ADMIN Network & Security

Sovereign Cloud Stack | Copilot | RaMP

ADMIN Network & Security

ISSUE 82

Sovereign Cloud Stack

Digital freedom, both foreign and domestic

Emulating and Mocking AWS and GCP Services

Rapid Modernization Plan Securing Active Directory

Legal or Litigation Holds and eDiscovery in Microsoft Teams

Copilot: Control or block GPT-4-based AI chat

Go Testing Frameworks

LINUX NEW MEDIA The Pulse of Open Source

Bacula
Flexible backup for large environments

AIX Automation
IBM AIX collections from Ansible Galaxy

Topgrade
Update and upgrade your Linux apps

Spacelift
IaC management platform

WWW.ADMIN-MAGAZINE.COM

Premium-class business companion

TUXEDO InfinityBook Pro 15 - Gen9

AMD or
Intel CPU

Matte-black
or silver-grey
aluminum chassis

Mobility

Up to 8 TB SSD and 96 GB RAM
2x PCIe 4.0 SSD | 2x DDR5-5600-RAM

Battery life

Linux compatible

Up to 5 Years Guarantee

Immediately ready for use

TUXEDO

tuxe.do/lxadmin82

Made in Germany

German Data Privacy

German Tech Support

The Best Laid Plans

The old saying that no one plans to fail, but many fail to plan is true. However, in the complex and ever-evolving field of IT, sometimes all the planning doesn't guarantee success.

I live by a few basic laws, one of which is, "Everything works on paper." Sometimes, all the IT governance and change control that an Information Technology Infrastructure Library- (ITIL) and agile-trained project manager offers isn't enough to deal with all the contingencies of system administration. The number of variables is too great. Sometimes, the number of dependencies is overwhelming to the point of boggling the most spotless of minds. Sometimes people make mistakes. Still, we march forward with the zeal and determination of a true warrior, never admitting defeat, googling until our fingers are sore, reading forums until our eyes bleed, and arguing our points in a few subreddits for good measure.

In the early days of the pre-Google Internet, I was updating systems one starry evening, hopped up on Diet Dr. Pepper and pepperoni pizza, when I got an error I couldn't get past. I tried several things to bypass the fatal error preventing my progress. Something was missing – a dependency, a permission, a setting, or something I couldn't pinpoint. I even resorted to searching Yahoo for an answer. Sadly, none came. The body of knowledge that exists now didn't exist at the time, and I was stuck. I had to back out all my changes and email my manager that tonight's maintenance had failed. I knew that the next day wasn't going to be a pleasant one for me.

In those days, it was common for managers and senior staff members to call out junior members as incompetent or, worse, to label them "clueless newbies." I took the instructions home to study them carefully again to see where I could have made a mistake that disrupted my entire evening of patching bliss. I found no such error.

I braced myself as I walked onto the floor and headed to my cubicle, hoping not to see anyone until I arrived safely. I made it, but the phone rang to summon my presence almost as soon as I logged into my workstation. My manager obviously received messages when his staff members logged in. Can you say "micromanagement?"

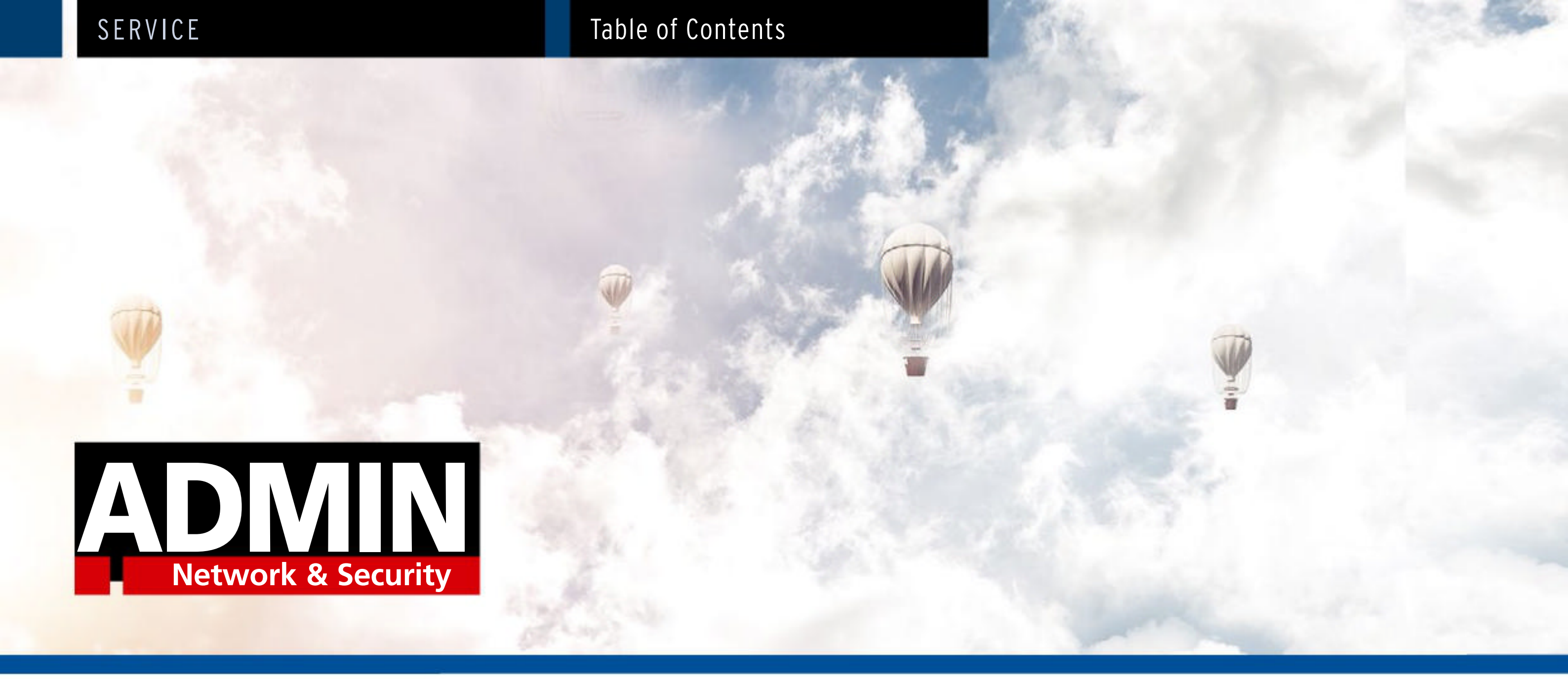
"So, how'd it go last night?" came the rhetorical query that forced me to admit my failure. "Not good," I answered. "I had a problem here in the instruction." I highlighted the failure point on the sheet of paper where I'd scribbled a few notes to document what I'd done in response to a couple of system messages that I didn't recognize and didn't have time to Yahoo!. My manager was already picking up the phone to summon over a senior member of the team. Chris was the senior guy and everyone's guru. You went to him when you needed tools, software, hardware, or the vast occult and ungoogleable knowledge of the Windows operating system. "Hey, Chris, can you come to my desk for a minute?" He appeared quickly to begin his "I know more than you do" assault on my already damaged psyche.

I reluctantly handed him my instruction sheet while he sat down to study what I'd done. He said, "Good notes." I never lifted my gaze from the floor and said, "Thanks." He continued his ponder. After a few minutes, he looked at me and then my manager and stood up, saying, "There's a step missing in the instructions I gave you. I'm sorry. I skipped the step where you should've changed the address of a memory block."

You see, in those days, with Windows 3.11, we had to alter "upper" memory constantly and do a lot of remapping to get everything to work. We used the Quarterdeck Expanded Memory Manager (QEMM) `OPTIMIZE` utility, but that was only a start. Because of our many drivers, you often had to go in and remap specific areas to maximize the other 384KB of available RAM, above the base 640KB worth. He forgot to mention the remap and address space I should have used. To my surprise, he admitted it – freely and without coercion.

The point of this story is that even if you follow the instructions, abide by every rule, and rely on your knowledge and experience, sometimes things still go wrong. Don't beat yourself up for it. Don't allow anyone else to beat you up for it. There's always another maintenance window awaiting you.

Ken Hess • Senior *ADMIN* Editor


ADMIN
Network & Security

@adminmagazine

@adminmag

ADMIN magazine

@adminmagazine

News for Admins

Tech News

MySQL 9.0 Released

The MySQL team has announced open source MySQL 9.0 (Innovation Release), along with the first update of the 8.4 LTS (8.4.1): https://dev.mysql.com/doc/relnotes/mysql/8.4/en/news-8-4-1.html.

MySQL 9.0 represents a major release with several important updates. For example, this release removes the `mysql_native_password` authentication plugin. This plugin was previously deprecated but now has been removed altogether.

According to the 9.0 release notes (https://dev.mysql.com/doc/relnotes/mysql/9.0/en/news-9-0-0.html#mysqld-9-0-0-deprecation-removal): “The mysql_native_password authentication plugin, deprecated in MySQL 8.0, has been removed, and the server now rejects mysql_native authentication requests from older client programs which do not have CLIENT_PLUGIN_AUTH capability. For backward compatibility, mysql_native_password remains available on the client; the client-side built-in authentication plugin has been converted into a dynamically loadable plugin.”

A recent blog post (https://blogs.oracle.com/mysql/post/mysql-90-its-time-to-abandon-the-weak-authentication-method) from MySQL Community Manager Frederic Descamps explains further, noting that the `mysql_native_password` is considered weak compared to modern authentication methods because it:

- Uses the SHA-1 hashing algorithm, which is vulnerable to certain types of cryptographic attacks.
- Does not use salting when hashing passwords.
- Does not use multiple iterations of the hash function, which makes it faster to compute and therefore easier to brute force.

Learn more in the MySQL 9.0 release notes: https://dev.mysql.com/doc/relnotes/mysql/9.0/en/.

NordVPN Launches File Checker Tool

NordVPN has launched File Checker (https://www.globenewswire.com/Tracker?data=el9wEU_rBQaOVA4YBDAz-MUm-twUCIKS38iW1CpZ6mv6Q1bUSqGzedAdTWU-x1-tZIWRzZXNED4HqODkq5OnH-_YOFvTipPocixmD6EL9hdO=), a new online tool for scanning files for malware and viruses.

According to the announcement, “File Checker helps prevent malicious codes from invading user’s devices through infected or corrupted files downloaded online.”

With File Checker, users can easily download the file without opening it and scan it for malware.

Read more at NordVPN: https://nordvpn.com/file-checker/.

Critical OpenSSH Vulnerability Affects Linux Systems

Researchers at the Qualys Threat Research Unit (TRU) have found a critical security flaw in OpenSSH’s server in glibc-based Linux systems.

The “regreSSHion” vulnerability (CVE-2024-6387 – https://www.cve.org/CVERecord?id=CVE-2024-6387), is “a signal handler race condition in OpenSSH’s server (sshd),” which allows unauthenticated remote code execution (RCE) as root on glibc-based Linux systems, says Bharat Jogi in a Qualys TRU blog post. “This race condition affects sshd in its default configuration.”

OpenSSH (https://www.openssh.com/) is “a suite of secure networking utilities based on the Secure Shell (SSH) protocol, which is vital for secure communication over unsecured networks,” Jogi explains. “OpenSSH versions earlier than 4.4p1 are vulnerable to this signal handler race condition unless they are patched for CVE-2006-5051 and CVE-2008-4109.”

Read more at Qualys: https://blog.qualys.com/vulnerabilities-threat-research/2024/07/01/regresshion-remote-unauthenticated-code-execution-vulnerability-in-openssh-server.

Lead Image © vlastas, 123RF.com

IT Pros See Shrinking Job-Related Benefits Despite Salary Increases

The demand for IT expertise remains strong, leading to a general increase in base compensation levels, according to the 2024 IT Salary Survey Report from ITPro Today (https://www.itprotoday.com/career-management/itpro-today-2024-it-salary-survey-report). However, decreases in job-related benefits have led to less overall job satisfaction.

The median total compensation for IT professionals this year was $130,000, compared with $125,000 in 2023. Unfortunately, a breakdown of salaries by gender shows that the median salary for women was only $123,329 this year. "This persistent disparity echoes trends observed in previous years," the report notes.

Despite increases in base pay, non-salary benefits (such as healthcare insurance, training and certification reimbursements, stock options) did not see similar growth:

- Only 68% of respondents reported receiving health insurance, down 9% from 2023.
- 56% of respondents reported receiving a 401(k) savings match, a decrease of 10%.
- 3% of respondents reported access to day care or day-care subsidies at their organizations, down from 4% last year.

Only 39% of survey respondents said they were "satisfied" with their total compensation, which is down from 50% last year. "Although base compensation may have increased, it barely offset the 3.4% inflation rate in 2023. Moreover, with decreasing non-compensation benefits like health insurance, IT professionals may need to budget their base salary for these services," the report states.

Average reported salaries by job title are:

- CIO/CTO: $313,581
- CEO/president/owner: $246,478
- IT director: $150,861
- Engineer/systems engineer: $148,819
- Software/web developer: $144,810
- IT consultant/SI/VAR: $138,855
- IT manager: $131,170
- Systems analyst/systems admin: $101,171
- IT staff: $92,006
- Technical staff $87,978
- IT operations: $86,486

Read more at ITPro Today: https://www.itprotoday.com/career-management/itpro-today-2024-it-salary-survey-report.

Top Trends Driving Observability Adoption

Nearly half (44%) of IT/telco respondents said it takes at least 30 minutes to detect high-impact outages, according to a recent report from New Relic (https://newrelic.com/resources/report/state-of-observability-it-telco), and 23% said it takes at least an hour.

Moreover, 60% of respondents said it takes at least 30 minutes to resolve these outages, and 35% said it takes at least an hour.

According to New Relic, observability solutions can help. Specifically, survey respondents said that observability helps:

- Improve collaboration (55%)
- Increase productivity (48%)
- Improve system uptime and reliability (40%)
- Mitigate service disruptions and business risk (37%)

Additionally, the top technology strategies or trends driving observability adoption are:

- Focus on security, governance, risk, and compliance (50%)
- Development of cloud-native application architectures (48%)
- Adoption of AI technologies (43%)
- Migration to a multicloud environment (40%)
- Adoption of open source technologies (36%)

Read more at New Relic: https://newrelic.com/resources/report/state-of-observability-it-telco.

Containers Dominate in Both Development and Production, According to Docker Report

Docker has released its 2024 State of Application Development Report (https://www.docker.com/blog/docker-2024-state-of-application-development-report/), providing an overview of current application development, along with insights into various DevOps trends.

General findings in the report include:

- Nearly 80% use containers in application development.
- 51% work on microservices-based applications.
- 36% of respondents said their main development environment is remote, pointing to the growing popularity of developing software in the cloud.

When asked where they primarily use containers, respondents cited:

- Development (76%)
- Deployment and testing (65% each)
- Production (61%)

The survey also asked about security-related tasks, security tools, and which roles or teams focus on software security at their organization.

"Fixing vulnerabilities is clearly a key responsibility that is shouldered by many different individuals within an organization; it was the top security task reported by back-end, front-end, and fullstack developers, DevOps and platform engineers, and most security-focused roles (e.g., Security Manager, DevSecOps)," the report states. Specific security-related tasks include:

- Fixing vulnerabilities common security tasks (49%)
- Running security scans (34%)
- Dealing with the scan results (32%)
- Monitoring security incidents (30%)
- Logging data analysis (26%)

Regarding security tools they use, respondents cited:

- SonarQube (24%)
- AWS Security Hub (20%)
- Snyk (18%)
- JFrog (15%)
- Docker Scout (14%)

Read the complete report at Docker: https://www.docker.com/blog/docker-2024-state-of-application-development-report/.

Ubuntu Core 24 Released for Edge and IoT

The Ubuntu team has released Ubuntu Core 24, built on Ubuntu 24.04 LTS (Noble Numbat): https://releases.ubuntu.com/24.04/.

Ubuntu Core 24 (https://ubuntu.com/core), which is especially designed and engineered for IoT and embedded systems, offers 12 years of long-term support (LTS).

Ubuntu Core 24 also includes:

- Improved GPU integration
- Device management with Landscape
- Edge and cloud integration
- New ROS integration for robotics developers
- New documentation and documentation structure
- Raspberry Pi 5 support

According to the team, the "strictly-confined OS enables developers to build production-grade images for embedded devices on various architectures."

Learn more at Ubuntu: https://ubuntu.com/core/docs/uc24.

Yocto Project Releases 5.0 LTS Version

The Yocto Project (https://www.yoctoproject.org/) has released version 5.0 LTS (scarthgap) as a major release (https://docs.yoctoproject.org/next/migration-guides/release-notes-5.0.html), which will be maintained with bug fixes and security updates for four years.

The open source Yocto Project, which is aimed at helping developers create custom Linux-based systems, provides tools to create tailored Linux images for embedded and IoT devices.

According to the announcement (https://www.yoctoproject.org/blog/2024/05/31/latest-long-term-support-release-new-platinum-member-boeing-and-developer-day-2024/), the 5.0 "mega-release" includes Linux kernel 6.6, gcc 13.2, glibc 2.39, and LLVM 18.1. It also includes more than "300 recipe upgrades and improvements to a variety of critical areas including core workflow, security, testing, Toaster web UI, packaging, and the roll-out of a new plug-in for VS Code among other available IDEs."

OpenSSF Introduces Siren Security Platform

The Open Source Security Foundation (OpenSSF) has announced Siren, "a collaborative effort to aggregate and disseminate threat intelligence specific to open source projects."

According to the announcement (https://openssf.org/blog/2024/05/20/enhancing-open-source-security-introducing-siren-by-openssf/), the Siren intelligence sharing list "provides a secure and transparent environment" for keeping the open source community informed of threats and activities.

Key features of OpenSSF Siren include:

- Open source threat intelligence: Info about actively exploited public vulnerabilities and threats is shared with the community.
- Real-time updates: List members receive notifications via email about emerging threats.
- TLP:CLEAR: The list follows the Traffic Light Protocol (TLP) – https://www.first.org/tlp/, with clear guidelines for the sharing and handling of intelligence.
- Community-driven: Leverages community knowledge and expertise to foster a culture of shared responsibility and collective defense.

Learn how to sign up (https://lists.openssf-vuln.org/g/siren) and get involved at OpenSSF (https://openssf.org/).

Raspberry Pi Announces Intent to Go Public

Raspberry Pi has announced its intent to become a publicly traded company known as Raspberry Pi Limited.

The London Stock Exchange filing (https://www.londonstockexchange.com/news-article/market-news/expected-intention-to-float/16470316) states that "Raspberry Pi has a strong track record of revenue growth and profitability. For the year ended 31 December 2023, revenues were $265.8 million, with gross profit of $66.0 million."

According to Raspberry Pi CEO Eben Upton, "Raspberry Pi enthusiasts will see the next phase of our development offer unprecedented opportunities for creativity and innovation. Our commitment to low-cost computing, a fundamental part of what is special about Raspberry Pi, is unchanged."

Red Hat Introduces Image Mode for RHEL

Red Hat is now offering Red Hat Enterprise Linux (RHEL) as a container image (https://thenewstack.io/red-hat-enterprise-linux-8-1-released-with-live-kernel-patching/). According to the website, this new deployment method reduces complexity and "harnesses the power of containers to bring all aspects of IT management into a single workflow."

Additionally, image mode for RHEL (https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux/image-mode) means:

- Red Hat Enterprise Linux users can benefit from greater simplification and portability across all of their environments that span the hybrid cloud.
- DevOps teams can more easily plug RHEL into their CI/CD and GitOps workflows.
- Security teams can apply container security tools, from scanning and validation to cryptography and attestation to the base elements of the operating system.
- Solution providers can more easily build, test, and distribute RHEL-based applications.

This technical blog post (https://www.redhat.com/en/blog/image-mode-red-hat-enterprise-linux-quick-start-guide) by Ben Breard explains the concepts behind image mode as well as "foundational concepts required to package operating systems in Open Container Initiative (OCI) container images."

Sovereign Cloud Stack – a genuine alternative for Europe

Freedom Fighter

The Sovereign Cloud Stack promises no less than liberation from the shackles of vendor tie-in. Cloud users who rely on this technology are free to choose their provider and switch, without further investment in training or financial penalty. By Kurt Garloff

The European Union (EU) is extremely dependent on platforms from the US in particular when it comes to basic technologies. Recent reports indicate that the EU has lagged in terms of digitalization [1], although it "has narrowed the gap with the United States in adopting advanced digital technologies (EU 69% vs. US 71%)," despite disparities within the EU [2].

The EU and member governments have launched major funding programs in the field of basic digital hardware (chip technology), which is an ambitious project that will probably take more than a decade and require tens of billions of euros in investment. Dependencies are no better in other areas. The platforms for state-of-the-art, automated provisioning of virtual IT environments with cloud and container technology, which has enabled an enormous boost in quality and productivity in the development and provisioning of software solutions, are in the hands of a few large high-tech companies in the US.

If Europeans want to have the freedom to set and enforce their rules without being cut off from digital progress, they need technology that is under their control. Otherwise, the game will probably continue, with the EU Commission ignoring the rules in a business-friendly manner and Max Schrems [3] having to remind the European Court of Justice (CJEU) of the applicable law.

Here, too, the remedy is a lengthy process because an entire ecosystem is built on platforms on which Europe has become dependent. However, one piece of good news is that the basic technology is available as open source software and is in active use in many different (albeit fragmented) applications. This technology can be expanded into a competitive open platform without investing billions.

Germany is putting theory into practice with the Sovereign Cloud Stack (SCS) project, born at the end of 2019 as an extension to the then new Gaia-X initiative. After a successfully completed evaluation contract from the German Federal Agency for Disruptive Innovation (SPRIND), the idea was financed by the Federal Ministry for Economic Affairs and Climate Protection (BMWK) with funding in the low double-digit million range as an Open Source Business Alliance (OSBA) project. The project is managed by a small project team at OSBA and executed by means of development contract awards. As a technical manager on the project team, I describe the project from an insider's perspective.

Digitally Sovereign?

When SCS chose its name, digital sovereignty was not part of many marketing departments' vocabulary. It should probably be considered a success that everyone is now claiming their own digital sovereignty. At the same time, however, the term is being watered down and softened beyond recognition, just like "green" and "open" in previous years. The "Sovereign Cloud Stack" trademark registered by the OSBA in 2021 would possibly no longer be capable of registration today.

Sovereignty is about not having to enter into dependencies, but being able to choose consciously where to avoid them and where to consider them unproblematic. These choices ensure freedom of design in what is an increasingly important digital world. This definition was established as early as the Digital Summit in 2018. The SCS team developed a model, published in a 2022 issue of *The Cloud Report* [4], that lets users assess the extent to which a platform lets them retain their sovereignty (Figure 1).

Compliance with the General Data Protection Regulation (GDPR) is fundamental (level 1). Anyone processing personal data cannot simply do so in the public clouds of providers outside the scope of European data protection regulations. The EU Commission's adequacy decisions may disguise this fact in the short term until the CJEU's next Schrems ruling, but they are no more than an all-too-obvious fig leaf.

Level 2 relates to having a choice of different cloud operators and, where applicable, setting up and operating a platform yourself. This choice must continue to exist even after you have chosen and optimized the workload and automation for one provider, which requires defined, provider-agnostic interfaces and identical system behavior, and it opens up another option: connecting several providers and using them as a large federated cloud.

Level 3 refers to the option of designing the platform in line with your own wishes. How exactly does the technology work? What configuration options do you have? Can the source code be changed to open up new possibilities? Why was a technology decision made? Can you successfully put forward and introduce changes? Only a platform that uses completely and permanently open source code developed by several parties in an open process can offer these options. The SCS team relies on the Open Infra Foundation's "Four Opens" [5] – Open Source, Open Design (decisions), Open Development, Open (and diverse) Community – and has developed a checklist for open source health [6].

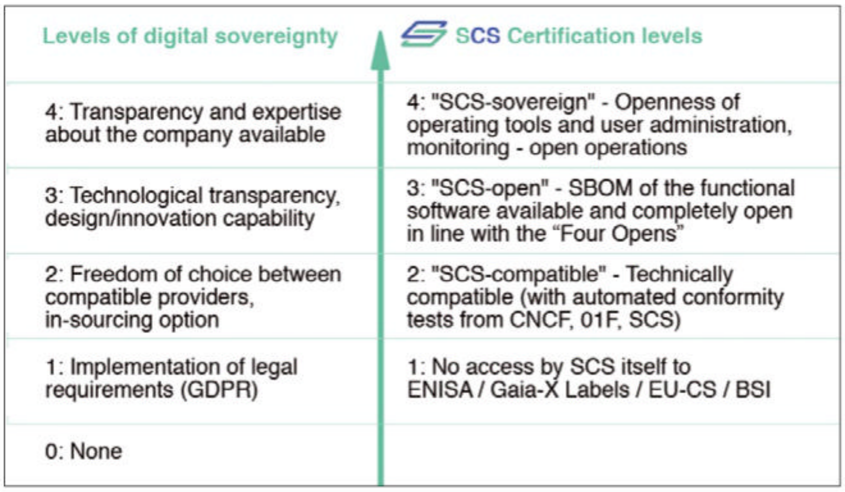
Highly reliable operation of a cloud infrastructure is a major challenge that necessitates building up knowledge and expertise and establishing the right tools and processes. Level 4, therefore, is about transparency for the operational aspects of the platform that make it possible to pass on knowledge so that knowledge can be established across all providers. The Open Operations initiative, along with the SCS team that played a key role in initiating it, has set itself precisely this goal.

This model can now be used to evaluate what are known as sovereign cloud offerings. On the one hand are the offshoots of the major American providers that are technologically equivalent to the originals, but where operations are handled by a European partner (e.g., Delos, T-Systems) or a European offshoot (e.g., Amazon, Oracle). The US authorities must not have technical access to customer data in these constructs, and the Cloud Act does not apply to partners. The European offshoots would need to be sufficiently independent of the parent company so that the parent would have no authority to issue instructions, meaning that the US Clarifying Lawful Overseas Use of Data (CLOUD) Act and, more importantly, the US Foreign Intelligence Surveillance Act (FISA) orders would not apply. Both the ability to rule out access technically and the non-applicability of the CLOUD Act need to be looked at very closely. If everything checks out, these offers would reach level 1 and therefore enable legally compliant processing of personal data.

However, the freedom of choice for any of these offerings is lacking with only one provider in each case (plus a partner in Europe, if applicable) that specifies the technology, available features, prices, and terms of use. As soon as you buy in, you cannot switch without major technical overhead. VMware could occupy a special position. The technology is available from several providers, and you can operate it yourself. Level 2, freedom of choice, would be achievable, but you cannot help getting the impression that its new owner, Broadcom, is no longer interested and would prefer to focus on rapid growth with a few large partners.

Of course, European operators of SCS platforms are subject to European data protection regulations. The European Union Agency for Cybersecurity (ENISA) rules or the related labels from Gaia-X can be used as a guide. The SCS standards define freedom of choice (level 2). These technical standards stipulate interfaces and system behavior and are validated by automated tests, guaranteeing a high degree of compliance. SCS platforms with SCS-compatible certification therefore offer genuine freedom of choice. Thanks to the ability to federate, users also have the option of combining several providers to create one large virtual cloud.

The SCS software (reference implementation) implements all SCS standards. It is a complete cloud



Levels of digital sovereignty	SCS Certification levels
4: Transparency and expertise about the company available	4: "SCS-sovereign" - Openness of operating tools and user administration, monitoring - open operations
3: Technological transparency, design/innovation capability	3: "SCS-open" - SBOM of the functional software available and completly open in line with the "Four Opens"
2: Freedom of choice between compatible providers, in-sourcing option	2: "SCS-compatible" - Technically compatible (with automated conformity tests from CNCF, 01F, SCS)
1: Implementation of legal requirements (GDPR)	1: No access by SCS itself to ENISA / Gaia-X Labels / EU-CS / BSI
0: None	

Figure 1: The four-stage model for digital sovereignty and the associated SCS certifications (translated from the German). © SCS

and container platform, unlike some other reference implementations that claim to be production-ready software. Developed entirely in accordance with the principles of the Four Opens, development takes place in close cooperation with the upstream communities. Users can get involved in a variety of ways and help shape the technology. Providers who deliver the functionality through the SCS software or use software developed in an equally open way can demonstrate their adherence through SCS open certification. Similarly, SCS sovereign certification will provide evidence of openness both in the operating tools and in the establishment of knowledge and transparency in the operating processes (Open Operations).

Architecture

Agile development methods have enabled a leap in productivity in software development over the past 20 years. In addition to the right culture, an agile infrastructure is important to leverage the benefits: a platform that empowers development teams (or rather DevOps or DevSecOps teams) to automate all steps, including deployment and operation, and to carry out all integration steps continuously and early in the development process. With good test coverage, you not only achieve higher speeds, but also better quality. Empowering teams to use a self-service infrastructure offers a very different motivation than does waiting for resources to be requested, approved by change boards, and provisioned manually.

Of course, this empowerment is the core promise of cloud environments – real clouds, mind you. Sometimes virtualization platforms without all the cloud attributes are wrongly marketed as private clouds. Applications can be developed in such cloud environments in a modular, scalable, and highly automated way (i.e., genuine cloud-native applications can be created). New cloud-native workloads are now mostly provided as containers. Kubernetes (K8s) has become widely accepted in the IT industry for container orchestration, making it the most important interface for application development teams and DevOps teams.

A focus on the provisioning of Kubernetes is the most important technical goal of SCS, but simply setting up and operating a K8s cluster is not enough to achieve the SCS goals. The clusters must have well-defined properties so that the operator can change without causing any issues. In most cases, a large centralized cluster does not meet the users' security requirements. Isolation, as provided by the lightweight containers, offers many benefits; however, it does not offer the level of isolation that you need to protect yourself against potentially malicious users, making it unsuitable for everyday use as a public cloud without taking further action.

The SCS project aims to serve as a basis for both private and public clouds, which has prompted a number of fundamental architectural decisions. The project facilitates the process of creating, managing (growing, shrinking, updating to a new K8s version), and removing your own K8s clusters – known in SCS as Kubernetes as a service (KaaS). These clusters exclusively belong to the customer, although the platform

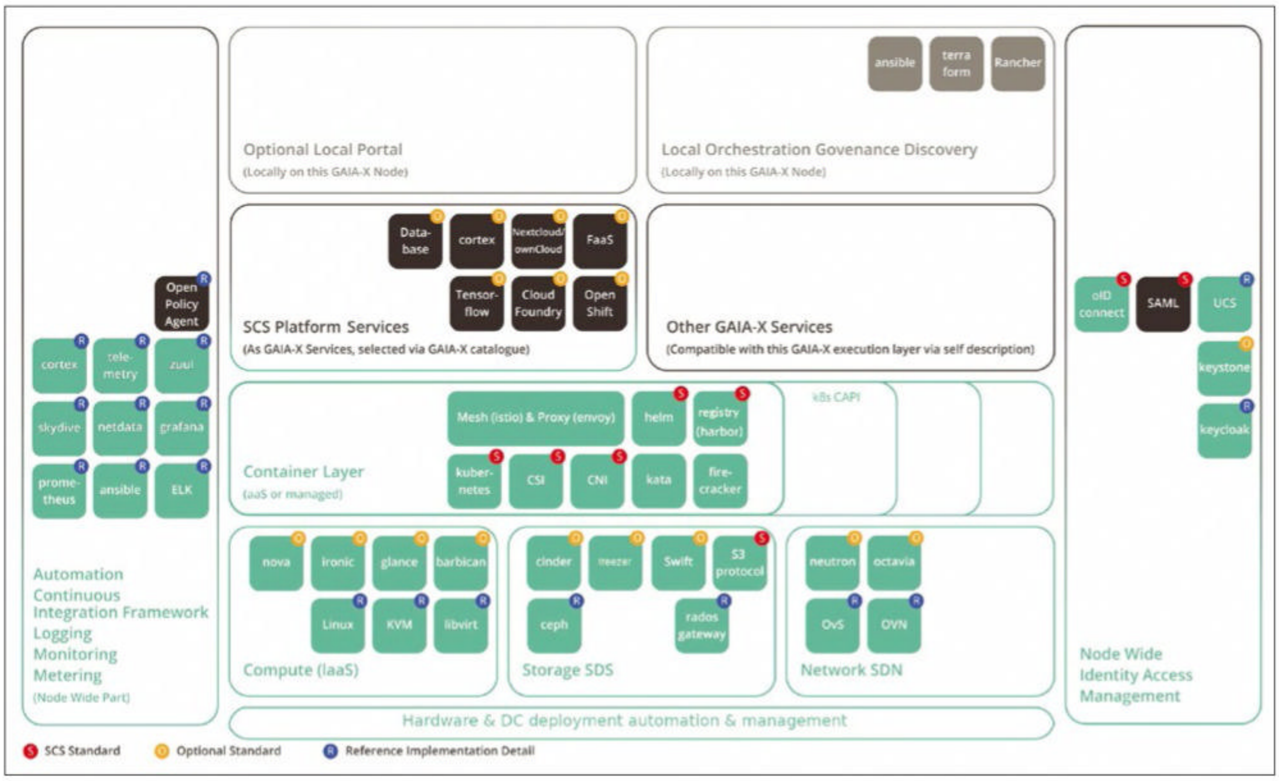


Figure 2: The different layers of SCS with IaaS and a container layer, along with the operating tools (left) and IAM (right). The dark tiles show potential add-ons that SCS itself has not (yet) implemented. © SCS

operator can support the operation of such K8s clusters.

Cluster API, the cross-provider Cloud Native Computing Foundation (CNCF) project, is used to provision clusters. It supports many cloud technologies as underlying platforms, and its use on bare metal is also possible. Cloud platforms make it easier to manage K8s clusters, which is why the vast majority of users choose this option and why the Kubernetes clusters in the SCS reference implementation are based on a sovereign cloud technology (infrastructure as a service, IaaS) by default.

OpenStack is used as the IaaS platform. As the only proven platform, it fulfills all openness requirements and has achieved a high degree of maturity in its core components, as well as a considerable degree of distribution, which is also in line with market studies on European cloud providers.

The IaaS and KaaS platforms can also be used individually if compliance with the standards is complete. However, the SCS team favors the use of both layers because it ensures the greatest sovereignty and synergy in the collaboration.

In addition to the IaaS and KaaS layers, setting up and operating a platform requires additional software, including operating tools – life-cycle management, monitoring, logging, auditing, metering, continuous integration (CI), etc. – and user and authorization management (identity and access management, IAM). These tools need to be developed with the same priority and, where possible, be able to support both layers. Figure 2 provides an overview of the architecture.

Some elements have been left out deliberately. The decision in favor of the Kubernetes Cluster API was controversial because the Gardener standard Kubernetes extension (SAP Gardener) is a very mature open source alternative. However, it fully depends on the goals of the enterprise resource planning (ERP) software company SAP. So far, neither has attempted to achieve technological convergence

Award Example: SCS Container Layer

The Cluster API (CAPI) makes it possible to manage complete K8s clusters as Kubernetes objects with custom resources, working on many different infrastructure layers with a provider (e.g., the OpenStack, CAPO, provider). To make all of this usable, however, you still need a few more elements: the cloud controller manager (CCM), network integration (container network interface, CNI), storage integration (container storage interface, CSI), node images, and automation for all of these components.

The project team (building on preliminary work by B1 Systems) implemented version 1 for OpenStack with a few scripts for an SCS KaaS. Tender package 5 included a call for tenders to implement the whole enchilada in a cloud-native way, to make things more generic, and to improve ease of use. The tender was won by Syself GmbH, which has since been working closely with the SCS community to develop the version for the SCS KaaS.

The result is the Cluster Stacks framework (Figure 3), which has an operator that uses K8s objects to control the entire automation setup while also managing the node images. The configuration is ideally obtained from a Git repository (GitOps). To date, the SCS project has implemented Cluster Stacks for OpenStack and Docker. However, the architecture is open to other environments, and SCS and Syself look forward to further supported infrastructures.

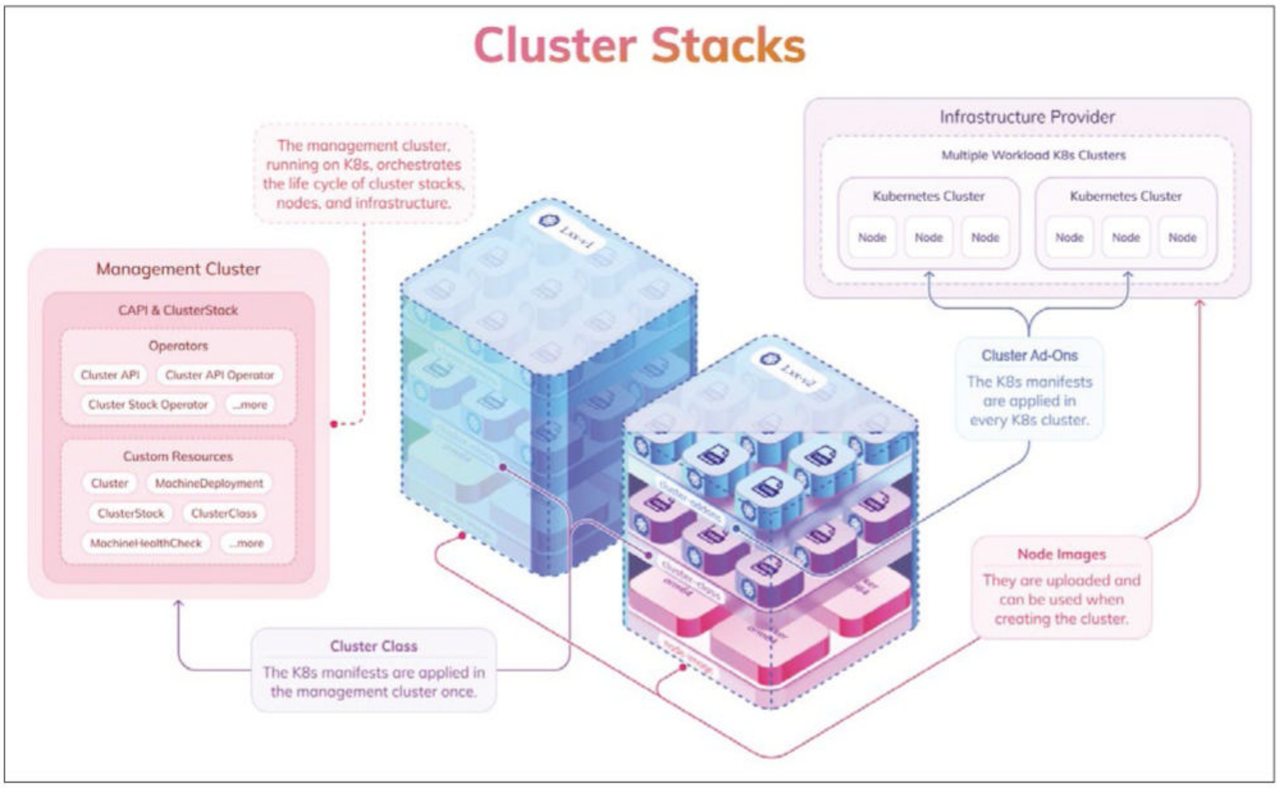


Figure 3: Cluster Stacks manage the Kubernetes clusters with the Cluster API and provide the node images and integration as cluster add-ons. © SCS

Award Example: Federation

In a target scenario with several SCS cloud providers used as a large virtual cloud, operators do not want to manage the users and their authorizations on each cloud individually. In OpenStack, the Keystone component is set up for federation by OpenID Connect (OIDC), whereas K8s uses OIDC by default. However, because things are not quite that simple, Univention is already working on improving federability in the scope of the SCS IAM award package. To enable user federation across layers, the OIDC-compliant Keycloak component was integrated into SCS to act as an identity provider and broker, which meant getting the device authorization grant workflow in Keystone up and running; the SCS project successfully incorporated this into the OpenStack project.
Customers of SCS operators should be able to manage users, authorizations, and projects within these domains themselves as administrators of OpenStack domains, which required contributions to the OpenStack project to help the manager role work for domains in OpenStack. The improvements therefore also benefit OpenStack operators independent of SCS and reduce the divergence between them. After the construction of sites on the IaaS layer, the focus is now on user management on the container layer.

with the Cluster API, nor have ideas to structure Gardener to be more independent of SAP met with success. The openness still exists, but the SCS reference implementation now depends a lot on the Cluster API. At least compatibility with the standards remains an easy option.
The solution does not include higher level platform services such as databases, email, messaging, big data, or artificial intelligence (AI). The SCS project is too small for that. On the other hand, the IaaS plus KaaS solution should provide an excellent platform for open source platform-as-a-service (PaaS) solutions, which in fact has already been demonstrated with individual examples such as ownCloud and Nextcloud.
The vision is that third-party providers have the opportunity to offer and integrate platform services, including automatic deployment, monitoring, user management, and billing. As soon as solutions become established as standards here, SCS can work with the partners and establish standards of higher value. Ideas are available from the Eclipse Xpanse project [7] and the developers of Glasskube [8].

Work Packages

All the components from the layers and tools must be configured and maintained consistently so that the result is a product that feels as if it has been cast from a single mold. Calls were made for tenders for maintenance, further development, and integration of the individual technology focuses (see the award examples in the "SCS Container Layer" and "Federation" boxes); companies were invited to develop concepts and contribute their expertise to the project. Contracts for the major blocks – IaaS, KaaS, IAM, and OpsTooling – were awarded to OSISM GmbH, Syself GmbH, Univention GmbH, and (for the operating tools) Gonicus GmbH and dNation s.r.o. (Slovak Republic).
Additionally, specific areas were identified that require further development in line with the project's objectives and will make the network layer (based on Open Virtual Network, OVN) more scalable and secure and enable connections between clouds. Plans are in place to improve client isolation and encryption in the storage solution, and the same applies to the components for integrating the Cluster API and the underlying OpenStack layer.
Also essential is analyzing the security architecture, carrying out attack tests, optimizing utilization data collection and system monitoring, and developing standards with automated tests. A test laboratory has been established in which tests can be carried out on physical hardware in close-to-production conditions. Tenders were awarded to B1 Systems, Cloud&Heat, SecuStack, Proventa International, and JH-Computers GmbH (see overviews on the SCS website [9]).

CI and Cloud in a Box

SCS development takes place in virtual teams that meet on a weekly basis over video conferencing to coordinate progress and resolve dependencies. The topics are prepared under the leadership of the product owner (PO, a member of the SCS project team); the decisions at the team meetings are typically reached by consensus. All dates are public and published in the Community Calendar. A Product Board for interdisciplinary topics is made up of POs and other community members (e.g., those who have assumed responsibility as leaders of special interest groups).
As you can already guess from the terminology, the process is based on scrum, and user stories that typically belong to larger epics and are fed from the product backlog are processed in 14-day sprints. Backlog refinements in smaller groups and short-term touchpoints take the place of team-wide dailies. Video conferences for consultations or hacking sessions can occur at any time on SCS's own Jitsi server. GitHub is used intensively during development to collect, discuss, and merge code. Much of the development work flows back into the upstream open source projects. Wherever possible, the SCS team seeks coordination with the upstream communities and also prefers to contribute code there: upstream first!
The meeting minutes are also stored on GitHub, created live and collaboratively during the appointments in HedgeDoc, a collaborative real-time editor with Markdown rendering. With Nextcloud, almost 200 members can flexibly edit and share files, contacts, surveys, and more. The members are automatically added to mailing lists, and a great deal of collaboration takes place in the SCS matrix chat rooms.
SCS delivers a complete cloud of its own. OSISM automation uses Ansible playbooks (the OSISM framework uses Kolla Ansible) to install infrastructure containers and

OpenStack containers automatically on the hardware and to start and manage the services. Even if this means the installation process (and in particular the upgrade) on hardware can be highly automated, it is typically not accessible to developers, most of whom simply don't have a small data center in their basement for testing purposes. Therefore, whereas applications for SCS can be tested easily in existing SCS environments, the SCS developers themselves have a more difficult slog, especially if they are working on the infrastructure layer.

OSISM developers, however, have a trick up their sleeve: Instead of installing the services on physical hardware, they can also run on virtual machines (VMs) in a cloud with the help of OpenTofu (formerly Terraform), which creates a virtual infrastructure on which a complete SCS can then be rolled out by normal mechanisms. An SCS cloud can be up and running in less than two hours. If the underlying cloud supports nested virtualization, the performance of the inner VMs is very much usable despite this double virtualization.

Of course, this type of deployment is not suitable for production purposes, but in terms of configuration and behavior, it comes quite close to a genuine deployment from a logical point of view and is therefore ideally suited to act as a test or demo environment for developers or to train operating personnel. This testbed (Figure 4) is currently implemented for OpenStack clouds, which in turn offer the huge advantage that SCS can host an SCS testbed. Implementation on other clouds (e.g., AWS or Azure) would be possible but has not yet been prioritized by the project team or implemented by a member of the SCS community out of their own interest.

This testbed is installed daily and tested automatically. The current code verison can therefore never be buggy for more than one day without it going noticed. In addition to a testbed reinstallation, an upgrade test is also carried out daily that runs through the update process from the last release to the current code version. Automation is handled by the very flexible Zuul CI framework.

Sometimes testing on physical hardware is more practical or even necessary, which requires another installation variant for this purpose: cloud in a box. A specially configured deployment profile is installed on a single workstation that, as a single-node deployment, differs more greatly from a genuine installation than the testbed. For a hardware investment of a few thousand euros, you can have your own small cloud without double virtualization that can also be expanded as an option for genuine, production-ready edge clouds.

Testing the K8s layer requires less effort because it is normally based on existing infrastructure. You can easily create a Kubernetes cluster in less than a quarter of an hour and then put it through its paces. Automation in version 1 of the KaaS implementation was limited to OpenStack as the underlying infrastructure layer, but the Cluster API can do more, managing clusters on VMware, AWS, OpenStack, Azure, and many other environments; this capability was implemented in version 2 of the SCS KaaS (Cluster Stacks), as well. The SCS project provides support for OpenStack and local deployments in Docker, and cooperation partners are being sought for further integration.

The roll-out of clusters can be integrated easily into the existing Zuul. CNCF end-to-end (E2E) tests, which are run by the Sonobuoy diagnostic tool, are typically used as the default tests in the SCS project. However, these tests take two hours, which means that, despite a fast cluster roll-out, the cycle time still cannot be reduced to less than a few hours if the full test program is run. Therefore, a shorter version exists for short tests.

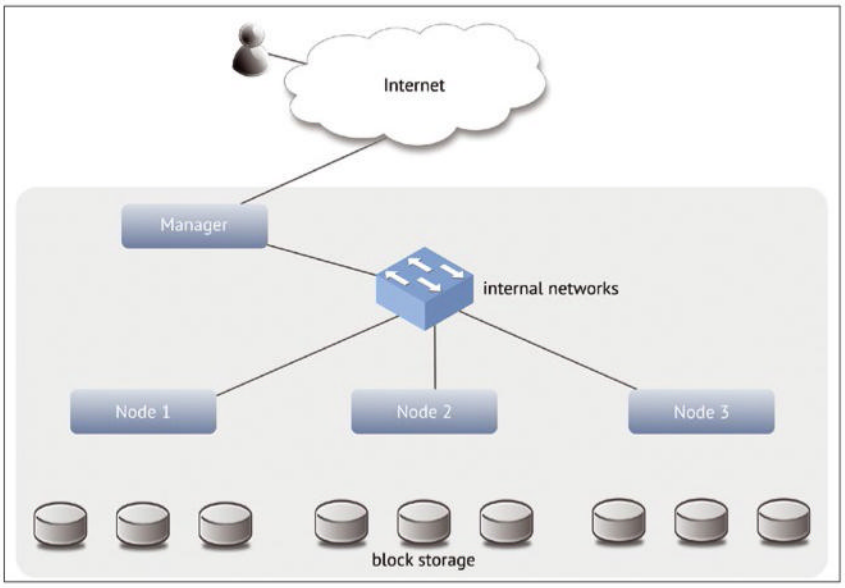


Figure 4: The SCS testbed with three hyperconverged nodes and a manager in an existing cloud. © SCS

Standards and Releases

Concepts are not enough. Developed and usable software are required to make cloud operators and ultimately their users an offer that they can accept and to make real progress toward digital sovereignty. Use by and feedback from real users is a basic prerequisite for the software to mature and meet the needs and requirements of users. Although this may sound like a truism, it is often overlooked.

Software releases were introduced in the SCS project right from the start. The CI was basically set up to enable cloud operators to update the platform on a minimally invasive weekly basis, which does not match the way the current SCS partners

Example of Flavors Standardization

A flavor in the IaaS implementation is a template for a VM: It specifies the number of virtual CPUs and the amount of memory, and it determines the size needed for an optional local hard disk from which the VM boots. You may define more virtual hardware properties for graphics cards, special CPU types, and special hypervisors, as well.
SCS also defines the virtual properties of the CPU and memory, which means that if they have a flavor with a V type CPU, the user can be sure of getting at least one fifth of a real CPU core (alternatively, one third of a hyperthread). Users know that the latest microcode must be installed and that the protection mechanisms, enabled by default on Linux and KVM to mitigate CPU security vulnerabilities, are used. The RAM allocation is also important.
The SCS flavor naming scheme lets users define flavors that do not provide these guarantees; however, they can be clearly identified by their names. SCS is working with the upstream community to encode these properties in standardized attributes so that this information no longer needs to be included in the names in the long term.

work. Upstream, Ubuntu, OpenStack, and Kubernetes are constantly being released with changes that require the operator, or even the user, to make adjustments, which is why the SCS project publishes releases at regular intervals; it also tests the releases manually and documents important changes in the release notes.
After the first release R0, which took place out of sequence almost immediately after the delayed project start in July 2021, regular SCS releases take place in September and March of each year. The current version is SCS R6 from the end of March 2024. Each release is given maintenance updates up to six weeks after the release of the subsequent version to fix bugs and address vulnerabilities in particular (for more information, see the SCS blog [10]).
Parallel to the releases, the project has developed standards for users of the cloud offerings that allow workloads with identical automation to be rolled out on different SCS infrastructures without having to adjust too many parameters or – worse still – having to study the service descriptions in advance to understand what system behavior is guaranteed for which resources.
One example is the SCS flavor standard (see the "Example of Flavors Standardization" box), which does not require a painstaking study of the flavor properties of each cloud operator. The results of the standardization efforts can be found in the SCS documentation pages [11]. Like the entire code, the standards on GitHub are developed, reviewed, and finally stabilized through pull requests (Figure 5). The standardization process is described in a separate standard (number 0001). Each individual standard has a different status, which is noted in the metadata of the respective document. The technical teams decide on the pull requests by vote.
Certification scopes summarize the individual standards and are broken down into two layers: IaaS and container (KaaS). The layers are subject to a life cycle in which new requirements replace old ones. In most cases, the new rules are a superset of the old ones so that users do not have to make any adjustments. Thanks to the standards, users can develop applications, operating processes, and automation once and then install and operate them on all SCS-compliant clouds. In this way, you can simply move or work in a federated way on several clouds – even in your own private SCS cloud if you want.
The standards are developed with a close tie-in to the reference implementation, ensuring that standards can be implemented in practice. Because some cloud operators are already working with the reference implementation, cloud operators and their users have provided feedback. This kind of proximity to the reference implementation promotes the quality of the standards; on the other hand, it carries the risk that they can be fulfilled without any difficulties, solely by the reference implementation. For this reason, the project deliberately seeks contact with upstream communities and projects in the SCS environment that use similar technologies but not the reference implementation. The desire is that these people create their own implementations of SCS-compatible standards.
These efforts have led to close cooperation with the Association for Operational, Open Cloud Infrastructures (ALASCA) e.V. [12] standardization association. The association was initially only intended to provide a neutral home for the Yaook project but was then launched with the greater ambition of improving digital sovereignty for cloud users; thus, the association shares many of the basic principles and objectives of the SCS project, making close cooperation a natural choice. This association is now bearing fruit: Many employees from ALASCA member companies contribute to SCS. In particular, the work package relating to standardization is being implemented by Cloud&Heat employees.
The first Yaook-based environments (which explicitly do not use most of the SCS IaaS reference implementation, OSISM) successfully implement SCS-compliant standards at the IaaS level.
All standards come with automated tests that check all essential

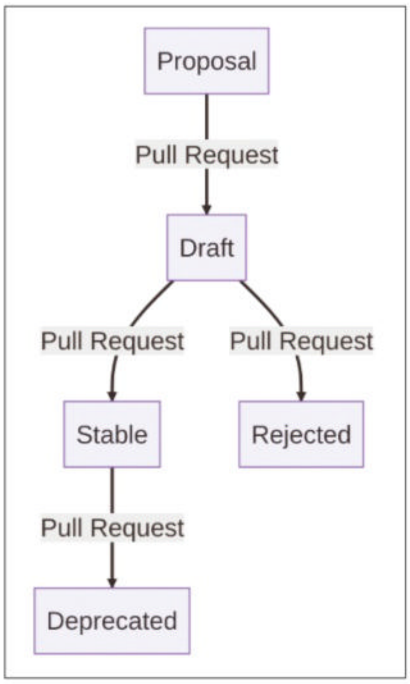


Figure 5: Status diagram of the individual standards. © SCS

properties of the standards wherever possible, which means that cloud operators know at all times whether they meet all standards and where gaps may exist. The tests do not require any special authorizations, so users can also run them at any time to check compliance.

SCS Cloud Offerings

Without seeing production use, software can achieve neither relevance nor practical applicability. SCS was fortunate that, besides OSISM GmbH with its Betacloud test cloud, a provider with greater ambitions also quickly backed SCS: plusserver GmbH with pluscloud open (PCO). The first versions of PCO were based on SCS before the R0 release. The implementation was updated after each SCS release.

OpenStack upgrades are generally considered difficult. Many providers do not manage to keep up to date and thus contribute to the fragmentation of the market. In addition to peculiarities in the configuration of the OpenStack cloud, users also have to deal with legacy versions – which is not the case with SCS.

