>]
- [2] NVMe Express: [https://en.wikipedia.org/wiki/NVMe_Express]

```

USB Serial — 166x44 — 115200.8.N.1
root@hippolita:~$ fiozone -e -I -a -s 100M -r 4k -i 0 -i 1 -i 2 -f /home/federico/test | tail -12
Record Size 4 kB
Command line used: fiozone -e -I -a -s 100M -r 4k -i 0 -i 1 -i 2 -f /home/federico/test
Output is in kBytes/sec
Time Resolution = 0.000001 seconds.
Processor cache size set to 1024 kBytes.
Processor cache line size set to 32 bytes.
File stride size set to 17 * record size.

      kB  reclen  write  rewrite  read  reread  random  random  bkwd  record  stride  fwrite  frewrite  fread  freread
      102400    4    4319    4425    21243    21032    16535    4060

fiozone test complete.
root@hippolita:~$ fiozone -e -I -a -s 100M -r 4k -i 0 -i 1 -i 2 -f /media/federico/PCIe\ SSD/test | tail -n 12
Record Size 4 kB
Command line used: fiozone -e -I -a -s 100M -r 4k -i 0 -i 1 -i 2 -f /media/federico/PCIe SSD/test
Output is in kBytes/sec
Time Resolution = 0.000001 seconds.
Processor cache size set to 1024 kBytes.
Processor cache line size set to 32 bytes.
File stride size set to 17 * record size.

      kB  reclen  write  rewrite  read  reread  random  random  bkwd  record  stride  fwrite  frewrite  fread  freread
      102400    4   185234   226257   155114   155381   77439   225341

fiozone test complete.
root@hippolita:~$ fiozone -e -I -a -s 100M -r 4k -i 0 -i 1 -i 2 -f /media/federico/USB\ DISK/test | tail -n 12
Record Size 4 kB
Command line used: fiozone -e -I -a -s 100M -r 4k -i 0 -i 1 -i 2 -f /media/federico/USB DISK/test
Output is in kBytes/sec
Time Resolution = 0.000001 seconds.
Processor cache size set to 1024 kBytes.
Processor cache line size set to 32 bytes.
File stride size set to 17 * record size.

      kB  reclen  write  rewrite  read  reread  random  random  bkwd  record  stride  fwrite  frewrite  fread  freread
      102400    4    1251    1291    28184    28056    10319    839

fiozone test complete.
root@hippolita:~$

```

Figure 12: Read and write random IOPS for SD card, NVMe storage, and USB drive, respectively.

- [3] M.2, formerly the Next Generation Form Factor (NGFF): [\[https://en.wikipedia.org/wiki/M.2\]](https://en.wikipedia.org/wiki/M.2)
- [4] Raspberry Pi M.2 HAT+ product brief: [\[https://datasheets.raspberrypi.com/m2-hat-plus/raspberry-pi-m2-hat-plus-product-brief.pdf\]](https://datasheets.raspberrypi.com/m2-hat-plus/raspberry-pi-m2-hat-plus-product-brief.pdf)
- [5] Raspberry Pi M.2 HAT+, Adafruit.com: [\[https://www.adafruit.com/product/5902?src=raspberrypi\]](https://www.adafruit.com/product/5902?src=raspberrypi)
- [6] Inland TN446 NVMe M.2 Gen 4x4 SSD: [\[https://www.microcenter.com/product/663767/inland-tn446-512gb-3d-tlc-nand-pcie-gen-4-x4-nvme-m2-2230-internal-ssd-compatible-with-steam-deck\]](https://www.microcenter.com/product/663767/inland-tn446-512gb-3d-tlc-nand-pcie-gen-4-x4-nvme-m2-2230-internal-ssd-compatible-with-steam-deck)
- [7] Heat sink for Raspberry Pi M.2 HAT+, The Pi Hut: [\[https://thepihut.com/products/heatsink-for-raspberry-pi-m2-hat\]](https://thepihut.com/products/heatsink-for-raspberry-pi-m2-hat)
- [8] Gnome Disks: [\[https://manpages.ubuntu.com/manpages/noble/en/man1/gnome-disks.1.html\]](https://manpages.ubuntu.com/manpages/noble/en/man1/gnome-disks.1.html)
- [9] Amos Waterland's stress: [\[https://manpages.ubuntu.com/manpages/noble/man1/stress.1.html\]](https://manpages.ubuntu.com/manpages/noble/man1/stress.1.html)
- [10] raspi-config(1) manual: [\[https://www.raspberrypi.com/documentation/computers/configuration.html\]](https://www.raspberrypi.com/documentation/computers/configuration.html)
- [11] Raspberry Pi PCIe Database project by Jeff Geerling: [\[https://pipci.jeffgeerling.com\]](https://pipci.jeffgeerling.com)
- [12] hdparm(8) man page: [\[https://manpages.ubuntu.com/manpages/noble/en/man8/hdparm.8.html\]](https://manpages.ubuntu.com/manpages/noble/en/man8/hdparm.8.html)
- [13] dd(1) man page: [\[https://manpages.ubuntu.com/manpages/noble/man1/dd.1.html\]](https://manpages.ubuntu.com/manpages/noble/man1/dd.1.html)
- [14] iotzone(1) man page: [\[https://manpages.ubuntu.com/manpages/nobles/en/man1/iotzone.1.html\]](https://manpages.ubuntu.com/manpages/nobles/en/man1/iotzone.1.html)
- [15] "Finding the fastest SD cards for the Raspberry Pi" by Federico Lucifredi, *ADMIN*, issue 76, 2023, [\[https://www.admin-magazine.com/Archive/2023/76/Finding-the-fastest-SD-cards-for-the-Raspberry-Pi\]](https://www.admin-magazine.com/Archive/2023/76/Finding-the-fastest-SD-cards-for-the-Raspberry-Pi)
- [16] "Raspberry Pi 5 review - Hands-On with the Most Powerful Raspberry Pi Yet" by Gareth Halfacree, *hackster.io* HW101, September 2023, [\[https://www.hackster.io/news/raspberry-pi-5-review-hands-on-with-the-most-powerful-raspberry-pi-yet-57efaf61b10f\]](https://www.hackster.io/news/raspberry-pi-5-review-hands-on-with-the-most-powerful-raspberry-pi-yet-57efaf61b10f)