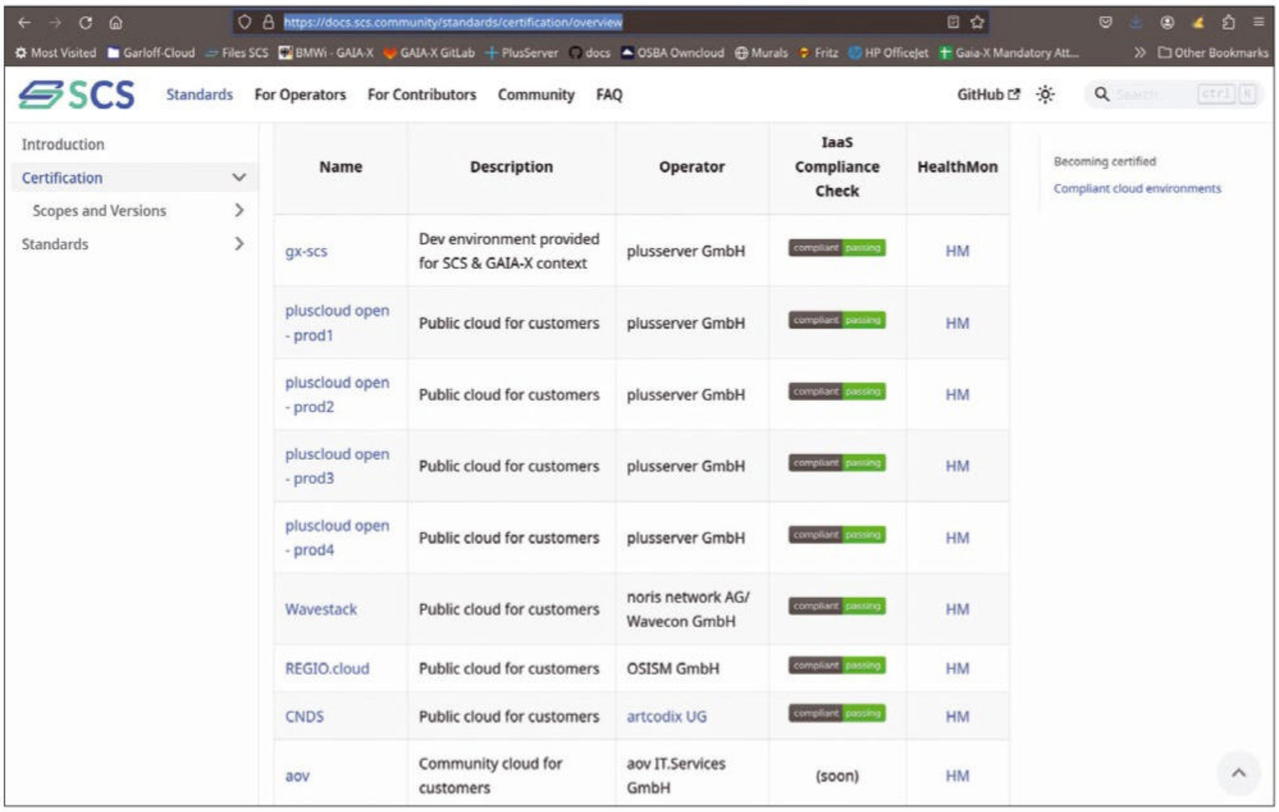
Containerization of the OpenStack services with Kolla Ansible, automation by OSISM, and the nightly upgrade tests have removed worries about the update process. At the time this article was written (the end of March 2024), plusserver was already running the various test and development platforms on the newly released SCS R6 (with OpenStack 2023.2 "Bobcat"), and preparations for upgrading what are by now four production regions were in full swing.

Yes, plusserver now has four regions, and PCO growth is massive. According to public statements at pluscon in fall 2023, growth is significantly greater than that of the existing VMware-based pluscloud solution. One flagship project, the BayernCloud School (ByCS) [13], required a large, scalable platform for hundreds of thousands of pupils, their teachers, and parents; plusserver implemented their infrastructure with PCO, which can be seen as a baptism of fire for the scalability of SCS.

The SCS promise to offer cross-provider solutions would be hollow if the only successful PCO was plusserver. From fall 2022 to March 2024, Wavestack (from noris network AG), Regio Cloud (JH-Computers/OSISM), Community Cloud (aov IT.Services GmbH), and CNDS Cloud (artcodix GmbH) were added. Many other SCS clouds are currently being set up, including those at Kyndryl, Dataport, T-Systems, and ScaleUp. Industry has also set up SCS projects as private clouds, all based on the SCS reference implementation.

For public clouds, compliance with SCS standards is automated and continuously monitored and can be viewed publicly on the SCS documentation pages [14] (Figure 6). The first clouds not based on the SCS reference implementation, but nevertheless fully SCS-compatible at the IaaS level, are likely to appear soon.

In the list of clouds with an SCS compliance test result, you will also see an element of Open Operations.



Name	Description	Operator	IaaS Compliance Check	HealthMon
gx-scs	Dev environment provided for SCS & GAIA-X context	plusserver GmbH	compliant passing	HM
pluscloud open - prod1	Public cloud for customers	plusserver GmbH	compliant passing	HM
pluscloud open - prod2	Public cloud for customers	plusserver GmbH	compliant passing	HM
pluscloud open - prod3	Public cloud for customers	plusserver GmbH	compliant passing	HM
pluscloud open - prod4	Public cloud for customers	plusserver GmbH	compliant passing	HM
Wavestack	Public cloud for customers	noris network AG/ Wavecon GmbH	compliant passing	HM
REGIO.cloud	Public cloud for customers	OSISM GmbH	compliant passing	HM
CNDS	Public cloud for customers	artcodix UG	compliant passing	HM
aov	Community cloud for customers	aov IT.Services GmbH	(soon)	HM

Figure 6: The daily compliance check of public SCS clouds from March 28, 2020. In addition to the compliance check (second to last column), all providers also offer an OpenStack Health Monitor (last column), which permanently monitors the functionality of the clouds. © SCS

Monitoring cloud environments with the OpenStack Health Monitor makes the status of the clouds visible within the SCS community. Instead of faking a perfect world, the real situation is relentlessly made transparent, which makes any challenges that arise visible to everyone; they can be tackled jointly with everyone benefitting from the lessons learned. The very good results from monitoring of the production clouds are a welcome side effect. Only quality measured and reliably made transparent gets the attention it deserves.

The SCS community has organized formats (e.g., Lean Coffee and Open Operations Meetups) for the operator community; the knowledge exchange there helps cloud operator employees build on the experience of a large community instead of having to learn from their own mistakes, alleviating the shortage of highly qualified operating personnel.

Digital sovereignty is a basic prerequisite for the use of digital platforms for public administration – at least at level 1 (data protection). At the same time, the fragmentation of the IT landscape because of federalism has tended to hinder the emergence of efficiency-enhancing platforms and has therefore proven to be of little help in digitalizing public authorities. The highly sensible one-for-all principle, supporting the reuse of applications by other municipalities or countries and joint development on Open CoDE, a platform of public administration for the exchange of open source software, will only really work with the presence of similar execution environments. The IT Planning Council is working on this problem and has defined suitable guidelines with the German Administration of Cloud Strategy (DVC) [15]. They reference SCS as an example of a successful sovereign and federated approach.

In a digitalization competition in the state of Schleswig-Holstein, SCS was required as a cloud layer, and the OSBA WG Cloud is working with Federal IT Cooperation (FITKO) to validate the portability of applications with SCS standards. The requirement to comply permanently with SCS-compliant standards gives the user the security of not being dependent on a single provider. A number of other providers, particularly in the public sector, will certainly ensure their interoperability through SCS standards, whether at the IaaS level or increasingly at the container level with SCS Cluster Stacks and the associated standards.

This environment is ideal for a project like Opendesk, which is being tested together with ZenDIS. The first major provider in the public sector is the Thuringian State Computing Center, which has been building a large platform based on the SCS reference implementation since 2023.

Demand in other European countries is confirmed by invitations to Switzerland, cooperation with Cleura AB (Sweden), contacts in The Netherlands (Organisation for Applied Scientific Research, TNO), and cooperation with the international GovStack project [16].

Future

VMware's new licensing terms after its acquisition by Broadcom were a wake-up call for IT infrastructure operators – a painful reminder that dependencies can have costly consequences. Smaller users of VMware are more likely to find a VMware replacement in Proxmox, whereas SCS can be a good choice for larger environments – whether it continues to be operated in-house or in combination with public SCS offerings. On the basis of discussions with interested parties, the SCS project has prioritized the VM and high-availability (HA) feature for the roadmap, which involves the infrastructure automatically restarting virtual machines elsewhere if a host fails.

Funding for the SCS project from the BMWK runs until September 30, 2024 (Figure 7), which, however, does not signal the end of the project; it just needs to survive on its own from then onward. In numerous discussions, the SCS team was told that the not-for-profit structure in a cross-provider association with public funding generates a great deal of credibility. The intent is therefore for ongoing work not to take place solely in a commercial enterprise. Instead, there will be two entities.

Trademark rights (currently held by OSBA), standardization efforts, certification, participation in upstream communities, and neutral management of the ecosystem of partners and the community will be operated by a nonprofit entity, which ensures the openness of the community and the processes. Still unclear is whether it will become a separate organization or a largely independent department within OSBA.

On the other hand, users of the reference implementation expect further development and maintenance (for

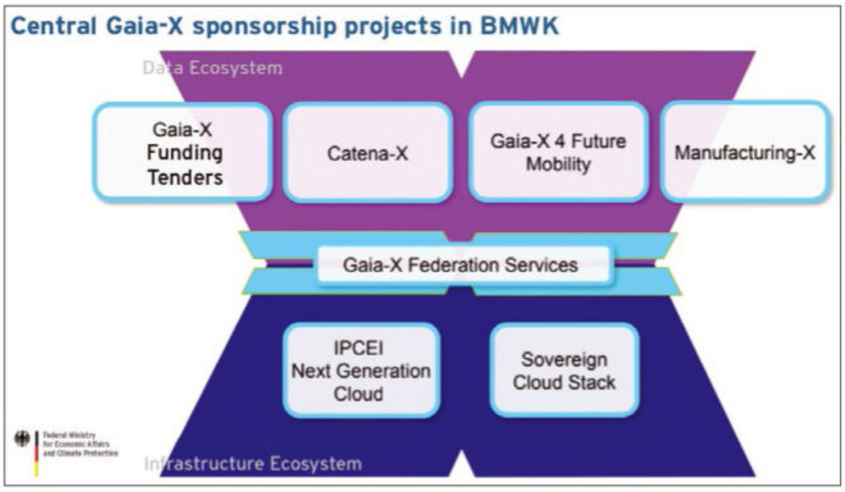


Figure 7: Central Gaia-X funding projects in BMWK. © SCS

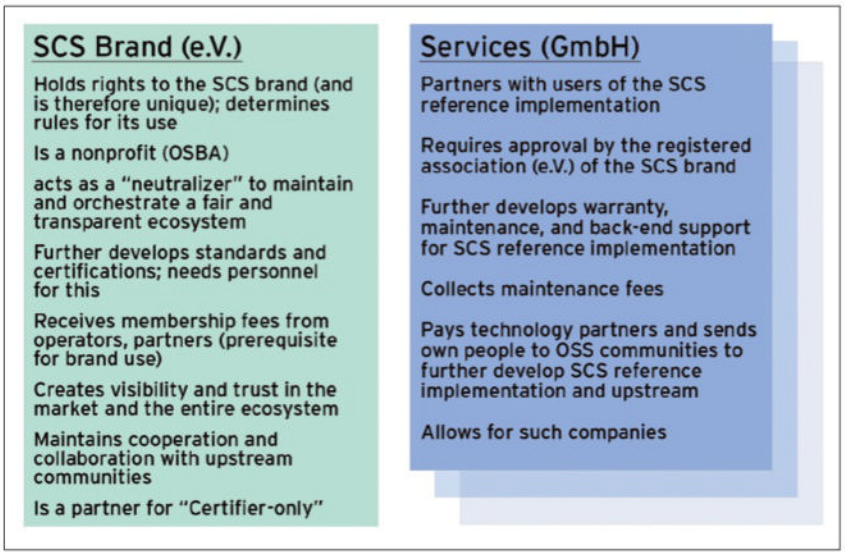
upstream projects, as well) and professional maintenance and support offerings. This classic business model for an open source company (Figure 8) can effectively bundle and ensure these services through cooperation with the technology partners and payment. In return, it will receive maintenance fees from the cloud operators.

Conclusions

US policy and its choice of allies directly affects security, particularly in Eastern European countries. Not even deployment planning for the newly procured F35 fighter jets works without support from the AWS cloud platform. Without the goodwill of platform operators based in the US or the People's Republic of China, European industry cannot produce the goods needed. Europeans must ask whether they will be able to defend their own security – politically, militarily, technologically, and economically.

Even without such gloomy prospects, good reasons abound to prompt nations to build up or reestablish their own expertise and innovative capacity for basic digital technology, relating not only to compliance with country-specific rules and laws (e.g., data protection), but to having a strategic background (risk management). All nations should be able to participate in the value that increasingly important digital platforms add and achieve cost efficiencies to match through a functioning market without oligopolies.

The SCS project took care from the outset to ensure that individual manufacturers alone did not define the software and that the copyright remained distributed, so that monopolization along the lines of Redis is not possible. The company needs to meet all the openness requirements (Four Opens) with the implementation to be allowed to use the SCS brand. This route is also open to other companies. The future of SCS is secure for the time being without further public funding. Its exact size from October 2024 will still depend on the outcome of some talks. It is clear that there are many ideas for making SCS more comprehensive and meaningful – for example, for standard solutions in the field of classic platform services such as database services or initiatives in the area of artificial intelligence, where it is not desirable to disclose the data of members of the public or the enterprise to third parties to fine-tune large language models (LLMs) – or even carry out basic training. In any case, the SCS project is looking forward to support, both private and public, to tackle such ideas with the required speed. ■



SCS Brand (e.V.)

- Holds rights to the SCS brand (and is therefore unique); determines rules for its use
- Is a nonprofit (OSBA)
- acts as a "neutralizer" to maintain and orchestrate a fair and transparent ecosystem
- Further develops standards and certifications; needs personnel for this
- Receives membership fees from operators, partners (prerequisite for brand use)
- Creates visibility and trust in the market and the entire ecosystem
- Maintains cooperation and collaboration with upstream communities
- Is a partner for "Certifier-only"

Services (GmbH)

- Partners with users of the SCS reference implementation
- Requires approval by the registered association (e.V.) of the SCS brand
- Further develops warranty, maintenance, and back-end support for SCS reference implementation
- Collects maintenance fees
- Pays technology partners and sends own people to OSS communities to further develop SCS reference implementation and upstream
- Allows for such companies

Figure 8: Duality between the organization for the SCS brand and its standards and the possible service companies (GmbH) that support specific implementations (reference implementation). © SCS

Info

[1] EU Commission study on digitalization in Europe, 2023: [https://ec.europa.eu/eurostat/web/interactive-publications/digitalisation-2023]

[2] Digitalization in the European Union. Press Release, May 2023: [https://www.eib.org/en/press/all/2023-203-digitalisation-in-the-european-union-progress-challenges-and-future-opportunities]

[3] Max Schrems: [https://en.wikipedia.org/wiki/Max_Schrems]

[4] "Digital Sovereignty," Friederike Zelke, editor. *The Cloud Report*, issue 1/2022: [https://the-report.cloud/downloads]

[5] Four Opens: [https://openinfra.dev/four-opens]

[6] OSS Health Check: [https://github.com/SovereignCloudStack/standards/blob/main/Drafts/OSS-Health.md]

[7] Eclipse Xpanse: [https://projects.eclipse.org/projects/technology.xpanse]

[8] Glasskube: [https://glasskube.eu/en/]

[9] SCS tenders: [https://scs.community/tenders]

[10] SCS blog: [https://scs.community/blog]

[11] SCS standards: [https://docs.scs.community/standards]

[12] ALASCA e.V.: [https://alasca.cloud/en/]

[13] ByCS: [https://bycs.de] (in German)

[14] SCS compatibility: [https://docs.scs.community/standards/certification/overview]

[15] DVC strategy (v3.0): [https://www.cio.bund.de/Webs/CIO/DE/digitale-loesungen/digitale-souveraenitaet/deutsche-verwaltungscloud-strategie/deutsche-verwaltungscloud-strategie-node.html] (in German)

[16] GovStack: [https://govstack.global]

The Author

After training as a physicist, Kurt Garloff has spent his professional life in the open source world, including some time at SUSE as a Linux kernel developer and a team, departmental, and development manager. He has spent the past 12 years working with Open Infrastructure and was responsible for the development of the Open Telekom Cloud as Chief Architect at T-Systems. Since 2020, he has headed the Open Source Business Alliance's Sovereign Cloud Stack project.

Managing GPT-4 on the Network

Watch Your Step

Now that Microsoft has integrated its GPT-4-based AI chat functions directly into the operating system with Windows Copilot, which now includes Bing Chat and Bing Chat Enterprise, you might want to know how to control or even block its use on the network. By Thomas Joos

Copilot is integrated as part of Windows 11 22H2 Moment 4 or Windows 11 23H2. As with all new functions, Microsoft does not enable new services on all computers at once, but in several steps. Because of legal regulations in Europe, it might take some time before Copilot is available everywhere in Windows 11.

That said, not all companies will want to use the artificial intelligence (AI) function in Windows 11. Wherever the feature is desired, effective management – including data protection – should always be in place. Windows Copilot is currently still at a very early stage. Microsoft can be expected to make significant improvements, also in terms of configuration options for administrators. At present, though, the control functions are fairly few and far between; I look at some control approaches in more detail in this article.

In principle, Microsoft automatically enables all functions in the AI services as soon as they are available and all users can access them. If you want to restrict usage, you need to make some changes by Group Policy, registry values, the Microsoft 365 Admin Center, the Bing Chat Enterprise Admin Center, Microsoft Intune, or other services. It generally makes sense to have a set of rules for the use of AI services, but this is not a task for the IT department.

Windows Copilot is accompanied by the integration of AI chat functionality in Microsoft Edge and Google Chrome in the form of the new Bing Chat and Bing Chat Enterprise for businesses. As with ChatGPT, users can communicate with the GPT-4 large language model (LLM). In the case of Bing Chat, the AI accesses the Internet directly. You have various ways in which to make use of the AI features on Windows 10 and Windows 11: Windows Copilot, Bing Chat, Bing Chat Enterprise, and the option to communicate directly with ChatGPT, as well as with Microsoft 365 Copilot.

Corporate Data in the Crosshairs

The various LLMs on which AI bots such as ChatGPT, Windows Copilot, and Google Bard are based do not just learn from predefined training data, but also from the information that users of the AI bots enter at the various prompts to retrieve information from the AI and to create texts. If they use critical or personal data when interacting with the AI, this is equivalent to transferring the data to the Internet and releasing the data to the AI, which then learns by processing the information.

In a worst case scenario, this chain of events can lead to a company's competitors gaining access to internal information and profiting from it because employees have passed on the data voluntarily. Many company managers are not yet aware of these consequences, which is why Windows Copilot and Bing Chat need to be managed, such that it is only possible to use these services without disclosing sensitive data.

As a general rule, controlling services does not mean that AI chats are not used at all in companies – after all, they can be used to perform important and useful tasks. However, it is crucial to manage that use and to prevent company secrets from flowing first to the AI and then to other players.

Companies who opt for Bing Chat can rely on Bing Chat Enterprise to serve this function. Users benefit from a more feature-rich AI chat, but the requested information is not used for the AI's own learning process and therefore not passed on. Bing Chat Enterprise protects companies against the risk of sensitive data falling into the wrong hands. In general, the recommendation is not to use Bing Chat in a corporate environment, but instead to opt for Bing Chat Enterprise, possibly in conjunction with Windows Copilot, which is based on Bing Chat and can also use Bing Chat Enterprise.

Windows Copilot in Europe

Put simply, Windows Copilot is the direct integration of GPT-4 into Windows 11. Redmond is rolling out Copilot in the Moment 4 update in Windows 11 22H2, and you can expect to see it in Windows 11 23H2 at the latest. Although the feature set is already installed in Europe, it cannot be used right now because of the Digital Markets Act, a European Union (EU) regulation intended to ensure fairer competition in the EU and give users more choice when it comes to online offerings.
You can launch Copilot by clicking on the icon in the taskbar or typing in the shortcut,

```
microsoft-edge:///?ux=copilot&tcp=1↩
  &source=taskbar
```

which works on computers in Europe even before the feature is officially enabled by Microsoft. You simply need the latest updates in place. If they are, you can use Copilot on the basis of Bing Chat Enterprise, provided the computer meets the requirements and your company has taken out a subscription to match (Figure 1).
Windows Copilot appears as a sidebar in Windows 11 that is not hidden by other programs, which means you can work with Copilot and other programs in parallel. If Copilot is not yet available on your computer, enabling the *Get the latest updates as soon as they're available* setting in Windows Update in the Windows 11 configuration should help. No admin rights are required – one more good reason to come to grips with the issue now.

Windows Copilot in Practice

You can press the Copilot icon and then type AI prompts directly in Windows (e.g., to control the operating system). Examples might be, *The screen is too bright*, *The volume is too high*, or *I want to activate BitLocker*. The AI is initially intended to help you control Windows quickly and easily by avoiding the need for deep diving into menus and letting you type your questions and problems directly at the chat prompt. Copilot can then run the programs itself or display step-by-step instructions.
At the same time, you can work directly with conventional AI prompts, entering the prompts in the same way as in ChatGPT. Note that the data does not remain on your network; instead, Copilot phones home to Microsoft. Copilot is based on GPT-4, but Microsoft has stated that it will not transfer any data to OpenAI. Of course, Microsoft uses the data itself. When you use Windows Copilot, the communication exchange takes place over the Internet, and Copilot, or the underlying AI, learns from the input. As I mentioned earlier, it does not make sense to share company secrets with Windows Copilot and Bing Chat – or with ChatGPT, Google, Bard, or any other AI bot, for that matter. However, if your computer is connected to Azure AD/Entra ID and you log in there, you can use Copilot with Bing Chat Enterprise. In this case, the company's data is protected against use for AI training, as indicated by the *Protected* status at the top of the Copilot window.
If you do not see this status indicator, Copilot is using Bing Chat. In this case, you should avoid using the chat feature and enable Bing Chat Enterprise up front. You need a matching Microsoft 365 subscription to make this available in the browser and in Windows Copilot.

Disabling Windows Copilot

If you want to disable Copilot for Windows 11 completely, you can use Group Policy in Active Directory. Of course, the corresponding settings are also available in local policy management, which you can launch by typing `gpedit.msc`. You will need the latest Group Policy templates for Windows 11 or the templates for Windows 11 22H2 Moment 4. The settings are under *User Configuration\Administrative Templates\Windows Components\Windows Copilot*. Currently, only the *Turn off Windows Copilot* policy setting can be found here (Figure 2).
If you enable this setting, you are not allowed to work with Windows Copilot. By default, Windows automatically allows access after activating Windows Copilot in Europe. Make sure you disable Windows Copilot first and do not enable it again until extensive testing of the function is complete.
Like many other Group Policy entries, you can alternatively configure the settings directly in the registry by creating a new DWORD value named *HideCopilotButton* in the *HKEY_ LOCAL_MACHINE*

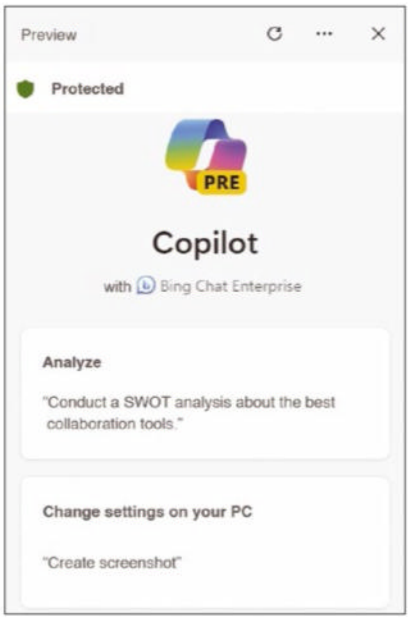
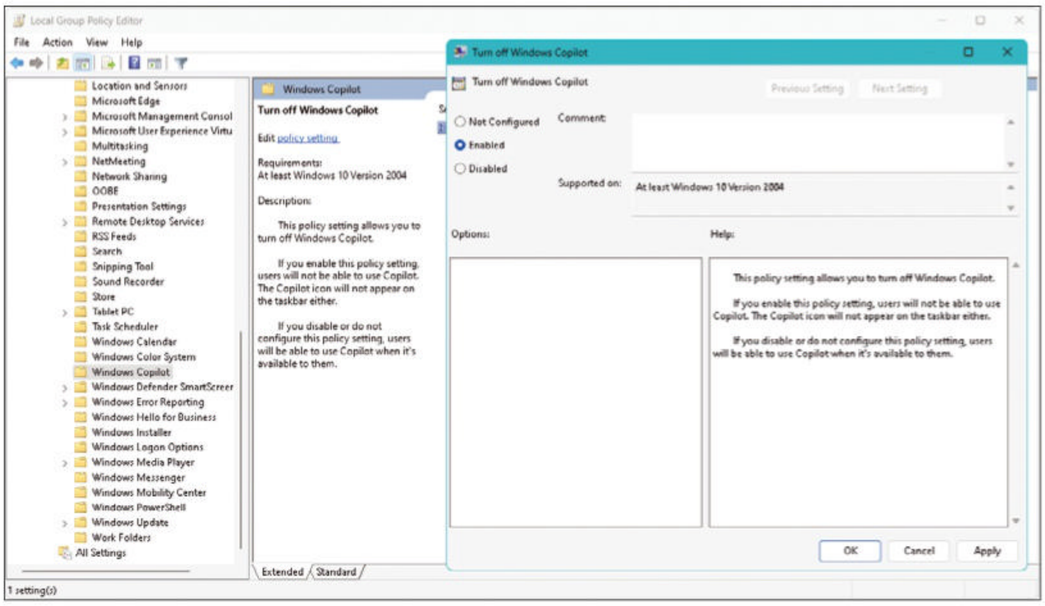

Preview
Protected
PRE
Copilot
with Bing Chat Enterprise
Analyze
"Conduct a SWOT analysis about the best collaboration tools."
Change settings on your PC
"Create screenshot"

Figure 1: In combination with Bing Chat Enterprise, Windows Copilot does not send user prompts to Microsoft, as indicated by the *Protected* label.


Local Group Policy Editor

File Action View Help

- Location and Sensors
- Microsoft Edge
- Microsoft Management Consol
- Microsoft User Experience Virtu
- Multitasking
- NetMeeting
- Network Sharing
- OOBE
- Presentation Settings
- Remote Desktop Services
- RSS Feeds
- Search
- Snipping Tool
- Sound Recorder
- Store
- Tablet PC
- Task Scheduler
- Windows Calendar
- Windows Color System
- Windows Copilot
- Windows Defender SmartScreer
- Windows Error Reporting
- Windows Hello for Business
- Windows Installer
- Windows Logon Options
- Windows Media Player
- Windows Messenger
- Windows Mobility Center
- Windows PowerShell
- Windows Update
- Work Folders
- All Settings

Windows Copilot

Turn off Windows Copilot

Edit policy setting

Requirements:
At least Windows 10 Version 2004

Description:

This policy setting allows you to turn off Windows Copilot.

If you enable this policy setting, users will not be able to use Copilot. The Copilot icon will not appear on the taskbar either.

If you disable or do not configure this policy setting, users will be able to use Copilot when it's available to them.

Extended / Standard

1 setting(s)

Turn off Windows Copilot

Turn off Windows Copilot

Previous Setting Next Setting

Not Configured [✓ Enabled] Disabled

Comment:

Supported on: At least Windows 10 Version 2004

Options:

Help:

This policy setting allows you to turn off Windows Copilot.

If you enable this policy setting, users will not be able to use Copilot. The Copilot icon will not appear on the taskbar either.

If you disable or do not configure this policy setting, users will be able to use Copilot when it's available to them.

OK Cancel Apply

Figure 2: **Administrators can use Group Policy to disable Windows Copilot.**

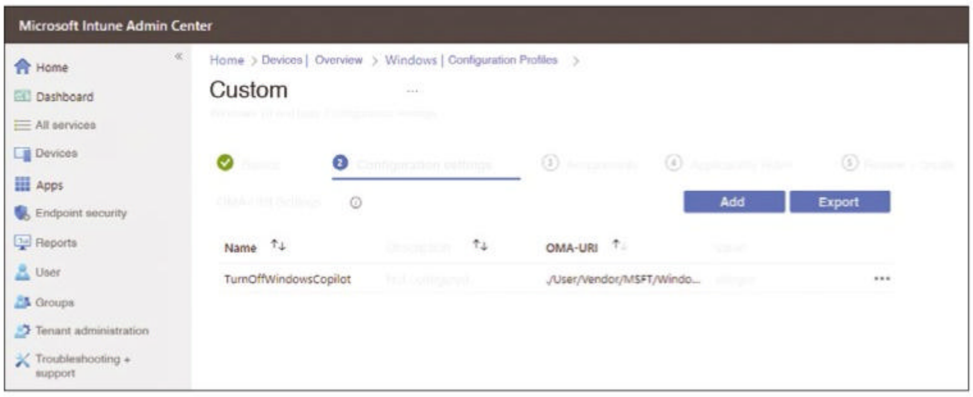
SOFTWARE\Policies\Microsoft\Windows\Explorer path and set the value to *1*. This value initially only hides the Copilot button in the taskbar but does not deactivate the AI service. You can manually hide Copilot in the taskbar in the Windows 11 settings under *Personalization* | *Taskbar* | *Copilot* (but, again, you cannot disable Copilot).

In *HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\WindowsCopilot*, you can disable Windows Copilot by setting the *TurnOffWindowsCopilot* DWORD value to *1*. After restarting the computer, the Copilot button has disappeared from the taskbar and can no longer be started – not even with a hack. Unfortunately, users can still work with Bing Chat or Bing Chat Enterprise in Edge.

Microsoft Intune

On mobile devices and home workstations, Copilot can be deactivated with Microsoft Intune by creating a new device profile; you can then use this profile to configure Copilot on the connected computers. The settings are under *Devices* | *Windows* | *Configuration Profiles*. Use *Create profile* and select *Windows 10 and higher* as the platform and *Templates* as the profile type, then click *Custom* and *Create*.

Assign a name to the new profile (e.g., *Windows Copilot*). Press *Add* on the *Configuration Settings* tab (Figure 3), then type *TurnOffWindowsCopilot* (e.g., under Name). In the OMA-URI field, enter *./User/Vendor/MSFT/WindowsAI/TurnOffWindowsCopilot* with *Integer* as the data type and *1* as the value. Save the settings and finish creating the configuration profile. Then assign the configuration profile to the desired computers to disable Copilot. No other configuration profiles are different from this point on.


Microsoft Intune Admin Center

Home > Devices | Overview > Windows | Configuration Profiles >

Custom

Name		OMA-URI	
TurnOffWindowsCopilot		./User/Vendor/MSFT/Windo...	

Figure 3: **Windows Copilot can also be disabled in Microsoft Intune.**

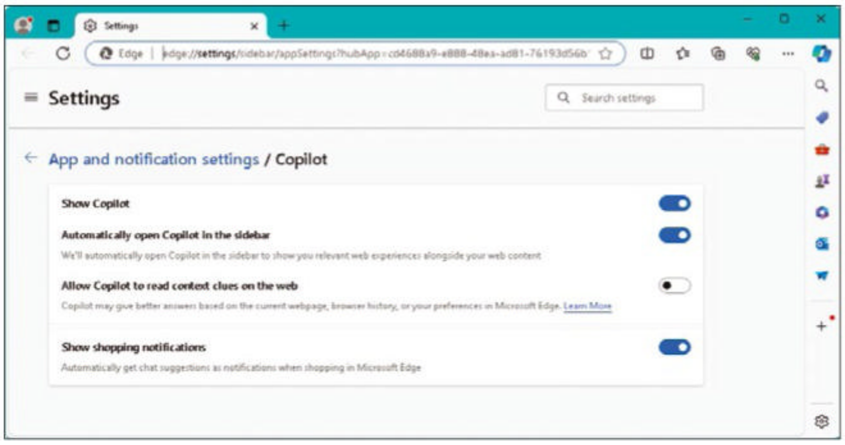
Disabling Copilot in Edge

By default, the Copilot icon is available in the top right corner of Microsoft Edge, so you can open Bing Chat or Bing Chat Enterprise. However, this service has nothing to do with Windows Copilot, except that Windows Copilot relies on Bing Chat and Bing Chat Enterprise as its basis. Even if you disable Windows Copilot, you can still access Copilot in the Edge browser. To hide the AI application manually, type *edge://settings/sidebar* in the address bar and disable *Show Copilot* (Figure 4).

These settings can also be automated with registry entries by adding the following two entries with the specified commands:

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\⏎
        Policies\Microsoft\Edge"
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\⏎
        Policies\Microsoft\Edge" ⏎
  /v HubsSidebarEnabled ⏎
  /t REG_DWORD /d 00000000 /f
```

After a restart, the sidebar with Bing Chat is no longer available in Edge. If you delete the key with the settings, the sidebar becomes available again after a restart – in Bing Chat, as well. The settings are available in the policies under *User configuration* | *Computer configuration* |


Settings

edge://settings/sidebar/appSettings?hubApp=cd4688a9-e888-48ea-ad81-76193d56b

Settings

Search settings

App and notification settings / Copilot

Show Copilot ●

Automatically open Copilot in the sidebar ●

We'll automatically open Copilot in the sidebar to show you relevant web experiences alongside your web content

Allow Copilot to read context clues on the web ○

Copilot may give better answers based on the current webpage, browser history, or your preferences in Microsoft Edge. Learn More

Show shopping notifications ●

Automatically get chat suggestions as notifications when shopping in Microsoft Edge

Figure 4: Copilot in Microsoft Chat, and therefore also Bing Chat and Bing Chat Enterprise, can be disabled in the browser settings.

Administrative templates | Microsoft Edge. However, the current ADMX files for Microsoft Edge for Business must be in place for this to work. You can pick them up from the Microsoft Edge download page [1]. These settings can be used locally or distributed by Group Policy.

If you use Edge in the enterprise, it definitely makes sense to secure the browser with the additional security settings. Edge version 116 or newer must be in place. From this version onward, Microsoft Edge for Business is enabled on signing in to Azure AD/ Entra ID, allowing the browser to separate business and private data. The briefcase symbol shows that Edge's business mode is active.

In the future, you will be able to manage Edge for Business in the Microsoft 365 Admin Center under *https:// admin.microsoft.com/ adminportal/ home#/Edge*, where you specify configuration profiles and define page lists. This development is significant for Bing Chat and Bing Chat Enterprise because the AI chat is also available in the sidebar. Again, the possibilities are currently limited.

Is Bing Chat Enterprise for You?

You need to think about whether it makes sense to use Bing Chat Enterprise. If you are logged in with an account in Azure AD/Entra ID on a computer that is part of a matching Microsoft 365 subscription, Bing Chat Enterprise can be used in Edge, and Copilot with Bing Chat Enterprise is available on Windows 11. At the same time, Edge activates business mode, which also protects the other data in the browser.

The Microsoft 365 E3/E5 and A3/A5 and Business Standard/Premium subscriptions all have access to Bing Chat Enterprise (see the box “Bing Chat Enterprise Is Not Microsoft 365 Copilot”). As soon as users with a license in this subscription open Bing Chat, Bing Chat Enterprise is automatically enabled. Of course, this requires a login to Azure AD/Entra ID.

You can enable Bing Chat Enterprise and manage it in the associated Admin Center, which is currently still under development; however, you can already see whether the service is active. The Admin Center can be accessed under *https://www.bing.com/business/bceadmin*. The feature set is currently still limited, but further options are likely to be introduced in the future.

Bing Chat Enterprise can only be used in the browser; API access is not possible. However, companies can register with OpenAI for ChatGPT Plus/Enterprise to enable API access. That said, Bing Chat Enterprise and Windows Copilot have nothing to do with ChatGPT Plus or Enterprise, although all three AI services are based on the OpenAI LLM GPT-4.

You can either launch Bing Chat from the Copilot icon in the top right-hand corner of Edge or log in with *Bing.com/chat*. Bing Chat Enterprise can also be used with Google Chrome.

Conclusions

Microsoft is significantly accelerating the spread of its AI services in the various applications. Windows Copilot lets you control the operating system directly in Windows with AI functions and lets you work with generative AI on the basis of OpenAI GPT-4 in the same way as with ChatGPT. Copilot is also available with Bing Chat in Edge and Chrome.

Bing Chat Enterprise is interesting for companies with a Microsoft 365 subscription because it protects company data. The control options for admins are currently still quite new and not very extensive. However, you can expect the management tools to become more powerful as the AI feature set grows. ■

Bing Chat Enterprise Is Not Microsoft 365 Copilot

Although the Microsoft 365 subscription plays an important role in Bing Chat Enterprise, it is not Microsoft 365 Copilot. Bing Chat Enterprise is a generative AI application that exclusively uses data from the Internet. The AI has no access to company resources or information from Microsoft 365. This access is added when you use Microsoft 365 Copilot. Questions that are answered in Microsoft 365 remain in the Microsoft 365 tenant domain.

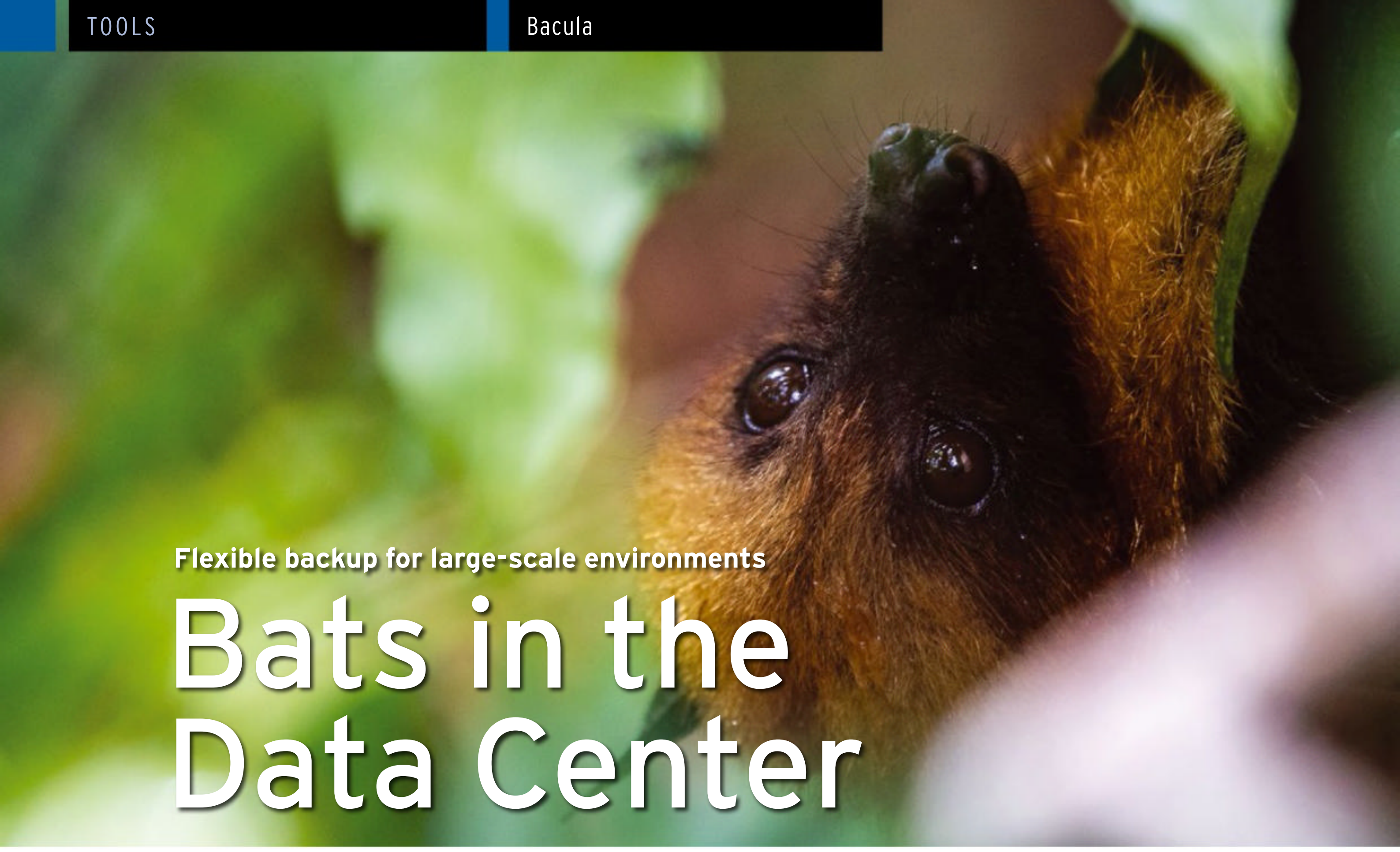
Important settings can be found on the *Configurations* tab in the *Settings/Search & intelligence* section of the Microsoft 365 Admin Center. The *Change Microsoft Search in Bing setting,* option under *Configurations*, lets you change how Bing Chat Enterprise is used in your company. Disabling this setting in the Microsoft 365 Admin Center means that users can no longer log in to Bing with their Azure AD/Entra ID accounts, preventing access to Bing Chat Enterprise.

Info

[1] Microsoft Edge download: [https://www.microsoft.com/en-us/edge/business/download?form=MA13FJ]

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].



Flexible backup for large-scale environments

Bats in the Data Center

Bacula is a powerful open source backup solution for large environments that offers automation, extension modules, and commercial support. By Harald Däubler

A network of universities (Universities of Ulm, Constance, and Tübingen, Germany) was looking for a new backup solution, but the requirements were tough. The new solution not only had to offer a simple and predictable license model, it also had to cope with billions of files and petabytes of data in all kinds of international character sets on thousands of computers with all kinds of operating systems at several locations in the country. Open and documented formats and interfaces were essential to ensure permanent, at least read-only, access to the data, and it had to allow the continued use of existing tape drives, including a large tape library. In the end, the people in charge opted for a proof of concept based on Bacula Enterprise Edition [1] by Switzerland's Bacula Systems [2] and, by doing so, for a combination of open source software with extension modules and commercial support.

Starting Point

The backup software originally used was IBM's Tivoli Storage Manager [3]. However, a review revealed that the license costs were difficult to calculate, with no central overview of how many CPUs were included in the backup in the individual decentralized systems in the branch offices and the data volume allowed for each CPU. Moreover, the figures could potentially change at any time. Inventorying the hardware data and constantly updating the figures looked like a very labor-intensive and time-consuming task, which is why license models by volume or processor performance are not a good fit for university operations.

The current hardware comprises two x86 servers running Solaris 11.4, which have access to three collections of hard drives (just a bunch of disks, JBODs). Each array comprises 90 SAS2 disks connected by several controllers, each with a capacity of 12TB. ZFS ensures the required filesystem redundancy. Each JBOD accommodates a zpool (one or more virtual devices, vdevs), which serves as a disk cache for one of the participating universities.

A tape library with more than 2,000 stations and eight IBM 3592-60 [4] tape drives completes the storage system. Each drive can write a tape with up to 20TB of uncompressed data. Hosts are connected to the tape drives over two redundant Fibre Channel paths at 32Gbps. Another storage server at the Tübingen site serves as a local backup-to-disk medium for selected systems, without a connection to the tape drives in Ulm. However, it would be easy to implement a link at a logical level with Bacula if required.

Bacula Architecture

The Bacula system comprises, among other things, what is known as a Director, which is responsible for controlling all the processes. The Director controls the tape library, initiates client backups, migrates data from disk to tape, and restores data.

To handle this task, the Director always needs to know what versions of which files have been saved by which clients and where the data is stored in the disk cache or on which tape it is stored at which location. This information is stored in a PostgreSQL database, which currently runs to 1.4TB and uses NVMe devices for performance reasons. Experience has shown that the database server also needs plenty of RAM – 512GB in this specific case.

Almost all the processes within the Bacula system (Figure 1) are driven by the Director and its configuration. The only exception to this is the backup encryption, which is handled by the clients themselves. All of the Bacula components can be configured with simple text files that contain full details of the objects and parameters.

The first category of objects is the client systems for which addresses, backup jobs, and file sets need to be defined. It makes sense to classify the systems by size and to pursue different backup strategies as a function of this attribute.

On some systems, too many files or too large a data volume prohibits a conventional backup. The initial backup of a mail system with more than 100 million files would take more than three weeks, in parallel to ongoing operations. Above all, however, it would take a similar amount of time to restore the system in an emergency – several days at least. These clients therefore require alternative strategies.

The next category of clients comprises systems that are so large or slow that a full backup would take several days, but on which an incremental backup can be completed in a single night. From experience, the limits of this category in the Ulm installation were defined as follows: The size of a full backup is between 500GB and 25TB, and an incremental backup takes less than eight hours. Experience has shown that such systems can be restored within three to five days. However, it is advisable to test this regularly, because the bottlenecks usually originate outside the backup system.

One of Bacula's features, known as Virtual Full backups, helps to backup such clients. After an initial backup, these systems never carry out full backups again – only daily incremental backups. Additionally, Full Virtual backups are performed every two months, which Bacula creates from the data available on the server without any intervention on the part of the client. The Full Virtual backups include the last (virtual) full backup along with the incremental or differential backups created by the client since then. A Virtual Full backup is seven times or more faster than a physical full backup.

This arrangement results in two scheduling classes. Large servers only create daily incremental backups of the client, a Virtual Full backup every two months, and a differential backup in between. On smaller systems, full monthly backups are created in addition to the daily incremental backups. All backup data, whether created by the client or by virtual backups, first ends up in the disk cache. Full backups older than one month or larger than 32GB are then migrated to tape. Incremental and differential backups, on the other hand, remain on the disks. This procedure reduces restore times, because a restore usually affects data on which work is being performed and is therefore available in the last incremental backups. The full backups needed for a complete restore can be read from tape at maximum speed, because they were written in a single piece and not fragmented during migration.

Configuration

As already mentioned, the Director controls all the processes as the central instance, which is why a large share of the configuration files resides there. The storage daemon is the server responsible for moving all data from the client to the disk cache, from the disk cache to tape, and vice versa. Client data is retrieved by the file daemon, an agent running on the client that the Director contacts.

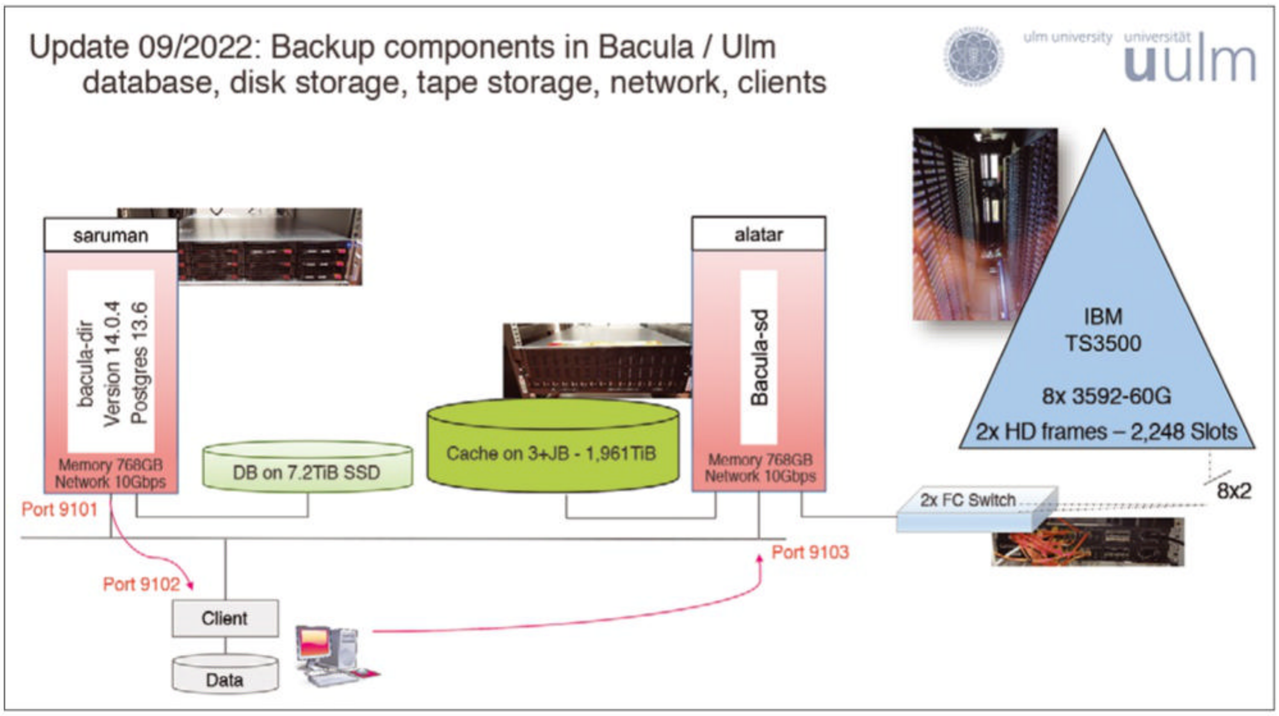


Figure 1: An overview of the Bacula installation on the Ulm backup network (SSD = solid state disk; FC = Fibre Channel). © H. Däubler/University of Ulm

The entire directory tree of the configuration could reside in /opt/bacula/etc/conf.d; for example:

```
# ls /opt/bacula/etc/conf.d/
clients.d          pools.d
filesets.d         schedules.d
jobs.d             storage.d
macos_excludes.inc unix_excludes.inc
messages.d         windows_excludes.inc
[...]
```

Listing 1: Client Configuration

```
Client {
  Name     = dbserv
  Address  = dbserv.rz.uni-ulm.de
  Password = "<XXXXXXXXXXXXXXXXXX>"
  @/opt/bacula/etc/conf.d/clients.d/client_defaults.inc
}
Job {
  Name     = dbserv-bck
  Client   = dbserv
  JobDefs  = kiz-job
  FileSet  = dbserv-fileset
  Schedule = daily-0500-sched
  Messages = dbserv-msg
  Client Run Before Job = "/pgsql/scripts/preschedule"
  Client Run After Job  = "/pgsql/scripts/postschedule"
}
Messages {
  Name     = dbserv-msg
  @/opt/bacula/etc/conf.d/messages.d/default_message.inc
  MailOnError = hostadm@uni-ulm.de = all, !skipped
}
FileSet {
  Name     = dbserv-fileset
  Ignore FileSet Changes = yes
  Include {
    Options {
      exclude  = yes
      @/opt/bacula/etc/conf.d/unix_excludes-exclude_
        cache_dir_too.inc
      WildFile = "/pgsql/log/*"
      WildFile = "/pgsql/logs/*"
      WildFile = "/pgsql/sw/logs/*"
      WildFile = "/pgsql/tmp/*"
    }
    Options {
      @/opt/bacula/etc/conf.d/filesets.d/default_
        fileset_options.inc
    }
    File = /pg_backup
    File = /pg_journal
    File = /pgsql
  }
  Exclude {
    @/opt/bacula/etc/conf.d/kiz_unix_file-excludes.inc
    File = /pgsql/sw/tapes/build
    File = /pgsql/data
    File = /pgsql/xlog
    File = /pgsql/tsm
  }
}
```

The object definitions can be clearly divided into clients, jobs, data pools, backup schedules, and so on. These categories can be further subdivided, for example, by grouping the clients by organizational unit. There are few limits to what you can do.

In the client definition, you can specify the properties of the associated backup job, the schedule on which it runs, and whether or not scripts are executed on the client before and after the backup. You can also specify which files are backed up and which are excluded from the backup. The example in Listing 1 shows the definition for backing up a PostgreSQL server, where Bacula leaves out the database files and simply retrieves database dump files.

The counterpart to this configuration file on the Director is the file daemon configuration on the client side (Listing 2). It must contain the correct password and contains logging specifications. Encryption can also be configured here, if required. The Director has no access to the data or the keys; however, metadata, such as file names, are available in the clear, because this information is essential for the system to work.

Restore

The very purpose of backups is to restore data – be it individual files that have been accidentally deleted or entire filesystems that have been lost because of a system failure or a ransomware attack.

To initiate a restore in the Bacula system, you must have access to the Director. Either an administrator starts the restore after a user has posted a corresponding request to the helpdesk, or the client offers the option of accessing the Director with the Bacula Console bconsole utility and running the required commands there. Access control lists (ACLs) can be used to define which commands are accessible from the Console.

In this context, it is advisable to prevent clients from initiating backups and only allow restores (e.g., to prevent the backups stored on the server being overwritten by uncontrolled backups that could be initiated by a malware-infected system).

Reporting

One of Bacula's great strengths is its reporting. In addition to predefined reports, further options are available thanks to documented database structures. You can also use bdirjson for read access to the active configuration or run scripts from bconsole. The SQL query from Listing 3, embedded in your choice of scripting language, provides an overview of the clients that require the most space, all told, and optionally match a name pattern.

A self-developed tool uses queries to detect anomalies. One of the most annoying problems occurs when external devices or filesystems are to be backed up on the client but are not mounted at the time of the backup.

Listing 2: File Daemon Configuration

```
FileDaemon {
  Name                   = "dbserv"
  FDAddress              = dbserv-fl-m.rz.uni-ulm.de
  FDport                 = 9102
  WorkingDirectory       = /opt/bacula/working
  Pid Directory          = /opt/bacula/working
  Maximum Concurrent Jobs = 4
}
Director {
  Name     = uniulm-dir
  Password = "<XXXXXXXXXXXXXXXXXX>"
}
Messages {
  Name     = Standard
  director = uniulm-dir = all, !skipped, !restored
}
```

Listing 3: SQL Query

```
SELECT
  Client.Name,
  sum(JobFiles) AS Files,
  sum(JobBytes) AS Bytes
FROM
  Job, Client, Pool
WHERE
      Client.Name ~* '$pattern'
  AND Client.ClientId = Job.ClientId
  AND Job.PoolID = Pool.PoolID
  AND Job.Type = 'B'
  AND JobStatus IN ('T', 'W')
GROUP by Client.Name
ORDER by Bytes DESC
LIMIT $number;
```

SFSCON FREE SOFTWARE CONFERENCE

NOI TECHPARK SÜDTIROL / ALTO ADIGE

NOV 8TH 9TH

2024

BE PART OF IT!

2 INSPIRING DAYS
1 FESTIVAL
100 SPEAKERS

WWW.SFSCON.IT

NOI TECHPARK
SÜDTIROL / ALTO ADIGE

GRUPPO FOS soluzioni ad alta tecnologia · TELMEKOM NETWORKS · SYMPHONIE PRIME · VATES Open infrastructure made simple · MADE IN CIMA · Zirkonzahn Human Zirconium Technology

1006.org · {catchsolve} · Christian Gapp · endian · ecosteer. · OBUS · peer · (studio hug)

NOI TECHPARK
SÜDTIROL / ALTO ADIGE

A.-VOLTA-STRASSE 13A, BOZEN
VIA A. VOLTA 13A, BOLZANO

VISIT US ON
WWW.NOI.BZ.IT

Listing 4: Early Detection

```
Job   Level Jobs    Avg.Time   Avg.Files  Avg.Bytes
=================================================
job-1     I   32    02:40:51     343,351   351.6 GB
job-1     F    1  4-18:50:41  30,734,592    34.2 TB
job-1    VF    1  1-05:54:54  28,946,254    35.2 TB

(C001) latest/average full-backup size is above acceptable limit of 25TB
(C101) latest/average VFull-backup size is above acceptable limit of 25TB
(W102) last VFull runtime is longer then acceptable limit of 22h
(C301) average incremental-backup size is above acceptable limit of 200GB
(W302) at least one incremental-backup size is above threshold of 200GB
```

Bacula interprets this as a deletion of the data and marks the data as deleted in the database at this point in time. If the data can be accessed again during the next backup on the client, they are all backed up again. This operation can take a long time and generates a noticeable load on the client.

You can counteract such problems with appropriate plausibility checks. For example, if more than 5,000 files have been deleted and less than 500KB of data has been backed up, something is probably wrong. Incidentally, the problem can be easily solved by early detection. The backup job that marked the files as deleted is itself deleted and the previous state is restored.

This test protects the clients, and others prevent unnecessary resource consumption and potential issues at restore time. Occasionally, the files belonging to a database or the file used by a virtual machine (VM) as a virtual hard drive are inadvertently backed up. Depending on the database management system (DBMS) or VM load, both can change considerably over the duration of the backup job and will therefore be inconsistent.

Also important is for the system operation to detect at an early stage whether or not systems exceed certain thresholds. Listing 4 shows an example of the output from this kind of check.

As already mentioned, further information can be obtained with the Bacula text-based console (`bconsole`). A little-known `.status storage running` command reveals the details of the data throughput for individual jobs; this command is particularly useful here by doing something that otherwise can only be done on an aggregated basis at the system level. The data can be combined and visualized with the other mechanisms (Figures 2 and 3).

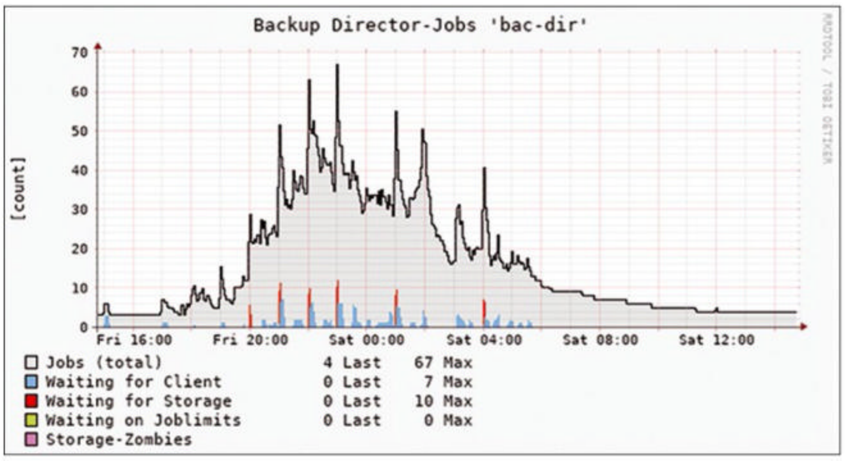


Figure 2: The number of backup processes and their status over time.

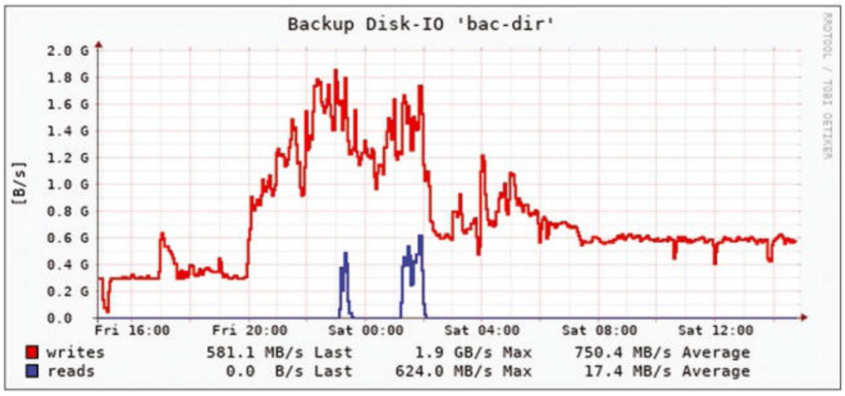


Figure 3: A visualization of read and write throughput.

Conclusions

Bacula proves to be a powerful backup solution that can efficiently back up large environments. Because it is open source, the formats and interfaces are open. Among other things, this means that, unlike proprietary backup software, the backed-up tapes can be read at any time, even if the license agreement is terminated at some point. Bacula also offers many options for automating processes and retrieving all kinds of useful information. When deciding on a new backup solution, you will definitely want to put Bacula on your shortlist. ■

Info

[1] Bacula Enterprise: [https://www.baculasystems.com/corporate-data-backup-software-solutions/bacula-enterprise-data-backup-software/]

[2] Bacula Systems: [https://www.baculasystems.com]

[3] IBM Tivoli Storage Manager: [https://www.ibm.com/docs/en/tsm/7.1.0?topic=servers-tivoli-storage-manager-overview]

[4] 3592 tape drives: [https://www.ibm.com/docs/en/ts3500-tape-library?topic=drives-3592-tape]

■

Supercharge your software upgrade routine

Top of the Line

Topgrade steps forward as the hero to help you keep an upper hand over updating and upgrading your stable of Linux applications. By Daniel LaSalle

Ever since the last millennium, the most common methods of deploying applications revolved around compiling from source or dealing with each respective distributions' package manager. Terminal front ends then helped with the wider adoption of Linux by making software manipulation much easier for a range of users. Next, AppImage, Flatpak, and Snap stepped in, with literally no pause between their appearance and endorsement by mainstream distributions. Don't forget about self-updating third-party libraries from standalone processes or about all the various firmware incarnations that exist!

Even though it's always possible to automate these processes (e.g., with `anacron`), in reality, ain't nobody got time for that.

Your security posture begins by religiously applying software patches; however, keeping an upper hand over the update and upgrade process on your stable of applications, not to mention the many attack vectors this portability brings, can be a hair-pulling task. Thankfully, Topgrade comes to the rescue when it comes to running the latest and greatest.

Topgrade

In 2018 the Topgrade project debuted on GitHub [1], and before long it formed a very vibrant and active community. The `topgrade` tool is the terminal front end of all terminal front ends, englobing all of the terminal installation wizards you can imagine. Sadly, development came to a screeching halt in 2022. With an astonishing 220+ contributors having already pitched in to make this tool great, it was clear that it would leave a gigantic void, so somebody had to take over its life cycle and ensure that it would carry on its legacy of greatness.

Shortly after the announcement of Topgrade's demise, the mini-juggernaut was forked, and its development indeed continued under a different leadership. Behold: `topgrade-rs` [2] was born.

At the time of writing, the latest version was 14.0.1, and the number of contributors that rallied under this new fork exceeded 350. Those numbers should give you a strong opinion about such a new tool and how much it is really loved by its users.

Even with that kind of community behind it, you couldn't install either `topgrade` or `topgrade-rs` with Debian's Apt package manager or through official repositories. The most noble distribution that had picked up on this nifty tool was openSUSE [3]. Today, you can install it either with the usual `git clone` command or by going through a somewhat easier approach of a system-wide install with `cargo`, which is a part of the Rust *crates.io* package registry [4].

Topgrade Everything

For a system-level installation of `topgrade-rs` (the example in this article uses a freshly installed Ubuntu MATE 22.04.3 platform), first install all the requirements for compiling and deploying `topgrade-rs` (referred to as Topgrade moving forward): `curl`, `git`, `pkg-config`, and `rust`. Once in place, append the `$HOME/.cargo/env` absolute path to your already existing `$PATH` variable; then, install `cargo`. Listing 1

Photo by redcharlie on Unsplash

shows the download and installation of the official compiler for the Rust programming language and its package manager, Cargo.
Once that step is complete, you can then begin getting acquainted with this tool. As mentioned earlier, you can also chose to install Topgrade by pulling its GitHub repository and launching the process with

```
build-all.sh
```

from the root of the repo.

Before moving in with all of the previous steps involving your system-wide installation method with `cargo`, allow me to share a bit of advice. From my personal experience on deploying Topgrade on several Debian systems and derivatives over the course of the

Listing 1: Deploying Topgrade (Extract)

```
$ sudo apt install curl git pkg-config -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui
    gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  curl git pkg-config
0 upgraded, 3 newly installed, 0 to remove and 1 not upgraded.
Need to get 3,409 kB of archives.
After this operation, 19.5 MB of additional disk space will be used.
Get:1 http://gb.archive.ubuntu.com/ubuntu jammy-updates/main amd64
    curl amd64 7.81.0-1ubuntu1.15 [194 kB]
[...]
Setting up pkg-config (0.29.2-1ubuntu3) ...
Setting up git (1:2.34.1-1ubuntu1.10) ...
Setting up curl (7.81.0-1ubuntu1.15) ...
Processing triggers for man-db (2.10.2-1) ...
$
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
info: downloading installer

Welcome to Rust!

This will download and install the official compiler for the Rust
programming language, and its package manager, Cargo.

Rustup metadata and toolchains will be installed into the Rustup
home directory, located at:

/home/Dan/.rustup

This can be modified with the RUSTUP_HOME environment variable.

The Cargo home directory is located at:

/home/Dan/.cargo

This can be modified with the CARGO_HOME environment variable.

The cargo, rustc, rustup and other commands will be added to
Cargo's bin directory, located at:

/home/Dan/.cargo/bin

This path will then be added to your PATH environment variable by
modifying the profile file located at:

/home/Dan/.profile

You can uninstall at any time with rustup self uninstall and
these changes will be reverted.

Current installation options:

default host triple: x86_64-unknown-linux-gnu
default toolchain: stable (default)
profile: default
modify PATH variable: yes

1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
>1

info: profile set to 'default'
info: default host triple is x86_64-unknown-linux-gnu
[...]

stable-x86_64-unknown-linux-gnu installed - rustc 1.76.0 (07dca489a
  2024-02-04)

Rust is installed now. Great!

To get started, you may need to restart your current shell.
This would reload your PATH environment variable to include
Cargo's bin directory ($HOME/.cargo/bin).

To configure your current shell, run:
source "$HOME/.cargo/env"
$
$ source "$HOME/.cargo/env"
$
$ cargo install cargo-update
Updating crates.io index
Installing cargo-update v14.0.1
Updating crates.io index
Compiling libc v0.2.153
Compiling cc v1.0.86
[...]
Finished release [optimized] target(s) in 10m 27s
Replacing /home/Dan/.cargo/bin/cargo-install-update
Replacing /home/Dan/.cargo/bin/cargo-install-update-config
Replaced package `cargo-update v14.0.1` with `cargo-update
  v14.0.1` (executables `cargo-install-update`,
  `cargo-install-update-config`)
$ cargo install topgrade
Updating crates.io index
Downloaded topgrade v14.0.1
Downloaded 1 crate (4.4 MB) in 1.52s
Installing topgrade v14.0.1
Updating crates.io index
Downloaded async-io v1.13.0
Downloaded async-fs v1.6.0
[...]
Compiling proc-macro2 v1.0.78
Compiling unicode-ident v1.0.12
[...]
Finished release [optimized] target(s) in 15m 31s
Installing /home/Dan/.cargo/bin/topgrade
Installed package `topgrade v14.0.1` (executable `topgrade`)
$
```

past year, sometimes the Rust library is already installed yet can cause grief when proceeding with the regular installation of Topgrade. In that respect, I suggest purging everything related to Rust and reinstalling the latest from their official installation website [5] before attempting what follows.

Let the Upgrade Games Begin

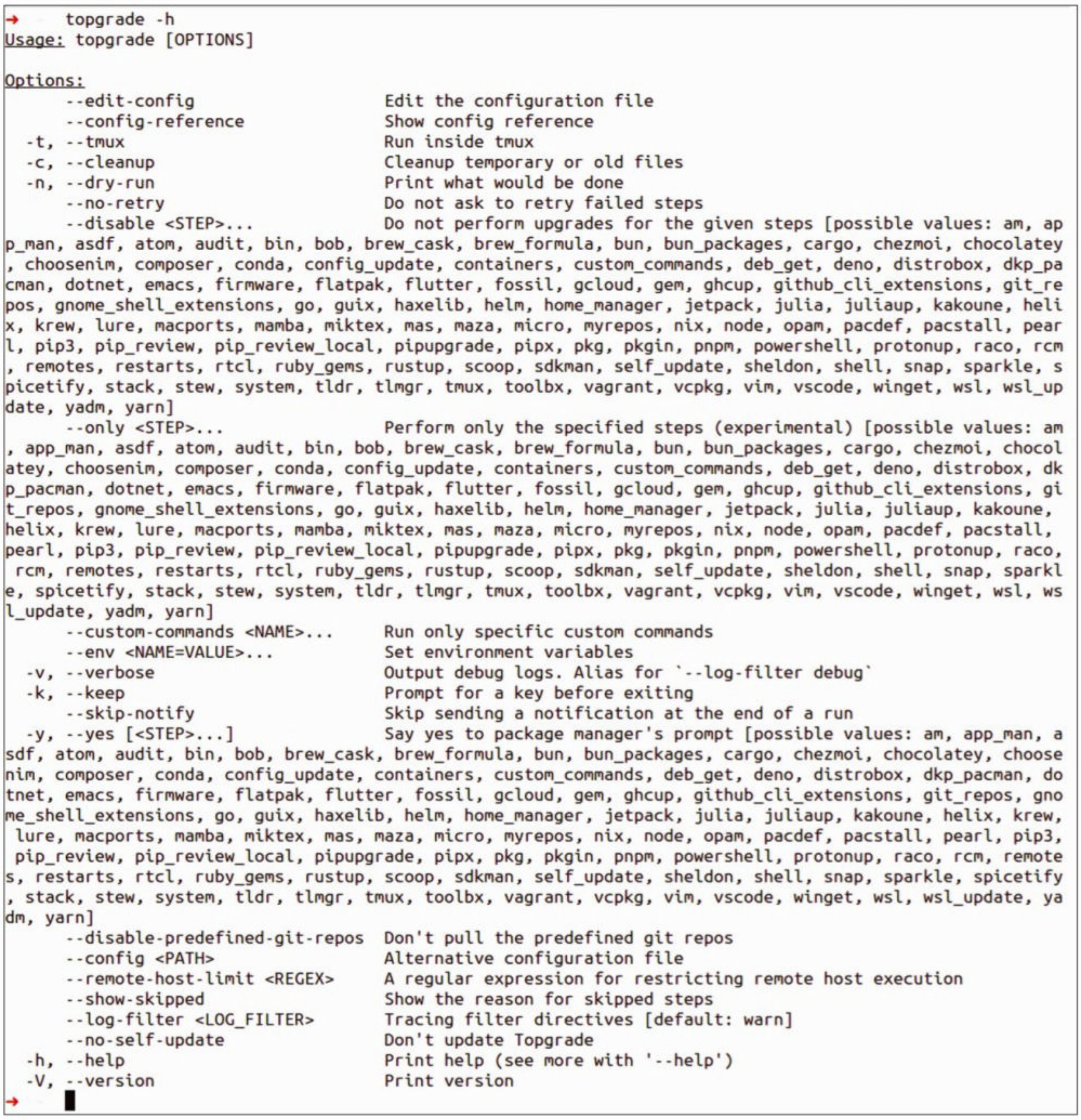
All of the previous steps take about five minutes on a modern system. Once all the building and installing is done successfully, you can confirm that the binary was installed successfully by issuing the `which topgrade` command. This should then result in showing the path into which the binary was installed (e.g., `~/.cargo/bin/topgrade`):

```
$ which topgrade
/home/Dan/.cargo/bin/topgrade
$
```

Another perk I deploy to keep on top of this never ending patching game is:

```
alias u="/home/Dan/.cargo/bin/topgrade -y"
```

in either my `.bashrc` or `.zshrc` file. Thereafter, begin your exploration journey of this new tool by issuing `topgrade -h` (Figure 1). The other switches you will mostly be using are `-y` (or `--yes`) to acknowledge automatically any questions that might arise (Figure 2), `-c` (or `--cleanup`) to purge any and all temporary and old files left by Topgrade, and perhaps `--only`, a still experimental switch as of the



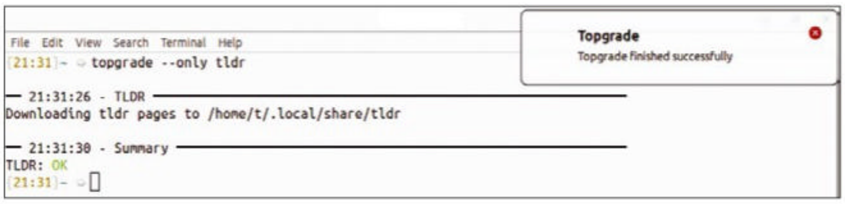
```
➜    topgrade -h
Usage: topgrade [OPTIONS]

Options:
      --edit-config                    Edit the configuration file
      --config-reference               Show config reference
  -t, --tmux                           Run inside tmux
  -c, --cleanup                        Cleanup temporary or old files
  -n, --dry-run                        Print what would be done
      --no-retry                       Do not ask to retry failed steps
      --disable <STEP>...              Do not perform upgrades for the given steps [possible values: am, ap
p_man, asdf, atom, audit, bin, bob, brew_cask, brew_formula, bun, bun_packages, cargo, chezmoi, chocolatey
, choosenim, composer, conda, config_update, containers, custom_commands, deb_get, deno, distrobox, dkp_pa
cman, dotnet, emacs, firmware, flatpak, flutter, fossil, gcloud, gem, ghcup, github_cli_extensions, git_re
pos, gnome_shell_extensions, go, guix, haxelib, helm, home_manager, jetpack, julia, juliaup, kakoune, heli
x, krew, lure, macports, mamba, miktex, mas, maza, micro, myrepos, nix, node, opam, pacdef, pacstall, pear
l, pip3, pip_review, pip_review_local, pipupgrade, pipx, pkg, pkgin, pnpm, powershell, protonup, raco, rcm
, remotes, restarts, rtcl, ruby_gems, rustup, scoop, sdkman, self_update, sheldon, shell, snap, sparkle, s
picetify, stack, stew, system, tldr, tlmgr, tmux, toolbx, vagrant, vcpkg, vim, vscode, winget, wsl, wsl_up
date, yadm, yarn]
      --only <STEP>...                 Perform only the specified steps (experimental) [possible values: am
, app_man, asdf, atom, audit, bin, bob, brew_cask, brew_formula, bun, bun_packages, cargo, chezmoi, chocol
atey, choosenim, composer, conda, config_update, containers, custom_commands, deb_get, deno, distrobox, dk
p_pacman, dotnet, emacs, firmware, flatpak, flutter, fossil, gcloud, gem, ghcup, github_cli_extensions, gi
t_repos, gnome_shell_extensions, go, guix, haxelib, helm, home_manager, jetpack, julia, juliaup, kakoune,
helix, krew, lure, macports, mamba, miktex, mas, maza, micro, myrepos, nix, node, opam, pacdef, pacstall,
pearl, pip3, pip_review, pip_review_local, pipupgrade, pipx, pkg, pkgin, pnpm, powershell, protonup, raco,
 rcm, remotes, restarts, rtcl, ruby_gems, rustup, scoop, sdkman, self_update, sheldon, shell, snap, sparkl
e, spicetify, stack, stew, system, tldr, tlmgr, tmux, toolbx, vagrant, vcpkg, vim, vscode, winget, wsl, ws
l_update, yadm, yarn]
      --custom-commands <NAME>...      Run only specific custom commands
      --env <NAME=VALUE>...            Set environment variables
  -v, --verbose                        Output debug logs. Alias for `--log-filter debug`
  -k, --keep                           Prompt for a key before exiting
      --skip-notify                    Skip sending a notification at the end of a run
  -y, --yes [<STEP>...]                Say yes to package manager's prompt [possible values: am, app_man, a
sdf, atom, audit, bin, bob, brew_cask, brew_formula, bun, bun_packages, cargo, chezmoi, chocolatey, choose
nim, composer, conda, config_update, containers, custom_commands, deb_get, deno, distrobox, dkp_pacman, do
tnet, emacs, firmware, flatpak, flutter, fossil, gcloud, gem, ghcup, github_cli_extensions, git_repos, gno
me_shell_extensions, go, guix, haxelib, helm, home_manager, jetpack, julia, juliaup, kakoune, helix, krew,
 lure, macports, mamba, miktex, mas, maza, micro, myrepos, nix, node, opam, pacdef, pacstall, pearl, pip3,
 pip_review, pip_review_local, pipupgrade, pipx, pkg, pkgin, pnpm, powershell, protonup, raco, rcm, remote
s, restarts, rtcl, ruby_gems, rustup, scoop, sdkman, self_update, sheldon, shell, snap, sparkle, spicetify
, stack, stew, system, tldr, tlmgr, tmux, toolbx, vagrant, vcpkg, vim, vscode, winget, wsl, wsl_update, ya
dm, yarn]
      --disable-predefined-git-repos   Don't pull the predefined git repos
      --config <PATH>                  Alternative configuration file
      --remote-host-limit <REGEX>      A regular expression for restricting remote host execution
      --show-skipped                   Show the reason for skipped steps
      --log-filter <LOG_FILTER>        Tracing filter directives [default: warn]
      --no-self-update                 Don't update Topgrade
  -h, --help                           Print help (see more with '--help')
  -V, --version                        Print version
➜    █
```

Figure 1: Topgrade's initial offering.

current version, that allows you to go through certain modules selectively (Figure 2 shows an upgrade of only the `tldr` portion) instead of going through the whole routine of upgrading everything (Figure 3). Do not forget the `-t` (or `--tmux`) switch, which facilitates `tmux` integration.

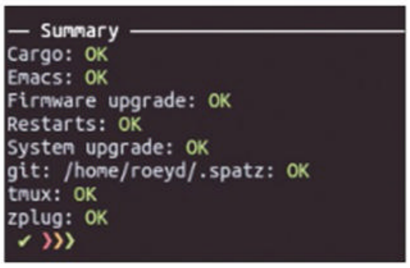
Configuration File

Your next action will be to access the config file located under `~/.config/topgrade.toml` (Figure 5). For this, you can either open it in your favorite editor or with the built-in `--edit-config` switch, which will take you directly to edit mode. For those not in the know, the `.toml` format flex is that it is user-friendly and already has been implemented in more than 40 programming languages [6].


File Edit View Search Terminal Help
Topgrade
Topgrade finished successfully

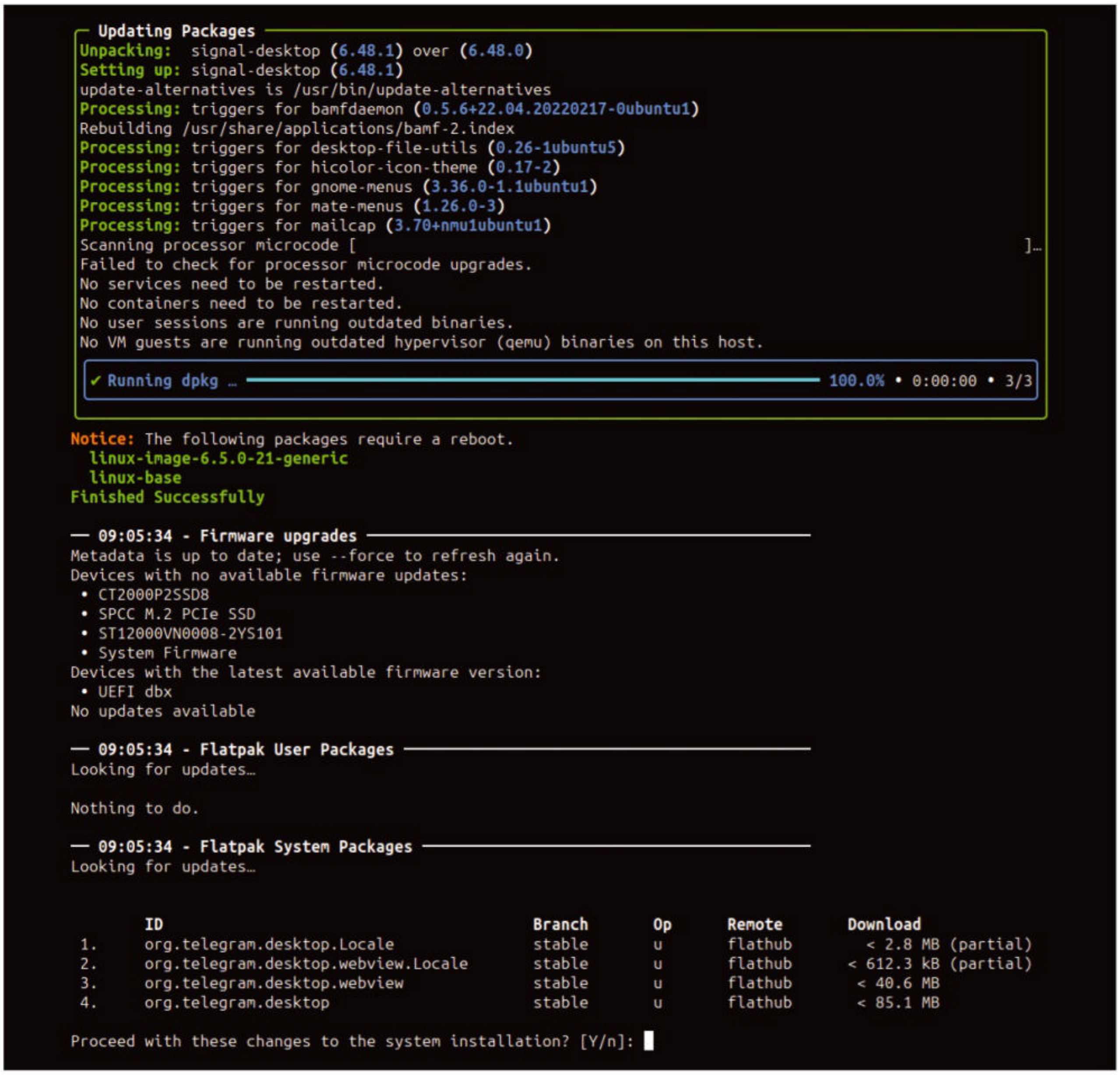
```
[21:31]~ ○ topgrade --only tldr

— 21:31:26 - TLDR ————————————————————————————
Downloading tldr pages to /home/t/.local/share/tldr

— 21:31:30 - Summary ——————————————————————————
TLDR: OK
[21:31]~ ○
```

Figure 2: Output for the `--only` switch.



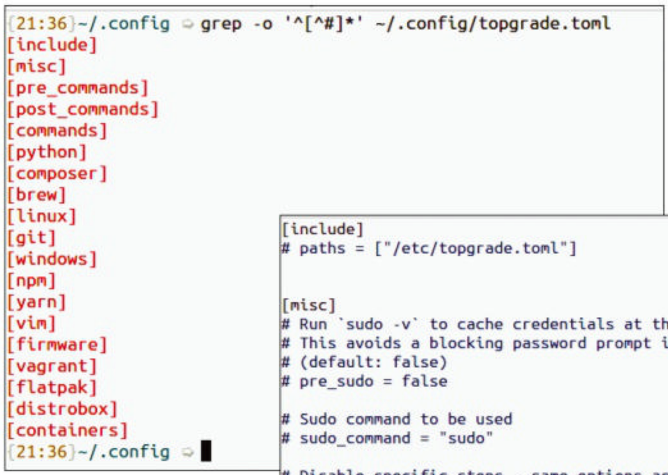
```
— Summary ——————————————————
Cargo: OK
Emacs: OK
Firmware upgrade: OK
Restarts: OK
System upgrade: OK
git: /home/roeyd/.spatz: OK
tmux: OK
zplug: OK
✔ ❯❯❯
```

Figure 4: When not enforcing the `--only` switch, Topgrade covers any outdated software. © Courtesy r-darwish GitHub page



```
┌ Updating Packages ─────────────────────────────────────────────────────────────
│Unpacking:  signal-desktop (6.48.1) over (6.48.0)
│Setting up: signal-desktop (6.48.1)
│update-alternatives is /usr/bin/update-alternatives
│Processing: triggers for bamfdaemon (0.5.6+22.04.20220217-0ubuntu1)
│Rebuilding /usr/share/applications/bamf-2.index
│Processing: triggers for desktop-file-utils (0.26-1ubuntu5)
│Processing: triggers for hicolor-icon-theme (0.17-2)
│Processing: triggers for gnome-menus (3.36.0-1.1ubuntu1)
│Processing: triggers for mate-menus (1.26.0-3)
│Processing: triggers for mailcap (3.70+nmu1ubuntu1)
│Scanning processor microcode [                                                    ]…
│Failed to check for processor microcode upgrades.
│No services need to be restarted.
│No containers need to be restarted.
│No user sessions are running outdated binaries.
│No VM guests are running outdated hypervisor (qemu) binaries on this host.
│ ┌──────────────────────────────────────────────────────────────────────────────┐
│ │✔ Running dpkg … ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100.0% • 0:00:00 • 3/3│
│ └──────────────────────────────────────────────────────────────────────────────┘
└────────────────────────────────────────────────────────────────────────────────

Notice: The following packages require a reboot.
  linux-image-6.5.0-21-generic
  linux-base
Finished Successfully

— 09:05:34 - Firmware upgrades ———————————————————————————
Metadata is up to date; use --force to refresh again.
Devices with no available firmware updates:
 • CT2000P2SSD8
 • SPCC M.2 PCIe SSD
 • ST12000VN0008-2YS101
 • System Firmware
Devices with the latest available firmware version:
 • UEFI dbx
No updates available

— 09:05:34 - Flatpak User Packages ———————————————————————
Looking for updates…

Nothing to do.

— 09:05:34 - Flatpak System Packages —————————————————————
Looking for updates…


        ID                                    Branch     Op     Remote     Download
 1.     org.telegram.desktop.Locale           stable     u      flathub      < 2.8 MB (partial)
 2.     org.telegram.desktop.webview.Locale   stable     u      flathub    < 612.3 kB (partial)
 3.     org.telegram.desktop.webview          stable     u      flathub     < 40.6 MB
 4.     org.telegram.desktop                  stable     u      flathub     < 85.1 MB

Proceed with these changes to the system installation? [Y/n]:
```

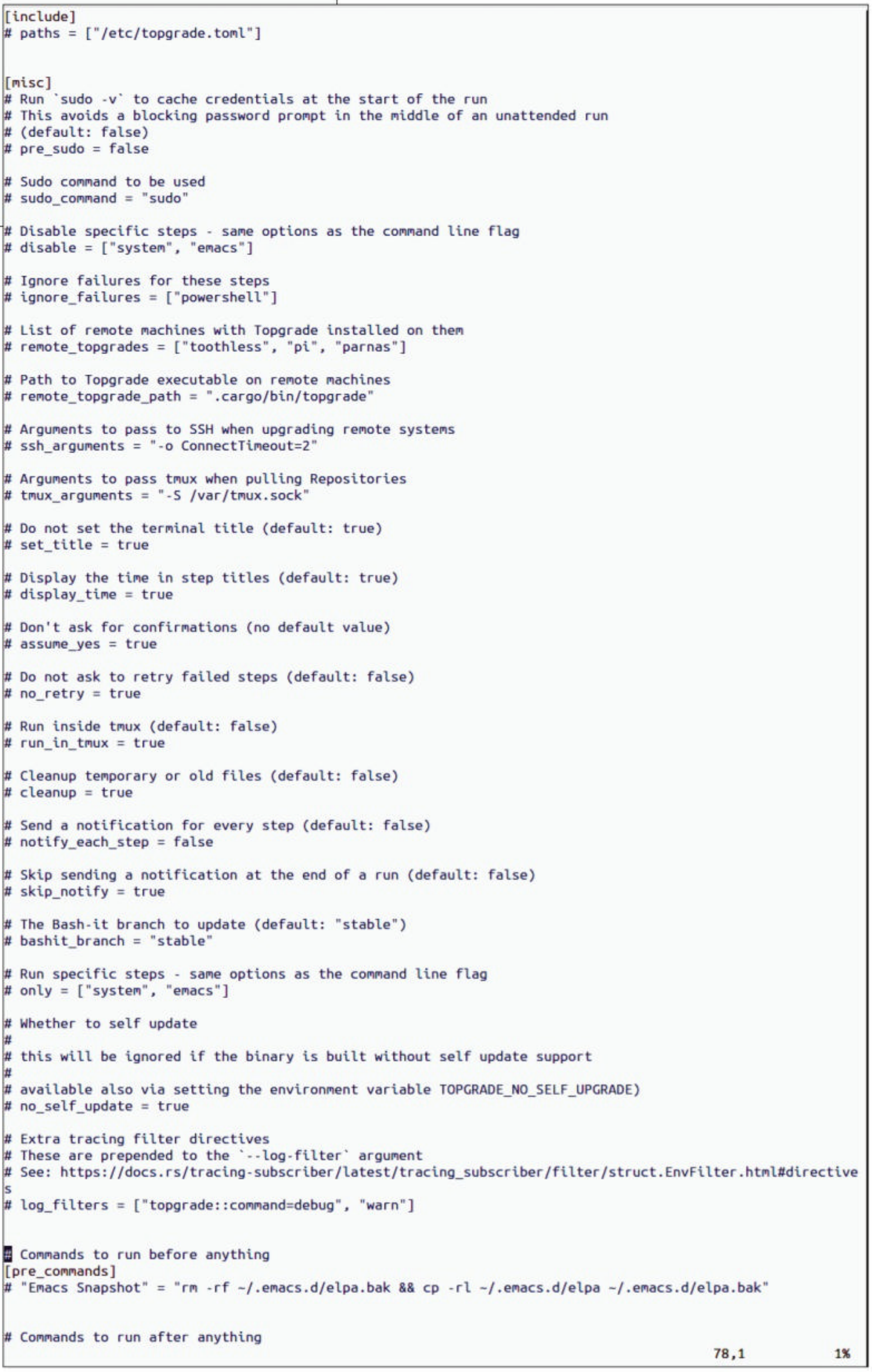
Figure 3: Failure to provide `-y` while executing `topgrade` will result in users having to accept or deny the forthcoming changes manually.



```
{21:36}~/.config ➾ grep -o '^[^#]*' ~/.config/topgrade.toml
[include]
[misc]
[pre_commands]
[post_commands]
[commands]
[python]
[composer]
[brew]
[linux]
[git]
[windows]
[npm]
[yarn]
[vim]
[firmware]
[vagrant]
[flatpak]
[distrobox]
[containers]
{21:36}~/.config ➾ █
```

Figure 5: By default, the topgrade.toml config file contains all these uncommented entries.

From the configuration file, notice that the rule of thumb for most supported software is to offer `sudo` execution. Another interesting point is that Topgrade is cross-platform; therefore, it also offers customization options for other operating systems that are not POSIX compliant. That said, I have only run Topgrade on Linux systems with the defaults, without bothering much about changing the content of the config file. The results have been quite lucrative: Topgrade caught many updates that were not explicitly listed anywhere in the `.toml` file. Such is true notably for `cargo`, `oh-my-zsh`, and `tldr`. Although the `[misc]` section possibly contains the most options (Figure 6), the `[linux]` group also has a number of options that fall into the category from all the popular choices (Figure 7).



```
[include]
# paths = ["/etc/topgrade.toml"]


[misc]
# Run `sudo -v` to cache credentials at the start of the run
# This avoids a blocking password prompt in the middle of an unattended run
# (default: false)
# pre_sudo = false

# Sudo command to be used
# sudo_command = "sudo"

# Disable specific steps - same options as the command line flag
# disable = ["system", "emacs"]

# Ignore failures for these steps
# ignore_failures = ["powershell"]

# List of remote machines with Topgrade installed on them
# remote_topgrades = ["toothless", "pi", "parnas"]

# Path to Topgrade executable on remote machines
# remote_topgrade_path = ".cargo/bin/topgrade"

# Arguments to pass to SSH when upgrading remote systems
# ssh_arguments = "-o ConnectTimeout=2"

# Arguments to pass tmux when pulling Repositories
# tmux_arguments = "-S /var/tmux.sock"

# Do not set the terminal title (default: true)
# set_title = true

# Display the time in step titles (default: true)
# display_time = true

# Don't ask for confirmations (no default value)
# assume_yes = true

# Do not ask to retry failed steps (default: false)
# no_retry = true

# Run inside tmux (default: false)
# run_in_tmux = true

# Cleanup temporary or old files (default: false)
# cleanup = true

# Send a notification for every step (default: false)
# notify_each_step = false

# Skip sending a notification at the end of a run (default: false)
# skip_notify = true

# The Bash-it branch to update (default: "stable")
# bashit_branch = "stable"

# Run specific steps - same options as the command line flag
# only = ["system", "emacs"]

# Whether to self update
#
# this will be ignored if the binary is built without self update support
#
# available also via setting the environment variable TOPGRADE_NO_SELF_UPGRADE)
# no_self_update = true

# Extra tracing filter directives
# These are prepended to the `--log-filter` argument
# See: https://docs.rs/tracing-subscriber/latest/tracing_subscriber/filter/struct.EnvFilter.html#directive
s
# log_filters = ["topgrade::command=debug", "warn"]


# Commands to run before anything
[pre_commands]
# "Emacs Snapshot" = "rm -rf ~/.emacs.d/elpa.bak && cp -rl ~/.emacs.d/elpa ~/.emacs.d/elpa.bak"


# Commands to run after anything
                                                                        78,1          1%
```

Figure 6: The `[misc]` stanza contains the most options.

Troubleshooting as a Job

No matter how good your intentions, sooner or later something will go haywire, and when it does, Topgrade gives you the immediate opportunity to fix the sticky situation by prompting you to open a shell

(Figure 8). As stated before, because I have been using this front end of front ends for many months now, the problem I most commonly face is (1) when tunneling Topgrade over SSH without invoking a shell first or (2) when I have a `cargo` version mismatch.

For the first case, if issuing something like

```
ssh IP ⮐
  -t '/home/Dan/.cargo/bin/topgrade -y'
```

when a shell-dependant update is pending (e.g., an `oh-my-zsh` update), then the topgrade process will fail. In the second case of a `cargo` version mismatch, running

```
cargo install cargo-update
```

from a rescue shell has always managed to fix my problem. After completing these minor sessions of troubleshooting, exiting and coming back to the utility always prove to be successful until the process is completed.

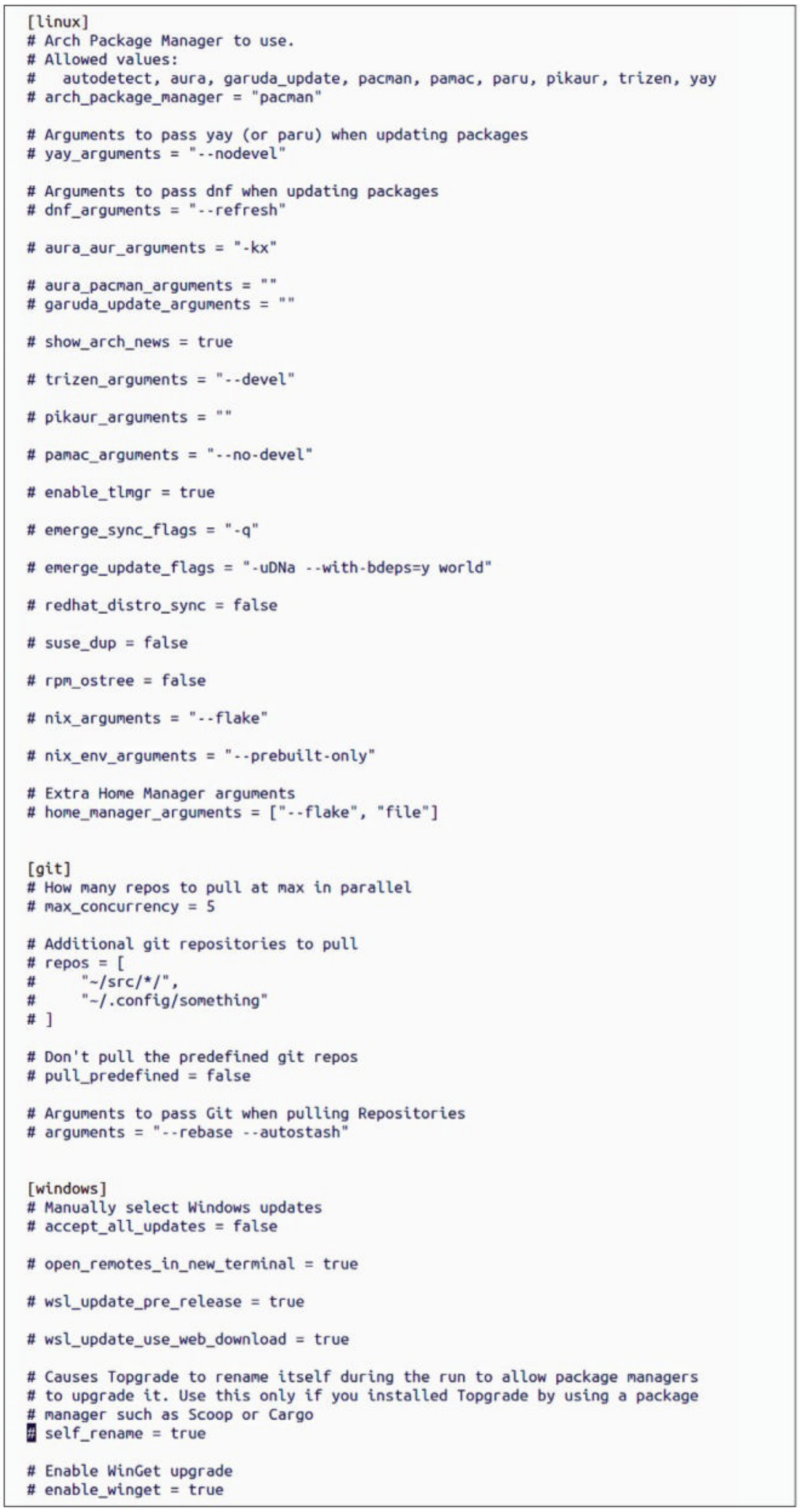
Pushing the Upgrade Game

Because of its nature, Topgrade will only be as good as the tools with which it is coupled. In that respect, and in the case of Debian and its derivatives, `apt` and its menu-driven `aptitude` sibling have been handling the job quite well for the last quarter century. Nevertheless, you still have room for improvement because even anything that has been in place that long should sooner or later be facing some sort of obsolence..

If you want to further enhance the toolset, the next level after deploying Topgrade would be to install something like Nala [7], a *libapt-pkg* front end (Figure 7), which, in turn, is much faster than `apt`. The tool offers not only a neater interface than the predecessors, but also parallel package downloads, the ability to select the fastest mirrors, and a history of package transactions. Although this step is optional, it will forever change your updating mindset.

Conclusion

Relying on Flatpak and Snap applications has never been fun from the get-go; however, you can trust

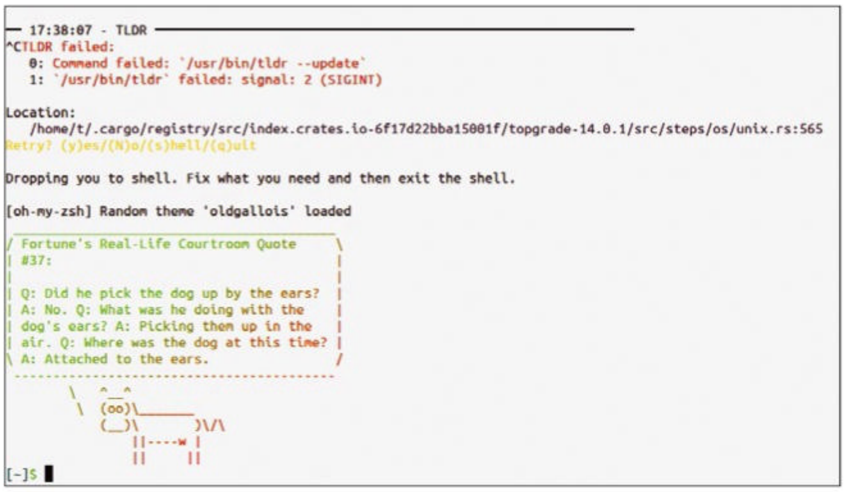


```
[linux]
# Arch Package Manager to use.
# Allowed values:
#   autodetect, aura, garuda_update, pacman, pamac, paru, pikaur, trizen, yay
# arch_package_manager = "pacman"

# Arguments to pass yay (or paru) when updating packages
# yay_arguments = "--nodevel"

# Arguments to pass dnf when updating packages
# dnf_arguments = "--refresh"

# aura_aur_arguments = "-kx"

# aura_pacman_arguments = ""
# garuda_update_arguments = ""

# show_arch_news = true

# trizen_arguments = "--devel"

# pikaur_arguments = ""

# pamac_arguments = "--no-devel"

# enable_tlmgr = true

# emerge_sync_flags = "-q"

# emerge_update_flags = "-uDNa --with-bdeps=y world"

# redhat_distro_sync = false

# suse_dup = false

# rpm_ostree = false

# nix_arguments = "--flake"

# nix_env_arguments = "--prebuilt-only"

# Extra Home Manager arguments
# home_manager_arguments = ["--flake", "file"]


[git]
# How many repos to pull at max in parallel
# max_concurrency = 5

# Additional git repositories to pull
# repos = [
#     "~/src/*/",
#     "~/.config/something"
# ]

# Don't pull the predefined git repos
# pull_predefined = false

# Arguments to pass Git when pulling Repositories
# arguments = "--rebase --autostash"


[windows]
# Manually select Windows updates
# accept_all_updates = false

# open_remotes_in_new_terminal = true

# wsl_update_pre_release = true

# wsl_update_use_web_download = true

# Causes Topgrade to rename itself during the run to allow package managers
# to upgrade it. Use this only if you installed Topgrade by using a package
# manager such as Scoop or Cargo
# self_rename = true

# Enable WinGet upgrade
# enable_winget = true
```

Figure 7: The `[linux]` stanza contains some options for Arch, Red Hat, and SUSE, among others.

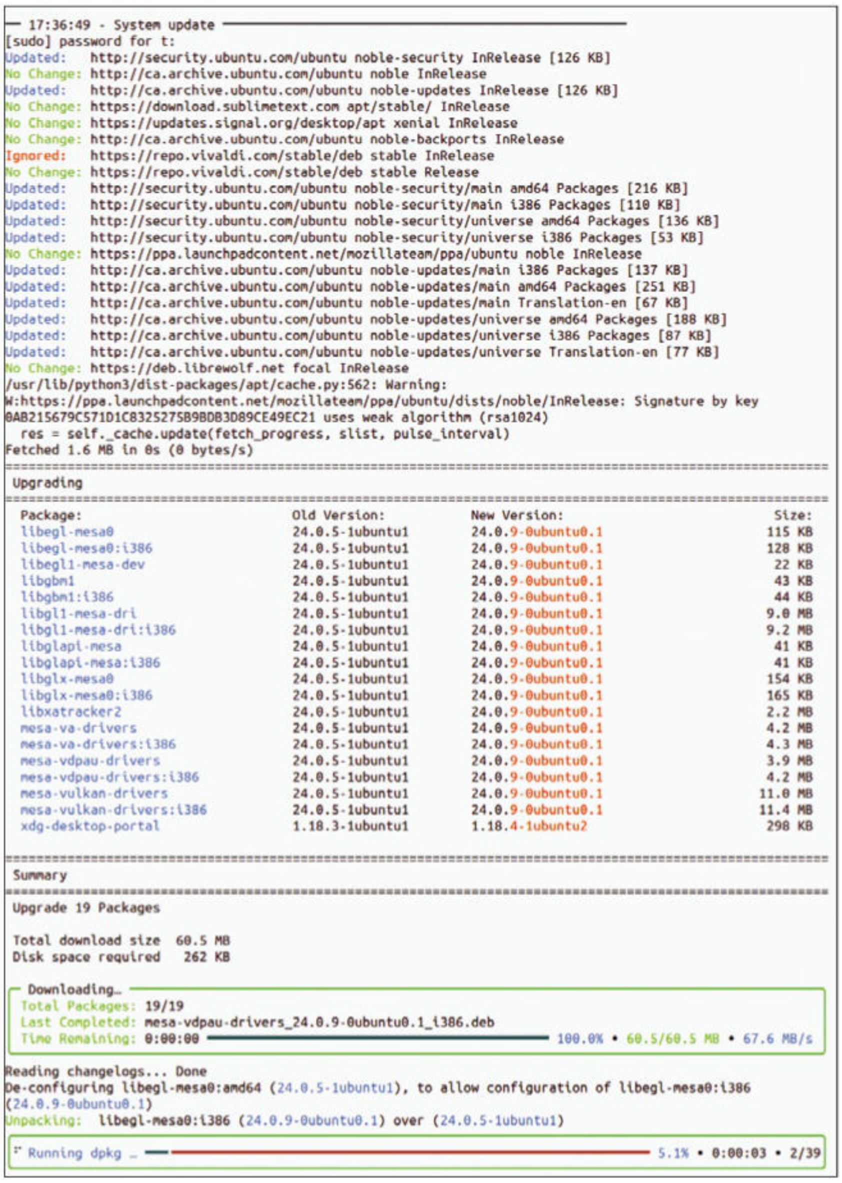
Topgrade to update and upgrade your software, regardless of its initial format. Coupled with high performance and a modern front-end package manager (e.g., Nala), you move from something that works OK out of the box to something slick and modern that, dare I say, might even make updating your systems somewhat fun. With the reality of continuous integration and continuous delivery processes offering never-ending bug fixes and functionalities with never-ending roll-outs, from its numerous projects down to its core software requirements, keeping everything up to the latest and greatest at the versioning level is a daunting task and, furthermore, a short-lived victory most of the time.

At the end of the day, any serious administrator will probably jump joyfully onto the Topgrade ride by entrusting their systems with a Ninite [8] equivalent that allows them to run the latest and the greatest. I hope this article has made you a believer. ■



```
— 17:38:07 - TLDR ————————————————————————————————————————
^CTLDR failed:
   0: Command failed: `/usr/bin/tldr --update`
   1: `/usr/bin/tldr` failed: signal: 2 (SIGINT)

Location:
   /home/t/.cargo/registry/src/index.crates.io-6f17d22bba15001f/topgrade-14.0.1/src/steps/os/unix.rs:565
Retry? (y)es/(N)o/(s)hell/(q)uit

Dropping you to shell. Fix what you need and then exit the shell.

[oh-my-zsh] Random theme 'oldgallois' loaded
 _______________________________________
/ Fortune's Real-Life Courtroom Quote   \
| #37:                                  |
|                                       |
| Q: Did he pick the dog up by the ears? |
| A: No. Q: What was he doing with the  |
| dog's ears? A: Picking them up in the |
| air. Q: Where was the dog at this time? |
\ A: Attached to the ears.              /
 ---------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
[-]$ █
```

Figure 8: When something goes wrong, users will be prompted to fix it immediately. In this case, a manual SIGINT was issued, so nothing requires fixing.