The Author

Federico Lucifredi (@0xf2) is the Product Management Director for Ceph Storage at IBM and Red Hat, formerly the Ubuntu Server Product Manager at Canonical and the Linux "Systems Management Czar" at SUSE. He enjoys arcane hardware issues and shell-scripting mysteries, and takes his McFlurry shaken, not stirred. You can read more from him in the new O'Reilly title *AWS System Administration*.

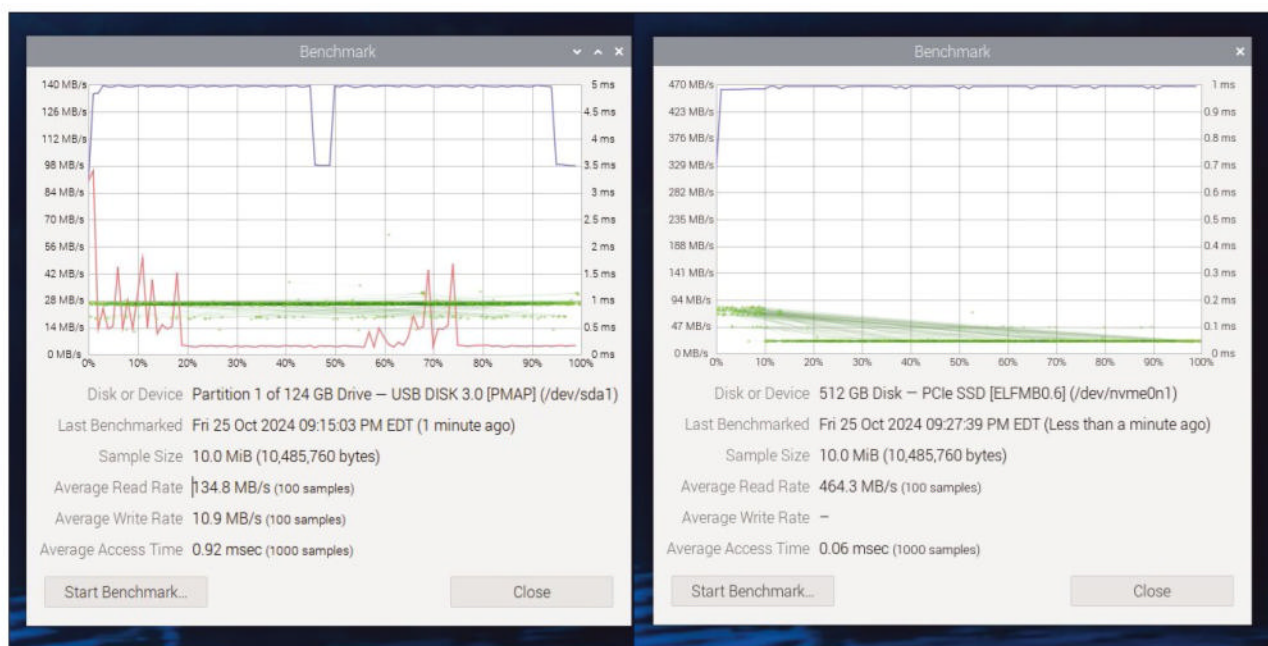


Figure 13: USB (left) and NVMe (right) gnome-disks benchmark comparing latency clouds.