```
— 17:36:49 - System update ————————————————————————————————
[sudo] password for t:
Updated:   http://security.ubuntu.com/ubuntu noble-security InRelease [126 KB]
No Change: http://ca.archive.ubuntu.com/ubuntu noble InRelease
Updated:   http://ca.archive.ubuntu.com/ubuntu noble-updates InRelease [126 KB]
No Change: https://download.sublimetext.com apt/stable/ InRelease
No Change: https://updates.signal.org/desktop/apt xenial InRelease
No Change: http://ca.archive.ubuntu.com/ubuntu noble-backports InRelease
Ignored:   https://repo.vivaldi.com/stable/deb stable InRelease
No Change: https://repo.vivaldi.com/stable/deb stable Release
Updated:   http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [216 KB]
Updated:   http://security.ubuntu.com/ubuntu noble-security/main i386 Packages [110 KB]
Updated:   http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [136 KB]
Updated:   http://security.ubuntu.com/ubuntu noble-security/universe i386 Packages [53 KB]
No Change: https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu noble InRelease
Updated:   http://ca.archive.ubuntu.com/ubuntu noble-updates/main i386 Packages [137 KB]
Updated:   http://ca.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [251 KB]
Updated:   http://ca.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [67 KB]
Updated:   http://ca.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [188 KB]
Updated:   http://ca.archive.ubuntu.com/ubuntu noble-updates/universe i386 Packages [87 KB]
Updated:   http://ca.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [77 KB]
No Change: https://deb.librewolf.net focal InRelease
/usr/lib/python3/dist-packages/apt/cache.py:562: Warning:
W:https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu/dists/noble/InRelease: Signature by key
0AB215679C571D1C8325275B9BDB3D89CE49EC21 uses weak algorithm (rsa1024)
  res = self._cache.update(fetch_progress, slist, pulse_interval)
Fetched 1.6 MB in 0s (0 bytes/s)
==========================================================================================
 Upgrading
==========================================================================================
  Package:                      Old Version:          New Version:                  Size:
  libegl-mesa0                  24.0.5-1ubuntu1       24.0.9-0ubuntu0.1            115 KB
  libegl-mesa0:i386             24.0.5-1ubuntu1       24.0.9-0ubuntu0.1            128 KB
  libegl1-mesa-dev              24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             22 KB
  libgbm1                       24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             43 KB
  libgbm1:i386                  24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             44 KB
  libgl1-mesa-dri               24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             9.0 MB
  libgl1-mesa-dri:i386          24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             9.2 MB
  libglapi-mesa                 24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             41 KB
  libglapi-mesa:i386            24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             41 KB
  libglx-mesa0                  24.0.5-1ubuntu1       24.0.9-0ubuntu0.1            154 KB
  libglx-mesa0:i386             24.0.5-1ubuntu1       24.0.9-0ubuntu0.1            165 KB
  libxatracker2                 24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             2.2 MB
  mesa-va-drivers               24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             4.2 MB
  mesa-va-drivers:i386          24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             4.3 MB
  mesa-vdpau-drivers            24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             3.9 MB
  mesa-vdpau-drivers:i386       24.0.5-1ubuntu1       24.0.9-0ubuntu0.1             4.2 MB
  mesa-vulkan-drivers           24.0.5-1ubuntu1       24.0.9-0ubuntu0.1            11.0 MB
  mesa-vulkan-drivers:i386      24.0.5-1ubuntu1       24.0.9-0ubuntu0.1            11.4 MB
  xdg-desktop-portal            1.18.3-1ubuntu1       1.18.4-1ubuntu2              298 KB

==========================================================================================
 Summary
==========================================================================================
 Upgrade 19 Packages

 Total download size  60.5 MB
 Disk space required   262 KB

┌ Downloading… ─────────────────────────────────────────────────────────────────────────┐
│ Total Packages: 19/19                                                                   │
│ Last Completed: mesa-vdpau-drivers_24.0.9-0ubuntu0.1_i386.deb                           │
│ Time Remaining: 0:00:00 ━━━━━━━━━━━━━━━━━━━━━━━━ 100.0% • 60.5/60.5 MB • 67.6 MB/s      │
└─────────────────────────────────────────────────────────────────────────────────────────┘
Reading changelogs... Done
De-configuring libegl-mesa0:amd64 (24.0.5-1ubuntu1), to allow configuration of libegl-mesa0:i386
(24.0.9-0ubuntu0.1)
Unpacking:  libegl-mesa0:i386 (24.0.9-0ubuntu0.1) over (24.0.5-1ubuntu1)
┌─────────────────────────────────────────────────────────────────────────────────────────┐
│ ⠋ Running dpkg … ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.1% • 0:00:03 • 2/39          │
└─────────────────────────────────────────────────────────────────────────────────────────┘
```

Figure 9: The supercharged `nala` utility shows a much more verbose view of your current upgrade phase.

Info

[1] r-darwish Topgrade: [https://github.com/r-darwish/topgrade]

[2] topgrade-rs: [https://github.com/topgrade-rs/topgrade]

[3] SUSE Topgrade: [https://software.opensuse.org/download/package?package=topgrade&project=utilities]

[4] lib.rs alternative to crates.io: [https://lib.rs/]

[5] rustup download: [https://sh.rustup.rs]

[6] TOML: [https://toml.io/en/]

[7] Nala: [https://github.com/volitank/nala]

[8] Ninite: [https://ninite.com]

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/].

LINUX APP SUMMIT

Monterrey, Mexico | Online
October 4–5, 2024

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

Learn More
linuxappsummit.org

Mocking and emulating AWS and GCP services

Unstopped

Mock and emulate AWS services locally and use official Google Cloud Platform service emulators to unblock development and operations – all without paying a dime. By Ankur Kumar

Modern software-as-a-service (SaaS) offerings are highly dependent on external cloud providers to run and work reliably, but nothing comes without conditions or comes free, especially public cloud services. Nowadays, public cloud services are company cost centers, with strict measures in place to control costs. Like all modern distributed systems, all major cloud service providers go through major downtime now and then. Both cost-control measures and downtime hamper internal development, quality assurance (QA), and other dependent stakeholders by blocking productivity. These reasons are enough to consider emulating the required cloud services to unblock internal teams, keep them working on their goals with full control, and avoid any resistance or other unnecessary blocking requirements such as money, access policies, and so on. Cloud providers also started realizing these needs and started providing users the capabilities to launch or mock important services locally.

Local GCP with Emulators

Google Cloud Platform (GCP) provides libraries and tools to interact with their products and services through the Cloud software development kit (SDK). Pub/Sub, Spanner, Bigtable, Datastore, and Firestore cloud services have official emulators, exposed through the Google Cloud command-line interface (`gcloud` CLI) by Cloud SDK.

These emulators could be run natively on your laptop if the required dependencies are installed first, but I find that Docker containers are the best way to run these emulators out of the box because GCP provides various flavors of the official `cloud-sdk` base image. Therefore, you'll see various code snippets in the next sections to run various GCP cloud service emulators and their respective quick test routines in Docker containers.

Google Cloud Pub/Sub

Modern mass-scale web applications are mostly asynchronous and built around an asynchronous publish-subscribe pattern that requires some message-oriented middleware or broker service. With the help of such middleware or services, a producer can publish and a consumer can subscribe to events and take the necessary actions to process those events. Google Cloud Pub/Sub is such a hosted service with very low latency to craft modern microservices-based products.

To run the GCP Pub/Sub service locally through its emulator and test it to show basic functionality, without leaving your laptop or spending a single dime on a GCP bill, you need to create a Docker network:

```
docker network create gcpemulators-demo
```

To begin, create a Docker image to launch the Pub/Sub service locally after creating the Dockerfile in Listing 1 [1]:

```
docker build . -f Dockerfile_GCPCPSEMU ⮯
             -t gcpcpsemu
```

Now, create the script in Listing 2 for the Pub/Sub service to create a topic and a subscription to the topic, to publish messages to the topic, and to retrieve messages from the topic.

Listing 1: `Dockerfile_GCPCPSEMU`

```
FROM google/cloud-sdk:alpine

EXPOSE 8085

RUN apk add --no-cache openjdk8-jre && gcloud components install beta pubsub-emulator --quiet

ENTRYPOINT ["gcloud","beta","emulators","pubsub","start","--project=demo"]
CMD ["--host-port=0.0.0.0:8085"]
```

Listing 2: `gcpemu_cpstst.sh`

```
#! /bin/sh

echo
echo ' <Start of Cloud PubSub Quick Test>'
dockerize -wait tcp://gcpcpsemu:8085

cd python-pubsub/samples/snippets
python3 publisher.py demo create demo
python3 subscriber.py demo create demo demo
python3 publisher.py demo publish demo
python3 subscriber.py demo receive demo 20
echo ' <End of Cloud PubSub Quick Test>'
echo
```

Listing 3: `Dockerfile_GCPCPSTST`

```
FROM alpine:3.19 AS dockerize

ENV DOCKERIZE_VERSION v0.7.0
RUN wget https://github.com/jwilder/dockerize/releases/download/$DOCKERIZE_VERSION/
  dockerize-alpine-linux-amd64-$DOCKERIZE_VERSION.tar.gz && tar -C /usr/local/bin -xzvf
  dockerize-alpine-linux-amd64-$DOCKERIZE_VERSION.tar.gz && rm dockerize-alpine-linux-amd64-$DOCKERIZE_
  VERSION.tar.gz && echo "**** fix for host id mapping error ****" && chown root:root /usr/local/bin/
  dockerize

FROM google/cloud-sdk:alpine

SHELL ["/bin/ash", "-o", "pipefail", "-c"]
RUN apk add --no-cache --virtual .build-deps alpine-sdk libffi-dev openssl-dev python3-dev py3-pip &&
  gcloud config configurations create emulator --quiet && git clone https://github.com/googleapis/
  python-pubsub.git && cd python-pubsub/samples/snippets && pip3 install -r requirements.txt

COPY --from=dockerize /usr/local/bin/dockerize /usr/local/bin/

COPY gcpemu_cpstst.sh /usr/local/bin/run.sh

ENTRYPOINT ["run.sh"]
```

Please note that because of the current limitations of the Pub/Sub emulator, the test logic is implemented with Cloud Client Libraries in Python, not `gcloud pubsub` commands.

Next, create the Dockerfile in Listing 3 and execute the command

```
docker build . -f Dockerfile_GCPCPSTST ➥
               -t gcpcpstst
```

to create a Docker image containing the previously shown logic to run the quick test against the locally running Pub/Sub service. Finally, create the compose file in Listing 4, to start up the local Pub/Sub service, and execute the test logic against it:

```
docker run --rm ➥
  -v /var/run/docker.sock:/var/run/➥
     docker.sock:ro ➥
  -v ./gcp_emulators_cpsemu.yml:/etc/➥
     compose/gcp_emulators_cpsemu.yml:ro ➥
     docker docker compose ➥
  -f /etc/compose/gcp_emulators_➥
     cpsemu.yml up -d
```

You can see the log messages about the local Pub/Sub service and about the test logic that makes use of it with the respective commands in Listing 5. Figures 1 and 2 were taken on my laptop after executing these logging commands.

Now you have everything to make use of the local Pub/Sub service; just point your application to local port 28085 and reap the benefits of the emulator. Please feel free to explore the Pub/Sub emulator further through the online documentation [2]. Once done, use the command

```
docker run --rm ➥
  -v /var/run/docker.sock:/var/run/➥
     docker.sock:ro ➥
  -v ./gcp_emulators_cpsemu.yml:/etc/➥
     compose/gcp_emulators_cpsemu.yml:ro ➥
     docker docker compose ➥
  -f /etc/compose/gcp_emulators_➥
     cpsemu.yml down
```

to clean up.

Google Cloud Bigtable

Data services – especially databases – are the bread and butter of modern web-scale cloud-based products. On the other hand, the cloud-hosted versions of various kinds of databases are a huge cost center and dependency for any company running in the public cloud, so you should be interested in options provided by the GCP emulators to run databases services.

Indeed, the emulators provide local functionality to run Cloud Bigtable and Cloud Spanner services, as well. Cloud Bigtable is a key-value and wide-column store, ideal for low latency and fast access to structured, semi-structured, or unstructured data. Cloud Spanner, on the other hand, is an enterprise-grade, globally distributed, and strongly consistent database service built for the cloud specifically to combine the benefits of the relational database structure with non-relational horizontal scale.

Listing 4: `gcp_emulators_cpsemu.yml`

```
services:

  gcpcpsemu:
    image: gcpcpsemu
    container_name: gcpcpsemu
    hostname: gcpcpsemu
    ports:
      - "28085:8085"
    restart: unless-stopped

  gcpcpstst:
    image: gcpcpstst
    container_name: gcpcpstst
    hostname: gcpcpstst
    environment:
      - "PUBSUB_EMULATOR_HOST=gcpcpsemu:8085"
      - "PUBSUB_PROJECT_ID=demo"
      - "CLOUDSDK_AUTH_DISABLE_CREDENTIALS=true"
      - "CLOUDSDK_CORE_PROJECT=demo"
    depends_on:
      - gcpcpsemu

networks:
  default:
    name: gcpemulators-demo
    external: true
```

Listing 5: Logging Commands

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v ./gcp_emulators_cpsemu.yml:/etc/compose/
  gcp_emulators_cpsemu.yml:ro docker docker compose -f /etc/compose/gcp_emulators_cpsemu.yml logs gcpcpsemu
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v ./gcp_emulators_cpsemu.yml:/etc/compose/
  gcp_emulators_cpsemu.yml:ro docker docker compose -f /etc/compose/gcp_emulators_cpsemu.yml logs gcpcpst
```

First things first: Create the Dockerfile in Listing 6 and execute the command

```
docker build . -f Dockerfile_GCPCBTEMU ⮐
               -t gcpcbtemu
```

to create a Docker image for running a local Cloud Bigtable service.

Listing 6: `Dockerfile_GCPCBTEMU`

```
FROM google/cloud-sdk:alpine

EXPOSE 8086

RUN gcloud components install beta bigtable --quiet

ENTRYPOINT ["gcloud","beta","emulators","bigtable","start
  ","--quiet"]
CMD ["--host-port=0.0.0.0:8086"]
```

Next, create the script in Listing 7 containing quick test logic to execute against the locally running Bigtable service. You can clearly see that the cbt command-line utility is fully capable of executing various functionalities against the local Bigtable, unlike the local Pub/Sub limitation of using a language library only.

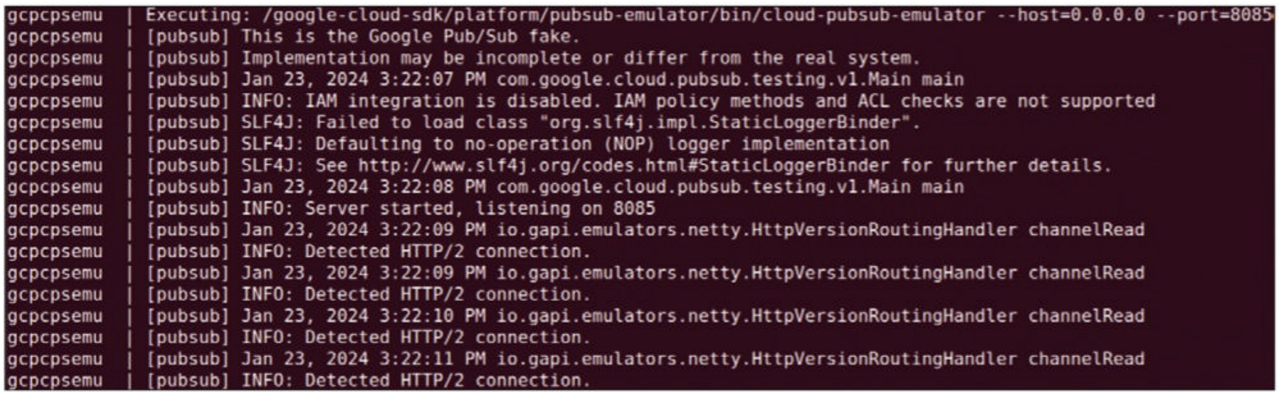
After creating the Dockerfile in Listing 8, create the test image with

```
docker build . -f Dockerfile_GCPCBTTST ⮐
               -t gcpcbttst
```

Finally, create the YAML file in Listing 9 to bring up the local Bigtable

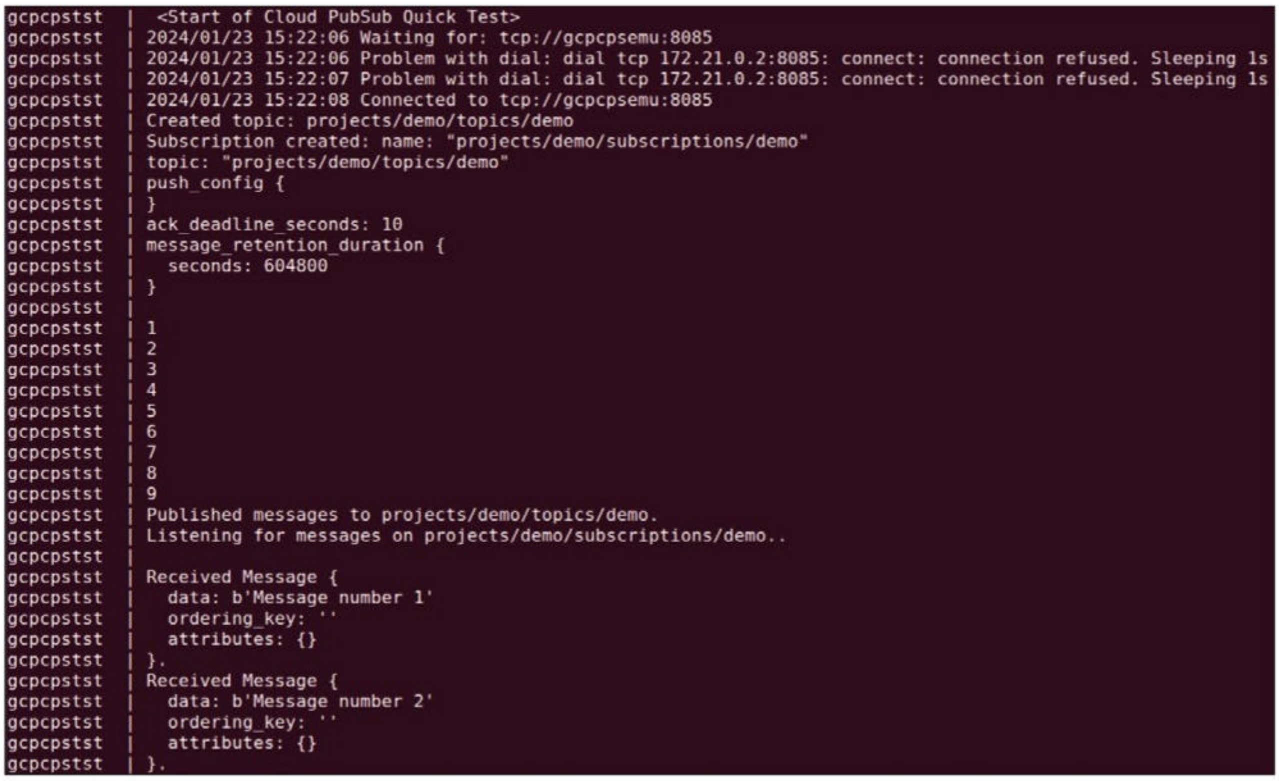
Listing 7: `gcpemu_cbttst.sh`

```
#! /bin/sh

(
echo ' <Start of Cloud Big Table Quick Test>'
dockerize -wait tcp://gcpcbtemu:8086

cbt createtable cbt-run-demo
cbt ls
cbt createfamily cbt-run-demo cf1
cbt ls cbt-run-demo
cbt set cbt-run-demo r1 cf1:c1=test-value
cbt read cbt-run-demo
cbt deletetable cbt-run-demo
cbt deleteinstance demo-instance
echo ' <End of Cloud Big Table Quick Test>'
echo
) 2>/dev/null
```



```
gcpcpsemu   | Executing: /google-cloud-sdk/platform/pubsub-emulator/bin/cloud-pubsub-emulator --host=0.0.0.0 --port=8085
gcpcpsemu   | [pubsub] This is the Google Pub/Sub fake.
gcpcpsemu   | [pubsub] Implementation may be incomplete or differ from the real system.
gcpcpsemu   | [pubsub] Jan 23, 2024 3:22:07 PM com.google.cloud.pubsub.testing.v1.Main main
gcpcpsemu   | [pubsub] INFO: IAM integration is disabled. IAM policy methods and ACL checks are not supported
gcpcpsemu   | [pubsub] SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
gcpcpsemu   | [pubsub] SLF4J: Defaulting to no-operation (NOP) logger implementation
gcpcpsemu   | [pubsub] SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
gcpcpsemu   | [pubsub] Jan 23, 2024 3:22:08 PM com.google.cloud.pubsub.testing.v1.Main main
gcpcpsemu   | [pubsub] INFO: Server started, listening on 8085
gcpcpsemu   | [pubsub] Jan 23, 2024 3:22:09 PM io.gapi.emulators.netty.HttpVersionRoutingHandler channelRead
gcpcpsemu   | [pubsub] INFO: Detected HTTP/2 connection.
gcpcpsemu   | [pubsub] Jan 23, 2024 3:22:09 PM io.gapi.emulators.netty.HttpVersionRoutingHandler channelRead
gcpcpsemu   | [pubsub] INFO: Detected HTTP/2 connection.
gcpcpsemu   | [pubsub] Jan 23, 2024 3:22:10 PM io.gapi.emulators.netty.HttpVersionRoutingHandler channelRead
gcpcpsemu   | [pubsub] INFO: Detected HTTP/2 connection.
gcpcpsemu   | [pubsub] Jan 23, 2024 3:22:11 PM io.gapi.emulators.netty.HttpVersionRoutingHandler channelRead
gcpcpsemu   | [pubsub] INFO: Detected HTTP/2 connection.
```

Figure 1: GCP Pub/Sub emulator log.



```
gcpcpstst   |  <Start of Cloud PubSub Quick Test>
gcpcpstst   | 2024/01/23 15:22:06 Waiting for: tcp://gcpcpsemu:8085
gcpcpstst   | 2024/01/23 15:22:06 Problem with dial: dial tcp 172.21.0.2:8085: connect: connection refused. Sleeping 1s
gcpcpstst   | 2024/01/23 15:22:07 Problem with dial: dial tcp 172.21.0.2:8085: connect: connection refused. Sleeping 1s
gcpcpstst   | 2024/01/23 15:22:08 Connected to tcp://gcpcpsemu:8085
gcpcpstst   | Created topic: projects/demo/topics/demo
gcpcpstst   | Subscription created: name: "projects/demo/subscriptions/demo"
gcpcpstst   | topic: "projects/demo/topics/demo"
gcpcpstst   | push_config {
gcpcpstst   | }
gcpcpstst   | ack_deadline_seconds: 10
gcpcpstst   | message_retention_duration {
gcpcpstst   |   seconds: 604800
gcpcpstst   | }
gcpcpstst   |
gcpcpstst   | 1
gcpcpstst   | 2
gcpcpstst   | 3
gcpcpstst   | 4
gcpcpstst   | 5
gcpcpstst   | 6
gcpcpstst   | 7
gcpcpstst   | 8
gcpcpstst   | 9
gcpcpstst   | Published messages to projects/demo/topics/demo.
gcpcpstst   | Listening for messages on projects/demo/subscriptions/demo..
gcpcpstst   |
gcpcpstst   | Received Message {
gcpcpstst   |   data: b'Message number 1'
gcpcpstst   |   ordering_key: ''
gcpcpstst   |   attributes: {}
gcpcpstst   | }.
gcpcpstst   | Received Message {
gcpcpstst   |   data: b'Message number 2'
gcpcpstst   |   ordering_key: ''
gcpcpstst   |   attributes: {}
gcpcpstst   | }.
```

Figure 2: GCP Pub/Sub test logic log.

service, as well as the quick test routine, with the command

```
docker run --rm ➥
  -v /var/run/docker.sock:/var/run/➥
  docker.sock:ro ➥
  -v ./gcp_emulators_cbtemu.yml:/etc/compose/➥
  gcp_emulators_cbtemu.yml:ro docker ➥
  docker compose ➥
  -f /etc/compose/gcp_emulators_cbtemu.yml ➥
  up -d
```

You can see the log messages for the local Bigtable service and the test logic making use of it with the command

```
docker run --rm ➥
  -v /var/run/docker.sock:/var/run/➥
     docker.sock:ro ➥
  -v ./gcp_emulators_cbtemu.yml:/etc/➥
     compose/gcp_emulators_cbtemu.yml:ro ➥
     docker docker compose ➥
  -f /etc/compose/gcp_emulators_➥
     cbtemu.yml logs
```

The screenshot in Figure 3 was taken on my laptop after executing this logging command. Now you have everything to make use of the Bigtable service, just point your application to local port 28086 and reap the benefits of the emulator. Please feel free to explore Cloud Bigtable further through the documentation [3].

Google Cloud Spanner

GCP provides a ready-to-use Docker image for the local Cloud Spanner service. The `gcloud` CLI commands in Listing 10 execute a quick test against a locally running spanner service. To build a Docker image to further launch a quick test routine against a local Cloud Spanner service, create the file in Listing 11 and execute the command

```
docker build . -f Dockerfile_GCPCSPTST ➥
               -t gcpcsptst
```

Finally, create the YAML file in Listing 12 and launch the local Bigtable service along with the quick test routine with the command

```
docker run --rm ➥
  -v /var/run/docker.sock:/var/run/➥
     docker.sock:ro ➥
  -v ./gcp_emulators_cspemu.yml:/etc/➥
     compose/gcp_emulators_cspemu.yml:ro ➥
     docker docker compose ➥
  -f /etc/compose/gcp_emulators_➥
     cspemu.yml up -d
```

You can see the log messages for the local spanner service and the test logic that uses it with the command

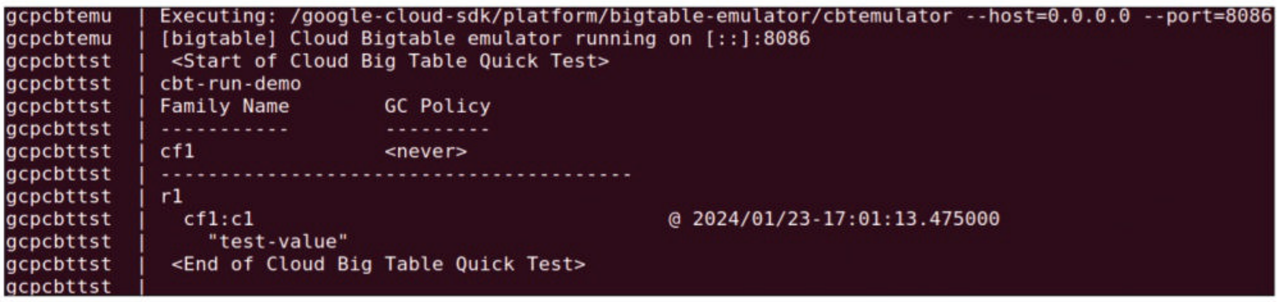
```
docker run --rm ➥
  -v /var/run/docker.sock:/var/run/➥
     docker.sock:ro ➥
  -v ./gcp_emulators_cspemu.yml:/etc/➥
```

Listing 8: `Dockerfile_GCPCBTTST`

```
FROM alpine:3.19 AS dockerize

ENV DOCKERIZE_VERSION v0.7.0
RUN wget https://github.com/jwilder/dockerize/releases/download/$DOCKERIZE_VERSION/
  dockerize-alpine-linux-amd64-$DOCKERIZE_VERSION.tar.gz && tar -C /usr/local/bin -xzvf
  dockerize-alpine-linux-amd64-$DOCKERIZE_VERSION.tar.gz && rm dockerize-alpine-linux-amd64-$DOCKERIZE_
  VERSION.tar.gz && echo "**** fix for host id mapping error ****" && chown root:root /usr/local/bin/
  dockerize

FROM google/cloud-sdk:alpine

SHELL ["/bin/ash", "-o", "pipefail", "-c"]
RUN gcloud components install cbt --quiet && printf "%s\n%s\n" "project=demo" "instance=demo"|tee
  ~/.cbtrc && gcloud config configurations create emulator --quiet

COPY --from=dockerize /usr/local/bin/dockerize /usr/local/bin/

COPY gcpemu_cbttst.sh /usr/local/bin/run.sh

ENTRYPOINT ["run.sh"]
```

Listing 9: `gcp_emulators_cbtemu.yml`

```
services:

  gcpcbtemu:
    image: gcpcbtemu
    container_name: gcpcbtemu
    hostname: gcpcbtemu
    ports:
      - "28086:8086"
    restart: unless-stopped

  gcpcbttst:
    image: gcpcbttst
    container_name: gcpcbttst
    hostname: gcpcbttst
    environment:
      - "BIGTABLE_EMULATOR_HOST=gcpcbtemu:8086"
      - "CLOUDSDK_AUTH_DISABLE_CREDENTIALS=true"
      - "CLOUDSDK_CORE_PROJECT=demo"
    depends_on:
      - gcpcbtemu

networks:
  default:
    name: gcpemulators-demo
    external: true
```



```
gcpcbtemu  | Executing: /google-cloud-sdk/platform/bigtable-emulator/cbtemulator --host=0.0.0.0 --port=8086
gcpcbtemu  | [bigtable] Cloud Bigtable emulator running on [::]:8086
gcpcbttst  |  <Start of Cloud Big Table Quick Test>
gcpcbttst  | cbt-run-demo
gcpcbttst  | Family Name         GC Policy
gcpcbttst  | -----------         ---------
gcpcbttst  | cf1                 <never>
gcpcbttst  | ----------------------------------------
gcpcbttst  | r1
gcpcbttst  |   cf1:c1                                   @ 2024/01/23-17:01:13.475000
gcpcbttst  |     "test-value"
gcpcbttst  |  <End of Cloud Big Table Quick Test>
gcpcbttst  |
```

Figure 3: **GCP Bigtable emulator quick test log.**

Listing 10: gcpemu_csptst.sh

```
#! /bin/sh

(
echo ' <Start of Cloud Spanner Quick Test>'
dockerize -wait tcp://gcpcspemu:9020

gcloud spanner instances create test-instance --config=emulator-config --description="Test Instance" --nodes=1
gcloud spanner instances list --configuration=emulator-config
gcloud spanner instances delete test-instance --configuration=emulator-config --quiet
echo ' <End of Cloud Spanner Quick Test>'
echo
)
```

Listing 11: Dockerfile_GCPCSPTST

```
FROM alpine:3.19 AS dockerize

ENV DOCKERIZE_VERSION v0.7.0
RUN wget https://github.com/jwilder/dockerize/releases/download/$DOCKERIZE_VERSION/
  dockerize-alpine-linux-amd64-$DOCKERIZE_VERSION.tar.gz && tar -C /usr/local/bin -xzvf
  dockerize-alpine-linux-amd64-$DOCKERIZE_VERSION.tar.gz && rm dockerize-alpine-linux-amd64-$DOCKERIZE_
  VERSION.tar.gz && echo "**** fix for host id mapping error ****" && chown root:root /usr/local/bin/
  dockerize

FROM google/cloud-sdk:alpine

SHELL ["/bin/ash", "-o", "pipefail", "-c"]
RUN gcloud config configurations create emulator --quiet

COPY --from=dockerize /usr/local/bin/dockerize /usr/local/bin/

COPY gcpemu_csptst.sh /usr/local/bin/run.sh

ENTRYPOINT ["run.sh"]
```

Listing 12: gcp_emulators_cspemu.yml

```
services:

  gcpcspemu:
    image: gcr.io/cloud-spanner-emulator/emulator
    container_name: gcpcspemu
    hostname: gcpcspemu
    ports:
      - "29010:9010"
      - "29020:9020"
    restart: unless-stopped

  gcpcsptst:
    image: gcpcsptst
    container_name: gcpcsptst
    hostname: gcpcsptst
    environment:
      - "SPANNER_EMULATOR_HOST=gcpcspemu:9020"
      - "CLOUDSDK_AUTH_DISABLE_CREDENTIALS=true"
      - "CLOUDSDK_CORE_PROJECT=demo"
      - "CLOUDSDK_API_ENDPOINT_OVERRIDES_SPANNER=http://gcpcspemu:9020/"
    depends_on:
      - gcpcspemu

networks:
  default:
    name: gcpemulators-demo
    external: true
```

```
    compose/gcp_emulators_cspemu.yml:ro ↩
    docker docker compose ↩
  -f /etc/compose/gcp_emulators_↩
    cspemu.yml logs
```

Figure 4 was taken on my laptop after executing the above logging command. Now you have everything you need to make use of the spanner service; just point your application to local port 29010/ 29020 and reap the benefits of the emulator. Feel free to explore the Cloud Spanner emulator further in the documentation [4]. Once done, you can use the command

```
docker run --rm ↩
  -v /var/run/docker.sock:/var/run/↩
    docker.sock:ro ↩
  -v ./gcp_emulators_cspemu.yml:/etc/↩
    compose/gcp_emulators_cspemu.yml:ro ↩
    docker docker compose ↩
  -f /etc/compose/gcp_emulators_↩
    cspemu.yml down
```

to clean up.

Cloud Datastore and Firestore

You're in further luck if you use Google Cloud Datastore, its next-generation Cloud Firestore services, or both in your products. These services can also be launched locally through their respective emulators. The Dockerfiles to create the images to be launched locally are shown in Listings 13 and 14.

Listing 13: Dockerfile_GCPCDSEMU

```
FROM google/cloud-sdk:alpine

EXPOSE 8081

RUN apk add --no-cache openjdk8-jre && gcloud components install beta
  cloud-datastore-emulator --quiet

ENTRYPOINT ["gcloud","beta","emulators","datastore","start","--project=demo"]
CMD ["--host-port=0.0.0.0:8081"]
```

Listing 14: Dockerfile_GCPCFSEMU

```
FROM google/cloud-sdk:alpine

EXPOSE 8721

RUN apk add --no-cache openjdk8-jre && gcloud components install beta
  cloud-firestore-emulator --quiet

ENTRYPOINT ["gcloud","beta","emulators","firestore","start"]
CMD ["--host-port=0.0.0.0:8721"]
```

Local AWS Mocking with Moto

AWS is the most popular public cloud services provider, with hundreds of infrastructure-, platform-, and software-as-a-service (IaaS, PaaS, SaaS) offerings, and more. The need of mocking and emulating AWS services becomes more important from financial and dependency perspectives for any company of any size. Because of the power of free and open source software, you can find solutions to mock and emulate almost any AWS service comfortably on your own terms, without paying a single dime.

The first solution to explore is a Python library and server known as Moto [5], with which you can mock a number of AWS services and features in a simple, straightforward way. To begin, create a Docker network from the terminal to start playing with Moto:

```
docker network create awsmockemu-demo
```

Now, create the script in Listing 15 to see Moto's basic capabilities as a library in action. This test code simply runs AWS calls between the `mock.start` and `mock.stop` methods (lines 16-29). That's all you need to know for raw AWS mocking with Python code.

Two other ways to mock AWS calls in your code are to use a decorator (by preceding `test_s3_save` with `@mock_aws`) or a context manager (all AWS calls after the line with `mock_aws`). The diffs shown in Listings 16 and 17 are the necessary code changes needed to mock with the Moto decorator and context manager, respectively. The command

```
docker run --rm ⮯
  -v ./moto_python_test.py:/etc/⮯
     motopython/moto_python_test.py:ro ⮯
  -w /etc/motopython ⮯
  --entrypoint=python motoserver/moto ⮯
    moto_python_test.py
```

quickly executes the Moto library mocking example code. If no error is thrown, AWS mocking is successful.

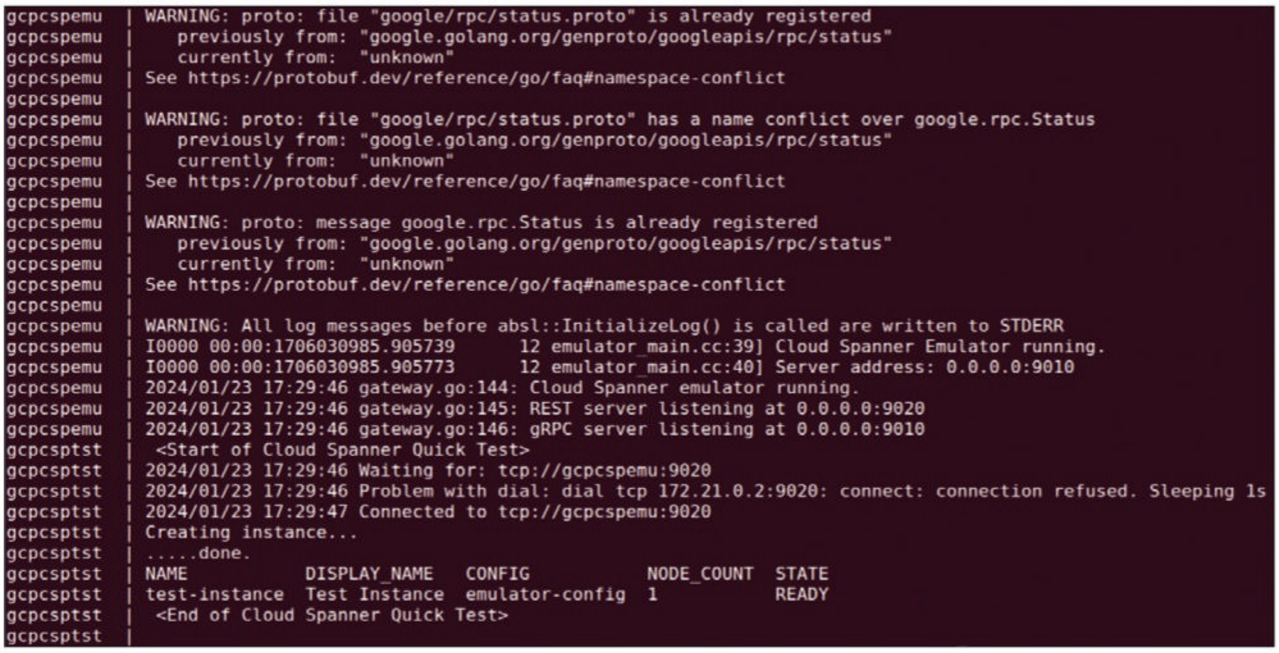
If your application is not programmed in Python or you're not in a position to edit your application or use popular Infrastructure as Code (IaC) tools such as Ansible or Terraform, then Moto's server mode comes into the picture.

To launch a Moto server along with an AWS CLI container able to execute some basic actions against your locally running AWS mocking server, create the file in Listing 18. Then execute the commands in Listing 19 to bring up the Moto stack and dump information about your default virtual private cloud (VPC) and subnets. Voila, you can see local AWS mocking by running the Moto server (Figure 5).

Next, dump the IDs and descriptions of Linux Amazon machine images (AMIs) returned by the Moto server and the instance types in the mocked environment (Listing 20). Then create an AWS instance in the mocked environment.

Listing 15: `moto_python_test.py`

```
import boto3
from moto import mock_aws

class MyBucket:
    def __init__(self, name, value):
        self.name = name
        self.value = value

    def save(self):
        s3 = boto3.client("s3", region_name="us-east-1")
        s3.put_object(Bucket="mybucket", Key=self.name,
           Body=self.value)

def test_s3_save():
    mock = mock_aws()
    mock.start()

    conn = boto3.resource("s3", region_name="us-east-1")
    conn.create_bucket(Bucket="mybucket")

    bucket = MyBucket("PinkFloyd", "is awesome")
    bucket.save()

    body = conn.Object("mybucket", "PinkFloyd").get()[
        "Body"].read().decode("utf-8")

    assert body == "is awesome"

    mock.stop()

if "__main__" == __name__:
  test_s3_save()
```



```
gcpcspemu   | WARNING: proto: file "google/rpc/status.proto" is already registered
gcpcspemu   |    previously from: "google.golang.org/genproto/googleapis/rpc/status"
gcpcspemu   |    currently from:  "unknown"
gcpcspemu   | See https://protobuf.dev/reference/go/faq#namespace-conflict
gcpcspemu   |
gcpcspemu   | WARNING: proto: file "google/rpc/status.proto" has a name conflict over google.rpc.Status
gcpcspemu   |    previously from: "google.golang.org/genproto/googleapis/rpc/status"
gcpcspemu   |    currently from:  "unknown"
gcpcspemu   | See https://protobuf.dev/reference/go/faq#namespace-conflict
gcpcspemu   |
gcpcspemu   | WARNING: proto: message google.rpc.Status is already registered
gcpcspemu   |    previously from: "google.golang.org/genproto/googleapis/rpc/status"
gcpcspemu   |    currently from:  "unknown"
gcpcspemu   | See https://protobuf.dev/reference/go/faq#namespace-conflict
gcpcspemu   |
gcpcspemu   | WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
gcpcspemu   | I0000 00:00:1706030985.905739      12 emulator_main.cc:39] Cloud Spanner Emulator running.
gcpcspemu   | I0000 00:00:1706030985.905773      12 emulator_main.cc:40] Server address: 0.0.0.0:9010
gcpcspemu   | 2024/01/23 17:29:46 gateway.go:144: Cloud Spanner emulator running.
gcpcspemu   | 2024/01/23 17:29:46 gateway.go:145: REST server listening at 0.0.0.0:9020
gcpcspemu   | 2024/01/23 17:29:46 gateway.go:146: gRPC server listening at 0.0.0.0:9010
gcpcsptst   |  <Start of Cloud Spanner Quick Test>
gcpcsptst   | 2024/01/23 17:29:46 Waiting for: tcp://gcpcspemu:9020
gcpcsptst   | 2024/01/23 17:29:46 Problem with dial: dial tcp 172.21.0.2:9020: connect: connection refused. Sleeping 1s
gcpcsptst   | 2024/01/23 17:29:47 Connected to tcp://gcpcspemu:9020
gcpcsptst   | Creating instance...
gcpcsptst   | .....done.
gcpcsptst   | NAME           DISPLAY_NAME   CONFIG           NODE_COUNT  STATE
gcpcsptst   | test-instance  Test Instance  emulator-config  1           READY
gcpcsptst   |  <End of Cloud Spanner Quick Test>
gcpcsptst   |
```

Figure 4: Google Cloud Spanner emulator quick test log.

Please note that the created instance assumes a default key pair, VPC, security group, and so on, but feel free to experiment with creating those new resources and launching your instance under them. Finally, dump the properties of the created instance (Figure 6) and terminate the created instance, if desired, by providing the instance ID (could be different in your case). These working examples should be enough to help you get going with AWS mocking, and it's just the tip of the iceberg, because Moto covers a number of AWS services [6].

Local AWS Emulation with LocalStack

Now comes the big daddy of all the AWS cloud mocking and emulation frameworks. LocalStack is a drop-in single-container solution to provide 80+ AWS services at your immediate disposal. In fact, it's a full-fledged enterprise-level solution to emulate AWS services and Lambdas

Listing 16: Decorator Diffs

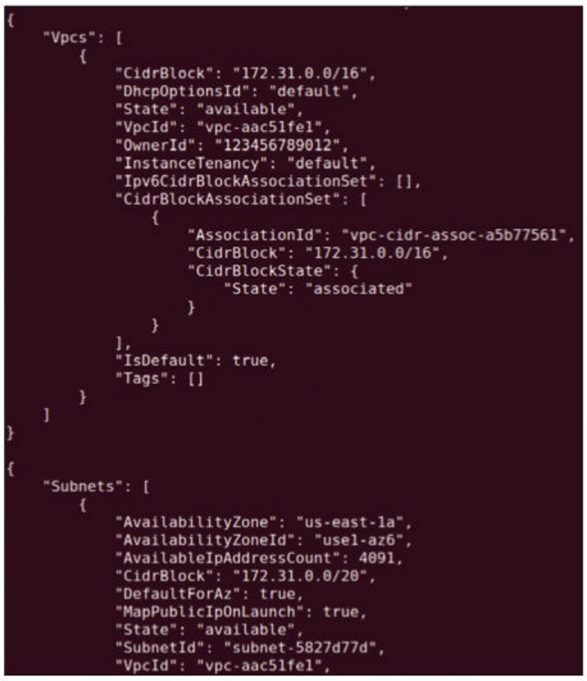
```
12a13
> @mock_aws
14,15d14
<     mock = mock_aws()
<     mock.start()
27,28d25
<
<     mock.stop()
```

Listing 17: Context manager Diffs

```
<     mock = mock_aws()
<     mock.start()
---
>   with mock_aws():
27,28d25
<
<     mock.stop()
```

Listing 18: `moto_server_stack.yml`

```
services:

  motoserver:
    image: motoserver/moto:latest
    container_name: motoserver
    hostname: motoserver
    ports:
      - "9500:5000"
    environment:
      - MOTO_PORT=5000
    healthcheck:
      test: ["CMD", "curl", "-I", "localhost:5000"]
      interval: 5s
      timeout: 3s
      retries: 5
    restart: unless-stopped

  awsclitest:
    image: amazon/aws-cli
    container_name: awsclitest
    hostname: awsclitest
    environment:
      - AWS_ACCESS_KEY_ID=foo
      - AWS_SECRET_ACCESS_KEY=foo
      - AWS_DEFAULT_REGION=us-east-1
      - AWS_ENDPOINT_URL=http://motoserver:5000
    entrypoint: "sh"
    command: "-c 'while true; do sleep 5; done'"
    depends_on:
      motoserver:
        condition: service_healthy

networks:
  default:
    name: awsmockemu-demo
    external: true
```



```
{
    "Vpcs": [
        {
            "CidrBlock": "172.31.0.0/16",
            "DhcpOptionsId": "default",
            "State": "available",
            "VpcId": "vpc-aac51fe1",
            "OwnerId": "123456789012",
            "InstanceTenancy": "default",
            "Ipv6CidrBlockAssociationSet": [],
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-a5b77561",
                    "CidrBlock": "172.31.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": true,
            "Tags": []
        }
    ]
}

{
    "Subnets": [
        {
            "AvailabilityZone": "us-east-1a",
            "AvailabilityZoneId": "use1-az6",
            "AvailableIpAddressCount": 4091,
            "CidrBlock": "172.31.0.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-5827d77d",
            "VpcId": "vpc-aac51fe1",
```

Figure 5: AWS CLI output showing the default local VPCs and subnets returned by the Moto server.

Listing 19: Moto in Server Mode

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v ./moto_server_stack.
  yml:/etc/compose/moto_server_stack.yml:ro docker docker compose -f /etc/compose/moto_
  server_stack.yml up -d
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v ./moto_server_stack.
  yml:/etc/compose/moto_server_stack.yml:ro docker compose -f /etc/compose/moto_server_
  stack.yml exec awsclitest sh -c 'aws ec2 describe-vpcs && aws ec2 describe-subnets'
```

Listing 20: AWS Instance

```
# dump IDs and descriptions of Linux AMIs
docker run --rm v /var/run/docker.sock:/var/run/docker.sock:ro -v ./moto_server_stack.yml:/
  etc/compose/moto_server_stack.yml:ro docker compose -f /etc/compose/moto_server_stack.yml
  exec awsclitest sh -c 'aws ec2 describe-images --filters "Name=Description,Values=* Linux *"
  "Name=root-device-type,Values=ebs" --query "Images[*].[ImageId,Description]" --output text'
# dump instance types
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v ./moto_server_stack.yml:/
  etc/compose/moto_server_stack.yml:ro docker compose -f /etc/compose/moto_server_stack.
  yml exec awsclitest sh -c 'aws ec2 describe-instance-types --query "InstanceTypes[*].
  [InstanceType]" --output text'
# create an AWS instance
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v ./moto_server_stack.yml:/
  etc/compose/moto_server_stack.yml:ro docker compose -f /etc/compose/moto_server_stack.yml
  exec awsclitest sh -c 'aws ec2 run-instances --image-id ami-002068ed284fb165b --count 1'
# dump instance properties
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v ./moto_server_stack.yml:/
  etc/compose/moto_server_stack.yml:ro docker compose -f /etc/compose/moto_server_stack.yml
  exec awsclitest sh -c 'aws ec2 describe-instances'
# terminate instance
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v ./moto_server_stack.yml:/
  etc/compose/moto_server_stack.yml:ro docker compose -f /etc/compose/moto_server_stack.yml
  exec awsclitest sh -c 'aws ec2 terminate-instances --instance-ids i-6ce75b7f81a552ce0'
```

(serverless compute services) entirely on your local machine without any external dependencies. LocalStack is simply your local cloud sandbox for development, testing, and experimentation.

To begin with LocalStack, I use a very popular IaC automation tool known as Terraform. If Terraform is new to you, it provides a declarative configuration language with tunable properties to create resources on services such as AWS. The Terraform ecosystem also provides official modules to create various cloud resource combos as configurable stacks. I'll use the official Terraform AWS module test examples here to familiarize you with LocalStack.

First things first: Create the Dockerfile in Listing 21, the Terraform module in Listing 22, and the script in Listing 23 to create an image to run Terraform actions against the localhost. Then create the necessary Docker image:

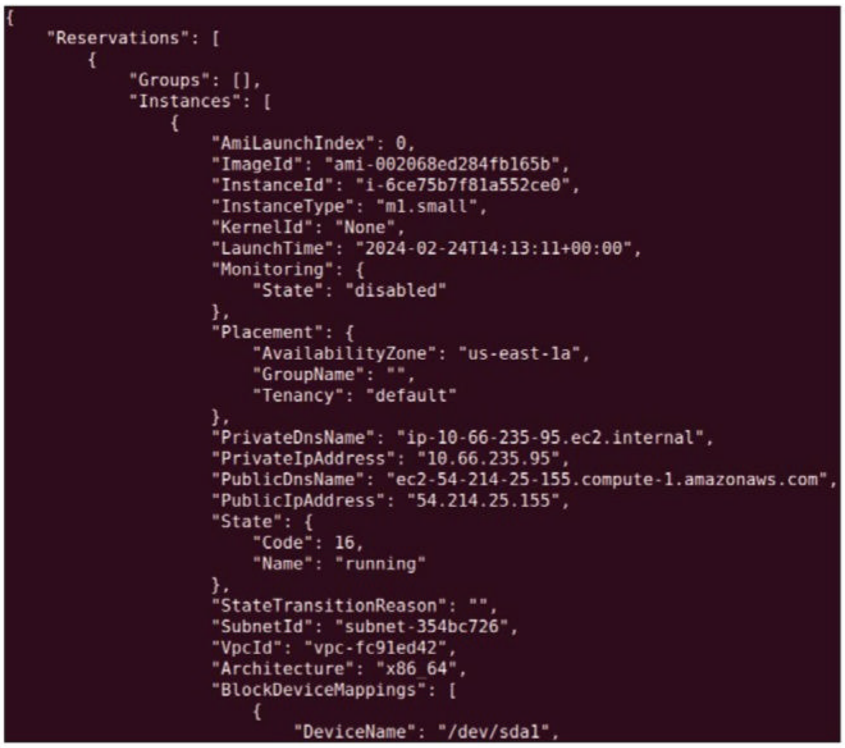
```
docker build -f Dockerfile_TerraformWS . ➥
             -t terraformws
```

Next, create the YAML file in Listing 24 and execute the command

```
docker run --rm ➥
  -v /var/run/docker.sock:/var/run/➥
     docker.sock:ro ➥
  -v ./terraform_localstack_stack.yml:➥
     /etc/compose/terraform_localstack_➥
     stack.yml:ro docker docker compose ➥
  -f /etc/compose/terraform_localstack_➥
     stack.yml up -d
```

to bring up the necessary containers. You are now fully ready to demonstrate the power of LocalStack to create AWS resources with the official Terraform modules. The first demo uses the AWS Relational Database Service (Amazon RDS) designed to simplify the setup, operation, and scaling of a relational database for use in applications. Administration processes such as patching the database software, backing up databases, and enabling point-in-time recovery are managed automatically. To start, execute

```
docker run --rm ➥
  -v /var/run/docker.sock:/var/run/➥
     docker.sock:ro ➥
  -v ./terraform_localstack_stack.yml:➥
     /etc/compose/terraform_localstack_➥
     stack.yml:ro docker docker compose ➥
  -f /etc/compose/terraform_localstack_➥
     stack.yml exec terraformws sh ➥
  -c "tfrmawstst rds ➥
      complete-postgres apply"
```



```
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-002068ed284fb165b",
                    "InstanceId": "i-6ce75b7f81a552ce0",
                    "InstanceType": "m1.small",
                    "KernelId": "None",
                    "LaunchTime": "2024-02-24T14:13:11+00:00",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-east-1a",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
                    "PrivateDnsName": "ip-10-66-235-95.ec2.internal",
                    "PrivateIpAddress": "10.66.235.95",
                    "PublicDnsName": "ec2-54-214-25-155.compute-1.amazonaws.com",
                    "PublicIpAddress": "54.214.25.155",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "StateTransitionReason": "",
                    "SubnetId": "subnet-354bc726",
                    "VpcId": "vpc-fc91ed42",
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/sda1",
```

Figure 6: AWS instance properties returned by the Moto server.

Terraform first shows which AWS resources will be created for a complete RDS Postgres module (Figure 7) and then creates everything.

Next, try your hand with the very popular Amazon Redshift service used by tens of thousands of customers every day to modernize their data analytics workloads. Just execute the command

```
docker run --rm ➥
  -v /var/run/docker.sock:/var/run/➥
     docker.sock:ro ➥
  -v ./terraform_localstack_stack.yml:➥
     /etc/compose/terraform_localstack_➥
     stack.yml:ro docker docker compose ➥
  -f /etc/compose/terraform_localstack_➥
     stack.yml exec terraformws sh ➥
  -c "tfrmawstst redshift complete apply"
```

Listing 21: Dockerfile_TerraformWS

```
FROM hashicorp/terraform

COPY aws_override.tf /etc/terraform/

COPY terraform_awsemu_test.sh /usr/local/bin/tfrmawstst
```

Listing 22: aws_override.tf

```
provider "aws" {
  region                      = "us-west-1"
  access_key                  = "mock_access_key"
  s3_use_path_style           = true
  secret_key                  = "mock_secret_key"
  skip_credentials_validation = true
  skip_metadata_api_check     = true
  skip_requesting_account_id  = true

  endpoints {
    acm            = "http://localstack:4566"
    apigateway     = "http://localstack:4566"
    cloudformation = "http://localstack:4566"
    cloudwatch     = "http://localstack:4566"
    dynamodb       = "http://localstack:4566"
    es             = "http://localstack:4571"
    ec2            = "http://localstack:4566"
    firehose       = "http://localstack:4566"
    iam            = "http://localstack:4566"
    kinesis        = "http://localstack:4566"
    kms            = "http://localstack:4566"
    lambda         = "http://localstack:4566"
    route53        = "http://localstack:4566"
    redshift       = "http://localstack:4566"
    s3             = "http://localstack:4566"
    secretsmanager = "http://localstack:4566"
    ses            = "http://localstack:4566"
    sns            = "http://localstack:4566"
    sqs            = "http://localstack:4566"
    ssm            = "http://localstack:4566"
    stepfunctions  = "http://localstack:4566"
    sts            = "http://localstack:4566"
  }
}
```

and profit from playing with Redshift locally without spending even a penny.

The next demo is for DynamoDB, a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. If you execute

```
docker run --rm ↩
  -v /var/run/docker.sock:/var/run/↩
     docker.sock:ro ↩
-v ./terraform_localstack_stack.yml:↩
   /etc/compose/terraform_localstack_↩
   stack.yml:ro docker docker compose ↩
-f /etc/compose/terraform_localstack_↩
   stack.yml exec terraformws sh ↩
-c "tfrmawstst dynamodb-table ↩
    autoscaling apply"
```

Listing 23: `terraform_awsemu_test.sh`

```
#! /bin/sh

TAMD=${1}
EXDR=${2}
OPRN=${3}
NUMOPTNMX=4
TRFRMCNFL=${TRFRMCNFL:-'/etc/terraform/aws_override.tf'}
TRFRMWLOC=${TRFRMWLOC:-'/tmp/terraform'}

printUsage() {
  cat <<EOF
 Usage: $(basename "${0}") <terraform aws module> <example dir>
        <plan|apply|destroy>
EOF
  exit 0
}

parseArgs() {
  if [[ $# -gt ${NUMOPTNMX} ]]
  then
    printUsage
  fi

  if [[ "${OPRN}" != "plan" ]] && [[ "${OPRN}" != "apply" ]] && [[
       "${OPRN}" != "destroy" ]]
  then
    printUsage
  fi
}

preProcess() {
  if [[ -d "/tmp/terraform-aws-${TAMD}" ]]
  then
    cd "/tmp/terraform-aws-${TAMD}"
    git pull
    cd "examples/${EXDR}/"
    terraform init -input=false
  else
    if git clone "https://github.com/terraform-aws-modules/
      terraform-aws-${TAMD}.git" "/tmp/terraform-aws-${TAMD}"
    then
      if [[ -e "${TRFRMCNFL}" ]]
      then
        if [[ -d "/tmp/terraform-aws-${TAMD}/examples/${EXDR}/" ]]
        then
          if cp -f "${TRFRMCNFL}" "/tmp/terraform-aws-${TAMD}/
            examples/${EXDR}/"
          then
            cd "/tmp/terraform-aws-${TAMD}/examples/${EXDR}/"
            terraform init -input=false
          fi
        fi
      fi
    fi
  fi
}

runOprtn() {
  cd "/tmp/terraform-aws-${TAMD}/examples/${EXDR}/"

  if [[ "${OPRN}" = "apply" ]] || [[ "${OPRN}" = "destroy" ]]
  then
    terraform ${OPRN} -input=false -auto-approve
  else
    terraform "${OPRN}"
  fi
}

main() {
  parseArgs
  preProcess
  runOprtn
}

main 2>&1
```

Listing 24: `terraform_localstack_stack.yml`

```
services:

  localstack:
    image: localstack/localstack:${LSTKTAG:-latest}
    container_name: localstack
    hostname: localstack
    volumes:
      - "${TMPDIR:-/tmp/terraform}:/tmp/terraform"
      - /var/run/docker.sock:/var/run/docker.sock:ro
    ports:
      - "14566:4566"
      - "14510-14559:4510-4559"
    environment:
      - DEBUG=${DEBUG:-0}
    healthcheck:
      test: ["CMD", "curl", "-I", "localhost:4566/_localstack/health"]
      interval: 5s
      timeout: 3s
      retries: 5
    restart: unless-stopped

  terraformws:
    image: terraformws:${TFRMTAG:-latest}
    container_name: terraformws
    hostname: terraformws
    volumes:
      - ./aws.tf:/etc/terraform/aws.tf:ro
      - ./terraform_awsemu_test.sh:/usr/local/bin/tfrmawstst:ro
    entrypoint: ash
    command: "-c 'while true; do sleep 5; done'"
    depends_on:
      localstack:
        condition: service_healthy

networks:
  default:
    name: awsmockemu-demo
    external: true
```

you create an AWS DynamoDB table with autoscaling and let Terraform create all the necessary cloud resources locally (Figure 8).

To wrap up this quick exploration of LocalStack, the next example uses the Elastic Kubernetes Service (Amazon EKS), a managed Kubernetes service to run Kubernetes in the AWS cloud and on-premises data centers. The command

```
docker run --rm ⮐
  -v /var/run/docker.sock:/var/run/⮐
     docker.sock:ro ⮐
  -v ./terraform_localstack_stack.yml:⮐
     /etc/compose/terraform_localstack_⮐
     stack.yml:ro docker docker compose ⮐
  -f /etc/compose/terraform_localstack_⮐
     stack.yml exec terraformws sh ⮐
  -c "tfrmawstst eks eks_managed_node_⮐
      group apply"
```

illustrates how easy it is to create any cloud service locally with LocalStack. All these applications are just the tip of the LocalStack iceberg which is full of great functionalities for AWS services, along with a number of integrations, extensions, tools, and more. Please browse through LocalStack's extensive documentation [7] to become familiar with its power.

Conclusion

Mocking and emulation are essential ingredients in the modern API-driven, cloud-native world. Sooner or later, development, QA, and other experimental environments require cost-effective solutions to unblock development and operations teams. GCP provides free official emulators to unblock teams. In the AWS world, Moto is an elegant library to mock cloud services. LocalStack is an enterprise-level solution for full emulation of hundreds of AWS services. These effective solutions will enable your company to move fast on the cloud-native journey without burning holes in your pockets. ■

Info

[1] Code for this article: [https://linuxnewmedia.thegood.cloud/s/9nFQcFb2p8oRMEJ]

[2] Pub/Sub: [https://cloud.google.com/pubsub/docs/emulator]

[3] Bigtable quickstart: [https://cloud.google.com/bigtable/docs/create-instance-write-data-cbt-cli]

[4] Cloud Spanner: [https://cloud.google.com/spanner/docs/emulator]

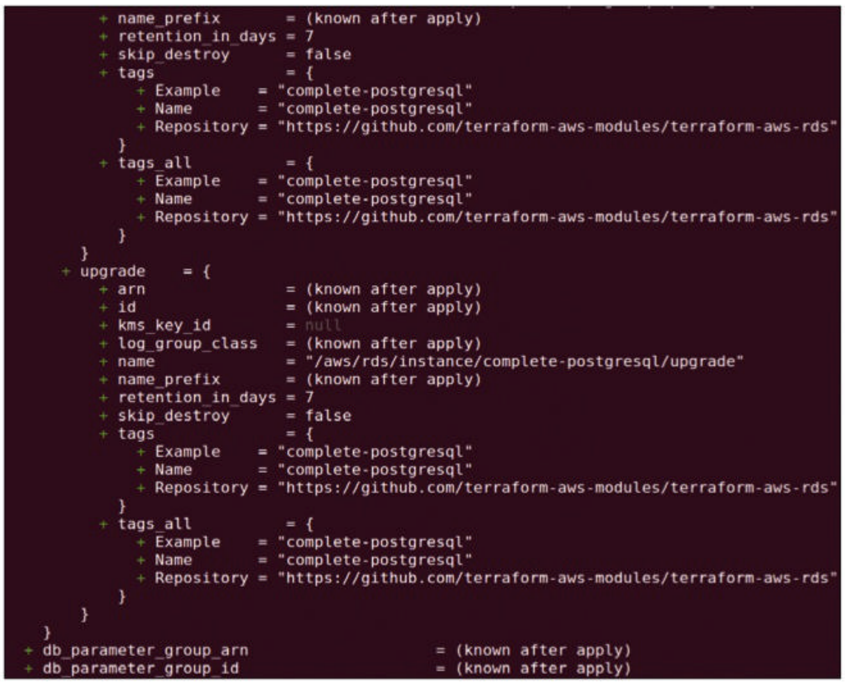
[5] Moto: [https://docs.getmoto.org/en/latest/]

[6] Moto services implementation coverage: [https://github.com/getmoto/moto/blob/master/IMPLEMENTATION_COVERAGE.md]

[7] LocalStack: [https://docs.localstack.cloud/overview/]

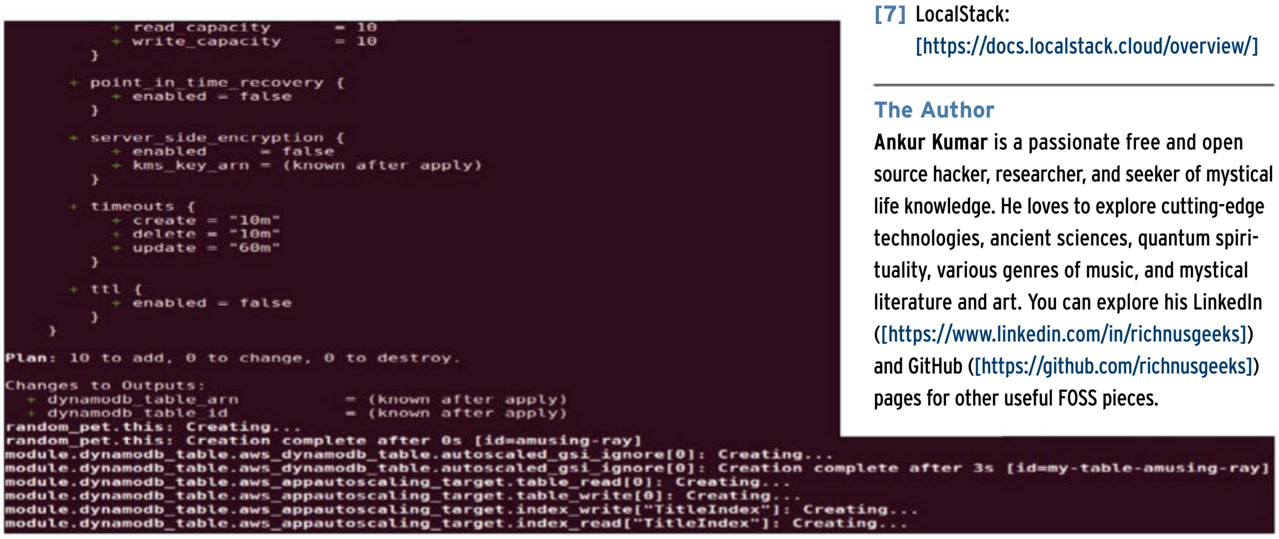
The Author

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



```
        + name_prefix        = (known after apply)
        + retention_in_days = 7
        + skip_destroy       = false
        + tags               = {
            + Example    = "complete-postgresql"
            + Name       = "complete-postgresql"
            + Repository = "https://github.com/terraform-aws-modules/terraform-aws-rds"
          }
        + tags_all           = {
            + Example    = "complete-postgresql"
            + Name       = "complete-postgresql"
            + Repository = "https://github.com/terraform-aws-modules/terraform-aws-rds"
          }
      }
    + upgrade    = {
        + arn                = (known after apply)
        + id                 = (known after apply)
        + kms_key_id         = null
        + log_group_class    = (known after apply)
        + name               = "/aws/rds/instance/complete-postgresql/upgrade"
        + name_prefix        = (known after apply)
        + retention_in_days = 7
        + skip_destroy       = false
        + tags               = {
            + Example    = "complete-postgresql"
            + Name       = "complete-postgresql"
            + Repository = "https://github.com/terraform-aws-modules/terraform-aws-rds"
          }
        + tags_all           = {
            + Example    = "complete-postgresql"
            + Name       = "complete-postgresql"
            + Repository = "https://github.com/terraform-aws-modules/terraform-aws-rds"
          }
      }
  }
+ db_parameter_group_arn                  = (known after apply)
+ db_parameter_group_id                   = (known after apply)
```

Figure 7: **Terraform in action for an RDS Postgres module.**



```
          + read_capacity       = 10
          + write_capacity      = 10
        }

      + point_in_time_recovery {
          + enabled = false
        }

      + server_side_encryption {
          + enabled     = false
          + kms_key_arn = (known after apply)
        }

      + timeouts {
          + create = "10m"
          + delete = "10m"
          + update = "60m"
        }

      + ttl {
          + enabled = false
        }
    }

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

Changes to Outputs:
  + dynamodb_table_arn              = (known after apply)
  + dynamodb_table_id               = (known after apply)
random_pet.this: Creating...
random_pet.this: Creation complete after 0s [id=amusing-ray]
module.dynamodb_table.aws_dynamodb_table.autoscaled_gsi_ignore[0]: Creating...
module.dynamodb_table.aws_dynamodb_table.autoscaled_gsi_ignore[0]: Creation complete after 3s [id=my-table-amusing-ray]
module.dynamodb_table.aws_appautoscaling_target.table_read[0]: Creating...
module.dynamodb_table.aws_appautoscaling_target.table_write[0]: Creating...
module.dynamodb_table.aws_appautoscaling_target.index_write["TitleIndex"]: Creating...
module.dynamodb_table.aws_appautoscaling_target.index_read["TitleIndex"]: Creating...
```

Figure 8: **Terraform in action for an autoscaled *dynamodb-table* module. Note that Terraform shows the Plan for adding, changing, and destroying before proceeding.**

Kubernetess networking in the kernel

Core Connection

Cilium and eBPF put Kubernetes networking down in the kernel where it belongs. By Abe Sharp

For many self-managed Kubernetes clusters, networking is little more than an afterthought. Administrators install whichever network plugin they've used in the past, and as long as pods and services can be contacted, that's the end of the matter. In this article, I describe the networking requirements of a Kubernetes cluster and how the Container Network Interface (CNI) fosters innovation and choice in this vital area.

I focus on the open source Cilium network plugin, which is one of a few CNI choices that leverages eBPF (the successor to the Berkeley Packet Filter) to provide high performance, control, and observability. You'll install Cilium into a test cluster and compare its performance in unencrypted and encrypted forms with that of Flannel, implement network policies, and observe their effectiveness with the help of Hubble, Cilium's companion user interface (UI).

Networking in Kubernetes

Kubernetes is sometimes described as an orchestration *layer*, and that term is helpful when you think of deploying an application in a pod (or container) whose environment is abstracted from the underlying cluster of physical nodes, to the point where you don't have to know or care about those nodes. To realize this abstraction, any pod in the cluster should be able to communicate with any other pod as though they were independent "hosts" on a routable IP network, which offers many advantages over other ways that containers can interact with networks (e.g., mapping ports on a host to ports in the container) by increasing capacity in the address and port spaces.

Another requirement is that external users should be able to access application pods by consistent ingress points regardless of which node the pods are currently running on, and with no intervention required if a pod is rescheduled from one host to another. In this case, Services, another vital object in Kubenetes networking, comes into play.

Administrators might want to encrypt pod-to-pod traffic, as well, to further ensure the cluster's isolation from external influences (virtual extensible LAN (VxLAN) traffic is trivial to collect and snoop on, as you'll see later); to exert granular control over which pods can talk to which other pods and for what purposes, especially in a multitenant environment; and to have detailed observability of packet flows and network policy decisions.

Kubernetes delegates the implementation of all of these requirements to a network plugin (or add-on), with which container runtimes interact by means of the CNI, a Cloud Native Computing Foundation (CNCF) construct intended to foster choice and innovation in this important area of orchestration. The construct defines a simple set of methods that a network plugin must implement.

Network plugins must be able to provide interfaces and IP addresses to newly created pods (ADD method), handle the removal of a pod and the freeing up of its IP address (DEL method), and track the use of an IP (CHECK method). As long as a network plugin implements CNI correctly, it can be installed in a Kubernetes cluster and have complete freedom over how it gets the job done. For example, some network plugins have no way to encrypt VxLAN traffic, and others don't implement Kubernetes *NetworkPolicy* objects. (Cilium does both, and much more!)

Self-managed Kubernetes clusters encompass a broad scope, with many control plane implementation choices available to the administrator. In a "vanilla" `kubeadm` install, the core control plane services of `kube-apiserver`, `etcd`, `kube-scheduler`, and `kube-controller-manager` run as static pods on the master host and provide their services over the master host's own network interfaces. For example, `kube-apiserver` uses default port 6443 of the master host's primary interface. Therefore, the static pods come to the Running state before any CNI has been installed. However, other pods that need to use the *pod network* cannot be scheduled until a network plugin has been installed on the cluster. If you've ever followed the official `kubeadm` documentation, you'll have noticed a step after running the `kubeadm init` command that directs you to the Addons page [1] and tells you not to proceed until you've selected and installed a network plugin from a bewildering list of 19 possible options, all described in completely different terms. How do you choose? Unless you have vendor-specific requirements, then safe general choices include Calico, Flannel, Weave, and Cilium, which is the focus of this article. All these choices use VxLAN for internode communication. VxLAN is a reliable choice because its only requirement of the underlay network is that Layer 3 (L3) connectivity must exist between all hosts and that each host must be able to receive UDP packets on port 8472.

Pods on the Same Node

When a node's container runtime (e.g., `containerd`) creates a new pod, it leverages the Linux namespace concept to create a dedicated network namespace for that pod, isolated from all the other network namespaces on the node, including the default network namespace (which is the one containing the host's primary interface). At this stage, the pod remains disconnected from the pod network – not much use for participating in the world of microservices! To connect the pod, the container runtime invokes the network plugin with the CNI. The network plugin uses IP address management (IPAM) to obtain a unique IP address within the host's pod network subnet. It creates a network interface within the pod's network namespace and assigns the IP address (Figure 1; eth0@if … boxes). The plugin creates an interface on the host's Linux bridge (think of a virtual L2 switch inside each host) and connects it to the pod's network interface (Figure 1; lxc…@ boxes). If you exec into the pod and run `ip a`, you'll see the pod's interface and none of the host's interfaces; likewise, if you run `ip a` on the node, you'll see the host's interfaces, including the bridge port interfaces – one corresponding to each pod. Now the pod can send IP packets to other pods on the same node by `arping` for their MAC addresses. I haven't found a straightforward way of figuring out which Linux bridge port is connected to which pod, but if you generate some traffic from one pod to another, you can look at the pod's ARP table,

```
kubectl exec mypod -- arp -an
```

and see the MAC address corresponding to the port on the Linux bridge to which the pod is connected.

Pods on Different Nodes

If you run,

```
kubectl get po -o wide
```

you'll note that each node's pods are in a subnet unique to that node. In the examples given in this article, the pod network is 192.168.0.0/16; the pods for worker 1 have IP addresses assigned from 192.168.1.0/24, the pods for worker 2 are in 192.168.2.0/24, and so on. Figure 1 shows the interface on each node's Linux bridge called `cilium_host`, the default gateway over which traffic destined for another node is sent. Run `ip route` inside a pod and on the node to see that this is so. Although it all makes sense in the context of the pod network, clearly, a packet being sent from a pod on one node to a pod on another node has to traverse the "real" network between the hosts somehow. How

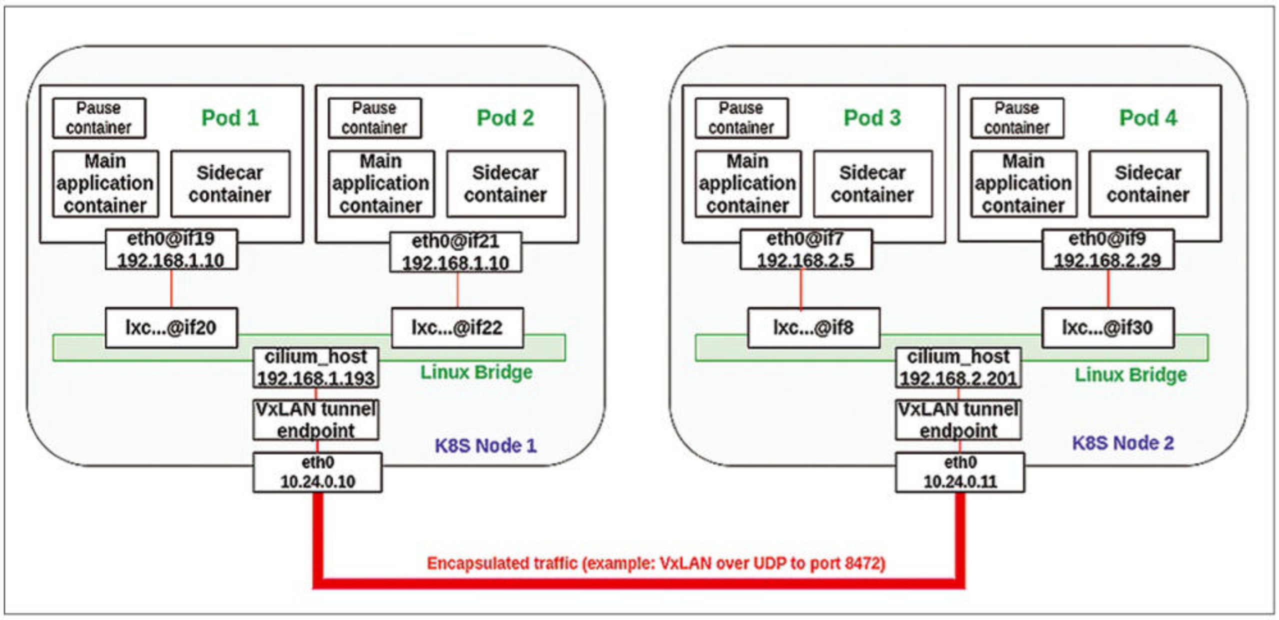


Figure 1: A Kubernetes network with a Linux bridge connecting pod namespaces and overlay networking for internode communication.

does this happen? There are two possible answers: native routing or encapsulation.

1. **Native routing.** Pod IP addresses exist in the same network as used by the hosts themselves, so the packets can traverse the network in their native form, just like a packet generated by the node. Interhost routing is performed by the native routing tables on the hosts. The size of that native subnet will limit the total number of pods that can be created in the cluster. With the Kubernetes cluster and the host network both trying to manage the same pool of IP addresses, conflicts could arise.
2. **Encapsulation.** To avoid the complexity and conflict potential of getting your pods to exist in the same network as the real hosts, you can leverage the kernel's built-in VxLAN functionality (or an alternative such as WireGuard or IPsec) to encapsulate the packets before sending them across the physical network to the node. Encapsulation is the default mode for most CNIs, including Cilium, so all of the following examples will use encapsulated traffic between nodes. Figure 1 shows encapsulated traffic (which could be TCP traffic on the pod network) being sent between nodes with UDP packets.

Listing 1: netshoot Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: netshoot
  name: netshoot
spec:
  replicas: 1
  selector:
    matchLabels:
      app: netshoot
  template:
    metadata:
      labels:
        app: netshoot
    spec:
      containers:
      - name: netshoot
        image: nicolaka/netshoot
        command: ["/bin/bash"]
        args: ["-c", "tail -f /dev/null"]
```

Baseline CNI and Testing Bandwidth

For the practical part of this article, I'll use the `iperf` tool for a general indication of Cilium's performance compared with that of Flannel. iPerf is a common tool for testing network throughput by means of a "server" mode at one end of the connection under test and a "client" mode at the other. It's not particularly representative of real-world performance, but it's a reproducible test and gives hard numbers as output, so it's a useful indicator. Earlier, I mentioned the point in a `kubeadm` installation, for which a network plugin is selected and installed. With my test cluster initialized with `kubeadm`, I installed Flannel directly from GitHub:

```
kubectl apply ↩
  -f https://github.com/flannel-io/↩
     flannel/releases/latest/download/↩
     kube-flannel.yml
```

When this command completed successfully and all pods were in the Running state, I created a simple deployment with a `netshoot` container (Listing 1). The `netshoot` [2] image comes with every network troubleshooting tool you might need (e.g., `tcpdump`, `iperf`, `netcat`, and various DNS tools). I then exposed port 5001 (the iPerf server port) on that deployment with a Kubernetes service:

```
kubectl expose deployment netshoot ↩
  --port=5001 --target-port=5001
```

The point of adding the Service (rather than having the iPerf client connect directly to the iPerf server's pod IP) is to make sure that Kube-Proxy's iptables rules are included in the routing of the test traffic (for the Flannel test) and to see whether Cilium's alternative approach to service endpoint mapping has any notable effect. Running

```
iptables -t nat --list KUBE-SERVICES
```

will show the service mapping rules created by Kube-Proxy.

I exec'd inside the `netshoot` pod and started the iPerf server with:

```
iperf -s
```

In this state, iPerf listens for incoming test requests from iPerf clients. The iPerf client was then run as a temporary shell inside another `netshoot` pod. An important control for the experiment is that the iPerf client pod should be on a different worker node from the pod running the iPerf server so that the test traffic runs across the underlay network and therefore has to be encapsulated. The bandwidth results for Flannel (for my particular LAN conditions, with no special effort paid to optimizing maximum transmission unit (MTU) settings or anything else) were up to 1.7Gbps.

Replacing Flannel with Cilium

The Cilium documentation describes a thorough process for migrating to the Cilium network plugin in production clusters where you don't want to disrupt existing workloads and you want to have connectivity between pre- and post-migration workloads [3]. If that's not a concern, you can migrate from Flannel to Cilium as follows:

```
kubectl delete -f kube-flannel.yaml
kubectl -n kube-system delete ds kube-proxy
kubectl -n kube-system delete cm kube-proxy
rm /etc/cni/net.d     # all hosts
reboot                # all hosts
```

After the hosts have been rebooted, all pods that require CNI will be in a Pending or Unknown state. Now install the Cilium command-line interface (CLI) on your master host as shown in the quick-start of the official Cilium documentation, then create a `values.yaml` file that will be used to configure the Cilium installation (Listing 2).
Next, use the Cilium CLI to install Cilium:

```
cilium install --values values.yaml ↩
               --version 1.15.2
watch kubectl get po -A
```

After the new Cilium agent pods are up and running, the pre-existing pods (e.g., the `netshoot` deployment's pod from the earlier test) will gradually return to the Running state. To re-run my performance test, I restarted the iPerf server in the pod of my `netshoot` deployment. It was contactable on the same Service IP address as before, but the `KUBE-SERVICES` iptables chain was gone, because Cilium had completely replaced Kube-Proxy and its iptables rules with its own eBPF programs (see the "How Network Plugins Use eBPF" box). Repeating the same iPerf bandwidth test gave marginally better results, at up to 1.8Gbps – again, taking care to ensure that the iPerf client pod was not running on the same worker node as the iPerf server.

Enabling WireGuard Encryption in Cilium

Up to this point, the encapsulation mode for internode traffic, for both the Flannel and the Cilium tests, has been standard VxLAN (as built into nearly every modern Linux kernel) with no encryption. Intercepting and decoding such traffic is trivial: Anyone able to capture VxLAN traffic on the underlay network can then decode it with the use of Wireshark's VxLAN analyzer to see the plain text content of the traffic between the pods (Figure 2). Many production Kubernetes workloads don't implement TLS at the pod level, expecting the cluster traffic to be secured automatically, but don't take that outcome for granted.

How Network Plugins Use eBPF

The eBPF technology allows programs to be run in kernel space without any need to recompile the kernel or compile and load a kernel module. Network plugins such as Cilium can leverage eBPF to execute packet-routing code directly in the network device drivers, reducing the need to analyze packets elsewhere in the operating system. For example, Cilium can replace the **kube-proxy** addon (which manages mappings between Service and Pod IPs by means of iptables rules) with an eBPF program that does the same job entirely in the kernel.

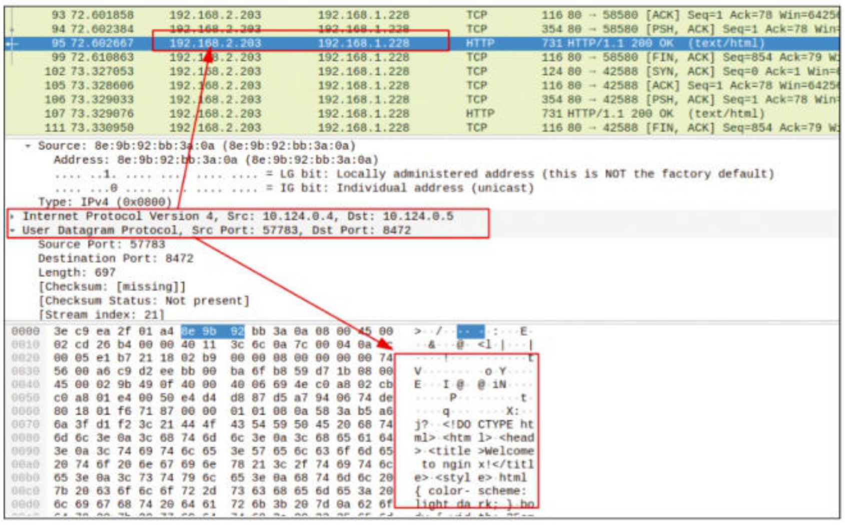
Luckily, enabling WireGuard encryption in Cilium is easy. You could have enabled it from the outset in the `encryption` section of `values.yaml`; however, because Cilium is already running, you can just use

```
kubectl -n kube-system ⮧
  edit cm cilium-config
```

then add `enable-wireguard: "true"` to the data list, and restart the Cilium agent daemonset. After that, internode traffic will be encrypted by the kernel's WireGuard VPN function and sent to the appropriate destination host on port 51871. You can also verify that encryption is in use with the `cilium-dbg` tool inside each Cilium agent pod:

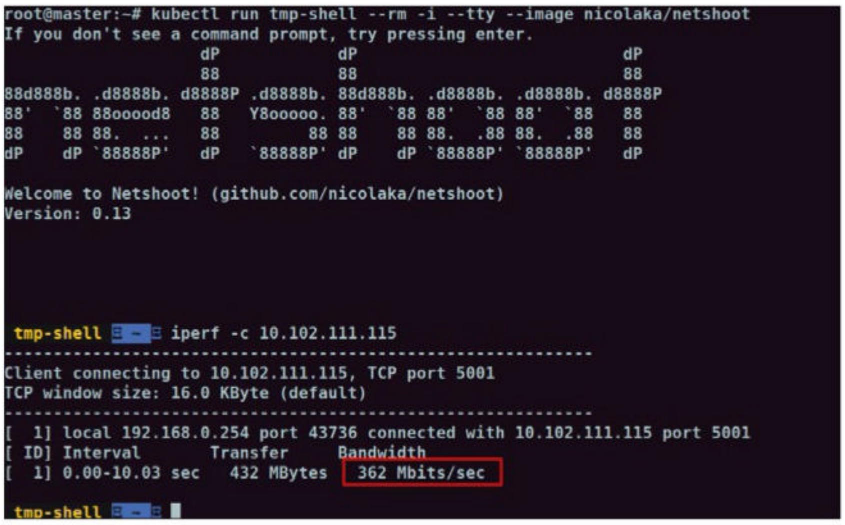
Listing 2: values.yaml

```
cluster:
  name: kubernetes
k8sServiceHost: 10.124.0.3
k8sServicePort: 6443
kubeProxyReplacement: strict
operator:
  replicas: 1
ipam:
  mode: "cluster-pool"
  operator:
    clusterPoolIPv4PodCIDRList: ["192.168.0.0/16"]
routingMode: tunnel
tunnelProtocol: vxlan
encryption:
  enabled: false
  nodeEncryption: false
```



```
   93 72.601858   192.168.2.203   192.168.1.228   TCP    116 80 → 58580 [ACK] Seq=1 Ack=78 Win=6425
   94 72.602384   192.168.2.203   192.168.1.228   TCP    354 80 → 58580 [PSH, ACK] Seq=1 Ack=78 Win
   95 72.602667   192.168.2.203   192.168.1.228   HTTP   731 HTTP/1.1 200 OK  (text/html)
   99 72.610863   192.168.2.203   192.168.1.228   TCP    116 80 → 58580 [FIN, ACK] Seq=854 Ack=79 W
  102 73.327053   192.168.2.203   192.168.1.228   TCP    124 80 → 42588 [SYN, ACK] Seq=0 Ack=1 Win=
  105 73.328606   192.168.2.203   192.168.1.228   TCP    116 80 → 42588 [ACK] Seq=1 Ack=78 Win=6425
  106 73.329033   192.168.2.203   192.168.1.228   TCP    354 80 → 42588 [PSH, ACK] Seq=1 Ack=78 Win
  107 73.329076   192.168.2.203   192.168.1.228   HTTP   731 HTTP/1.1 200 OK  (text/html)
  111 73.330950   192.168.2.203   192.168.1.228   TCP    116 80 → 42588 [FIN, ACK] Seq=854 Ack=79 W

  ▾ Source: 8e:9b:92:bb:3a:0a (8e:9b:92:bb:3a:0a)
      Address: 8e:9b:92:bb:3a:0a (8e:9b:92:bb:3a:0a)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.124.0.4, Dst: 10.124.0.5
▾ User Datagram Protocol, Src Port: 57783, Dst Port: 8472
    Source Port: 57783
    Destination Port: 8472
    Length: 697
    [Checksum: [missing]]
    [Checksum Status: Not present]
    [Stream index: 21]

0000  3e c9 ea 2f 01 a4 8e 9b  92 bb 3a 0a 08 00 45 00   >··/······:···E·
0010  02 cd 26 b4 00 00 40 11  3c 6c 0a 7c 00 04 0a      ··&···@·<l·|···|
0020  00 05 e1 b7 21 18 02 b9  00 00 08 00 00 00 00 74   ····!··········t
0030  56 00 a6 c9 d2 ee bb 00  ba 6f b8 59 d7 1b 08 00   V········o·Y····
0040  45 00 02 9b 49 0f 40 00  40 06 69 4e c0 a8 02 cb   E···I·@·@·iN····
0050  c0 a8 01 e4 00 50 e4 d4  d8 87 d5 a7 94 06 74 de   ·····P········t·
0060  80 18 01 f6 71 87 00 00  01 01 08 0a 58 3a b5 a6   ····q·······X:··
0070  6a 3f d1 f2 3c 21 44 4f  43 54 59 50 45 20 68 74   j?··<!DO CTYPE ht
0080  6d 6c 3e 0a 3c 68 74 6d  6c 3e 0a 3c 68 65 61 64   ml>·<htm l>·<head
0090  3e 0a 3c 74 69 74 6c 65  3e 57 65 6c 63 6f 6d 65   >·<title >Welcome
00a0  20 74 6f 20 6e 67 69 6e  78 21 3c 2f 74 69 74 6c    to ngin x!</titl
00b0  65 3e 0a 3c 73 74 79 6c  65 3e 0a 68 74 6d 6c 20   e>·<styl e>·html 
00c0  7b 20 63 6f 6c 6f 72 2d  73 63 68 65 6d 65 3a 20   { color- scheme: 
00d0  6c 69 67 68 74 20 64 61  72 6b 3b 20 7d 0a 62 6f   light da rk; }·bo
```

Figure 2: Wireshark decodes VxLAN traffic on the underlay network to see pod communications in plain text.



```
root@master:~# kubectl run tmp-shell --rm -i --tty --image nicolaka/netshoot
If you don't see a command prompt, try pressing enter.
                        dP            dP                           dP
                        88            88                           88
88d888b. .d8888b. d8888P .d8888b. 88d888b. .d8888b. .d8888b. d8888P
88'  `88 88ooood8   88   Y8ooooo. 88'  `88 88'  `88 88'  `88   88
88    88 88.  ...   88         88 88    88 88.  .88 88.  .88   88
dP    dP `88888P'   dP   `88888P' dP    dP `88888P' `88888P'   dP

Welcome to Netshoot! (github.com/nicolaka/netshoot)
Version: 0.13



 tmp-shell  ~  iperf -c 10.102.111.115
------------------------------------------------------------
Client connecting to 10.102.111.115, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  1] local 192.168.0.254 port 43736 connected with 10.102.111.115 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.00-10.03 sec   432 MBytes   362 Mbits/sec

 tmp-shell  ~ 
```

Figure 3: iPerf bandwidth results by Cilium with WireGuard encryption enabled.

```
exec cilium-6t6ss -- cilium-dbg status ⮐
  | grep Encryption
```

After enabling encryption on my test cluster, I repeated the same iPerf bandwidth test. Bandwidth dropped drastically, to less than 400Mbps (Figure 3) because of the extra overhead in encrypting and decrypting the content of each encapsulated packet. As you can see, security has a price.

Network Policy with Cilium

`NetworkPolicy` is a standard Kubernetes object that allows ingress and egress traffic to be restricted at Layers 3 and 4. An ingress `NetworkPolicy` allows a source (specified by IP address, namespace name, pod labels, or a combination thereof) to access numbered ports on pods containing specified labels running within the namespace to which the network policy was added.

When no network policies are applied to a namespace, the namespace's traffic is not restricted; however, as soon as one policy is created for ingress or egress, a `default deny` policy kicks in for that traffic direction, meaning that rules have to be created to allow all permitted traffic.

For `NetworkPolicy` to be enforced, the cluster's network plugin must implement it; however, not all plugins do. For example, if your cluster uses the Flannel plugin, its `NetworkPolicy` objects will not have any effect because Flannel doesn't support them.

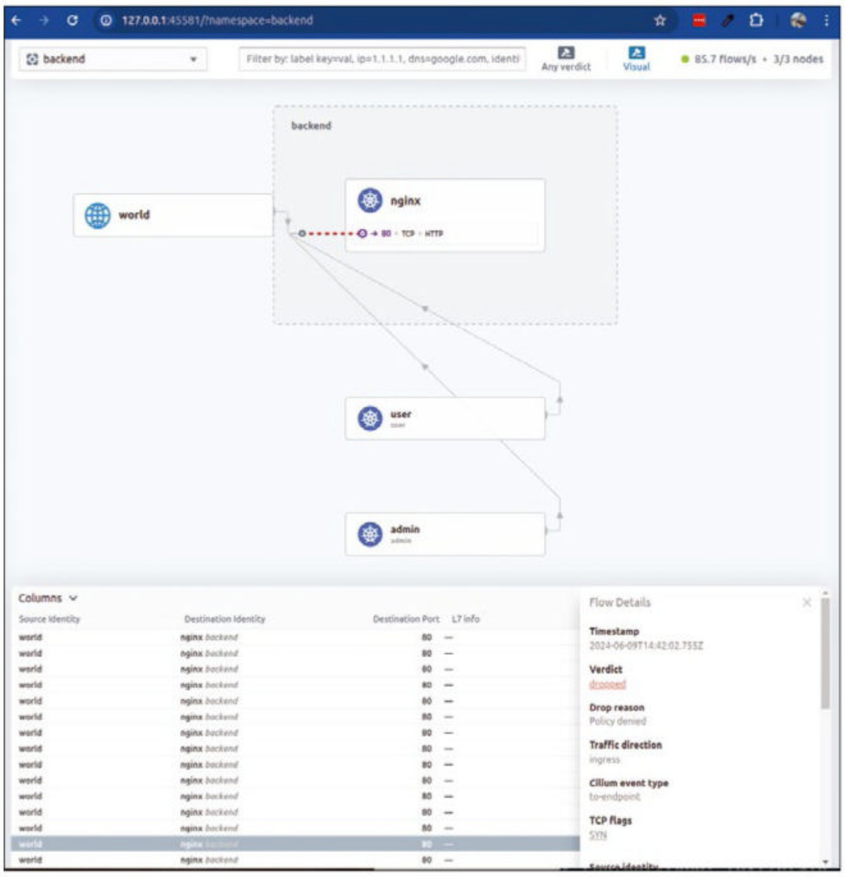
Cilium enforces both standard network policies and its custom resource `CiliumNetworkPolicy`, which is a superset of `NetworkPolicy` that adds support for L7 (application layer) policies with the aid of an Envoy proxy running inside the Cilium agent (the Cilium `daemonset` pod running on each node). L7 policies implement rules according to the application payload of packets, and Cilium supports them for HTTP, gRPC, Kafka, and DNS hostnames (which is useful, e.g., for limiting access to external websites and APIs in egress rules).

Imagine that an API-based application is running inside your cluster and you want to allow only pods running in a specified namespace to access particular endpoints of that API. The `CiliumNetworkPolicy` object in Listing 3 shows how you could allow pods in an `admin` namespace to access the API endpoints of pods in a `backend` namespace (represented by the `/api/v1/.*` regex in the HTTP rules), and allow all pods in a `user` namespace to access the docs page of the back-end pods, but not the API.

Understanding YAML arrays and lists is a great help in interpreting this policy. A single dash (-) can make the difference between a policy that allows traffic meeting the constraints of "this rule *and* that rule," compared with "this rule *or* that

Listing 3: Example CiliumNetworkPolicy

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-api-from-admin
  namespace: backend
spec:
  endpointSelector:
    matchLabels:
      app: nginx
  ingress:
  - fromEndpoints:
    - matchLabels:
        io.kubernetes.pod.namespace: admin
        app: adminapp
    toPorts:
    - ports:
      - port: "80"
        protocol: TCP
      rules:
        http:
        - method: GET
          path: /api/v1/.*
        - method: GET
          path: /docs
  - fromEndpoints:
    - matchLabels:
        io.kubernetes.pod.namespace: user
    toPorts:
    - ports:
      - port: "80"
        protocol: TCP
      rules:
        http:
        - method: GET
          path: /
```



Source Identity	Destination Identity	Destination Port	L7 info
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—
world	nginx *backend*	80	—

Flow Details

Timestamp
2024-06-09T14:42:02.755Z

Verdict
dropped

Drop reason
Policy denied

Traffic direction
ingress

Cilium event type
to-endpoint

TCP flags
SYN

Source identity

Figure 4: Viewing traffic flows and policy outcomes in the Hubble user interface.

rule." In the example, the `ingress` array contains two elements. One refers to pods labeled with `adminapp` in the `admin` namespace and allows access to the `/docs` path and any path starting with `/api/v1`. The other allows pods in the `user` namespace to access only the `/docs` path; attempts to access any other path will result in an *Access denied* return by Cilium's Envoy instance.

One notable point is the difference in denial behaviors between an L3/L4 policy rule compared with an L7 policy rule. When a connection attempt is denied at the L3/L4 level, the ingress request packet does not receive any response, so the request hangs and times out as though you were trying to send a request to a non-listening port. When the request is denied at the L7 level, it fails instantly with an *Access denied* response to the HTTP request and a 403 (forbidden) status code.

Traffic Flows and Policy Decisions

Cilium comes with a tightly coupled observability service and UI called Hubble, which gives an at-a-glance view of all your cluster traffic flows, filterable by namespace (Figure 4). It's easy to see whether one entity tried to access another and (if successful) which network policy allowed that connection to happen. The commands

```
cilium hubble enable --ui
cilium status --wait
kubectl -n kube-system port-forward ⮐
          service/hubble-ui :80
```

enable Hubble and connect to its UI.

Conclusion

Pod networking is often the forgotten child of Kubernetes. Exploring the configuration and performance of popular network plugins such as Cilium can help you select and configure one that meets your performance and security needs. ■

Info

[1] Kubernetes-compatible network plugins: [https://kubernetes.io/docs/concepts/cluster-administration/addons/#networking-and-network-policy]

[2] netshoot utility: [https://github.com/nicolaka/netshoot]

[3] Cilium installation and migration documentation: [https://docs.cilium.io/en/latest/installation/k8s-install-migration/]

Author

Abe Sharp is a Distinguished Technologist at Hewlett Packard Enterprise, where he works on containerized AI and MLOps platforms in the Ezmeral business unit.

Advanced monitoring techniques for Azure VMs on Linux

Observer

We delve into advanced monitoring methods for enhancing the performance and security of Linux-based Azure virtual machines.

By Marcin Gastol

Logging and monitoring are essential components of managing IT infrastructure, particularly for Ubuntu virtual machines (VMs) hosted on Microsoft Azure. These tasks play a crucial role in ensuring the performance, security, and reliability of these systems. Proper logging captures detailed records of system events, providing insights that are indispensable for troubleshooting, performance tuning, and security monitoring. On the other hand, monitoring involves real-time observation of system metrics, enabling proactive management and quick response to potential issues.

In the dynamic environment of cloud computing, where VMs are subject to varying loads and potential security threats, logging and monitoring become even more critical. They allow IT professionals to maintain a high level of operational excellence by providing the necessary tools to detect, diagnose, and resolve issues promptly.

Setting Up Azure Monitor

Introduction to Azure Monitor

Azure Monitor is a comprehensive monitoring solution provided by Microsoft Azure that enables you to collect, analyze, and act on telemetry data from both cloud and on-premises environments. Its primary capability includes data collection from various sources, such as Azure resources, applications, and on-premises infrastructure, providing a unified view of your entire IT environment.

With integrated tools like Azure Log Analytics and Azure Application Insights, Azure Monitor allows you to query and analyze collected data to gain insights into performance, detect anomalies, and troubleshoot issues. The service supports the creation of alerts on the basis of specific conditions, enabling you to respond quickly to potential problems by triggering actions such as sending notifications or executing automated remediation workflows. Additionally, Azure Monitor offers rich visualization capabilities through customizable dashboards and workbooks, helping you monitor your resources and applications effectively.

The integration with Azure Automation and other services allows for automated responses to detected issues, enhancing operational efficiency and reducing manual intervention.

Creating an Azure Monitor Account

To set up Azure Monitor for your Ubuntu VMs, start by signing in to the Azure Portal. Once signed in, navigate to the Monitor section from the left-hand navigation pane to access the Azure Monitor Overview page. Next, create a Log Analytics workspace, which is essential for storing and querying log data, by going to the *Log Analytics workspaces* item under the Insights section and selecting the option to create a new workspace (Figure 1). You will be asked to choose the appropriate subscription and resource group, provide a name for the workspace, and select a region; review the details before finalizing the creation of the workspace.

After setting up the Log Analytics workspace, configure diagnostic settings for your Ubuntu VM by navigating to the VM's settings and selecting *Diagnostic settings* under the Monitoring section (Figures 2 and 3). Here, you add a new diagnostic setting, name it, choose the logs and metrics you want to collect, and specify the Log Analytics workspace you created earlier. To complete the configuration, save the settings.

Configuring Azure Monitor for Ubuntu VMs

Once the Log Analytics workspace is set up and diagnostic settings are configured, you need to ensure that Azure Monitor is properly configured to collect data from your Ubuntu VMs. To begin, install the Azure Monitor Agent on your Ubuntu VM, and then SSH into your VM and run the following commands to install the agent:

```
wget https://raw.githubusercontent.com/➘
     Microsoft/OMS-Agent-for-Linux/master/➘
     installer/scripts/onboard_agent.sh
sh onboard_agent.sh -w <WorkspaceID> ➘
                  -s <PrimaryKey>
```

Be sure to replace `<WorkspaceID>` and `<PrimaryKey>` with the values from your Log Analytics workspace. Next, verify the installation by checking the status of the agent with the command,

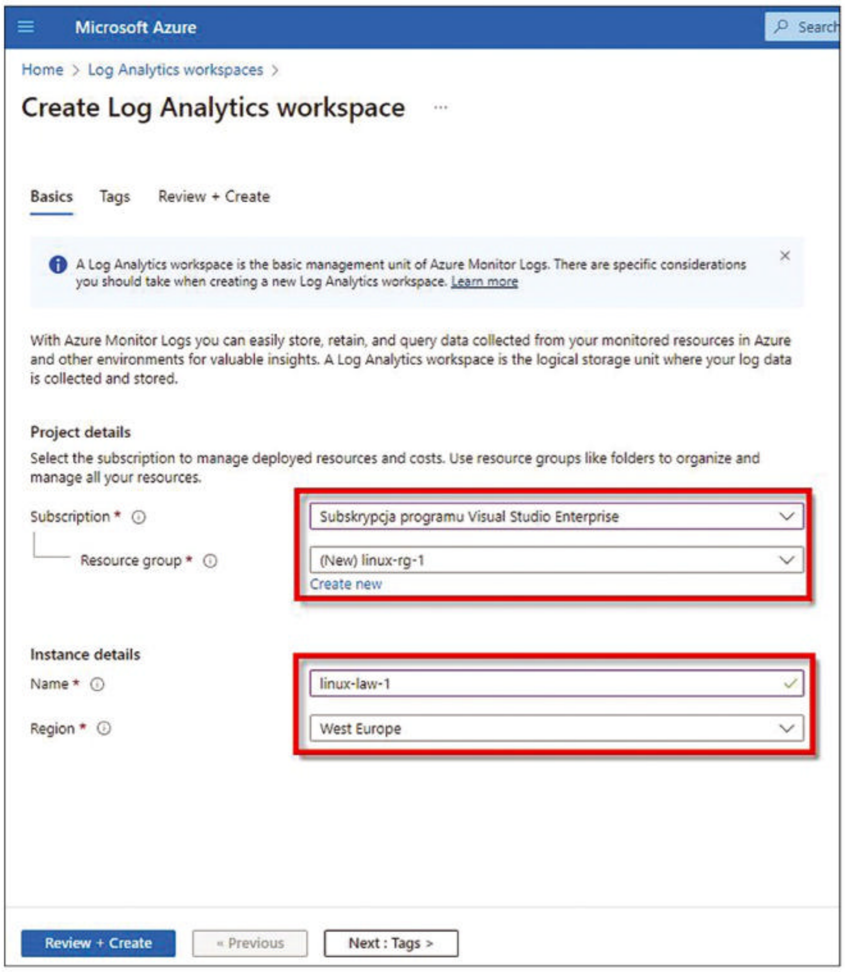
```
sudo /opt/microsoft/omsagent/bin/➘
  service_control status
```

and ensure that the agent is running and connected to the Log Analytics workspace.

Now configure data collection rules within the Azure Portal by navigating to your Log Analytics workspace and selecting *Agents configuration* under the Settings section. Make sure the necessary performance counters (e.g., CPU, memory, disk, and network) and syslog entries for your Ubuntu VMs are enabled.

To enhance your monitoring setup, create alerts and dashboards within Azure Monitor. Now navigate to the Alerts section and create alert rules according to the performance metrics and logs collected, such as alerts for high CPU usage, low available memory, or disk space issues. Customizing Workbooks and Dashboards to create visualizations for monitoring your Ubuntu VM's performance and health let you display key metrics and logs relevant to your environment.

Finally, test your configuration by generating log entries and performance metrics by running workloads on your Ubuntu VM. Verify that these logs and metrics appear in the Log Analytics workspace, and ensure that alerts are triggered as expected according to the conditions you have set. This comprehensive setup of Azure Monitor for your Ubuntu VMs will enable you to collect and analyze telemetry data effectively, set up proactive alerts, and visualize the health and performance of your VMs, ensuring a well-maintained and secure IT infrastructure.



Create Log Analytics workspace ...

A Log Analytics workspace is the basic management unit of Azure Monitor Logs. There are specific considerations you should take when creating a new Log Analytics workspace. Learn more

With Azure Monitor Logs you can easily store, retain, and query data collected from your monitored resources in Azure and other environments for valuable insights. A Log Analytics workspace is the logical storage unit where your log data is collected and stored.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *: Subskrypcja programu Visual Studio Enterprise

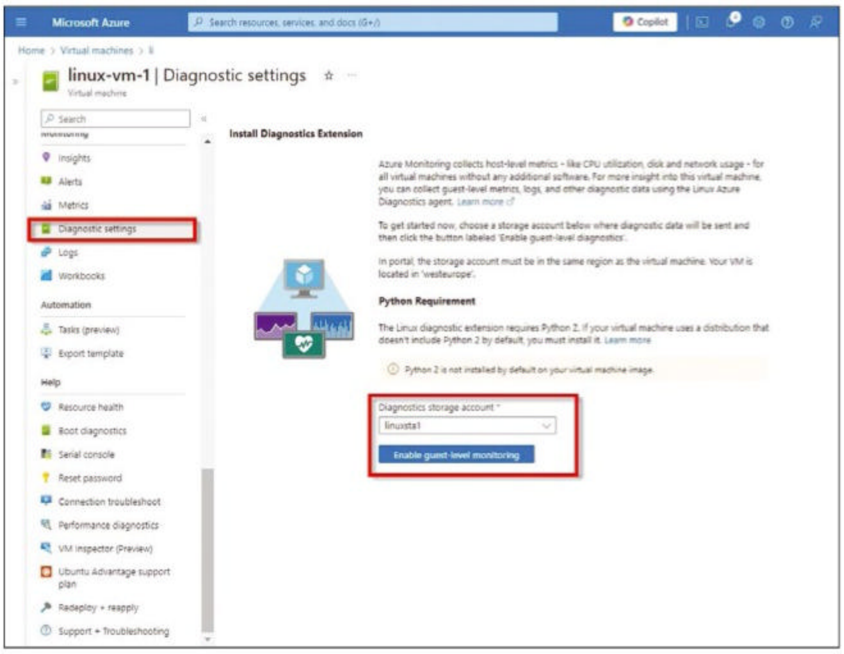
Resource group *: (New) linux-rg-1
Create new

Instance details

Name *: linux-law-1

Region *: West Europe

Figure 1: Creating a new Log Analytics workspace.



linux-vm-1 | Diagnostic settings

Install Diagnostics Extension

Azure Monitoring collects host-level metrics – like CPU utilization, disk and network usage – for all virtual machines without any additional software. For more insight into this virtual machine, you can collect guest-level metrics, logs, and other diagnostic data using the Linux Azure Diagnostics agent. Learn more

To get started now, choose a storage account below where diagnostic data will be sent and then click the button labeled 'Enable guest-level diagnostics'.

In portal, the storage account must be in the same region as the virtual machine. Your VM is located in 'westeurope'.

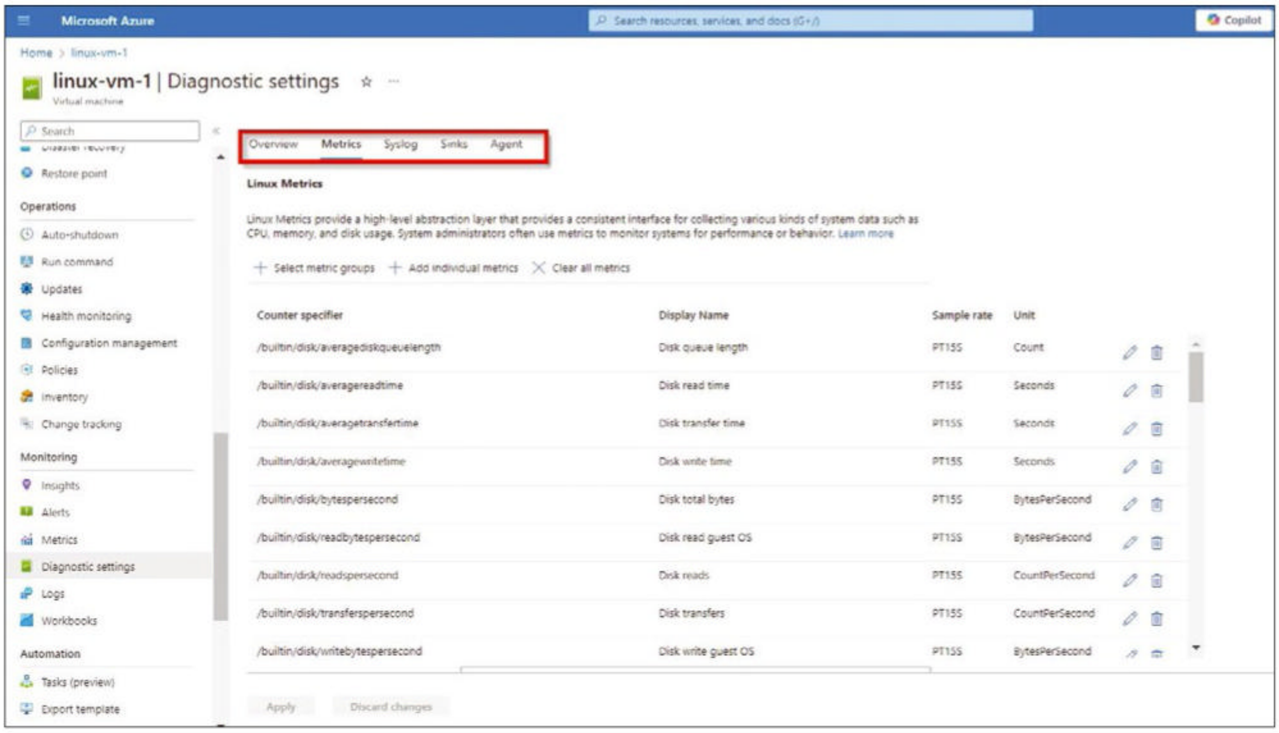
Python Requirement

The Linux diagnostic extension requires Python 2. If your virtual machine uses a distribution that doesn't include Python 2 by default, you must install it. Learn more

Python 2 is not installed by default on your virtual machine image.

Diagnostics storage account *: linuxsta1

Figure 2: Choosing the storage account for diagnostics.



Linux Metrics

Linux Metrics provide a high-level abstraction layer that provides a consistent interface for collecting various kinds of system data such as CPU, memory, and disk usage. System administrators often use metrics to monitor systems for performance or behavior. Learn more

Counter specifier	Display Name	Sample rate	Unit
/builtin/disk/averagediskqueuelength	Disk queue length	PT15S	Count
/builtin/disk/averagereadtime	Disk read time	PT15S	Seconds
/builtin/disk/averagetransfertime	Disk transfer time	PT15S	Seconds
/builtin/disk/averagewritetime	Disk write time	PT15S	Seconds
/builtin/disk/bytespersecond	Disk total bytes	PT15S	BytesPerSecond
/builtin/disk/readbytespersecond	Disk read guest OS	PT15S	BytesPerSecond
/builtin/disk/readspersecond	Disk reads	PT15S	CountPerSecond
/builtin/disk/transferspersecond	Disk transfers	PT15S	CountPerSecond
/builtin/disk/writebytespersecond	Disk write guest OS	PT15S	BytesPerSecond

Figure 3: Choosing the metrics for your Log Analytics workspace.

Log Collection and Storage

Azure Log Analytics

Azure Log Analytics is a powerful tool within the Azure Monitor suite that provides advanced data collection and querying capabilities. It acts as a central repository for storing and analyzing log data from various sources, including Azure resources, on-premises systems, and custom applications. By integrating with Azure Monitor, Log Analytics enables you to gain deep insights into the operational health, performance, and security of your IT environment.

Log Analytics works by collecting data from multiple sources and storing it in a Log Analytics workspace. This data can include performance metrics, diagnostic logs, activity logs, and custom log data. The collected data can then be queried by Kusto Query Language (KQL), which allows for complex data analysis and visualization. With Log Analytics, you can create custom dashboards, set up alerts, and generate reports to monitor your environment proactively.

Custom Logs

In addition to the default logs collected by Azure Monitor, you might need to collect custom logs specific to your applications or services. Custom logs allow you to extend Azure Monitor's capabilities to capture and analyze data that is unique to your environment.

To set up custom logs, first ensure that the Azure Monitor Agent is installed on the VM or server from which you want to collect custom logs. This agent is responsible for collecting and sending log data to your Log Analytics workspace.

The Azure Monitor Agent collects data from the specified paths and sends it to your Log Analytics workspace. You can use KQL to query the custom log data, create alerts, and visualize the data alongside other collected logs. This capability allows you to monitor and analyze specific aspects of your applications and services that are not covered by default logs, providing a comprehensive view of your environment.

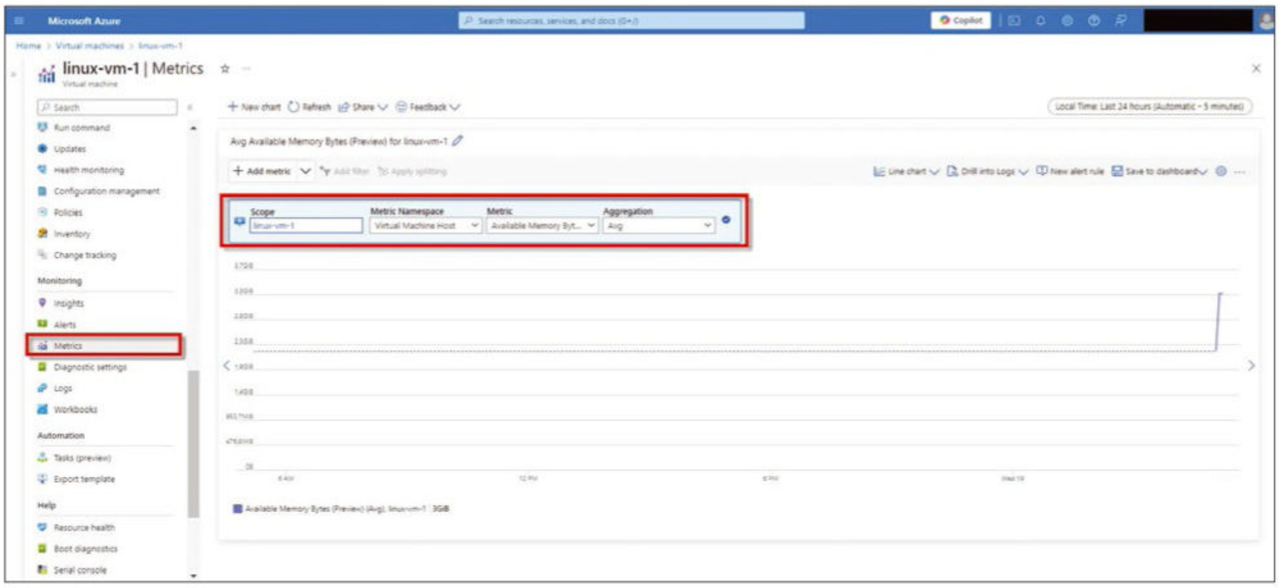
By leveraging Azure Log Analytics and custom logs, you can create a robust log collection and storage solution that meets the unique needs of your organization. This setup enables proactive monitoring, detailed analysis, and effective troubleshooting, ensuring that your IT operations run smoothly and securely.

Performance Monitoring

Metrics and Alerts

Performance monitoring of your Ubuntu VMs in Azure starts with configuring metrics and setting up alerts to keep track of key performance indicators. Metrics are numerical data points that represent the performance of your resources, such as CPU utilization, memory usage, disk I/O, and network traffic. By monitoring these metrics, you can ensure your VMs are operating within acceptable thresholds and quickly identify any deviations that may indicate potential problems.

To configure metrics, start by navigating to the Monitoring section in the Azure Portal, select *Metrics* from the menu, and choose the resource you want to monitor (Figure 4). For Ubuntu VMs, you will typically select metrics related to compute, memory, and storage. To filter, use the metric namespace to add the relevant available metrics to your monitoring view. Once you have selected the metrics to monitor, the next step is to set up



Avg Available Memory Bytes (Preview) for linux-vm-1

Scope: linux-vm-1 · Metric Namespace: Virtual Machine Host · Metric: Available Memory Byt... · Aggregation: Avg

Figure 4: Configuring metrics for your resources.

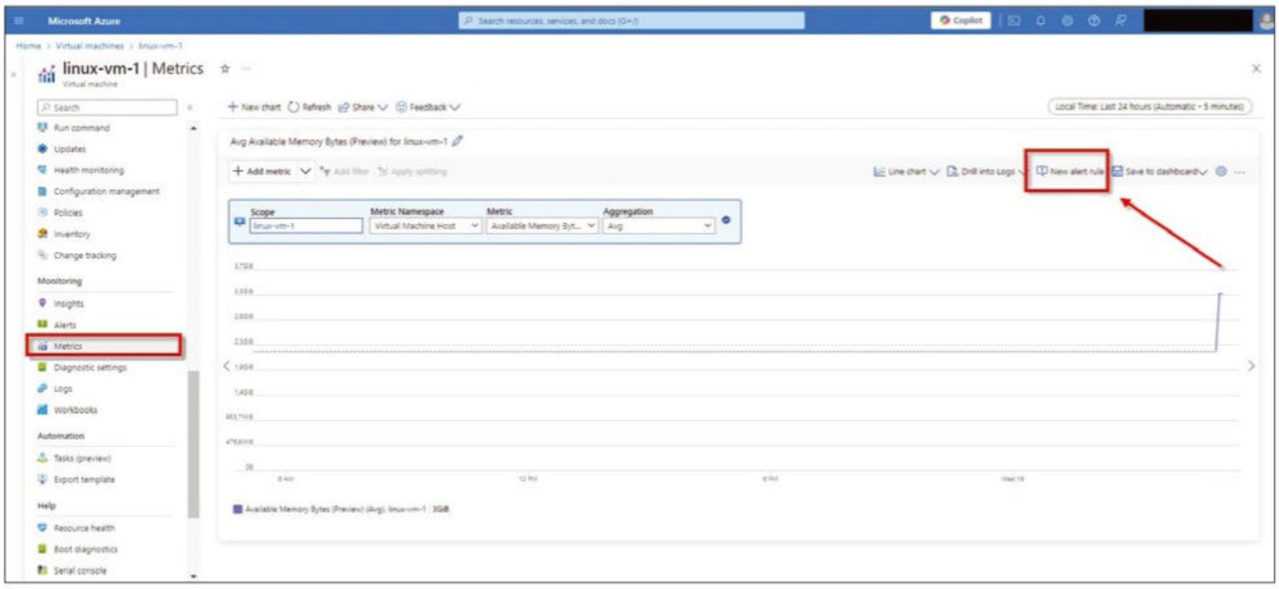
alerts. Alerts notify you when a metric crosses a specified threshold, allowing you to take immediate action. In Azure Monitor, navigate to the Alerts section and click on *New alert rule* (Figure 5). Define the scope by selecting the resource (your Ubuntu VM), and choose the metric on which you want to base the alert. Configure the condition by setting the threshold value and the duration for which the metric should breach this threshold to trigger the alert.

To ensure you are promptly informed of any performance issues, so you can enable a quick resolution, specify the action group that will be notified when the alert is triggered. An action group can include email addresses, SMS numbers, or webhook URLs. Next, define a name for the alert rule and provide a description before reviewing the settings and creating the alert rule.

Dashboards and Visualization

Visualizing performance data in dashboards is essential for effective monitoring. Dashboards provide a centralized view of your VM performance metrics, making it easier to track the health and status of your resources in real time. Azure Monitor offers robust dashboard capabilities that you can customize to fit your specific needs. You can also integrate Log Analytics queries into your dashboard for more detailed insights. Use the *Log Analytics* tile to display query results directly on your dashboard. This integration allows you to visualize complex queries and correlate data from different sources.

To provide a comprehensive view of your VM's performance, group related metrics together. For instance, create a section for compute metrics, another for memory metrics, and



Avg Available Memory Bytes (Preview) for linux-vm-1

Scope: linux-vm-1 · Metric Namespace: Virtual Machine Host · Metric: Available Memory Byt... · Aggregation: Avg

Figure 5: You can add an alert under *Metrics*, as well as under *Alerts*.

one for network metrics. This organization makes it easier to identify quickly which part of the system might be experiencing issues.
You can save and share your dashboards with team members to ensure everyone has access to the same performance data. This collaborative approach enhances team awareness and facilitates coordinated responses to potential issues.

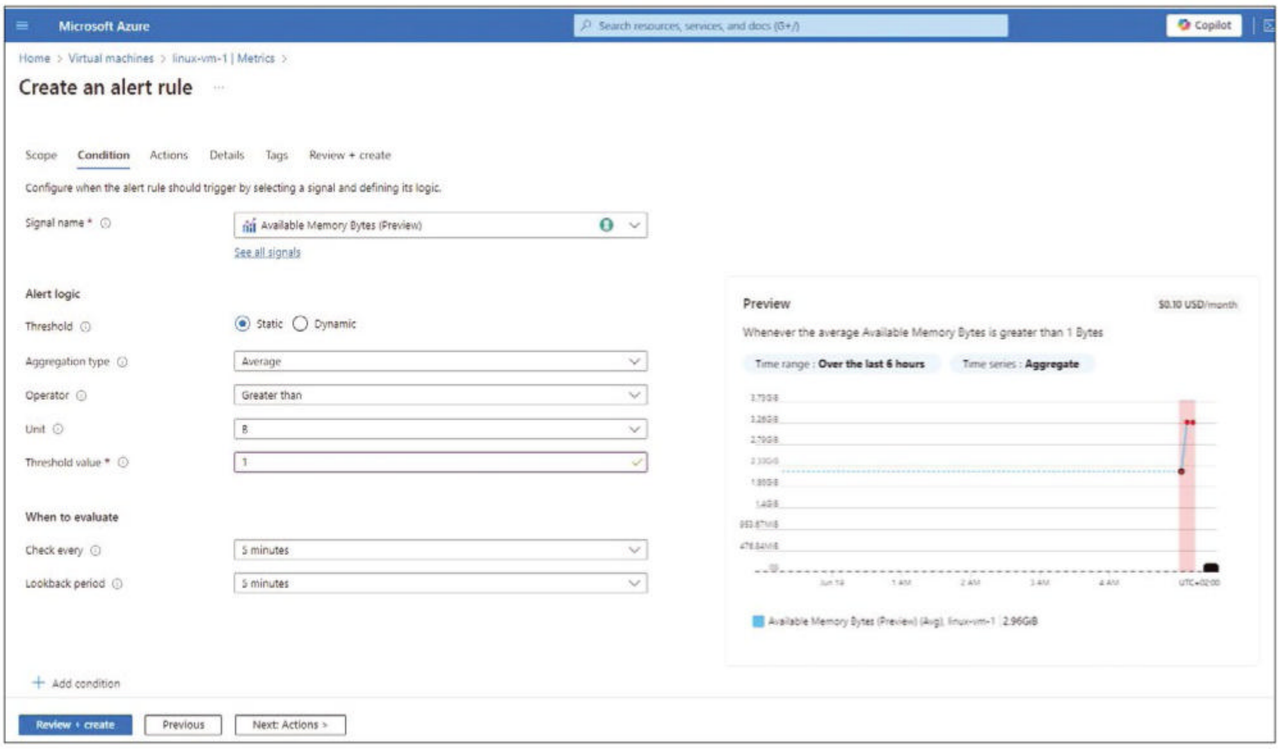
Analyzing Performance Data

Analysis of performance data is crucial for identifying potential issues and optimizing the performance of your Ubuntu VMs. Techniques for effective analysis include historical data comparison, anomaly detection, and correlation analysis.
You can begin by using Azure Monitor's built-in analytics tools. To view historical data, navigate to the *Metrics* page and select the relevant time range. To identify trends and deviations, compare current performance metrics with historical data. This comparison helps you understand normal performance baselines and spot anomalies that could indicate problems.
For more in-depth analysis, use the Azure Log Analytics workspace. Queries written in KQL can drill down into specific performance metrics. For example, you can query CPU usage over time to identify patterns of high utilization that might correlate with specific workloads or times of day.
Anomaly detection is another powerful technique. Azure Monitor's machine learning-based insights let you detect unusual patterns in your metrics automatically. These insights can alert you to potential issues before they escalate, allowing for proactive management.
Correlation analysis helps you understand the relationships between different metrics. For instance, you might find that high CPU usage correlates with increased network traffic. By identifying these correlations, you can troubleshoot issues more effectively and implement targeted optimizations.
By effectively configuring metrics and alerts, creating insightful dashboards, and using advanced analysis techniques, you can maintain optimal performance of your Ubuntu VMs in Azure. This proactive approach to performance monitoring ensures your infrastructure remains robust, responsive, and capable of meeting the demands of your applications and users.

Security Monitoring

Security Center Integration

Integrating Azure Security Center with Azure Monitor enhances the security monitoring capabilities of your Ubuntu VMs by providing a unified view of security alerts and recommendations. Azure Security Center continuously assesses the security posture of your environment, identifies potential vulnerabilities, and provides actionable recommendations to mitigate risks.
To integrate Azure Security Center with Azure Monitor, start by enabling Azure Security Center for your subscription. Navigate to the Azure Portal, select *Security Center* from the left-hand navigation pane and click *Getting started* (Figure 6). Choosing the standard tier unlocks advanced security features, including threat detection and automated response capabilities.
Once Azure Security Center is enabled, configure the security policies and continuous export of security alerts and recommendations to Azure



Create an alert rule

Configure when the alert rule should trigger by selecting a signal and defining its logic.

Signal name *: Available Memory Bytes (Preview)

Alert logic

Threshold: [✓ Static] Dynamic
Aggregation type: Average
Operator: Greater than
Unit: B
Threshold value *: 1

When to evaluate

Check every: 5 minutes
Lookback period: 5 minutes

Whenever the average Available Memory Bytes is greater than 1 Bytes

Time range : **Over the last 6 hours** Time series : **Aggregate**

🟦 Available Memory Bytes (Preview) (Avg), linux-vm-1 2.96GB

Figure 6: Creating an alert rule in Azure to trigger when average Available Memory Bytes exceeds 1 byte.

Listing 1: Sending Metrics to Azure Monitor

```
from azure.monitor.opentelemetry.exporter import AzureMonitorMetricsExporter
from opentelemetry import metrics

# Set up the exporter
exporter = AzureMonitorMetricsExporter.from_connection_string("Your_Connection_String")

# Create a meter and record metrics
meter = metrics.get_meter("my_application")
counter = meter.create_counter("custom_metric")

# Record a metric
counter.add(1, {"custom_dimension": "value"})

# Export the metrics
exporter.export()
```

Monitor for your resources. In the Security Center, choose *Security policy* in the left sidebar and assign policies to your subscriptions or resource groups. These policies define the security baseline and compliance requirements for your environment.
This integration ensures that all security-related data from Azure Security Center is available in your Log Analytics workspace, allowing you to leverage Azure Monitor's advanced querying and visualization capabilities to monitor and analyze security incidents effectively.

Setting Up Security Alerts

Configuring security alerts is a crucial step in detecting potential threats and responding promptly. Security alerts notify you when suspicious activities or potential vulnerabilities are detected in your environment, enabling you to take immediate action to mitigate risks.
To set up security alerts in Azure Monitor, navigate to the *Alerts* selection in the Azure Portal and click *+ New alert rule*. To define the scope, select the Log Analytics workspace that receives data from Azure Security Center. You choose the condition by specifying the KQL query that identifies security threats. For example, to detect failed SSH login attempts, use the query:

```
SecurityEvent
| where EventID == 4625 and LogonType == 3
```

Setting the threshold value and the frequency of evaluation configures the alert logic. For instance, you might want to trigger an alert at more than five failed login attempts within five minutes.
Next, define the action group that will be notified when the alert is triggered. Action groups can include email addresses, SMS numbers, or webhook URLs. After specifying the details, click *Create* to finalize the alert rule.
By setting up these alerts, you ensure that any suspicious activity is promptly detected and you are notified immediately, allowing you to respond quickly and effectively to potential threats.

Investigating Security Incidents

When a security incident occurs, the use of logs to investigate and respond is essential for understanding the scope and effect of the threat and for implementing measures to prevent future occurrences. Azure Monitor and Log Analytics provide powerful tools for log analysis and incident investigation.
To begin, access the Log Analytics workspace where your security data is stored, and use KQL to query the logs and gather relevant information about the incident. For example, to investigate a potential brute force attack, you can query for all failed login attempts:

```
SecurityEvent
| where EventID == 4625
| project TimeGenerated, Account, IPAddress
| order by TimeGenerated desc
```

This query retrieves the timestamp, account name, and IP address of each failed login attempt, helping you identify patterns and potential sources of the attack.
Next, analyze the sequence of events leading up to the incident. Use additional queries to gather context and correlate different log entries. For instance, to check whether any accounts were compromised you can query for successful logins after the failed attempts:

```
SecurityEvent
| where EventID == 4624 and IPAddress in (⮐
  SecurityEvent | where EventID == 4625 | ⮐
  distinct IPAddress)
| project TimeGenerated, Account, IPAddress
| order by TimeGenerated desc
```

By correlating successful and failed login attempts, you can identify compromised accounts and take immediate action to secure them, such as resetting passwords or blocking the IP addresses involved.
Be sure to document the findings and steps taken during the investigation. This documentation is crucial for auditing purposes and for improving your security posture. On the basis of the insights gained from the incident, review and update your security policies and alert configurations.
Ensure ongoing compliance and identify any gaps in your security monitoring setup by regularly reviewing security logs and performing audits. Leveraging the full capabilities of Azure Monitor and Log Analytics effectively lets you investigate and respond to security incidents, enhancing the overall security of your Ubuntu VMs in Azure.

Advanced Monitoring Techniques

Custom Metrics and Alerts

Creating custom metrics and alerts allows you to tailor monitoring to your specific needs, ensuring you can track and respond to the unique performance and operational characteristics of your Ubuntu VMs in Azure. Custom metrics provide additional insights that default metrics might not cover, and custom alerts help you stay proactive in addressing potential issues.
To create custom metrics, first identify the specific data points you need to monitor. For example, you might

want to track application-specific performance indicators or custom log entries. Azure Monitor allows you to send custom metrics with the Azure Monitor Metrics API. Start by installing the Azure SDK for Python if it's not already installed:

```
pip install ➘
  azure-monitor-opentelemetry-exporter
```

Next, use the SDK to send custom metrics to Azure Monitor (Listing 1). After defining your custom metrics, you should configure alerts for these metrics in the Azure Portal. Navigate to *Alerts* under Azure Monitor and create a new alert rule; then, select your custom metric as the signal, set the threshold conditions, and define the action group for notifications. This setup ensures you are alerted when your custom metrics indicate a potential issue.

Automating Responses

Automating responses to alerts can significantly reduce the time to resolution for common issues, enhancing the resilience and stability of your environment. Azure Automation allows you to create runbooks that execute predefined tasks in response to alerts.

To set up automated responses, begin by creating a runbook in Azure Automation by navigating to the Azure Portal, choosing *Automation Accounts*, and selecting or creating an automation account. Under the *Process Automation* section, select *Runbooks* and create a new runbook. As the runbook type, choose *PowerShell* or *Python*. Listing 2 shows an example of a PowerShell runbook that restarts a VM when an alert is triggered.

Next, you publish the runbook and then link it to an alert action. In the Azure Portal, go to *Alerts*, create a new alert rule, and configure the action group to trigger the runbook in response to the alert. This setup automates the response, ensuring timely action is taken without manual intervention.

Integrating with Third-Party Tools

Integrating Azure Monitor with third-party monitoring tools can enhance your monitoring capabilities by combining the strengths of multiple platforms and is particularly useful if your organization uses a diverse set of tools for monitoring different aspects of your infrastructure. To integrate Azure Monitor with third-party tools like Splunk, Datadog, or Prometheus, you can use various methods, including APIs, data export, and native integrations.

For example, to send logs to Splunk, you can use the HTTP Event Collector (HEC). To begin, ensure you have a properly configured Event Hub:

- Navigate to the Azure Portal
- Go to *Event Hubs*
- Create a new Event Hub namespace
- Create an Event Hub within the namespace

In Azure Monitor, go to *Diagnostic settings* for the resources from which you want to export logs and configure it to send logs to the Event Hub.

Next, set up Splunk to receive logs from the Event Hub. Use the Azure Event Hub add-on for Splunk, which allows Splunk to ingest logs from the Azure Event Hub. Configure the add-on with the necessary Event Hub connection details.

For Datadog integration, use the Datadog Azure integration. In the Azure Portal, go to the Datadog integration page and follow the instructions to connect your Azure subscription to Datadog. This action enables Datadog to collect and visualize Azure Monitor metrics and logs.

The Prometheus integration available through Azure Monitor allows Prometheus to scrape metrics from Azure Monitor. To set it up, configure the Azure Monitor metrics exporter and set up Prometheus to scrape these metrics.

By creating custom metrics and alerts, automating responses with Azure Automation, and integrating Azure Monitor with third-party tools, you can build a comprehensive and responsive monitoring solution tailored to the specific needs of your IT infrastructure. This approach ensures you have the visibility, control, and automation necessary to maintain optimal performance and security in your Azure environment.

Best Practices for Logging and Monitoring

Regular Audits and Reviews

Regularly reviewing and auditing logs is a critical practice for maintaining the health and security of your IT environment. Logs provide detailed records of system events, which are invaluable for identifying issues, understanding system behavior, and ensuring compliance with security policies. Regular audits help detect anomalies, unauthorized access attempts, and performance bottlenecks, enabling proactive management and quick resolution of potential problems.

To conduct effective log audits, establish a schedule that fits your operational needs: daily, weekly, or monthly reviews, depending on the criticality of the systems being monitored. You can use tools like Azure Monitor and Log Analytics to set up automated queries that flag unusual activities or errors. For example, a simple query to check for failed login attempts might look like:

```
SecurityEvent
| where EventID == 4625
| summarize count() ➘
  by bin(TimeGenerated, 1h)
| where count_ > 10
```

Listing 2: Restart a VM on Alert

```
param(
    [string] $VMName,
    [string] $ResourceGroupName
)

# Connect to Azure
Connect-AzAccount

# Restart the VM
Restart-AzVM -ResourceGroupName $ResourceGroupName -Name $VMName -Force
```

This query aggregates failed login attempts over one-hour bins and highlights any hours with more than 10 failed attempts, which could indicate a brute-force attack.

From each audit, you can document findings and take appropriate actions on the basis of the results. This documentation not only helps in tracking the health of your systems over time but also serves as evidence during compliance audits. Regularly updating your auditing processes and queries help you adapt to new threats and changes in your environment.

Optimizing Log Retention

Managing log retention policies is essential to balance the cost of storage with the need for data availability. Logs can quickly accumulate, leading to increased storage costs, especially in cloud environments. However, retaining logs for an adequate period is crucial for compliance, auditing, and historical analysis.

A first step is to determine the regulatory requirements and business needs for log retention. Different types of logs might have different retention periods. For example, security logs might need to be retained for a year, whereas performance logs might be kept for a shorter period.

In Azure Monitor, you can configure log retention policies directly in your Log Analytics workspace. Azure charges by the amount of data stored and the retention period, so choose settings that optimize cost while meeting your compliance requirements.

If your data needs to be retained but is accessed infrequently, use export features to move older logs to cheaper storage options. For instance, you can export logs to Azure Blob Storage for long-term, cost-effective storage and even set up a retention policy that automatically moves logs older than a certain period of time to Blob Storage:

```
az monitor log-analytics ➘
  workspace data-export create ➘
  --workspace-name MyWorkspace ➘
  --resource-group MyResourceGroup ➘
  --name ExportToBlob ➘
  --destination /subscriptions/➘
    {subscription-id}/resourceGroups/➘
    {resource-group-name}/providers/➘
    Microsoft.Storage/storageAccounts/➘
    {storage-account-name} ➘
  --tables SecurityEvent ➘
  --export-all-tables
```

You should regularly review and adjust your retention policies to ensure they remain aligned with your business needs and cost constraints.

Ensuring Data Security

Securing log data is paramount to protect sensitive information and ensure compliance with data protection regulations. Logs can contain critical information, such as user activities, configuration details, and system vulnerabilities, making them a prime target for malicious actors.

Implementing access controls restricts who can view and manage logs. In Azure, you use role-based access control (RBAC) to assign permissions. For example, you could grant read-only access to security analysts and full access to system administrators. To define roles and assign them at the resource group or Log Analytics workspace level, use:

```
az role assignment create ➘
  --assignee {user-principal-name} ➘
  --role "Log Analytics Reader" ➘
  --scope /subscriptions/➘
    {subscription-id}/resourceGroups/➘
    {resource-group-name}/providers/➘
    Microsoft.OperationalInsights/➘
    workspaces/{workspace-name}
```

You can encrypt logs both in transit and at rest. Azure Log Analytics automatically encrypts data at rest, but you need to ensure that all communications with the Azure Monitor APIs use HTTPS to protect data in transit.

Regularly review access logs to ensure that only authorized users are accessing log data, and set up alerts for any unauthorized access attempts or changes to log configurations.

Additionally, implement a log integrity monitoring system to detect any tampering with logfiles. To ensure log integrity, use hash functions to create cryptographic checksums of log entries and periodically verify these checksums:

```
import hashlib
def calculate_checksum(log_entry):
  return hashlib.sha256(➘
    log_entry.encode('utf-8')).hexdigest()
# Example log entry
log_entry = "2024-06-15 12:00:00 Failed ➘
             login attempt from 192.168.1.1"
checksum = calculate_checksum(log_entry)
print(f"Checksum: {checksum}")
```

By following these best practices – conducting regular audits, optimizing log retention policies, and ensuring data security – you can create a robust logging and monitoring framework that enhances the security, performance, and compliance of your IT infrastructure.

Conclusion

Effective logging and monitoring are foundational to maintaining a secure, performant, and compliant IT environment. By implementing regular audits and reviews, optimizing log retention policies, and ensuring the security of log data, IT professionals can proactively manage their Ubuntu VMs in Azure. Leveraging Azure Monitor, Log Analytics, and best practices in log management enables you to detect and respond to issues promptly, mitigate risks, and ensure your infrastructure operates smoothly. A robust logging and monitoring strategy not only enhances operational efficiency but also provides critical insights that support continuous improvement and resilience in your IT operations. ■

The Author

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

Secure Active Directory with the rapid modernization plan

Shields Up!

The rapid modernization plan by Microsoft is a practical guide to securing Active Directory, so criminals cannot gain access to privileged user accounts. By Matthias Wübbeling

Microsoft defined the logical separation of user accounts with different authorizations at different levels in the Enhanced Security Admin Environment (ESAE) recommendation. Often referred to as "Red Forest," it is still used in many companies today. Privileged company-wide administrator accounts are managed in their own forest and therefore isolated from the local administrator accounts on servers, workstations, and other devices. If attackers gains access to a local administrator account, their scope of action is limited to the validity of this one account; above all, they cannot get up to any mischief in the entire Active Directory (AD) enterprise.

The continuation of this policy in the rapid modernization plan (RaMP) [1] supports admins in implementing the most important steps of Microsoft's privileged access strategy as a replacement for ESAE. This plan and the associated documents offer admins a step-by-step guide for securing access to enterprise resources. Of course, the most important prerequisite is that you are using Microsoft's Entra ID, formerly known as Azure Active Directory.

Separate Admin Accounts

As in ESAE, the various accounts for administrative function are strictly segregated. Figure 1 shows the strategy for breaking accounts by privileged and non-privileged, along with reducing the attack surface.

In contrast to a local AD, restoring access to the Entra ID directory in an emergency is not simple; the first step in the implementation is to create two or more emergency access accounts that are basically normal accounts with global administrator authorizations assigned to the *onmicrosoft.com* cloud domain – but not to any user – and whose passwords are then stored securely for emergencies. Microsoft's suggestion is to write down these passwords on different pieces of paper and keep them in different places in the company.

With Entra privileged identity management (PIM) enabled (alternatively from the PowerShell API), you can then search for all accounts in the organization that are global or local administrators and have administration rights for Exchange or SharePoint servers. To help you select the relevant accounts, you can use the role overview built into Entra [3].

Before you proceed to classify the accounts you discover, you need to remove any accounts that are no longer required. Next, break the accounts into user accounts for admins, shared administrative access, scripts, and administrative accounts for external employees.

Any local AD operated in addition to Entra needs to be secured, as well. Above all, Microsoft recommends creating separate admin accounts in AD for on-premises administrator activities, exclusively with privileged access workstations (PAWs) to access them. You will also want to create separate local administrator accounts with individual passwords for local computers and servers.

Microsoft offers Defender for Identity to protect privileged accounts [4] – and not just admin accounts, although protecting them is more important. Defender for Identity monitors security-related activities of user accounts, such as those caused by hackers moving around the corporate network. The activities can include brute force login attempts, unusual changes to group memberships of accounts, or unexpected access to resources in other areas of the company. Of course, Defender can only log these incidents and notify you; as the administrator, you need to define and implement the responses to these events yourself, which is why you need separate accounts for admins who handle administrative work – so they can perform their standard tasks (e.g., documentation, web surfing, processing email) with less privileged access.

Nevertheless, these separate accounts must not be used on the employees' non-privileged workstations. Microsoft envisages the installation of separate workstations for administrative tasks, preferably one computer for each administrative account and for the activities to be carried exclusively by the linked account [5].

Account Management in Entra

Microsoft sees the user self-service offered by Entra as the first step in updating RaMP's account management. If users can assign passwords and enable multifactor authentication independently, they have a better user experience and a reduction in help desk workload. Don't forget to email the automatic notices about activities to users, as well, so that they themselves notice unusual activities without delay. You will also want to enable notifications for administrative accounts if passwords are reset by another administrator. Please note that self-service is not available for on-premises administrators, who can only manage their passwords locally.

To ensure the account security of privileged users in Entra ID, you will want to enforce multifactor authentication (MFA) for every admin. Alternatively, you can get rid of passwords altogether and switch to passwordless login with Windows Hello or Microsoft Authenticator. Microsoft recommends checking this setting for all active admin accounts once a quarter. Access to outdated applications is also possible in Entra ID with legacy protocols for authentication, including the SMTP, POP3, and IMAP mail server protocols, along with Autodiscover, Exchange ActiveSync (EAS) or the Exchange offline address book (OAB), Exchange Web Services (EWS), and MAPI over HTTP. Unfortunately, none of these protocols can take advantage of MFA, even if it's been enabled, so only password-based login is supported. In particular, this means attackers can use these protocols for credential stuffing or password spraying attacks.

Risk Assessment

Practical implementation of IT security means constantly assessing the risks and weighing up which risks your company is able to and wants to take on. In Entra, you can configure and monitor the risk level for your users and logins separately [6]. A higher risk level minimizes the number of restrictions for users in their daily work but obviously involves greater administrative effort and affects security.

Conclusions

If you are following Microsoft's path to the cloud and your company is already using Entra ID or a hybrid Active Directory, it makes sense to follow the security suggestions. Some of the comprehensive documents provide step-by-step instructions that you only need to implement. ■

Info

[1] RaMP: [https://learn.microsoft.com/en-us/security/privileged-access-workstations/security-rapid-modernization-plan]

[2] Privileged access strategy: [https://learn.microsoft.com/en-us/security/privileged-access-workstations/privileged-access-strategy]

[3] Roles built into Entra: [https://learn.microsoft.com/en-us/entra/identity/role-based-access-control/permissions-reference]

[4] Defender for Identity: [https://learn.microsoft.com/en-us/defender-for-identity/what-is]

[5] Workstations for privileged access: [https://learn.microsoft.com/en-us/security/privileged-access-workstations/privileged-access-deployment]

[6] Configuring the risk level: [https://learn.microsoft.com/en-us/entra/id-protection/howto-identity-protection-configure-risk-policies]

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.

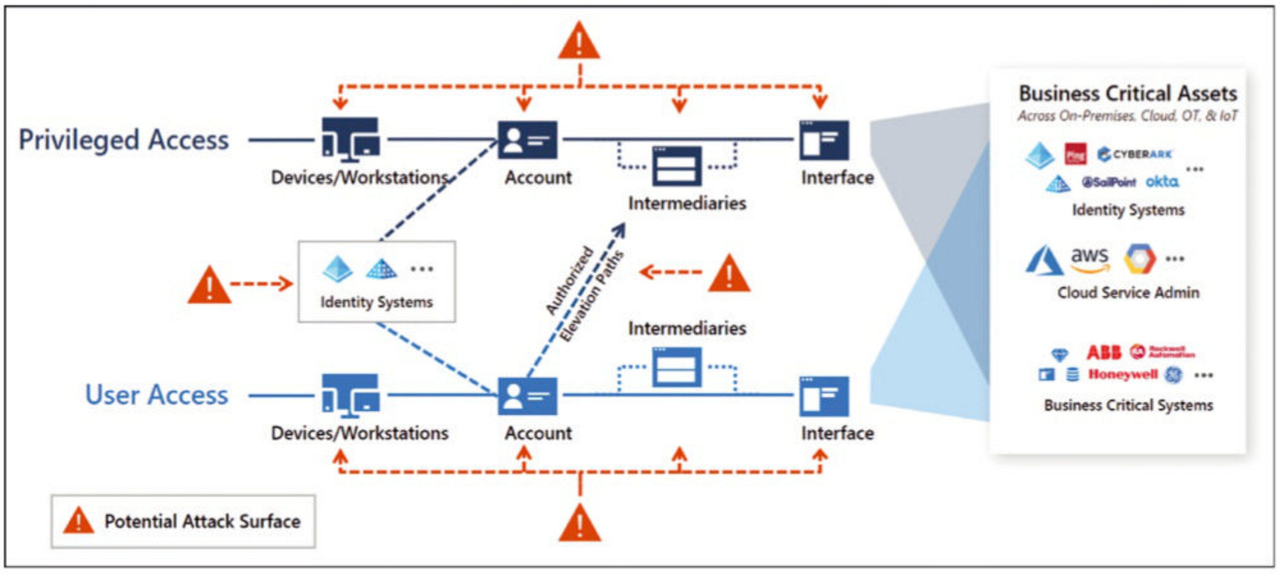
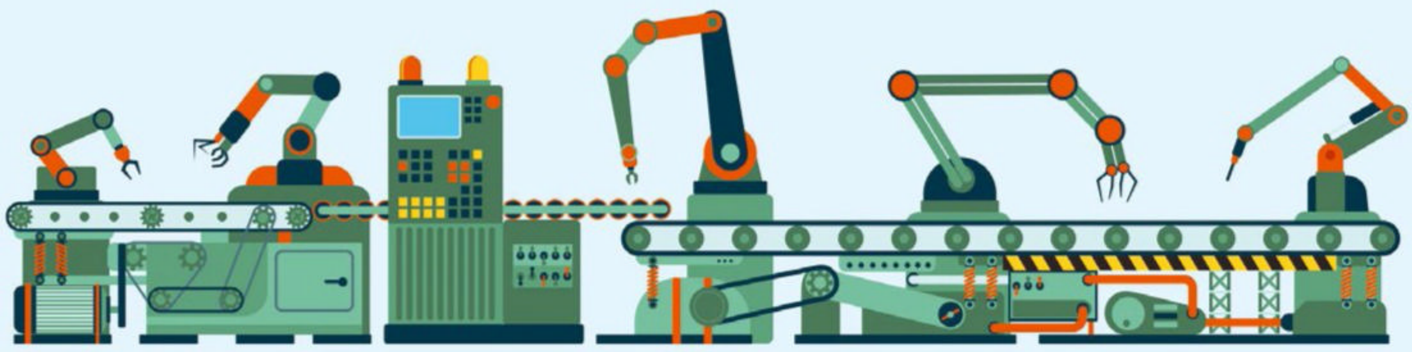


Figure 1: Accounts on company networks can be attacked in many ways. You need to make a distinction between user access and privileged access. © Microsoft [2]



Ansible collections simplify AIX automation

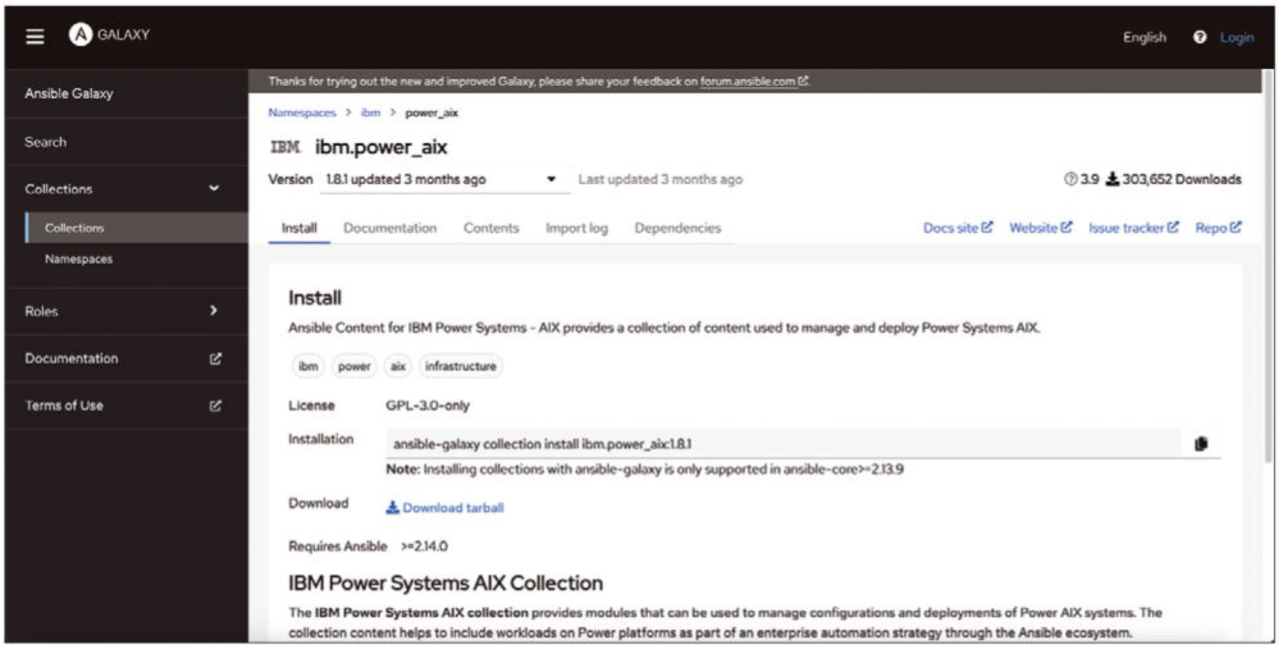
Collective Power

Starting your AIX automation journey has never been easier with the IBM AIX collections from Ansible Galaxy. By Chris Gibson

Ansible is one of the most popular tools for managing servers and other infrastructures. The use of Ansible to automate the management of AIX systems is an attractive idea. Although setting up Ansible to work with AIX might sound like a daunting task, it is far from complicated. Given that IBM has embraced Ansible as its tool of choice for automating the management of its Power servers, if you are looking for a flexible, powerful tool to automate your AIX administration tasks, Ansible is the way to go.

Ansible collections are a way to package and distribute Ansible content, such as playbooks, roles, modules, and plugins. These collections help organize Ansible projects and share them with other users or teams. Ansible Galaxy provides collections for IBM Power AIX that are maintained by the Ansible community [1]. The overall set of IBM Power collections provide modules that can be used to manage IBM Power servers; collections are also available for IBM VIOS, HMC, IBM i, and Linux on Power [2]. A full list of components and modules for the AIX collection are available on the Ansible Galaxy and IBM GitHub sites [3]. The content in these collections helps you manage workloads running on IBM Power platforms as part of an enterprise automation strategy within the Ansible ecosystem.

The IBM AIX collection provides Ansible modules that can be used to



Thanks for trying out the new and improved Galaxy, please share your feedback on forum.ansible.com

ibm.power_aix

Version 1.8.1 updated 3 months ago — Last updated 3 months ago — 3.9 — 303,652 Downloads

Install

Ansible Content for IBM Power Systems - AIX provides a collection of content used to manage and deploy Power Systems AIX.

License: GPL-3.0-only

Installation:

```
ansible-galaxy collection install ibm.power_aix:1.8.1
```

Note: Installing collections with ansible-galaxy is only supported in ansible-core>=2.13.9

Download: Download tarball

Requires Ansible >=2.14.0

IBM Power Systems AIX Collection

The **IBM Power Systems AIX collection** provides modules that can be used to manage configurations and deployments of Power AIX systems. The collection content helps to include workloads on Power platforms as part of an enterprise automation strategy through the Ansible ecosystem.

Figure 1: The IBM AIX collection is available for download from Ansible Galaxy.

manage AIX configurations and deployments. The collection is available to download and install from the Ansible Galaxy site [4] (Figure 1). Both Ansible-certified and community-supported IBM Power collections are provided [5].

Getting Started

The AIX collection [6] contains a large number of modules and roles to help you manage AIX. The collection can be installed on your Ansible controller node with the `ansible-galaxy` command. The Ansible controller node must have, at a minimum, Ansible version 2.14.x installed to support the AIX collection requirements. In the examples that follow, I show you how to install the AIX collection on an existing (i.e., already installed and configured) Ansible controller node on Red Hat Enterprise Linux (RHEL) running Ansible version 2.14.2 (Listing 1).

To install the AIX collection, run the

```
ansible-galaxy collection install ⮐
  ibm.power_aix
```

command on your Ansible controller (Listing 2). If your controller does not have Internet access, you can instead download the AIX collection TAR file, transfer it to your controller, and install the collection from the local TAR file with the command:

```
ansible-galaxy collection install ⮐
  ibm-power_aix-1.8.1.tar.gz
```

To confirm the AIX collection is installed, run the `ansible-galaxy` command with the `collection list` option:

```
# ansible-galaxy collection list ⮐
  ibm.power_aix

# /root/.ansible/collections/⮐
  ansible_collections
Collection    Version
------------- -------
ibm.power_aix 1.8.1
```

By default, collections are installed in `~/.ansible/collections/`. After installation, the collection content will resemble the hierarchy shown in Listing 3.

After a collection is installed, you can access the collection content for a playbook by referencing the namespace `ibm` and the collection's fully qualified name (e.g., `power_aix`); then, you can specify which AIX module you want to use in your playbook. The following specification

```
- hosts: all

tasks:
- name: Query and install latest updates
  ibm.power_aix.suma:
      oslevel: 'latest'
```

accesses the AIX `suma` module with the `oslevel: 'latest'` parameter under the `tasks:` directive. Understanding what this (`suma`) module does is not important (right now), but understanding how you "call" the AIX modules from the Ansible YAML code to perform a certain task is.

In Ansible 2.9 and later, the added `collections` keyword reduces the need to refer to the collection repeatedly. For example, you can use the `collections` keyword in your playbook:

```
- hosts: all
  collections:
  - ibm.power_aix

tasks:
- name: Query and install latest updates
  suma:
      oslevel: 'latest'
```

The AIX collection comes with many demo playbooks. If you navigate to the `~/.ansible/collections/ansible_collections/ibm/power_aix` directory and list the content, you'll find the `demo_*.yml` files (Listing 4). You can copy these files and modify them to your liking. They're an excellent starting point for configuring

Listing 1: Displaying Ansible Version

```
# ansible --version
ansible [core 2.14.2]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.11/site-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.11.2 (main, Feb 17 2023, 09:48:02) [GCC 8.5.0 20210514 (Red Hat 8.5.0-18)]
                  (/usr/bin/python3.11)
  jinja version = 3.1.2
  libyaml = False
```

Listing 2: Installing the AIX Collection

```
# ansible-galaxy collection install ibm.power_aix
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/api/v3/plugin/ansible/content/published/collections/artifacts/
   ibm-power_aix-1.8.1.tar.gz to /root/.ansible/tmp/ansible-local-575019yfq1xg61/tmpj1vtkv5x/ibm-power_
   aix-1.8.1-jcbtn5qn
Installing 'ibm.power_aix:1.8.1' to '/root/.ansible/collections/ansible_collections/ibm/power_aix'
ibm.power_aix:1.8.1 was installed successfully
```

Listing 3: Collection Hierarchy

```
+-- collections/
|  +-- ansible_collections/
|      +-- ibm/
|          +-- power_aix/
|              +-- docs/
|              +-- playbooks/
|                  +-- group_vars/
|                  +-- host_vars/
|              +-- plugins/
|                  +-- action/
|                  +-- module_utils/
|                  +-- modules/
|                  +-- filter/
|              +-- roles/
```

your own playbooks for your AIX environment.

You can access Ansible documentation for the modules in the AIX collection with the `ansible-doc` command. For example, the command

```
ansible-doc ibm.power_aix.filesystem
```

lets you view the documentation for the `filesystem` module in the `ibm.power_aix` collection.

AIX Ansible Client

Ansible is supported on AIX versions 7.1, 7.2, and 7.3. Ansible communicates with AIX hosts over SSH, with authentication through either password or public and private key pairs. All AIX nodes need to have Python3 installed and have either the YUM or DNF package managers installed and configured. Note that Python3 is included and installed by default starting with AIX 7.3 and that YUM is not supported on AIX 7.3 (so you must install DNF instead). In other words, don't waste your time with YUM, just install DNF on all your AIX hosts [7].

If your AIX systems do not have these packages installed, the collection provides an Ansible role to bootstrap and manage an AIX host. The `demo_dnf_bootstrap.yml` playbook (from the AIX collection) can prepare a new AIX server for Ansible management (Listing 5).

The playbook will download and install DNF to the AIX host and install the necessary Python3 packages required by Ansible.

If your AIX host doesn't have Internet access to download and install DNF, you can, instead, manually download the DNF bundle (`dnf_bundle_aix_73.tar`) from the AIX toolbox PPC repository [8], then unpack the TAR file and run the (extracted) `install_dnf.sh` script to install DNF from the locally extracted RPMs.

AIX Modules

With the AIX collection installed, you can now write playbooks that use its components (modules, roles, etc.). Table 1 provides a list of some of the modules called in the example playbooks that follow. Many more AIX modules are included in the collection than shown here.

Ansible User

Some modules included in the `ibm.power_aix` collection perform privileged operations. Many actions require Ansible to perform tasks as the root superuser. Only authorized users can run privileged operations. If required, you can create an Ansible (non-root) user with the proper roles on the AIX system.

AIX role-based access control (RBAC) can be employed to provide non-root users the ability to execute Ansible tasks if needed. For example, the `alt_disk` module runs the `alt_disk_copy` and `alt_rootvg_op` commands, which require the `aix.system.install` RBAC authorization, and the `chpv` command requires the `aix.lvm.manage.change` RBAC authorization. You can obtain a list of commands run as part of the `alt_disk` module with the command:

```
# awk -F\' '/cmd =/ {print $2}' ↩
  .ansible/collections/↩
  ansible_collections/ibm/power_aix/↩
  plugins/modules/alt_disk.py | sort -u
```

Listing 4: Demo Playbooks

```
# cd /root/.ansible/collections/ansible_collections/ibm/power_aix/playbooks/
# ls -ltr
total 208
lrwxrwxrwx. 1 root root    8 Feb 11 19:45 roles -> ../roles
drwxr-xr-x. 2 root root   42 Feb 11 19:45 group_vars
-rw-r--r--. 1 root root 3407 Feb 11 19:45 dynamic_tunables.yml
-rw-r--r--. 1 root root 1996 Feb 11 19:45 dynamic_mktun.yml
[...etc...]
-rw-r--r--. 1 root root 1246 Feb 11 19:45 demo_mkfilt.yml
-rw-r--r--. 1 root root 1448 Feb 11 19:45 demo_lvol.yml
-rw-r--r--. 1 root root  587 Feb 11 19:45 demo_lvm_facts.yml
-rw-r--r--. 1 root root 2129 Feb 11 19:45 demo_lpp_facts.yml
-rw-r--r--. 1 root root  893 Feb 11 19:45 demo_flrtvc.yml
-rw-r--r--. 1 root root  318 Feb 11 19:45 demo_dnf.yml
-rw-r--r--. 1 root root 2482 Feb 11 19:45 demo_apache-server.yml
```

Listing 5: AIX DNF Bootstrap Playbook

```
---
- name: "Bootstrap Yum on AIX"
  hosts: all
  gather_facts: false
  user: root
  collections:
  - ibm.power_aix
  tasks:

# CHECK for Yum on inventory host
  - import_role:
      name: power_aix_bootstrap
    vars:
      pkgtype: "dnf"
      download_dir: "~"
      target_dir: "/tmp/.ansible.cpdir"
```

Table 1: Sample of AIX Ansible Modules

Module	Description
`user`	Create new users or change/remove attributes of users on AIX by facilitating the creation of a new user with provided attributes, the modification of attributes, or the deletion of an existing user.
`group`	Manage the presence, attributes, and members of AIX groups by allowing you to create new groups, to change/remove attributes and administrators or members of a group, and to delete an existing group.
`tunables`	Modify/reset/show tunables for various components on AIX by facilitating the modification, reset, and display of tunable parameter(s) with provided inputs.
`lvol`	Configure AIX LVM logical volumes by allowing you to create, remove, and modify attributes of LVM logical volumes.
`filesystem`	Local and NFS filesystems management, allowing you to create, modify, and remove local and NFS filesystems.
`mount`	Mounts/unmounts a filesystem or device on AIX.
`inittab`	Manages `inittab` entries on AIX by allowing administrators to create, change, and remove entries in the `/etc/inittab` file.

A common practice is to create a service account (a user named *ansible*) with sudo access (i.e., if sudo is installed on your AIX hosts; Listing 6). However, in the lab environment used to create the examples in this article (for demonstration purposes only), I use root.

How you choose to implement privilege escalation for Ansible is very much dependent on the security needs and policies of your environment. I recommend you review the links in the references section [9] [10] for more information on how Ansible handles privilege escalation systems to allow a user to execute tasks as someone else. In this way, you can determine what method will work best for your particular environment.

Also note that the AIX hosts in the examples have already been configured to communicate with the Ansible controller over SSH. See the Ansible playbook documentation [11] for information on how to share the controller's public SSH key with a managed (AIX) node.

If you need guidance on installing and configuring your very first Ansible controller, the IBM Redbook, *Using Ansible for Automation in IBM Power Environments* [12], is a great reference; particularly section 3.3, "Installing your Ansible control node."

Automating AIX operations

AIX administrators perform a variety of tasks and operations on their servers every day. These tasks are known as "day 2 operations," and they are typically focused on post-deployment tasks such as managing, maintaining, monitoring, and optimizing a system. An example could be adding a new AIX user or creating a new filesystem. This is where the AIX collection makes automating these types of tasks easy. The playbooks that follow have been deliberately made to be simple to make it easy to understand the purpose and advantage of each. These examples are building blocks, which you can make far more complex and advanced with extensive use of variables and directives.

If you're not familiar with Ansible Playbooks and YAML syntax, to gain a better understanding, please take a moment to read "Intro to Playbooks" [13] on the Ansible website. Put simply, Ansible Playbooks are a way to use commands to perform operations on remote computers in a scripted manner. An Ansible Playbook contains one or more "plays," which map hosts to a certain function. Ansible does this through tasks, which are basically module calls. Figure 2 is a visual representation of a simple playbook example.

Rather than dissecting what each directive does in the code samples, I instead focus on the AIX operations being automated by the collection or module, with some details for clarity. You should be able to take one of the sample playbooks and (whilst referring to the available and extensive online documentation) build on it to fit your own needs.

Managing AIX Filesystems

To start the Ansible automation journey, I will work with AIX filesystems. For this specific example, you will create two new Ansible playbooks: one for the creation of a new filesystem and another for removal. I'll work through the example step by step to configure a playbook to achieve the goal of adding and removing a JFS2 filesystem with the ibm.power_aix.lvol, ibm.power_aix.filesystem, and ibm.power_aix.mount modules, from the AIX collection.

Start by navigating to the /etc/ansible/playbooks directory on your Ansible controller node and create a new directory named aix. Change to that directory and create a playbook named createfs.yml with the content shown in Listing 7. This playbook will create a new JFS2 filesystem named /testfs in rootvg with a size of 64MB on a logical volume named testlv. The filesystem will be set to mount automatically on boot.

Listing 6: Ansible sudo User

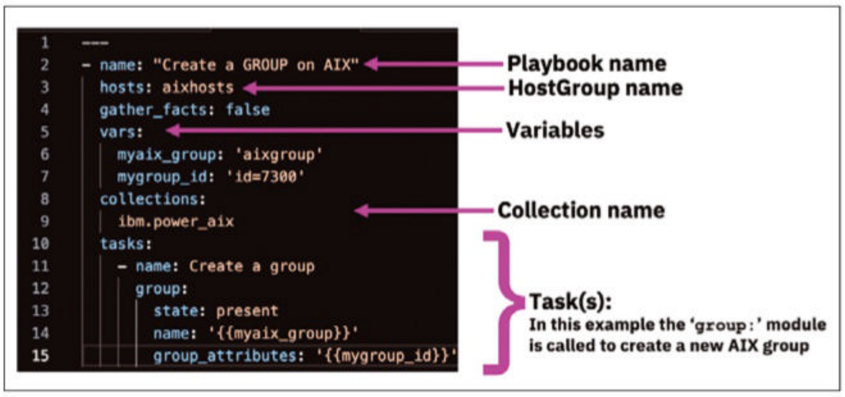
```
# grep -w ansible /etc/sudoers
ansible ALL=(ALL) NOPASSWD: ALL