NEWSSTAND

Order online:
<https://bit.ly/ADMIN-library>

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.

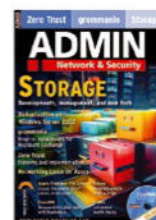


#83 - September/October 2024

Storage

Improved management, suitable drivers, standardized protocol structures, and advancements in future-proof hardware and software lead to more durable, more manageable, and easier-to-repair storage.

On the DVD: TrueNAS SCALE 24.04 "Dragonfish"



#82 - July/August 2024

Sovereign Cloud Stack

SCS liberates your data centers from monopolistic operations and companies beholden to out-country laws and regulations.

On the DVD: Kali Linux 2024.2

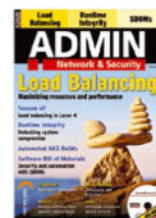


#81 - May/June 2024

Load Balancing

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

On the DVD: SystemRescue 11.01



#80 - March/April 2024

Threat Management

Digital infrastructures are vulnerable to all kinds of attacks. You need strategies and tools to detect and defend.

On the DVD: openSUSE Leap 15.5

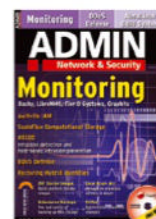


#79 - January/February 2024

Monitoring

This issue takes a deep dive into monitoring solutions for your IT infrastructure, including Dashy, LibreNMS, Tier 0 systems, and Graphite.

On the DVD: FreeBSD 14.0



#78 - November/December 2023

Domain-Driven Design

Business experts and developers collaborate to define domain models and business patterns that guide software development.

On the DVD: Fedora Server 39



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
Jens-Christoph Brendel	26
Markus Feilner	36
Otto Geissler	10
Ken Hess	3
Michael Höller	14
Thomas Joos	58
Daniel LaSalle	42
Rubén Llorente	50
Martin Gerhard Loschwitz	30, 85
Federico Lucifredi	91
David Myriel	20
Thomas Reuß	82
Andreas Stolzenberger	76
Wolfgang Weith	64
Matthias Wübbeling	70, 72

Contact Info

Editor in Chief

Joe Casad, jcasad@linuxnewmedia.com

Managing Editors

Rita L Sooby, rsooby@linuxnewmedia.com
Lori White, lwhite@linuxnewmedia.com

Senior Editor

Ken Hess

Localization & Translation

Ian Travis

News Editor

Amber Ankerholz

Copy Editors

Amy Pettie, Aubrey Vaughn

Layout

Dena Friesen, Lori White

Cover Design

Lori White, Illustration based on graphics by pollywa, 123RF.com

Advertising

Jessica Pryor, jpryor@linuxnewmedia.com

Publisher

Brian Osborn

Marketing Communications

Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription

For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2024 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by Kolibri Druck.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN (Print ISSN: 2045-0702, Online ISSN: 2831-9583, USPS No: 347-931) is published bimonthly by Linux New Media USA, LLC, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. November/December 2024. Application to Mail at Periodicals Postage Prices is pending at Philadelphia, PA and additional mailing offices. POSTMASTER: send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Coming

Soon

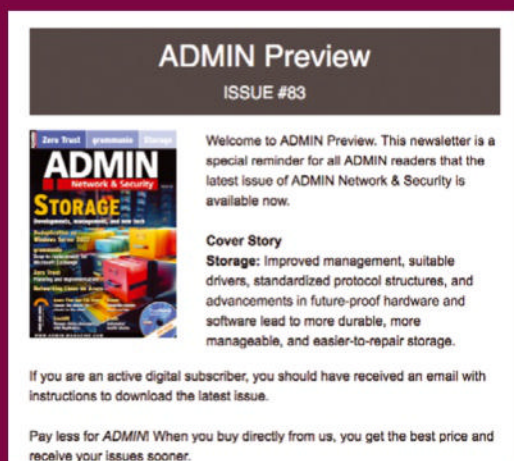
ADMIN 85

Our next issue will be packed with all the great content you expect from *ADMIN*. Here are a few of the upcoming articles:

- Bloonix Monitoring
- Kubeflow ML Toolkit
- Automating Microsoft Dev Box
- System Rescue Distro Compared
- Talos Minimal Distro for Kubernetes
- And much more!

Please note: Articles could change before the next issue.

Available Starting Febuary 7



BE THE FIRST TO SEE WHAT'S NEXT

Subscribe free to the *ADMIN* Preview newsletter and get a sneak peek at every article included in the next issue of *ADMIN*.



Sign up today at <https://bit.ly/admin-preview>

Image © hobbitfoot, 123RF.com



KICKSTART EUROPE

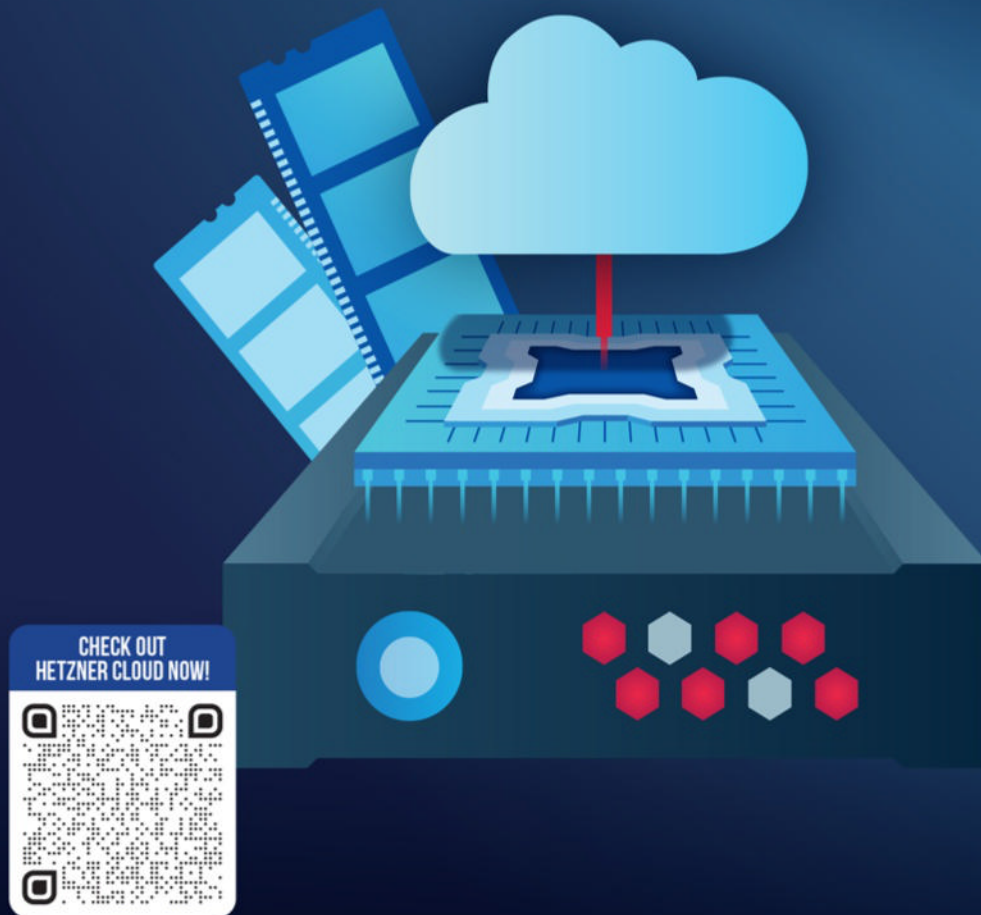
2025

GET YOUR TICKETS NOW

4TH & 5TH FEBRUARY 2025 - AMSTERDAM RAI - WWW.KICKSTARTCONF.EU

HETZNER

PURE POWER AT IMPOSSIBLY LOW PRICES



CLOUD SERVER CPX11

- ✓ 2 x **shared** vCPU
AMD EPYC™ 7002
- ✓ 2 GB RAM
- ✓ 40 GB NVMe SSD
- ✓ Location Germany, Finland, USA & Singapore
- ✓ Incl. IPv4
- ✓ Hourly billing
- ✓ No minimum contract

monthly from: **\$4.49**

CLOUD SERVER CCX13

- ✓ 2x **dedicated** vCPU
AMD Milan EPYC™ 7003 | Genoa EPYC™ 9654
- ✓ 8 GB RAM
- ✓ 80 GB NVMe SSD
- ✓ Location Germany, Finland, USA & Singapore
- ✓ Incl. IPv4
- ✓ Hourly billing
- ✓ No minimum contract

monthly from: **\$13.49**

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

cloud.hetzner.com