$ id
uid=2024(ansible) gid=2024(ansible) groups=1(staff)
$ sudo -l
User ansible may run the following commands on myaixhost1:
    (ALL) NOPASSWD: ALL
```

In this first example playbook, debug output was enabled (note the register: and debug: var= directives) for each step so that you can see what is happening as each operation is performed. The hosts: directive specifies the group of hosts on which to execute the tasks. In this example, it is aixhosts, as specified in the /etc/ansible/hosts file; in the example lab environment, only a single AIX host is listed in the hosts file. You can add more hosts as needed.

Also note that the ansible_python_interpreter directive specifies the location of the Python3 executable on the AIX host quark (i.e., /usr/bin/python3):

```
[aixhosts]
quark ansible_python_interpreter=⮐
  /usr/bin/python3
```

Including this parameter lets you avoid the *discovered Python interpreter*



```
---
- name: "Create a GROUP on AIX"
  hosts: aixhosts
  gather_facts: false
  vars:
    myaix_group: 'aixgroup'
    mygroup_id: 'id=7300'
  collections:
    ibm.power_aix
  tasks:
    - name: Create a group
      group:
        state: present
        name: '{{myaix_group}}'
        group_attributes: '{{mygroup_id}}'
```

Playbook name
HostGroup name
Variables
Collection name
Task(s):
In this example the 'group:' module is called to create a new AIX group

Figure 2: Anatomy of a simple Ansible Playbook that uses collections.

warning message (Listing 8). Refer to link in the warning message for other methods of handling this message through settings in the `ansible.cfg` file on the controller.

Now that the playbook has been created, you can run it to create the new AIX filesystem with the command:

```
# ansible-playbook createfs.yml
```

To confirm that the filesystem was created successfully, run `ansible -m shell` to verify the filesystem is mounted (Listing 9). From the output, you can observe that the `/testfs` filesystem is mounted and is 64MB in size.

Now, to create a playbook to remove the filesystem you just created, go to the `/etc/ansible/playbooks/aix` directory and edit a new file named `removefs.yml` with the content shown in Listing 10. This playbook will unmount and then remove the `/testfs` filesystem. To run this playbook, enter:

```
# ansible-playbook removefs.yml
```

Remember that AIX automatically deletes the underlying logical volume with all data on it when you remove a filesystem. The playbook will also remove the mount point directory.

Note that the `state:` directive is set to `absent`, meaning the filesystem should be removed if it exists. The `ibm.power_aix.mount` and `ibm.power_aix.filesystem` modules are used in this example. You can verify that the `/testfs`

Listing 7: Creating a Filesystem on AIX

```
# cd /etc/ansible/playbooks
# mkdir aix
# cd aix
# vi createfs.yml
---
- name: "Create a FILESYSTEM on AIX"
  hosts: aixhosts
  gather_facts: false
  vars:
    myfs_name: '/testfs'
    myvg_name: 'rootvg'
    mylv_name: 'testlv'
    mylv_size: '64M'
    myfs_type: 'jfs2'
    automount_fs: 'true'
  collections:
    ibm.power_aix
  tasks:
    - name: Create a logical volume of 64M
      lvol:
        vg: '{{myvg_name}}'
        lv: '{{mylv_name}}'
        size: '{{mylv_size}}'
        state: present
      register: output
    - debug: var=output

    - name: Creation of a JFS2 filesystem
      filesystem:
        state: present
        filesystem: '{{myfs_name}}'
        fs_type: '{{myfs_type}}'
        auto_mount: '{{automount_fs}}'
        device: '{{mylv_name}}'
      register: output
    - debug: var=output

    - name: Mount filesystems
      mount:
        state: mount
        mount_dir: '{{myfs_name}}'
      register: output
    - debug: var=output
```

Listing 8: Ansible Interpreter Warning

```
# ansible -m ping quark
[WARNING]: Platform aix on host quark is using the discovered Python interpreter at
   /usr/bin/python3.9, but future installation of another Python interpreter could change
   the meaning of that path. See https://docs.ansible.com/ansible-core/2.14/reference_
   appendices/interpreter_discovery.html for more information.
quark | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.9"
    },
    "changed": false,
    "ping": "pong"
}
```

Listing 9: Verifying /testfs Creation

```
# ansible -m shell -a 'df -m|grep testfs' aixhosts

quark | CHANGED | rc=0 >>

/dev/testlv       64.00     62.67    3%         4     1% /testfs
```

Listing 10: Removing a Filesystem

```
# vi removefs.yml
---
- name: "Remove a FILESYSTEM on AIX"
  hosts: aixhosts
  gather_facts: false
  vars:
    myfs_name: '/testfs'
    remove_mymount: 'true'
  collections:
    ibm.power_aix
  tasks:
    - name: Unmount filesystem
      mount:
        state: umount
        mount_dir: '{{myfs_name}}'
        mount_over_dir: '{{myfs_name}}'
      register: output
    - debug: var=output

    - name: Remove a filesystem
      filesystem:
        filesystem: '{{myfs_name}}'
        state: absent
        rm_mount_point: '{{remove_mymount}}'
      register: output
    - debug: var=output
```

filesystem was removed from the AIX hosts with the `ansible -m shell` command:

```
# ansible -m shell ↩
          -a 'df ↩
          -m|grep testfs' aixhosts
quark | FAILED | rc=1 >>
non-zero return code
```

This operation should fail with a return code of *1*, indicating the filesystem no longer exists on the AIX host.

Managing AIX Users and Groups

Next, you will create simple playbooks for managing AIX users and groups in Ansible (Listing 11). Once again, in the `/etc/ansible/playbooks/aix` directory, you create a playbook called `adduser.yml` and include the encrypted password (as shown) in the playbook. (This password will be copied into the `/etc/security/passwd` file on the AIX host. Note that this password decrypts to *abc123*.)
The playbook will create the user with attributes that allow for unlimited file sizes, user ID `730`, home directory `/home/aixguest`, shell `/usr/bin/ksh`, description ``AIX guest user'``, and primary group/groups `staff`. You also do not want the user to have to change their password when they first log in to the system, so you should set `change_passwd_on_login:` to `False`. The `ibm.power_aix.user` module is used here.
Now you can run the playbook to create the AIX user, `aixguest`,

```
# ansible-playbook adduser.yml
```

and verify that user `aixguest` was created on the AIX host by running

```
# ansible -m shell ↩
          -a 'lsuser aixguest' aixhosts
```

from the controller. To delete this AIX user, create the playbook in Listing 12 and name it `deluser.yml`. The playbook will not remove the user's home directory (i.e., `remove_homedir:` is set to `False`).

To delete the AIX user, run the playbook and then verify that the `aixguest` user was deleted:

```
# ansible-playbook deluser.yml
# ansible -m shell ↩
          -a 'lsuser aixguest' aixhosts
quark | FAILED | rc=2 >>
3004-687 User "aixguest" does not exist.↩
  non-zero return code
```

Listing 13 shows the steps for creating a group in AIX.

Managing AIX Configuration Files

Next up, you'll use Ansible to change an existing entry in the `/etc/inittab` file on an AIX host (Listing 14). The `chinittab.yml` playbook changes the entry for the `uprintfd` daemon so that its state is set to `off`, instead of `respawn`. Note that the `state:` directive is set to `modify`, meaning the `inittab` entry should be modified. The `ibm.power_aix.inittab` module is used here.
To modify `/etc/inittab`, run the playbook, then verify that the `/etc/inittab` entry for `uprintfd` was changed on the AIX host:

```
# ansible-playbook chinittab.yml
# ansible -m shell ↩
          -a 'lsitab uprintfd' aixhosts
quark | CHANGED | rc=0 >>

uprintfd:23456789:off:/usr/sbin/uprintfd
```

Next, create the `delinittab.yml` playbook to remove the `uprintfd` entry from the `/etc/inittab` file on the AIX

Listing 11: Adding an AIX User

```
# cd /etc/ansible/playbooks/aix
# vi adduser.yml
---
- name: "Create USER on AIX"
  hosts: aixhosts
  gather_facts: false

  vars:
    user_name: 'aixguest'
    userhome_dir: '/home/aixguest'
    user_id: '730'
    user_shell: '/usr/bin/ksh'
    user_gecos: 'AIX guest user'
    user_pgrp: 'staff'
    user_groups: 'staff'
    user_fsize: '-1'
    user_passwd: "{ssha256}06$FkFHd0q1hxVonC2a$sa1WA0G3m
                   PNWtz2GAhUkfcQ7BkD/mNngBu0Tn2.N.cl"
  collections:
    ibm.power_aix
  tasks:
    - name: Create local user
      user:
        state: present
        name: '{{user_name}}'
        change_passwd_on_login: False
        password: '{{user_passwd}}'
        load_module: files
        attributes:
          id: '{{user_id}}'
          shell: '{{user_shell}}'
          home: '{{userhome_dir}}'
          gecos: '{{user_gecos}}'
          pgrp: '{{user_pgrp}}'
          groups: '{{user_groups}}'
          fsize: '{{user_fsize}}'
      register: output
    - debug: var=output
```

Listing 12: Deleting the AIX User

```
# cd /etc/ansible/playbooks/aix
# vi deluser.yml
---
- name: "Delete USER on AIX"
  hosts: aixhosts
  gather_facts: false
  vars:
    user_name: 'aixguest'
  collections:
    ibm.power_aix
  tasks:
    - name: Delete Local user
      user:
        state: absent
        name: '{{user_name}}'
        remove_homedir: False
      register: output
    - debug: var=output
```

Listing 13: Creating an AIX Group

```
# cd /etc/ansible/playbooks/aix
# vi addgroup.yml
---
- name: "GROUP on AIX"
  hosts: aixhosts
  gather_facts: false
  vars:
    myaix_group: 'aixgroup'
    mygroup_id: 'id=7300'
  collections:
    ibm.power_aix
  tasks:
    - name: Create a group
      group:
        state: present
        name: '{{myaix_group}}'
        group_attributes: '{{mygroup_id}}'
      register: output
    - debug: var=output
```

Listing 14: Modifying /etc/inittab on AIX

```
# cd /etc/ansible/playbooks/aix
# vi chinittab.yml
---
- name: "Modify INITTAB on AIX"
  hosts: aixhosts
  gather_facts: false
  vars:
    inittab_name: 'uprintfd'
    inittab_level: '23456789'
    inittab_action: 'off'
    inittab_cmd: "/usr/sbin/uprintfd"
  collections:
    ibm.power_aix
  tasks:
    - name: Change entry for uprintfd
      inittab:
        state: modify
        name: '{{inittab_name}}'
        runlevel: '{{inittab_level}}'
        action: '{{inittab_action}}'
        command: '{{inittab_cmd}}'
      register: output
    - debug: var=output
```

Listing 15: Removing an /etc/inittab Entry

```
# cd /etc/ansible/playbooks/aix
# vi delinittab.yml
---
- name: "Remove INITTAB entry on AIX"
  hosts: aixhosts
  gather_facts: false
  vars:
    inittab_name: 'uprintfd'
    inittab_level: '23456789'
    inittab_action: 'off'
    inittab_cmd: "/usr/sbin/uprintfd"
  collections:
    ibm.power_aix
  tasks:
    - name: Remove entry for uprintfd
      inittab:
        state: absent
        name: '{{inittab_name}}'
        runlevel: '{{inittab_level}}'
        action: '{{inittab_action}}'
        command: '{{inittab_cmd}}'
      register: output
    - debug: var=output
```

host (Listing 15). Note that the `state:` directive is set to `absent`, meaning the `/etc/inittab` entry should be removed if it exists.
Run the playbook with the command:

```
# ansible-playbook delinittab.yml
```

To add the `uprintfd` entry back into the `/etc/inittab` file, create the `addinittab.yml` playbook (Listing 16) to create an

Listing 16: Adding an /etc/inittab Entry

```
# cd /etc/ansible/playbooks/aix
# vi addinittab.yml
---
- name: "Add INITTAB entry on AIX"
  hosts: aixhosts
  gather_facts: false
  vars:
    inittab_name: 'uprintfd'
    inittab_level: '23456789'
    inittab_action: 'respawn'
    inittab_cmd: "/usr/sbin/uprintfd"
    inittab_pos: 'writesrv'
  collections:
    ibm.power_aix
  tasks:
    - name: Remove entry for uprintfd
      inittab:
        state: present
        name: '{{inittab_name}}'
        runlevel: '{{inittab_level}}'
        action: '{{inittab_action}}'
        command: '{{inittab_cmd}}'
        insertafter: '{{inittab_pos}}'
      register: output
    - debug: var=output
```

entry named `uprintfd`, with runlevel `23456789`, action `respawn`, and command `/usr/sbin/uprintfd`. Also note that the `state:` directive is set to `present`, meaning the entry is created if it does not already exist. This new entry will be inserted into the `/etc/inittab` file after the `writesrv` entry, as specified by the `insertafter:` directive.

Conclusion

AIX administrators can use Ansible to automate tasks. I've only scratched the surface of what's possible, but I've tried to shine a light on the possibilities. Many more modules are available that you can leverage to perform far more interesting and powerful tasks, allowing you to perform administration and configuration tasks in a repeatable fashion across your entire AIX landscape. ■

Info

[1] Galaxy ibm.power_aix contents: [https://galaxy.ansible.com/ui/repo/published/ibm/power_aix/content/]

[2] IBM Power collections: [https://galaxy.ansible.com/ui/namespaces/ibm/?sort=name&keywords=power&page=1]

[3] Source code on GitHub: [https://github.com/IBM/ansible-power-aix]

[4] Galaxy ibm.power_aix install: [https://galaxy.ansible.com/ui/repo/published/ibm/power_aix/]

[5] Automation with Power Community: [https://community.ibm.com/community/user/power/communities/community-home?CommunityKey=57c56971-5794-4521-9a6e-dd98e6122435]

[6] AIX collection modules: [https://ibm.github.io/ansible-power-aix/modules.html]

[7] Configuring DNF and creating local repositories on IBM AIX: [https://developer.ibm.com/tutorials/awb-configuring-dnf-create-local-repos-ibm-aix/]

[8] PPC repository: [https://public.dhe.ibm.com/aix/freeSoftware/aixtoolbox/ezinstall/ppc/]

[9] Understanding privilege escalation: [https://docs.ansible.com/ansible/2.7/user_guide/become.html]

[10] become keyword: [https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_privilege_escalation.html]

[11] Ansible setting up SSH keys: [https://docs.ansible.com/ansible/latest/inventory_guide/connection_details.html]

[12] Simon, Tim, Jose Martin Abeleira, Shahid Ali, et al. *Using Ansible for Automation in IBM Power Environments*. IBM Redbooks, December 2023, draft SG24-8551-00: [https://www.redbooks.ibm.com/redpieces/abstracts/sg248551.html]

[13] Intro to Playbooks: [https://docs.ansible.com/ansible/2.9/user_guide/playbooks_intro.html]

Author

Chris Gibson is a technical specialist for the IBM Power Technical Training team. He creates technical training content for IBM Power, AIX, and PowerVM. Over his 25 years of working with AIX (as an administrator, consultant, instructor, presenter, speaker, blogger, and content creator), Chris has written and published many technical articles and has co-authored several IBM Redbooks on IBM AIX and Power. Chris resides in Melbourne, Australia. You can stay in touch by following his blog at [http://gibsonnet.net/blog/cgaix/] or from his Linkedin account, [https://www.linkedin.com/in/chris-gibson-9921253/].

DrupalCon
BARCELONA2024
24-27 SEPTEMBER

Last chance to secure your ticket at the Regular rate!

Join us from **24–27 September** at CCIB, Barcelona, for an unforgettable journey of learning and connection.

Secure your ticket now at the **Regular rate of €930** until 19 August 2024,

and get access to **3** days packed with insightful sessions divided in **6** tracks and BoFs, Inspiring keynotes including the **Driesnote**, contribution day, and much more.

Manage Windows images with the Windows ADK

Tools at Hand

The Windows Assessment and Deployment Kit is a universal verification and management suite for deployments, with tools for analyzing system performance and simplifying the deployment of Windows images. By Tam Hanna

Microsoft has been using the Windows Assessment and Deployment Kit (ADK) as a drop-off point for various useful tools and developments for some time now. Unfortunately, the matching documentation is distributed across the Microsoft Developer Network (MSDN) and the official documentation is incomplete. The collection of links [1] can be useful because it points you to relevant documents and tutorials for the respective components.

For the work in this article, I used a workstation with Windows 11 23H2; before proceeding, you should know that the ADK tools need substantial computing power. Microsoft last updated these tools in September 2023. The new features are not the main focus here, but I will point them out for more experienced ADK admins.

Installing Windows ADK

Microsoft's deployment SDK is very closely tied to the base version of the operating system running on your host. In other words, it is very much advisable to complete the Windows Update process before rolling out ADK, because outdated components will always provoke erratic behavior in the various ADK tools. The deployment is made more difficult by the proliferation of different versions.

For this article, I implement what follows with "ADK for Windows 11, version 22H2," but you should select the correct options on the ADK download page [2] for your situation. In general, Microsoft recommends you use the ADK that matches the latest version of Windows available in your environment. In a heterogeneous infrastructure, the ADK that matches the Windows version you use is the logical choice.

After pressing the button to confirm the download, the `adksetup.exe` file is dropped onto your disk; running it launches an installation wizard (Figure 1). Start by selecting the desired installation mode. Besides installing on your local host, you have the option of creating installation media for delivering ADK to target systems that are not connected to the Internet.

Also interesting to note is that the suggested installation path still references version 10 – probably an oversight on Microsoft's part. You should not correct this path, though, because it turns out that it is the default and is referenced in various examples and by the documentation. In the wizard, you need to approve the data transfer, among other things.

Microsoft is completely modularizing the Windows ADK. Tools for verifying application compatibility, for managing Windows and Office, and for the Windows evaluation console for determining performance metrics are not included by default – or for the Media eXperience Analyzer, which lets you draw conclusions on system behavior under multimedia load.

In this case, I opted for a full install, which is no problem as far as storage goes because it requires just 5GB. After successfully downloading the components and completing the installation wizard, you have the option of launching a Getting Started tutorial. Unfortunately, this only means an MSDN page with some additional information.

Analyzing Performance

The first most important new feature in Windows ADK is an update to Windows Performance Analyzer, which now uses version 6.0 of the .NET framework, meaning the program will also run on ARM hardware. Second, the Analyzer has been treated to a completely new user interface; I use it for all the examples in this article. Microsoft also provides a programming interface that enables more complex automation of performance tests in the form of the Windows Performance Recorder (WPR) API, WPRControl. Documentation can be found online [3], although it is targeted at system administrators who are familiar with Windows development.

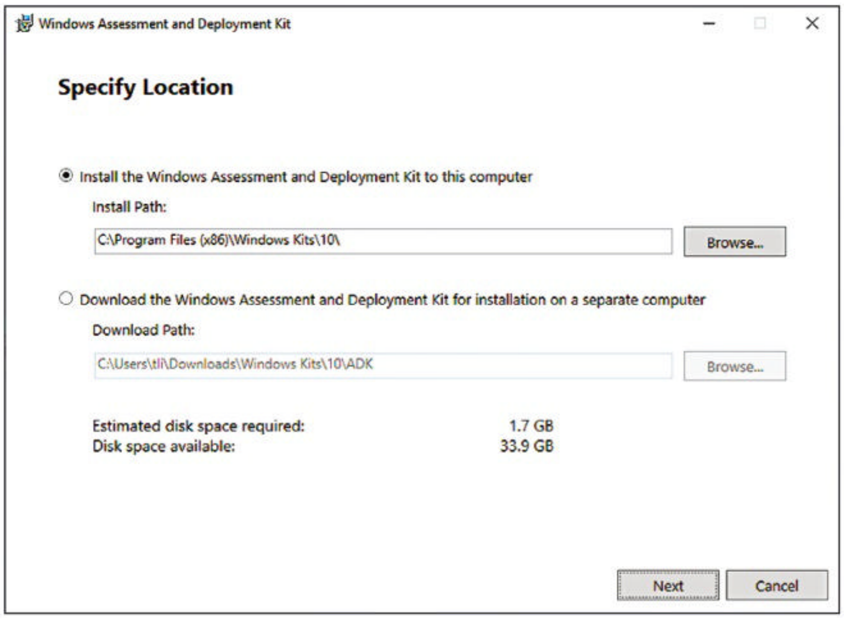
Performance data acquisition is split into two parts: Windows Performance Recorder is responsible for collecting the telemetry information, and the Windows Performance Analyzer component handles the visualization aspects. The Visualizer is not limited to the data generated by WPR; it will also field output from third-party files, as long as they use the ETL data format expected by WPR.

To start the graphical Recorder component (WPR also has a command-line version, but I don't use it here), type *Windows Performance Recorder* in the Start menu to open a window. The combo box is an interesting feature (Figure 2), offering three test categories: *First level triage* collects general information, *Resource Analysis* lets you evaluate the utilization levels of various resources on the test system, and *Scenario Analysis* contains options that classify common user experience (e.g., stuttering of multimedia content).

Pressing the *Start* button triggers the acquisition of status information. As an example of a workload to be tested, I called Visual Studio 2022 and loaded a fairly complex .NET application, Aspire. After loading and closing both in Visual Studio, I stopped the data acquisition session. The wizard then asked where I wanted to save the performance data and offered to create an ETL file named `DESKTOP-MLCJKNK.<TIMESTAMP>`. Selecting the `Compress` option at this point enabled data compression and saved space. Once the work was done, clicking *Save* stored an ETL file that could be evaluated by the graphical analysis component.

Note that processing the acquired data is a computationally intensive process. Even this brief run of Visual Studio and the subsequent file load resulted in a 250MB file. With compression enabled, the eight-core workstation took almost a minute to serve up the data. If the system fails to acquire the test data and reports that the resources are busy, a restart will typically help. If this does not do the trick, I recommend taking a look at a discussion online about fixing this problem [4].

The next step is to launch Windows Performance Analyzer (WPA), either by selecting the *Open in WPA* option in the recording component or from the Start menu on the host operating system. Once the work is complete, the Start menu appears; you can use it to analyze the loaded ETL file. The most important feature in this window is the Graph Explorer on the left, which groups the acquired information in categories. Much like a normal tree view display, the categories can be expanded; double-clicking on one


Windows Assessment and Deployment Kit

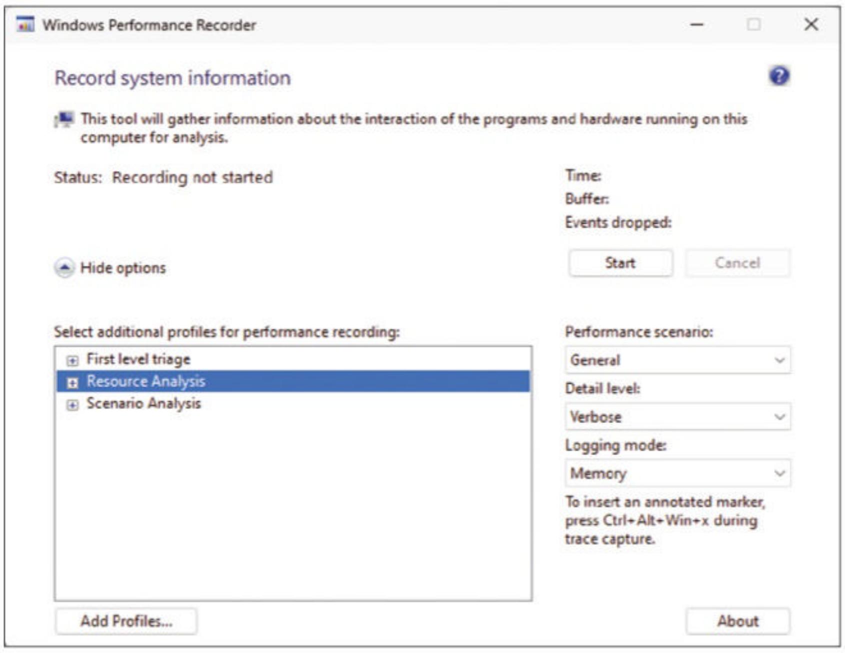
Specify Location

(●) Install the Windows Assessment and Deployment Kit to this computer
Install Path: C:\Program Files (x86)\Windows Kits\10\

( ) Download the Windows Assessment and Deployment Kit for installation on a separate computer
Download Path: C:\Users\tli\Downloads\Windows Kits\10\ADK

Estimated disk space required:	1.7 GB
Disk space available:	33.9 GB

Figure 1: The ADK installation wizard gives you flexible options for setting up the software.


Windows Performance Recorder

Record system information

This tool will gather information about the interaction of the programs and hardware running on this computer for analysis.

Status: Recording not started

Time:
Buffer:
Events dropped:

Select additional profiles for performance recording:
- First level triage
- Resource Analysis
- Scenario Analysis

Performance scenario: General
Detail level: Verbose
Logging mode: Memory

To insert an annotated marker, press Ctrl+Alt+Win+x during trace capture.

Figure 2: One of the ADK tools for recording system performance data.

of the categories loads it in the editing window in the pane to the right. Please note that WPA can trip over its own feet if numerous performance metrics are open at the same time. Close any categories that are not currently required by pressing the "X" buttons at the top right of each window. The timeline shown at the bottom applies for all telemetry data sources displayed. You can therefore correlate read-only memory utilization and CPU utilization over time, provided you have both on display.

Automating Tests

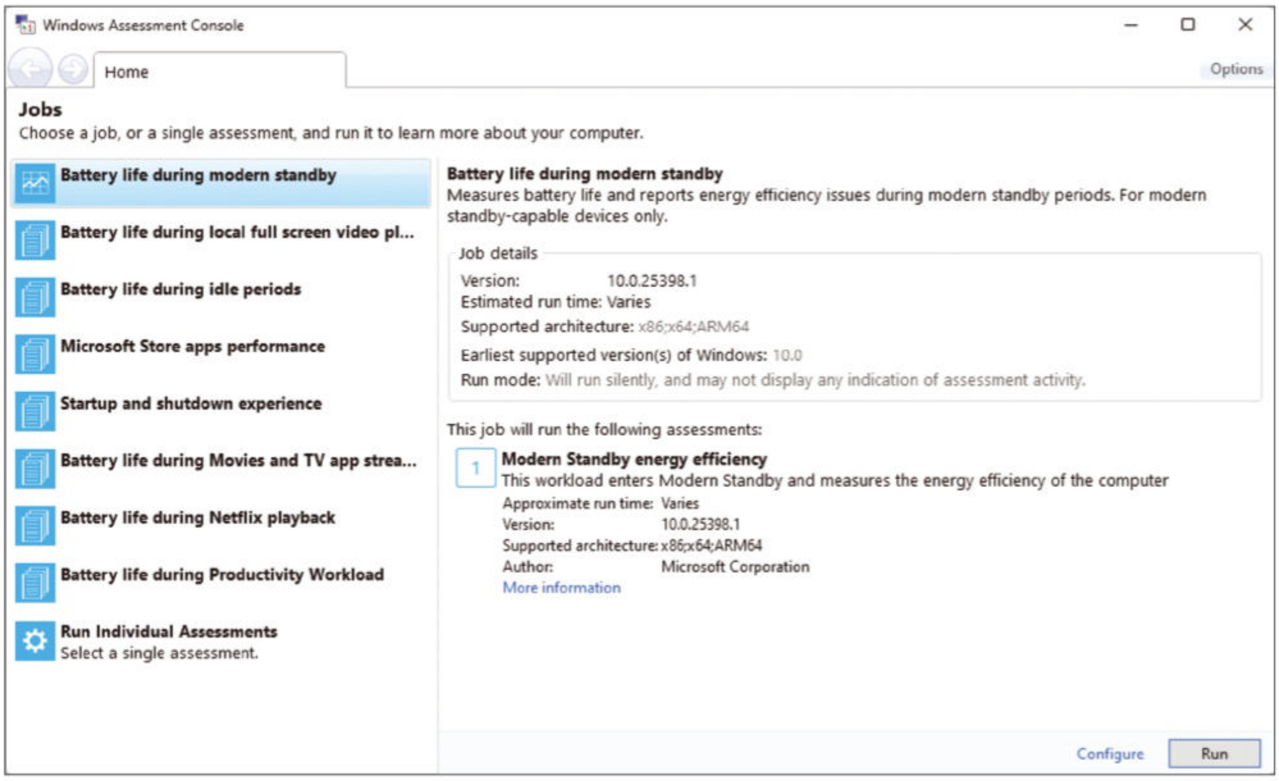
The steps carried out so far generate the load on the system when an integrated development environment is launched manually. Because Visual Studio relies on various caches and temporary storage, this operation is very difficult to reproduce and not exactly useful for the qualification of different system classes. It makes more sense to use an in-house benchmark as the operation to be tested. Windows Assessment Platform is a dedicated component for this purpose. As is my practice throughout this article, I use the graphical user interface. If you're interested in the command-line variant and the various options for advanced deployment scenarios, you can find the documentation online [5] [6].

To start the GUI, open the Start menu on the host workstation and search for *Windows Assessment Console* (WAC). When installed on a laptop, WAC offers tests mainly optimized for evaluating the system's battery life (Figure 3). To work around this problem, you can select the *Run Individual Assessments* option, which offers various advanced performance tests selectable with a group of checkboxes. The *File handling* option is a good choice for the next test; you need administrator authorization to avoid side effects when running the test. It is also important to note any *Warnings* that show up.

Some tests require adjustments to the configuration; for example, the bootloader can cause problems on dual-boot machines. Also note that the tests provided by Windows Assessment Console are generally intended for empty workstations (e.g., running the benchmark could clear the recycle bin). As with any benchmark, you need to avoid any disturbances to the workstation while the tests are running. When the test has completed, you will see a results report. In this example, the report was a filesystem performance analysis test (Figure 4).

Note that you do not need to install Windows Assessment Console fully on remote target systems. Microsoft lets you to export the test logic you want to run on a USB stick; when you then plug the stick into the target system, the performance data is collected automatically after running the `RunJob.cmd` file. To do this, you need to double-click on a task in the *Home* tag to open the matching settings page. The *Package* option then opens a wizard that lets you select a package list. The recommendation is to use a USB stick, as mentioned earlier, to give you a turnkey package that runs the desired assessment on the target system.

To load the information collected on the remote system, return to WAC on the workstation, click the *Options* button at top right, and select the *Import Results* option to load the results for further analysis.



Jobs
Choose a job, or a single assessment, and run it to learn more about your computer.

- Battery life during modern standby
- Battery life during local full screen video pl...
- Battery life during idle periods
- Microsoft Store apps performance
- Startup and shutdown experience
- Battery life during Movies and TV app strea...
- Battery life during Netflix playback
- Battery life during Productivity Workload
- Run Individual Assessments
  Select a single assessment.

Battery life during modern standby
Measures battery life and reports energy efficiency issues during modern standby periods. For modern standby-capable devices only.

Job details
Version: 10.0.25398.1
Estimated run time: Varies
Supported architecture: x86;x64;ARM64
Earliest supported version(s) of Windows: 10.0
Run mode: Will run silently, and may not display any indication of assessment activity.

This job will run the following assessments:
1 Modern Standby energy efficiency
This workload enters Modern Standby and measures the energy efficiency of the computer
Approximate run time: Varies
Version: 10.0.25398.1
Supported architecture: x86;x64;ARM64
Author: Microsoft Corporation
More information

Figure 3: Windows Assessment Console comes with preconfigured test payloads.

Provisioning Packages for Remote Systems

Microsoft also provides functions in the Assessment and Deployment Toolkit that automate the delivery of configuration settings, software packages required to complete business tasks, or both. Microsoft provides a total of three different operating modes for Windows Configuration Designer. They differ in name, in the type of target system to be set up, and in the options offered. For provisioning, I recommend following the workflow specified by Microsoft. In the first step, run through the wizard to the extent possible. With the tool's advanced options, you can then modify the image to handle the intended task in the best possible way.

Unfortunately, Microsoft puts a stumbling block in your way when you are getting started. Although the Designer component is listed as part of the ADK in the MSDN definition, it is delivered by the Microsoft Store [7]; you can enable the familiar workflow toolchain from standard application deployment in the Edge browser, which opens the dialog shown in Figure 5, first prompting you for the type of device to be provisioned. In the following steps, I use the *Provision desktop devices* option. A curiosity at this point is a surviving provisioning link for configuring Windows Phones, although they have not been supported for a long time. You will want to ignore this option.

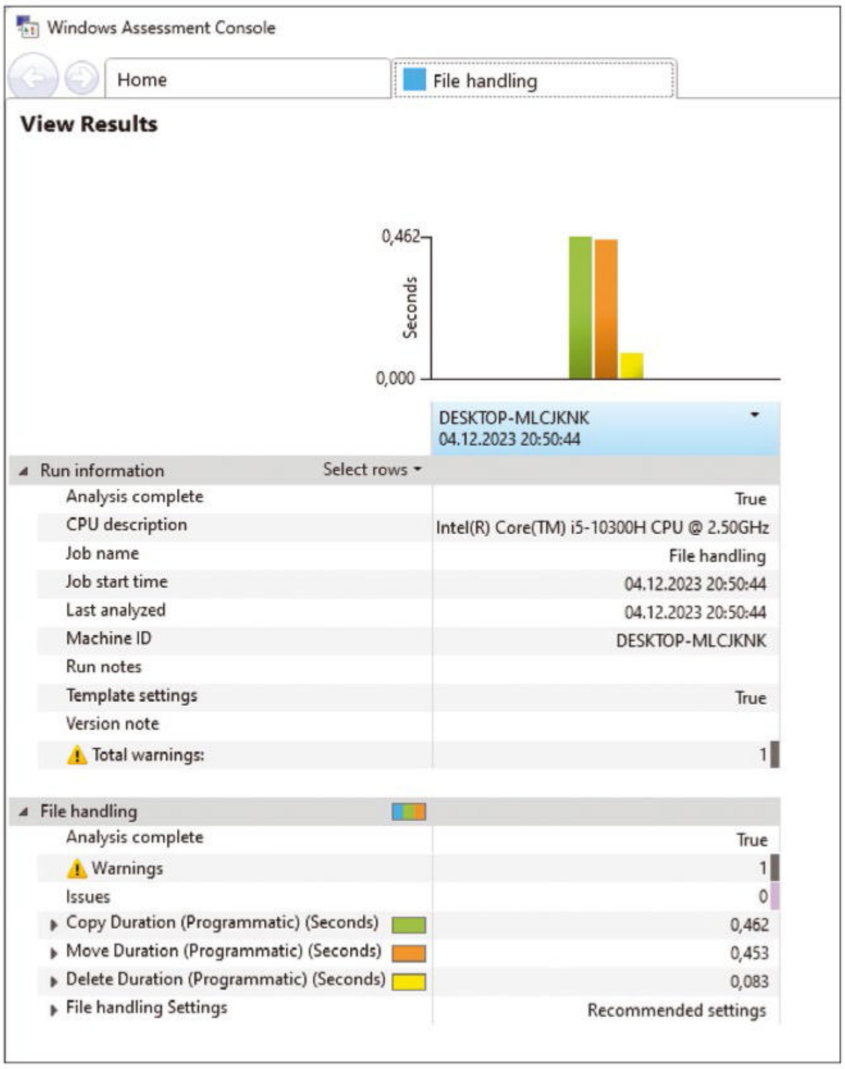
The setup begins by creating a project file, which – when you open it again later – groups all the settings of the matching provisioning profile. The next task on your list involves a wizard consisting of six steps. In the first step, the system prompts you for a naming scheme, which is then used to name the newly created instances. Microsoft creates some meta variables that either include the device serial number or a randomly generated string in the device name. The purpose of this procedure is to prevent network manager and provisioning systems from being flooded with a bunch of new device instances with the same name.

The second (optional) step relates to setting up credentials for a wireless local area network (WLAN). Microsoft currently only supports open wireless networks or wireless networks secured by WPA2 Personal. You then need to set up admin access to the newly created instances in the third step. The easiest way is to assign a username and password to create a local administrator account.

In the fourth step, Windows Configuration Designer adds one or more applications that the profile will execute on the target system in the scope of the deployment. The tool expects a standard executable file as the packaging format. The same applies to adding certificates (step five). The sixth and final step is exporting the package, which results in a PPKG file as output.

How you apply this PPKG package then depends on what stage of provisioning the target system has reached. If the system is still completing the initial setup, you just need to plug in a USB drive. In some cases, you might need to press the Windows key five more times to trigger the process. On pre-installed systems, you can either open the PPKG file in the system settings or by left-clicking in Windows Explorer.

The installation then generally follows the same process as any normal Windows application. I cannot cover all potential methods of using PPKG files here, but Microsoft provides a useful guide online [8] that explains the various types and details.



DESKTOP-MLCJKNK 04.12.2023 20:50:44

Run information	
Analysis complete	True
CPU description	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz
Job name	File handling
Job start time	04.12.2023 20:50:44
Last analyzed	04.12.2023 20:50:44
Machine ID	DESKTOP-MLCJKNK
Run notes	
Template settings	True
Version note	
Total warnings:	1

File handling	
Analysis complete	True
Warnings	1
Issues	0
Copy Duration (Programmatic) (Seconds) 🟩	0,462
Move Duration (Programmatic) (Seconds) 🟧	0,453
Delete Duration (Programmatic) (Seconds) 🟨	0,083
File handling Settings	Recommended settings

Figure 4: An example of the results achieved with Windows Assessment Console.

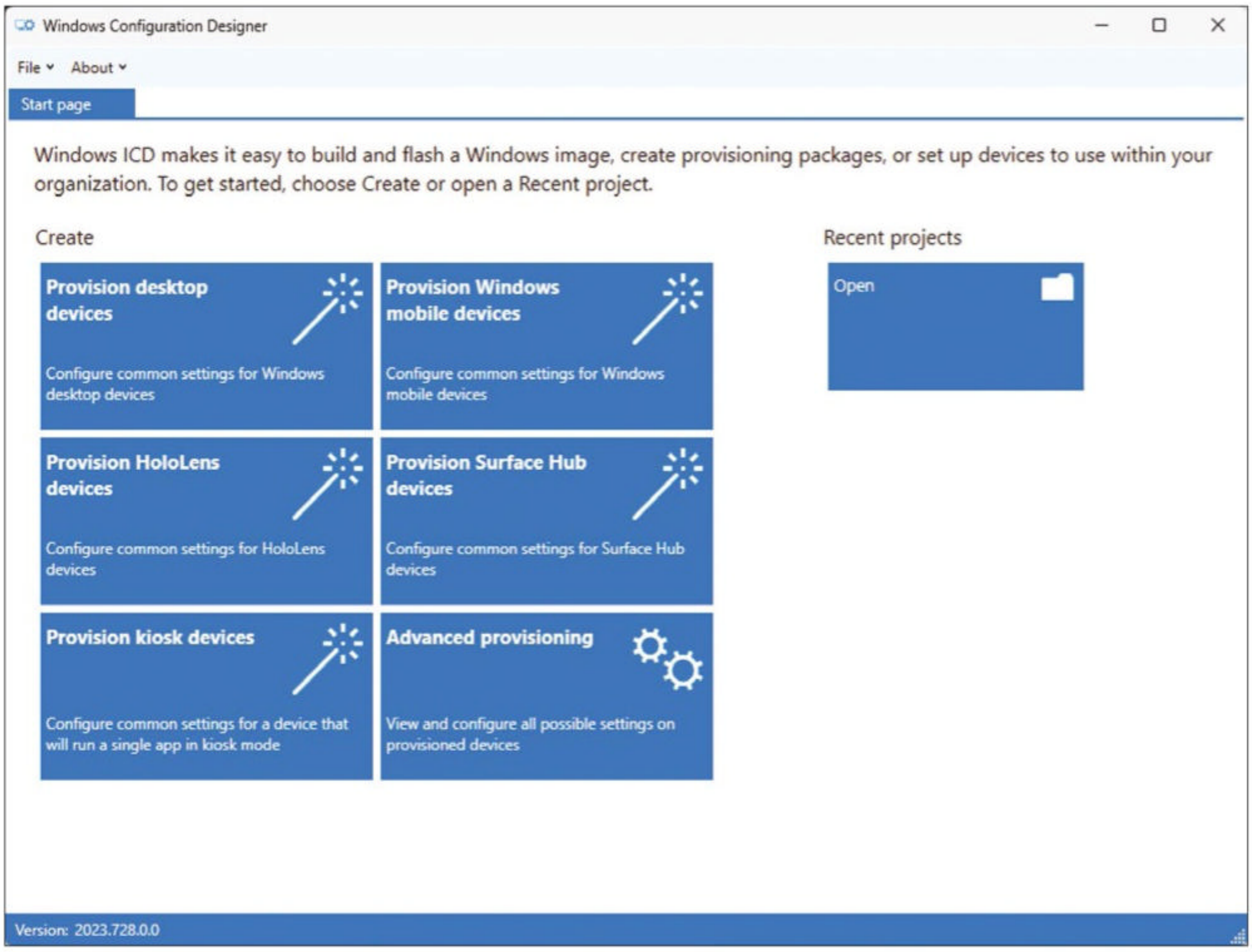
Advanced Settings for PPKG Files

This kind of basic system configuration, as discussed so far, is always

useful if you need to populate newly created systems with settings in the scope of a greenfield deployment. From a certain level of complexity, it might be advisable to use the files in a brownfield scenario to deliver configurations, applications, or the like. To do this, you will need to use the advanced editor to modify the content by selecting the *Switch to advanced editor* option. Please note that this permanently prevents the use of the step-by-step wizard for this project configuration file.
A configuration screen then appears with the properties that already exist in the file on the right and more options offered in the left section. The *Remove* button removes options created by the wizard; you can easily reset the project file to a completely empty state at this point.
Adding new elements is an idiosyncratic process. In both the left and right tree views, double-clicking populates the empty center of the screen with the settings of the element in question. You can then edit these in the same way as you would expect in Visual Studio or other rapid application development (RAD) tools. In many cases, selecting an option in the tree view Available Customizations pane also results in MSDN information appearing in the lower part of the center panel. When you use Windows Configuration Designer, it always makes sense to have an Internet connection. To add options, first select the desired category in the left-hand window. A configuration dialog then appears; it only contains a single combo box in this case. After selecting an option, the created category appears in the tree view on the right.

Windows PE

The final feature I want to discuss is Windows PE. Up to and including Windows 10 version 1809, the Windows Preinstallation Environment was an integral component of Windows ADK. In later versions, an additional `akdwinpesetup.exe` file launches another variant of the installation wizard that you are familiar with from the standard Windows ADK. The delivery then follows the familiar pattern – note that a full install requires a further 4GB of storage space.
Windows PE itself is a feature-reduced Windows that can be useful for various administration and management tasks. If you choose to use Windows PE, you must observe the restrictions documented in detail [9], including automatically restarting the system every 72 hours and discarding all changes made during runtime as part of the reboot. This problem can be avoided by modifying the Windows PE image [10].



Windows ICD makes it easy to build and flash a Windows image, create provisioning packages, or set up devices to use within your organization. To get started, choose Create or open a Recent project.

Create

Provision desktop devices – Configure common settings for Windows desktop devices
Provision Windows mobile devices – Configure common settings for Windows mobile devices
Provision HoloLens devices – Configure common settings for HoloLens devices
Provision Surface Hub devices – Configure common settings for Surface Hub devices
Provision kiosk devices – Configure common settings for a device that will run a single app in kiosk mode
Advanced provisioning – View and configure all possible settings on provisioned devices

Recent projects

Open

Version: 2023.728.0.0

Figure 5: Administrators can choose between different target platforms in the Windows Configuration Designer installer.

Conclusions

From a certain deployment size, every manual configuration action is a potential source of error and unnecessary overhead. With Windows ADK, Microsoft provides a rich admin toolbox that boosts the efficiency of managing Windows-based systems. Besides methods for quantifying system performance and logging data for those cases in which you encounter performance-related issues, Microsoft also offers tools to help automate the process of managing Windows systems. ■

Info

[1] Windows ADK components: [https://learn.microsoft.com/en-us/windows-hardware/get-started/kits-and-tools-overview]

[2] Windows ADK download: [https://learn.microsoft.com/en-us/windows-hardware/get-started/adk-install]

[3] WPRControl API: [https://learn.microsoft.com/en-us/windows-hardware/test/wpt/wprcontrol-api-reference]

[4] Fixing problems with WPR: [https://answers.microsoft.com/en-us/windows/forum/all/windows-performance-recorder-wont-start-recording/958c45c9-dd4f-4ba2-bbe2-b90ba0b01e56]

[5] Assessment Platform command-line options: [https://learn.microsoft.com/en-us/windows-hardware/test/assessments/assessment-platform-command-line-syntax]

[6] Windows Assessment Toolkit technical reference: [https://learn.microsoft.com/en-us/windows-hardware/test/assessments/windows-assessment-toolkit-technical-reference]

[7] Windows Configuration Designer: [https://apps.microsoft.com/detail/9NBLGGH4TX22?rtc=1&hl=en-us&gl=EN]

[8] Rolling out provisioning packages: [https://learn.microsoft.com/en-us/windows/configuration/provisioning-packages/provisioning-apply-vpackage]

[9] Restrictions of Windows PE: [https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/winpe-intro?view=windows-11]

[10] Boot media for Windows PE: [https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/winpe-create-usb-bootable-drive?view=windows-11]

The Author

Tam Hanna (*tam.hanna* on Instagram) has seen the embedded space inside and out. His multidecade work has involved coding games for early mobile phones, designing metrology systems, and tackling various projects for civil and military clients.

Legal or litigation hold and eDiscovery in Microsoft Teams

Holdover

A comprehensive suite of tools and features support legal or litigation hold and eDiscovery in Microsoft Teams. By Conor Fitzgerald

A legal or litigation hold is a process an organization uses to preserve all forms of relevant information when litigation is reasonably anticipated. In Microsoft 365, you can use the eDiscovery tool with the Microsoft Purview portal to place content locations like Exchange Online mailboxes, SharePoint Online and OneDrive for Business sites, and Microsoft Teams on hold. Legal or litigation hold should not be confused with data retention policies.

A *legal hold* is a process that an organization uses to preserve all forms of relevant information when litigation is reasonably anticipated. When a hold is placed on a content location, all existing content is preserved, and any content that is edited or deleted is also preserved, ensuring that all relevant data is available for the legal process.

Data retention policies in Microsoft 365 are used to manage the life cycle of data by specifying how long content should be retained and what should happen to it when the retention period expires. Retention policies can be applied to specific content locations or to the entire organization. Unlike legal hold, which is used to preserve data for legal purposes, retention policies are used to manage data for compliance, regulatory, or business reasons.

Sensitivity labels in Microsoft 365 allow you to classify and protect your organization's data, as well as apply protection settings such as encryption or visual markings to content. Although sensitivity labels do not directly affect the legal hold process, they can help ensure that sensitive data is properly protected and managed throughout its life cycle, including during a legal hold. By using sensitivity labels to classify and protect data, you can help ensure that sensitive information is not accidentally or intentionally deleted or altered during a legal hold.

When implementing a legal hold in Microsoft 365, you need to consider several things, including:

- *Scope of the hold*, which includes content locations and the types of content that need to be preserved
- *Communication of the hold* to relevant persons, including responsible parties and IT staff
- *Duration of the hold*, to ensure that it remains in place for as long as necessary
- *Documentation of the hold process*, including the steps taken to preserve data and any changes made to the hold over time
- *Compliance*, to ensure that the hold process complies with relevant legal and regulatory requirements
- *Process verification* through regular testing of the legal or litigation hold process by completing test eDiscovery exports to ensure consistency across a sample of users set on litigation hold.

Geographic regulations might also affect how legal hold is implemented in Microsoft 365. These regulations vary by country and region, and it is important for organizations to ensure that their legal hold processes comply with all relevant laws and regulations. Some examples of geographic regulations that could affect legal hold include:

- General Data Protection Regulation (GDPR) – applies to organizations operating within the European Union (EU) and governs the processing of personal data.

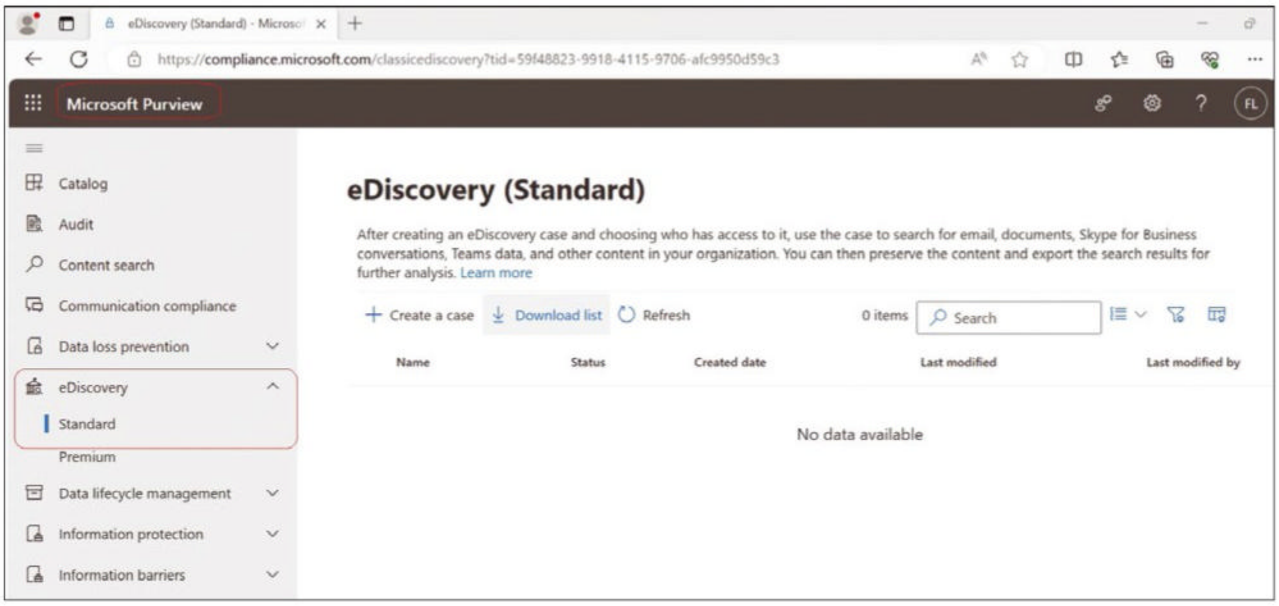
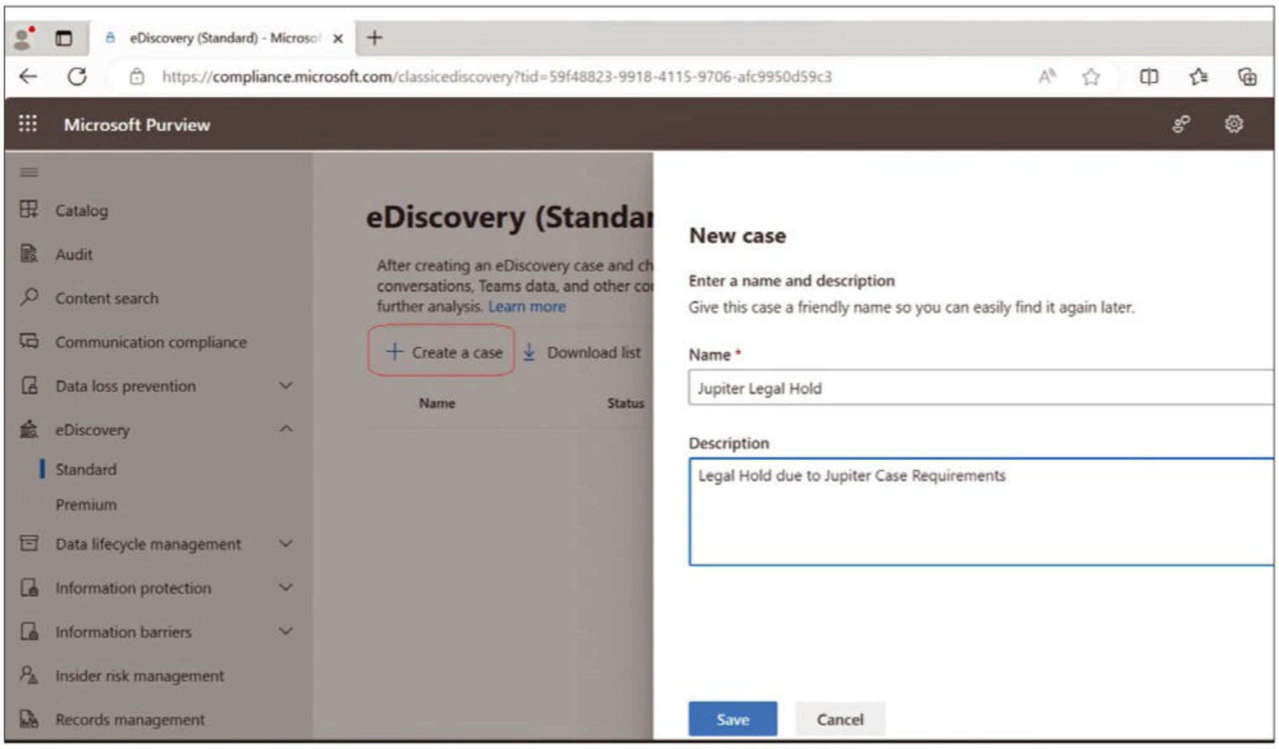

eDiscovery (Standard) - Microsoft
https://compliance.microsoft.com/classicediscovery?tid=59f48823-9918-4115-9706-afc9950d59c3
Microsoft Purview
Catalog
Audit
Content search
Communication compliance
Data loss prevention
eDiscovery
Standard
Premium
Data lifecycle management
Information protection
Information barriers
eDiscovery (Standard)
After creating an eDiscovery case and choosing who has access to it, use the case to search for email, documents, Skype for Business conversations, Teams data, and other content in your organization. You can then preserve the content and export the search results for further analysis. Learn more
Create a case
Download list
Refresh
0 items
Search
Name
Status
Created date
Last modified
Last modified by
No data available

Figure 1: To create an eDiscovery case, you need to be assigned the eDiscovery Manager or eDiscovery Administrator role.

- California Consumer Privacy Act (CCPA) regulation – applies to organizations doing business in California and governs the collection, use, and sharing of personal information.
- Personal Information Protection and Electronic Documents Act (PIPEDA) – applies to organizations operating in Canada and governs the collection, use, and disclosure of personal information.

In Microsoft 365, you can create an eDiscovery case from the Microsoft Purview portal [1] (Figures 1 and 2). It is important to point out that the eDiscovery Manager and eDiscovery Administrator roles and responsibilities are different. The *eDiscovery Manager* has the ability to create and manage eDiscovery cases, including placing content locations on hold, conducting searches, and exporting data. eDiscovery Managers can only access the cases they create or are members of. The *eDiscovery Administrator* has the same abilities as the eDiscovery Manager, but can also manage eDiscovery cases created by others and configure organization-wide eDiscovery settings.



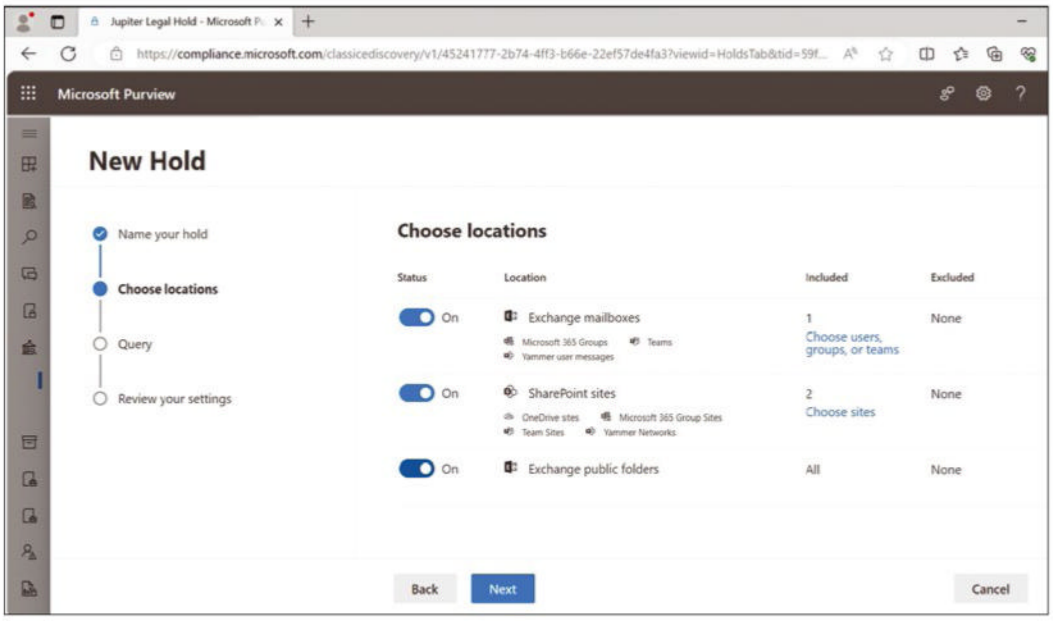
Name: Jupiter Legal Hold

Description: Legal Hold due to Jupiter Case Requirements

Figure 2: The eDiscovery Manager creates a new eDiscovery (Standard) case.

eDiscovery Administrators have access to all eDiscovery cases in the organization.

As part of the eDiscovery case, the eDiscovery Manager creates a new legal or litigation hold and, within that hold, selects which Exchange Online mailboxes to include, which SharePoint sites (and OneDrive locations) to include, and whether to include Exchange public folders (Figure 3).


Jupiter Legal Hold - Microsoft P...

https://compliance.microsoft.com/classicediscovery/v1/45241777-2b74-4ff3-b66e-22ef57de4fa3?viewid=HoldsTab&tid=59f...

Microsoft Purview

New Hold

- Name your hold
- Choose locations
- Query
- Review your settings

Choose locations

Status	Location	Included	Excluded
On	Exchange mailboxes Microsoft 365 Groups Teams Yammer user messages	1 Choose users, groups, or teams	None
On	SharePoint sites OneDrive sites Microsoft 365 Group Sites Team Sites Yammer Networks	2 Choose sites	None
On	Exchange public folders	All	None

Back | Next | Cancel

Figure 3: Choosing locations for the legal or litigation hold.

Now that you understand the steps involved in creating a litigation hold, you need to understand where the relevant data is stored (Figure 4) and create a hold policy on the basis of the relevant data components and their associated storage locations.

Challenges

The management of Microsoft Teams data is discussed in the "Microsoft Teams Operational Overview" box. One of the major challenges associated with creating a litigation hold for a Microsoft Teams user relates to the potential

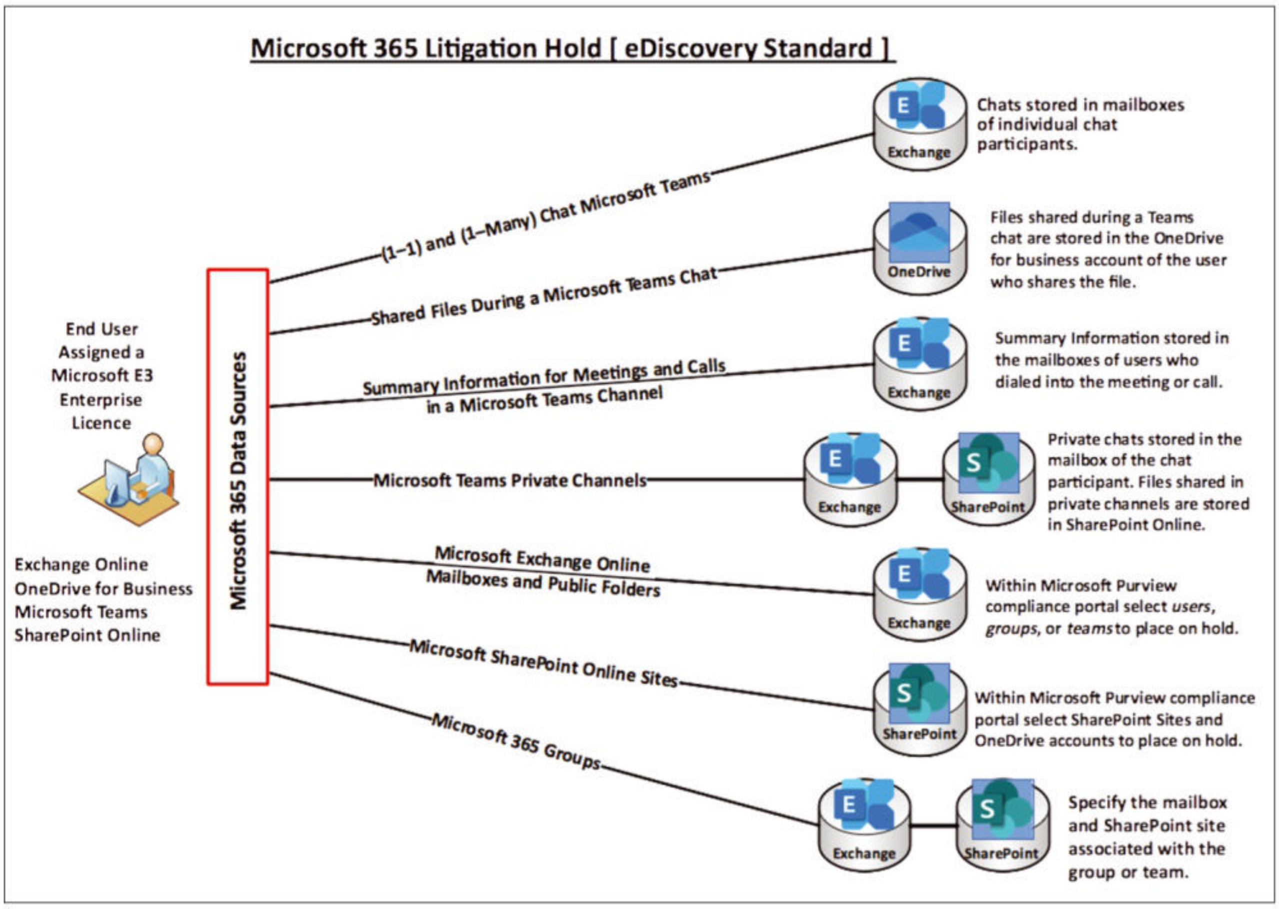

Microsoft 365 Litigation Hold [eDiscovery Standard]
End User Assigned a Microsoft E3 Enterprise Licence
Exchange Online
OneDrive for Business
Microsoft Teams
SharePoint Online
Microsoft 365 Data Sources
(1–1) and (1–Many) Chat Microsoft Teams
Exchange
Chats stored in mailboxes of individual chat participants.
Shared Files During a Microsoft Teams Chat
OneDrive
Files shared during a Teams chat are stored in the OneDrive for business account of the user who shares the file.
Summary Information for Meetings and Calls in a Microsoft Teams Channel
Exchange
Summary Information stored in the mailboxes of users who dialed into the meeting or call.
Microsoft Teams Private Channels
Exchange
SharePoint
Private chats stored in the mailbox of the chat participant. Files shared in private channels are stored in SharePoint Online.
Microsoft Exchange Online Mailboxes and Public Folders
Exchange
Within Microsoft Purview compliance portal select users, groups, or teams to place on hold.
Microsoft SharePoint Online Sites
SharePoint
Within Microsoft Purview compliance portal select SharePoint Sites and OneDrive accounts to place on hold.
Microsoft 365 Groups
Exchange
SharePoint
Specify the mailbox and SharePoint site associated with the group or team.

Figure 4: A topology diagram of where each data component is stored.

need to include a shared SharePoint site, a shared Exchange mailbox, or both in the legal hold, even if some of the team members using these resources are not subject to that hold. As a consequence, you could end up in a situation in which data that should not be discoverable now inadvertently becomes discoverable because the “shared” resources are included in the litigation hold.

Summary

Microsoft 365 provides a comprehensive suite of tools and features to support legal or litigation hold and eDiscovery in Microsoft Teams. By understanding the operational dependencies and data storage locations used by Microsoft Teams, you can effectively implement legal holds and manage data for compliance, regulatory, or business reasons. However, you need to be aware of the challenges associated with creating a litigation hold for a Microsoft Teams user, and you must consider carefully the scope of the hold process, especially in relation to shared resources (e.g., SharePoint sites and Exchange mailboxes) to ensure that only relevant data is preserved and discoverable. ■

Info

[1] Purview portal: [https://compliance.microsoft.com]

Author

Conor Fitzgerald is an IT Manager and has been working in IT infrastructure for more than 20 years. He is hugely interested in all aspects of IT.

Microsoft Teams Operational Overview

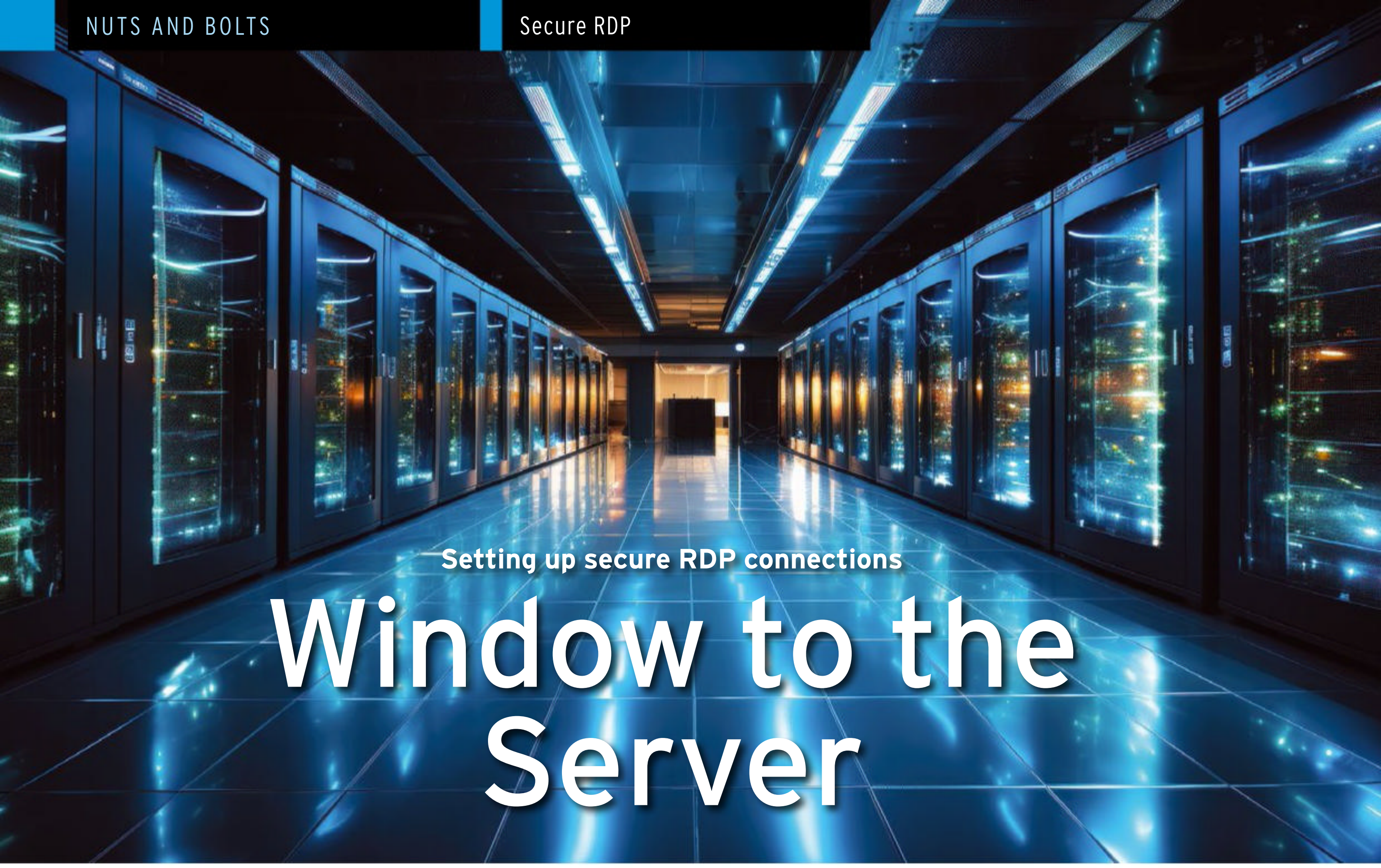
Microsoft Teams is part of the Microsoft 365 suite of products and is included as part of the Microsoft 365 E3 licence. From an operational perspective, Microsoft Teams has dependencies on the Exchange Online, SharePoint Online, and OneDrive for Business Microsoft 365 components.
Teams uses identities stored in Microsoft Azure Active Directory (Azure AD). When you create a new Team within Microsoft Teams, the following items are created by default:

- A new Microsoft 365 group
- A new SharePoint site and document library to store team files
- An Exchange Online shared mailbox and calendar
- A OneNote notebook
- Ties into other Microsoft 365 apps (Planner, Power BI, etc.)

Within the Microsoft Teams application, you can create teams and channels. Teams can be private, for which people need permission to join; public, for anyone in the organization; and org-wide, in which everyone in the organization automatically joins.

Channels are dedicated sections within a team to keep conversations organized by specific topics, projects, functional areas, and so on. They can be open to all team members (standard channel), selected team members (private channels), or selected people both inside or outside the team (shared channels). Within a channel you can chat with other team members, share files with the team, and meet over audio or video with colleagues.
Microsoft Teams uses OneDrive for Business and SharePoint for the storage of shared file data and uses Exchange Online for channel conversations; chat messages are journaled through Exchange Online. Files that you upload to a channel are stored in your Teams SharePoint folder. Those files are available in the *Files* tab at the top of each channel. Files that you upload to a one-on-one or group chat are stored in the *Microsoft Teams Chat Files* folder in your OneDrive for Business and are shared only with the people in that conversation. These files are available in the *Files* tab at the top of a chat. Meeting recordings are stored in OneDrive or SharePoint depending on the meeting type.

■



Setting up secure RDP connections

Window to the Server

Know when to make RDP available on the Internet for remote access to Windows Server and how to configure connections securely when you do. By Thomas Joos

The Remote Desktop Protocol (RDP) is still one of the most important methods for remote access. In this article, I show you how to enable RDP and use it in an effective and secure way. As an admin, this means you can manage your Windows servers with confidence and without having to compromise security. In combination with the Windows Admin Center, interesting options open up for managing Windows servers directly from the desktop.

Most of the settings described in this article also work on Windows Server 2016 and 2019, and on Windows 10 and 11, but I focus on Windows Server 2022 for the configuration. As things stand at present, the approach also works on Windows Server vNext/2025. RDP connections are primarily of interest for the graphical user interface (GUI) but can also be established on Server Core servers. In this case, you need to run `sconfig.exe` on the Server Core server to get started. In the *Remote desktop* menu item, you can then choose whether you want to enable or disable RDP, to which you can connect directly with the RDP client. This approach works with the Windows Admin Center (WAC) just as it does on servers with a GUI. In this case, you can access the command prompt and PowerShell over the RDP connection, along with the same GUI tools that you use on the Server Core console. The use of RDP in Windows Admin Center is interesting, in that it lets you access your servers over HTTPS in the web browser. If you connected your servers to Azure free of charge with Azure Arc, you can secure access to your servers over the Internet with Windows Admin Center in Azure over RDP without requiring a virtual private network (VPN).

RDP files can be digitally signed and protected with certificates, which enhances security while avoiding error messages popping up because of incorrect signing. All these options suggest that Microsoft will continue to rely on RDP in the future and even expand its support.

Enabling Remote Desktop

Without installing Remote Desktop Services, you can use two active RDP sessions for server management of Windows servers. On Windows 10/11 Pro/Enterprise, only one user can connect over RDP. During the remote session in Windows 11, the operating system locks the user's session on the desktop. This situation is not the case for servers, unless you use the same username for the RDP connection as for the console connection. RDP client access licenses (CALs) are not required for managing the server over RDP, provided the RDP connections are restricted to managing the server.

RDP access is disabled by default. You can enable the *Allow remote connections to this computer* option in the GUI by running `sysdm.cpl` and selecting the *Remote* tab. After confirming, you can connect to the computer over RDP as an admin user, for example, with the local RDP client, which you can launch by typing `mstsc.exe`. Other tools used for this purpose are Royal TS [1] or the Microsoft Remote Desktop Connection Manager [2].

Enabling the *Allow connections only from computers running Remote Desktop with Network Level Authentication* option ensures that the accessing PC first needs to authenticate regardless of the user login. The Network Level Authentication (NLA) used here is not a problem in Active Directory (AD) environments. However, if you are accessing a server from a PC outside the domain – from your home office or with an RDP app on a smartphone or tablet, for example – the server refuses the connection. Generally speaking, this option is useful and should be deployed wherever possible to prevent cyberattacks.

For access over RDP, the user must be a member of the server's Local Administrator or Remote Desktop Users group. You can set this up in `sysdm.cpl` on the *Remote* tab with the *Select User* button. The options are available under *System\Remote Desktop* in the Windows Server 2022 Settings app.

Windows Admin Center and PowerShell

If the Windows Admin Center is already in use on the network and the computer in question is connected, you can enable RDP over the network from *Settings\Remote Desktop* (Figure 1). Access is also possible by WAC and the Remote Desktop section. In this case, the connection is opened by the internal WAC gateway, not the RDP client. The client uses HTTPS to connect to the Admin Center, and the RDP connection to the server is opened from there. Fewer settings can be configured here, but the connection does not need to be configured.

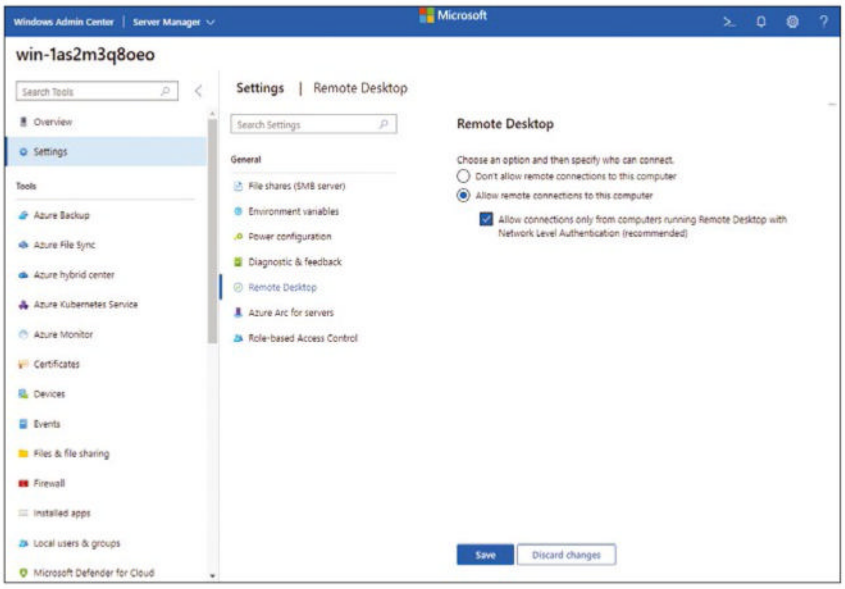
If you want to control access from the local *Remote Desktop Users* group in the Windows Admin Center, go to the *Local users & groups* sidebar item, where you can maintain local user accounts and their groups, and add user accounts to the group in your web browser. You do not need to do this step for admins.

The PowerShell `-computername` parameter lets you query whether or not RDP is running on a server and works over the network, as well:

```
Get-CimInstance ↩
 -Namespace "root\cimv2\TerminalServices" ↩
 -Class win32_terminalservicesetting | ↩
  select ServerName, AllowTSConnections
```

A value of *1* for *AllowTSConnections* shows that RDP is active on the server. If you see a value of *0*, you can enable RDP over the network or locally with PowerShell:

```
$rdp = Get-CimInstance ↩
 -Namespace "root/cimv2/TerminalServices" ↩
 -ClassName "Win32_TerminalServiceSetting" ↩
 -ComputerName <Servername>
$rdp | Invoke-CimMethod ↩
  -MethodName "SetAllowTSConnections" ↩
  -Arguments @{AllowTSConnections=1;↩
             ModifyFirewallException=1}
```



Remote Desktop

Choose an option and then specify who can connect.

○ Don't allow remote connections to this computer

● Allow remote connections to this computer

✓ Allow connections only from computers running Remote Desktop with Network Level Authentication (recommended)

Figure 1: In Windows Admin Center, you can enable the remote desktop and connect to a server in your web browser over RDP.

The command also sets up the required firewall rules to prevent Windows firewall blocking access.

Group Policies and Firewall Settings

Generally speaking, you can use group policies to control the remote desktop for managing a server. These settings can be found in Group Policy Management in *Computer Configuration\Policies\Administrative Templates\Windows Components\Remote Desktop Services\Remote Desktop Session Host\Connections*. The setting is named *Allow users to connect remotely by using Remote Desktop Services*. If you enable the settings, Windows disables the options in the GUI and enables RDP. These settings can no longer be changed in the user interface.

If you want to control the remote desktop with group policies, you also need to make sure the firewall rules are either defined manually or by a group policy. These settings are found in *Windows Firewall: Allow inbound Remote Desktop exceptions* , which resides in the path *Computer Configuration\Policies\Administrative Templates\Network\Network Connections\Windows Firewall\Domain Profile*.

RDP Files for Quick Access

In addition to various additional tools for establishing RDP connections, the local RDP client in Windows is often launched by typing `mstsc.exe`. You can configure various settings in the client, such as the desired resolution; pass-through of local resources such as the clipboard, drives, or printers; and the connection quality. All of these settings can be saved in an RDP file by pressing *Save as*. If you double-click on this file, Windows automatically opens the Remote Desktop Client and connects your user account after authentication. You can also save the login data. You will not find this feature in the Remote Desktop

Client, but in *Credential Manager* in the Control Panel (Figure 2). Login credentials can also be changed, deleted, or added here.

If you connect to a server by double-clicking on an RDP file, a security warning flashes up in most cases because of an unknown certificate. This warning can be avoided by signing the RDP file. You do not need a complete certificate configuration, but you can self-sign RDP files created with built-in Windows tools. The use of these files does entail some security risks. For example, an RDP file that you publish on the network can be used to lure other administrators to malware-infected servers and is a known attack vector. Therefore, it generally makes sense to sign frequently used RDP files – especially if you want other admins to use them, too.

Signing RDP Files

Windows comes with the `rdpsign.exe` command-line tool for signing RDP files. It is very easy to use and only has a few options. If a certificate authority (CA) is already in use on the network (e.g., from the AD certificate services), it makes perfect sense to use certificates from this CA to sign your RDP files.

The certificate of the root CA must be installed on all computers that want to connect to the signed RDP file. If the computers are all members of the AD, group policies automatically distribute the root CA's certificates on the network. You need to verify this in advance; the RDP client will not otherwise open the connection because it fails to recognize the certificate.

Signing certificates is a simple matter. Use the `/sha256` parameter to transfer the fingerprint of the certificate with which you want to sign the file. The issuing CA must be trusted by all computers that use this file, and the certificate must support AES 256-bit and be installed on the computer on which you are signing. You can find the thumbprint on the certificate's *Details* tab from either `certmgr.msc` for user certificates or `certlm.msc` for computer certificates.

When you pass the thumbprint in to `rdpsign.exe`, make sure you type it without spaces, except at the beginning and end of the character string:

```
rdpsign.exe ↵
  /sha256 df3e3a36e67a45e8e4cd↵
   96a26241ad871679ce1c ↵
  c:\temp\dl20.rdp
```

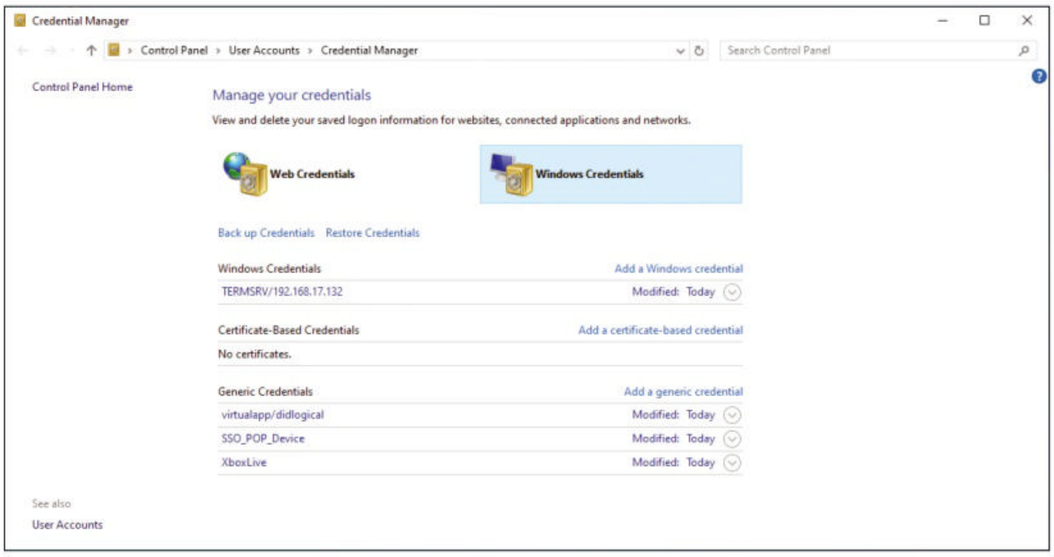
The command then reports the successful signing. If you want to distribute several RDP files, you can sign all of the files at once with a single certificate. To do this, simply create a comma-separated list. You will also notice that signing works when you use the file. Instead of an error message, you will see a message stating that the RDP file is signed and that you can view the publisher at this point. If this is not desired, you can use group policy to control the behavior.

Look for the *Specify SHA1 thumbprints of certificates representing trusted .rdp publishers* setting in *Computer Configuration* or *User Configuration* under *Administrative Templates\Windows Components\Remote Desktop Services\Remote Desktop Connection Client*. This is also where you enter the certificate's SHA-1 value.

Monitoring RDP Sessions on the Server

As soon as RDP is enabled on a server, up to two admins with different usernames can use the RDP client to connect to that server. A user can also be logged on at the console with a separate user account. You can run the `query session` command to discover whether any users are currently connected over RDP. After entering the command, you can see any existing RDP sessions or users logged in at the console, as well as the account names with which the users are connected. The `query user` command shows the connected users and when they logged in by RDP.

At this point, you can reset a session and disconnect from the server if you have any connection problems (e.g., an account is disconnected or a second admin cannot log in). Disconnecting can also be useful if you suspect that the second connection is an unwanted session. The `reset session<ID>` command is used for


Credential Manager
Control Panel › User Accounts › Credential Manager
Search Control Panel
Control Panel Home
Manage your credentials
View and delete your saved logon information for websites, connected applications and networks.
Web Credentials | Windows Credentials
Back up Credentials | Restore Credentials

Windows Credentials	Add a Windows credential
TERMSRV/192.168.17.132	Modified: Today
Certificate-Based Credentials	Add a certificate-based credential
No certificates.	
Generic Credentials	Add a generic credential
virtualapp/didlogical	Modified: Today
SSO_POP_Device	Modified: Today
XboxLive	Modified: Today

See also
User Accounts

Figure 2: Managing the login information, which is also important for RDP connections.

this purpose. Again, the `query user` command displays the ID. To view the processes launched in RDP sessions on the server, type `query process`.

Changing the RDP Port and Firewall Rules

By default, RDP listens for connections on port 3389. It might make sense to change the port, which means the server can still be reached by RDP. Of course, tools such as Nmap will also find the new port, but it still enhances security because malware and cybercriminals will initially try their luck on port 3389.

You can change the port in the registry. The settings are in *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp*. The port is specified by the *PortNumber* DWORD value. Change the value to *Decimal*; after making the change, save the setting, and restart the server, but also make sure you disable the old firewall rules for access to port 3389 and create new rules that allow access to the new port. In PowerShell, you can check the new port with the command:

```
Get-ItemProperty ➚
  -Path "HKLM:\SYSTEM\CurrentControlSet\➚
        Control\Terminal Server\➚
        WinStations\RDP-Tcp" ➚
  -name "PortNumber"
```

You can also change the port for RDP in PowerShell, which is useful, for example, if you want to script the change. For example, to use port 3390 for RDP, enter the command:

```
Set-ItemProperty ➚
  -Path "HKLM:\SYSTEM\CurrentControlSet\➚
        Control\Terminal Server\➚
        WinStations\RDP-Tcp" ➚
  -name "PortNumber" ➚
  -Value 3390
```

The firewall rules for TCP and UDP can be changed in PowerShell, as well. If you save the port as a variable up front (e.g., `$RDPPort = 3390`), you can script it all, which means you can pass in the port:

```
Set-ItemProperty ➚
  -Path "HKLM:\SYSTEM\CurrentControlSet\➚
        Control\Terminal Server\➚
        WinStations\RDP-Tcp" ➚
  -name "PortNumber" ➚
  -Value $RDPPort
```

Because you saved the port as a variable, you could also set matching firewall rules in the same script and use the variable to specify the port. Of course, you could enter the value directly, but a variable avoids errors and makes it easier to change the value:

```
New-NetFirewallRule ➚
  -DisplayName "RDP-TCP-In" ➚
  -Profile "Domain" ➚
  -Direction Inbound -Action Allow ➚
  -Protocol TCP -LocalPort $RDPPort
New-NetFirewallRule ➚
  -DisplayName "RDP-UDP-In ➚
  -Profile "Domain" ➚
  -Direction Inbound -Action Allow ➚
  -Protocol UDP -LocalPort $RDPPort
```

For `Profile`, specify the firewall profile in which the rule will apply. Besides the `Domain` value for the domain profile, you also have `Public` and `Private`. However, `Domain` is the right choice for networks. You will want to avoid public access to RDP if possible. If external access is necessary, routing with Azure Arc or a VPN is preferable.

TCP and UDP over RDP

RDP uses both TCP and UDP for data traffic and plays an important role for configuring firewall rules and in terms of performance. UDP can be faster than TCP data traffic in certain circumstances. The client and server negotiate which protocol to use for the current connection. Firewall rules for TCP and UDP therefore need to be enabled in parallel.

In general, group policy can be used to specify which protocol the server will use. If you only enable the use of TCP at this point, you do not need a firewall rule for UDP. The settings can be found in *Select RDP transport protocols* under *Computer Configuration\Policies\Administrative Templates\Windows Components\Remote Desktop Services\Remote Desktop Session Host\Connections*. If you select *Use UDP or TCP*, the client attempts primarily to communicate with UDP, which can improve the speed of the connections. TCP connections are only used in exceptional cases.

Conclusions

RDP is one of the most important protocols in the corporate landscape wherever remote access to Windows Server is required. Although it can be enabled in various ways, make sure you configure the connections as securely as possible. Also, change the port if you have this option to improve the server's security. If you do not need RDP, it makes sense to avoid enabling the protocol, which can ultimately create further attack vectors. Be particularly careful if you make RDP available on the Internet. Many attacks on servers target this avenue. It might be preferable to avoid the server being directly accessible by RDP over the Internet; instead, use an RDP gateway or Azure Arc for the connection. If you do, remote management does not rely on the RDP port; instead, RDP is available through the Windows Admin Center on the Azure portal – and is even free of charge. ■

Info

[1] Royal TS: [https://www.royalapps.com/ts/win/features]

[2] Microsoft Remote Desktop Connection Manager: [https://learn.microsoft.com/en-us/sysinternals/downloads/rdcman]

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].

Managing infrastructure as code with Spacelift

Universal Tool

The Spacelift infrastructure-as-code management platform quickly rolls out complete setups to the cloud, minimizing the possibility of failure and problems. By Martin Loschwitz

Administrators often first encounter the phrase “continuous development and continuous integration” (CI/CD) in the context of container-based setups with Kubernetes. Infrastructure as code (IaC) can also be on board as part of the strategy. IaC is exactly what the name suggests: the ability to describe a complete setup with a standardized format. Special software then reads this description and translates it into running resources, such as virtual instances or correctly configured containers.

Numerous corresponding tools are now on the market, some of which can be combined but that compete with each other in many respects, as well. Spacelift [1] looks to help administrators by offering nothing less than the glue for perfect integration of various tools in the form of standardized APIs and the ability to deploy initial production resources within minutes. Unlike other tools, the claim is that administrators don’t need to spend weeks reading books and manuals to even test the tool.

Support is available for classics such as Pulumi and Terraform, as well as its fork OpenTofu. The developers have also put some thought into the cloud connection, which is, to all extents and purposes, the software’s preferred deployment target. Spacelift can read its descriptions from Git, Azure DevOps, Bitbucket, and others. All kinds of major and minor helpers supplement the construct, supporting, for example, the enforcement of specific security rules, sophisticated monitoring, and monitoring of rolled-out resources. That sounds wonderful, but does it work in the real world? How complex is getting started with Spacelift, and how well does the published integration scope work?

Only for Hyperscalers

To get to the bottom of Spacelift, you first need a description of what Spacelift seeks to be and which features it doesn’t even want to take on. After a quick look at the tool’s website, you initially get the impression that it can also be used in on-premises installations and with bare metal. However, appearances are deceptive. Although the common definitions of IaC now also include on-premises hardware and bare-metal setups, and you can find tools capable of automatically handling physical installations with defined parameters (e.g., tools such as Canonical’s MaaS (metal as a service) or Foreman to handle lifecycle management for physical servers), Spacelift has stated that it does not want to go down these paths.

Native integration is only offered for the large hyperscalers (i.e., AWS, Azure, and GCP). If you don’t want to roll out your setups on one of these platforms, you can forget Spacelift. Supported deployment tools such as Pulumi or Ansible do offer the option of rolling out deployments on other platforms, such as OpenStack. In Spacelift, however, OpenStack will cause additional administrative overhead in many cases. Moreover, vendor support for this kind of setup is unlikely and is particularly unfavorable from a European perspective, because not every company in Europe is allowed to or wants to store its data with hyperscalers – the keyword here being digital sovereignty.

However, if you have no reservations about AWS, Azure, or GCP, you can include Spacelift in your short list of management tools for your virtual environment. In this case, the fact that Spacelift is usually a hosted service that runs in the cloud itself is irrelevant. The code and all company-specific configurations required for Spacelift also end up there. Spacelift does offer the option of renting your own instance of the service for larger setups. Again, this runs in AWS and is essentially a personalized version of the generic Spacelift offering that everyone else gets. Spacelift gives you virtually no alternative to hyperscalers and none at all to AWS.

Terminology

If Spacelift presents nothing fundamentally wrong from your point of view, the next step involves checking out the solution's terminology. You will find a whole host of Spacelift-specific terms, although you do not need to know them all (which would contradict the provider's mantra that Spacelift is geared for production use within a few minutes). A few of the terms from the CI/CD context will very likely already be familiar.

Two familiar items will be Terraform and OpenTofu, two deployment tools that abstract the CI/CD services of the major providers (e.g., AWS CloudFormation) and make them controllable with a standardized declaration language of their own. Terraform is undoubtedly the better known solution, and OpenTofu is strictly speaking also Terraform. This fork of the parent project took place shortly after a change in licensing by HashiCorp.

You will also have heard of terms such as Open Policy Agent (OPA), a standard for defining configuration parameters on systems as a kind of framework that allows more or less generic rules, and Git.

If you are familiar with typical CI/CD terms, the rest of the terms used by Spacelift should become clear quite quickly. The term "stack," for example, is central and is probably familiar to those having some experience with container deployments in cloud environments. In Spacelift, a stack refers to the complete set of resources and to the configuration and metadata of a rolled-out setup.

The precise purpose of Spacelift is to roll out complete environments, including the software running in them, to one of the hyperscalers in the shortest possible time. Spacelift identifies several components that add up to a stack. The stack can then be managed and maintained as an independent unit by Spacelift. Among the various ways of creating stacks in Spacelift, the developers recommend the built-in Terraform provider to convert configuration settings previously stored in a Git directory into a setup for one of the hyperscalers.

The Spacelift developers explicitly point out that this procedure requires you to write the corresponding configuration and all code required for execution in Terraform syntax. However, because Spacelift also supports other IaC tools such as Ansible, stacks can be created in other ways, such as a blueprint, which generates a stack in the same way that Spacelift's Terraform integration would but also contains instructions on the integration to be used for the supported IaC components.

In addition to Terraform and OpenTofu, components also include Ansible, Terragrunt, Kubernetes, and Pulumi (Figure 1). One thing is quite striking: Puppet and Chef, the first-generation automation tools, are missing from the list. The same applies to Salt, which is now also quite widespread, so anyone who primarily relies on these tools has some migration work ahead of them if they want to use Spacelift.

IaC and Version Management

As an IaC solution, Spacelift needs complete integration into standard CI/CD workflows; the current Spacelift development tool scenario makes things easy. Apart from Git, practically no other relevant solutions are out there in the field, and Spacelift integrates perfectly with both GitHub and local GitLab instances (Figure 2). The service also supports Bitbucket, but the interest of global users in this aspect of the service is likely to be limited.

From the perspective of the average user, it seems far more important that Spacelift is capable of implementing CI/CD pipelines itself and of being part of such pipelines. Spacelift's internal working methods are also fully in line with the principles of CI/CD. For example, Spacelift regards a fully rolled out stack as a CI/CD pipeline that can have several runs or calls – an excellent illustration of how Spacelift works under the hood.

In the first step and immediately after gaining access to Spacelift, the administrator creates a (still empty) stack in Spacelift. The next step is to link a Git directory containing all the code required to execute all the stack's necessary resources. Note that

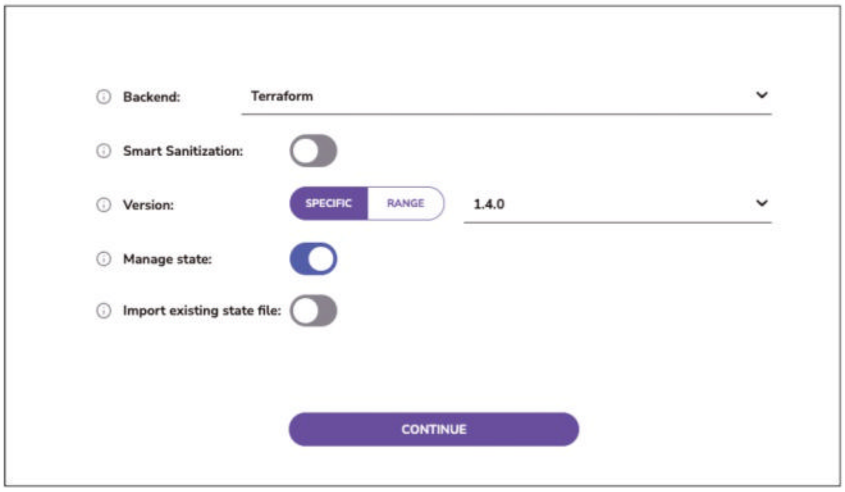
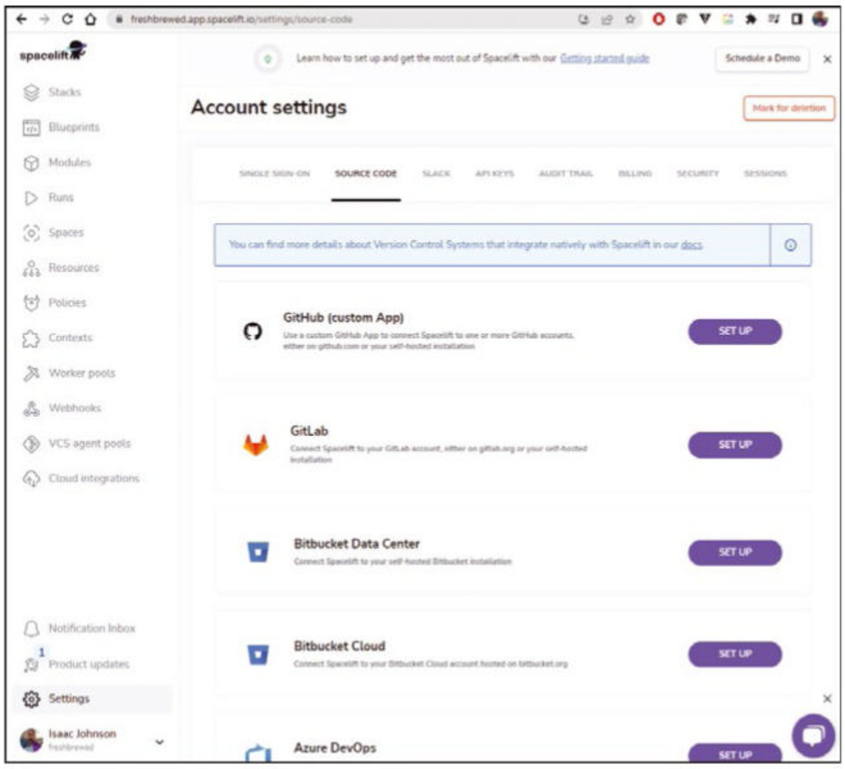

Backend: Terraform
Smart Sanitization:
Version: SPECIFIC RANGE 1.4.0
Manage state:
Import existing state file:

Figure 1: In addition to its own run engine for Terraflow, Spacelift supports various other tools for infrastructure as a service (IaaS), including Ansible and Pulumi. The popular Puppet and Chef tools are missing. © Spacelift



Learn how to set up and get the most out of Spacelift with our Getting started guide

Account settings

You can find more details about Version Control Systems that integrate natively with Spacelift in our docs

GitHub (custom App)
Use a custom GitHub App to connect Spacelift to one or more GitHub accounts, either on github.com or your self-hosted installation

GitLab
Connect Spacelift to your GitLab account, either on gitlab.org or your self-hosted installation

Bitbucket Data Center
Connect Spacelift to your self-hosted Bitbucket installation

Bitbucket Cloud
Connect Spacelift to your Bitbucket Cloud account hosted on bitbucket.org

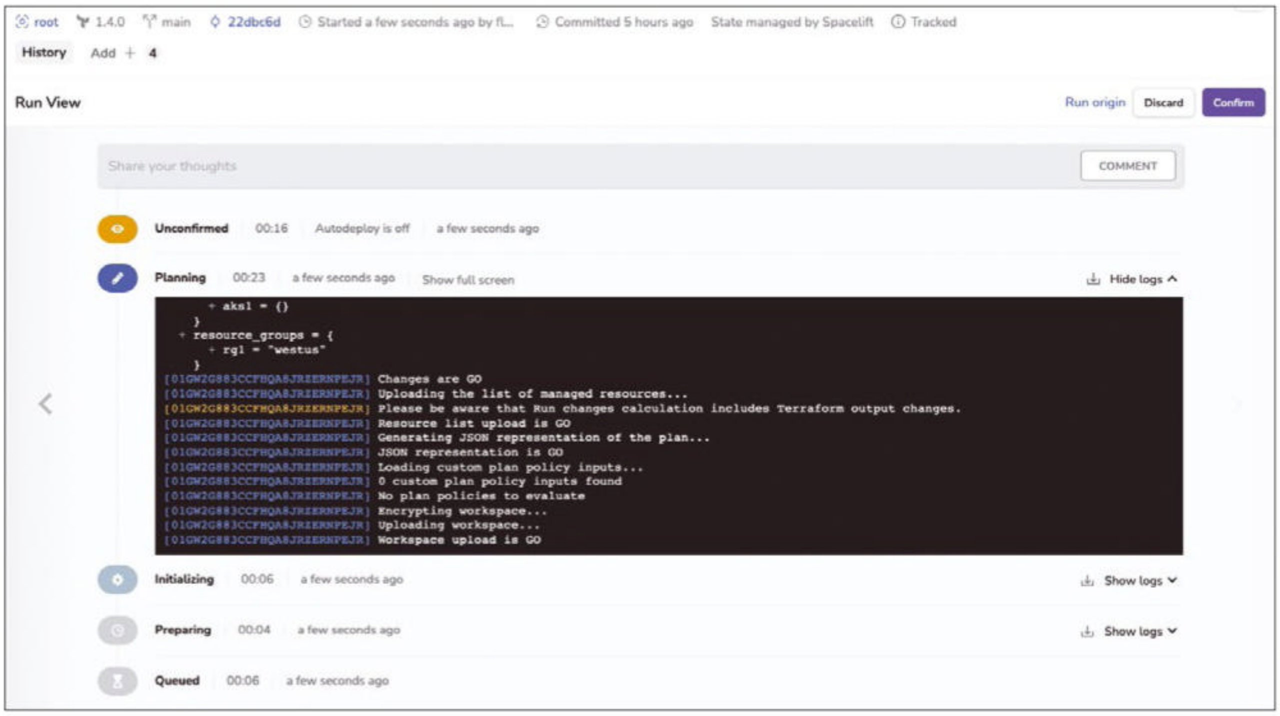
Azure DevOps

Figure 2: Spacelift promises an all-inclusive IaC experience and integrates the version control systems of various providers. © Isaac Johnson

Spacelift itself cannot create resource definitions for Kubernetes, Ansible, or AWS CloudFormation. Instead, you need to create the required definitions in the form of source code in one of the supported configuration formats. Once this step is done, however, Spacelift takes over the helm by retrieving and parsing the configuration data from the Git directory, then uses its internal functions to run tests defined by the administrator. These tests can relate to, say, the target platform or the source code. The situation is similar when working with a blueprint, where in addition to Terraform, the other IaC tools can be used, as well. Spacelift then creates the stack and evaluates a list of the tasks you define stored therein.

Once the stack has been configured, you need to create what is known as a run. Depending on the configuration, it can be a development run or a proposed run (Figure 3). If so desired, a proposed run automatically cleans up its resources when done, to avoid expensive surprises caused by test setups left behind in AWS. Any form of the run can also be repeated. If something goes wrong on the first attempt, you need to change the configuration in Git and then restart the run. Spacelift automatically retrieves the latest configuration and tries again.

Proposed runs designed for deployment in production simulate the changes they would make and indicate whether or not the run would be successful. Like the development run, the proposed run creates



```
      + aks1 = {}
    }
  + resource_groups = {
      + rg1 = "westus"
    }
[01GW2G883CCFHQA8JRZERNPEJR] Changes are GO
[01GW2G883CCFHQA8JRZERNPEJR] Uploading the list of managed resources...
[01GW2G883CCFHQA8JRZERNPEJR] Please be aware that Run changes calculation includes Terraform output changes.
[01GW2G883CCFHQA8JRZERNPEJR] Resource list upload is GO
[01GW2G883CCFHQA8JRZERNPEJR] Generating JSON representation of the plan...
[01GW2G883CCFHQA8JRZERNPEJR] JSON representation is GO
[01GW2G883CCFHQA8JRZERNPEJR] Loading custom plan policy inputs...
[01GW2G883CCFHQA8JRZERNPEJR] 0 custom plan policy inputs found
[01GW2G883CCFHQA8JRZERNPEJR] No plan policies to evaluate
[01GW2G883CCFHQA8JRZERNPEJR] Encrypting workspace...
[01GW2G883CCFHQA8JRZERNPEJR] Uploading workspace...
[01GW2G883CCFHQA8JRZERNPEJR] Workspace upload is GO
```

Figure 3: In Spacelift, a run refers to all the steps required to implement a complete deployment, starting with the plain-vanilla source code. There are different types of runs for development and production. © Spacelift

the required components but does not access production resources. It launches the configured services and resources in a fresh environment on the target platform with the respective IaC tool.

In the usual way for CI/CD systems, Spacelift tracks whether all the desired resources were created. If not (e.g., because the configuration is not perfect), you need to change the configuration in Git and relaunch the development or proposed run.

If everything works as desired, the roll-out takes the form of a tracked run (i.e., a monitored instance) in your stack's production environment. Spacelift can either start a new run or update an existing one with a new configuration on the fly from a new tracked run. Spacelift is idempotent.

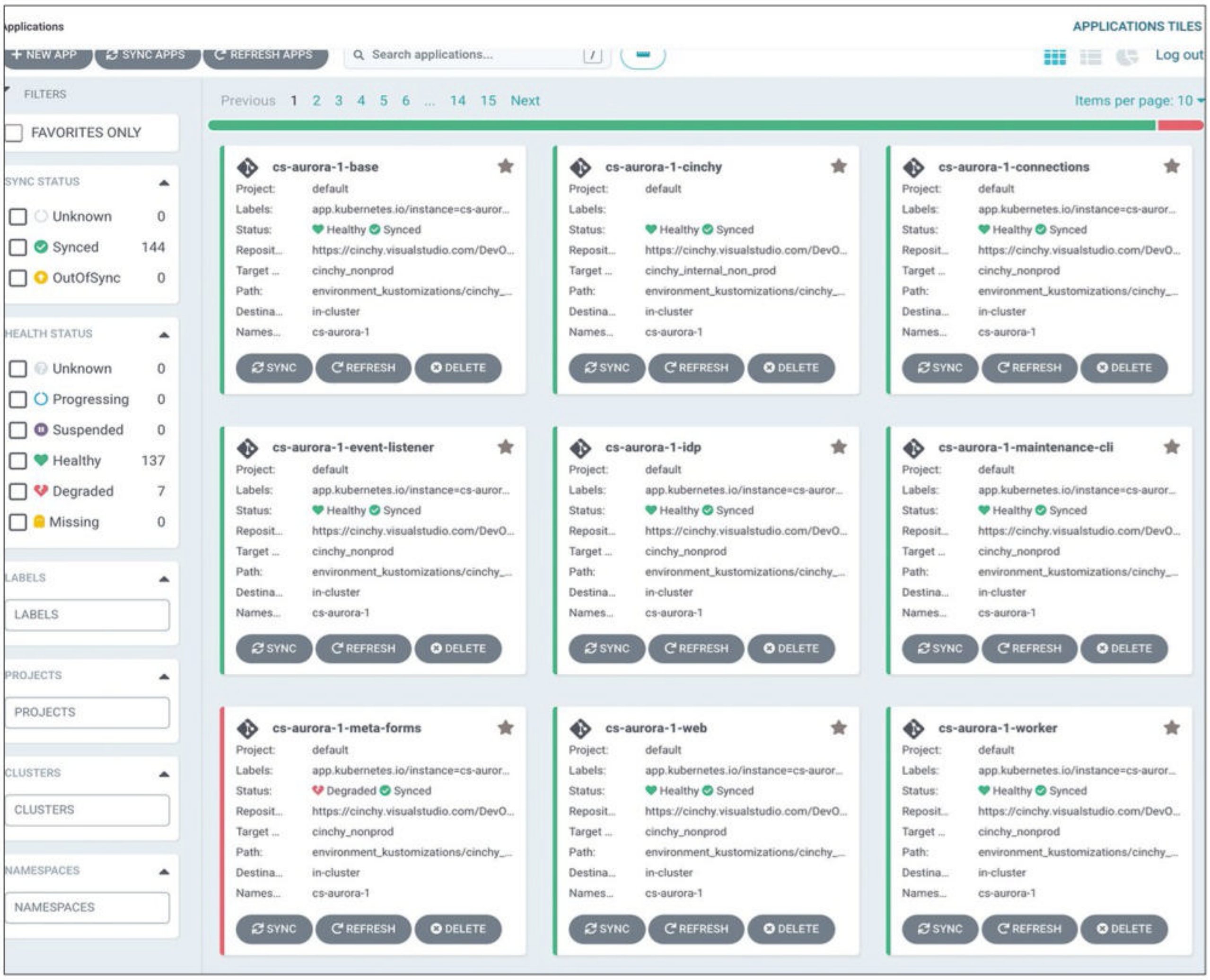
Even if all the resources configured by the administrator are available on the target platform, Spacelift's work is far from done. The tool's feature set includes all kinds of add-ons and extensions, not only to implement the deployment but to ensure security and monitoring. Spacelift can therefore roll out many components from the open source world (e.g., Prometheus) in addition to your defined setup. Spacelift largely works out the configuration for this itself on the basis of the known configuration, which means that complete monitoring can be set up very quickly for a distributed application in Kubernetes.

On top of this, Spacelift itself has a GraphQL interface, meaning that metrics can be exported to an existing Grafana installation in next to no time.

Important Additional Functions

In general, it is worth taking a closer look at the additional functions and convenience add-ons in Spacelift. After all – and the developers behind Spacelift know this – it's not overly complicated to implement a CI/CD solution for IaC that rolls out applications and monitors their lifecycles.

IT hipsters rely almost exclusively on Kubernetes for this task anyway with one of the countless solutions on the market, including Argo CD (Figure 4), which essentially pursues very similar goals to Spacelift and even works in a similar way under the hood. However, you need a running Kubernetes setup for Argo CD, which is offered along with superior alternatives by hyperscalers in most markets. Argo



cs-aurora-1-base
Project: default
Labels: app.kubernetes.io/instance=cs-auror...
Status: Healthy Synced
Reposit... https://cinchy.visualstudio.com/DevO...
Target ... cinchy_nonprod
Path: environment_kustomizations/cinchy_...
Destina... in-cluster
Names... cs-aurora-1

cs-aurora-1-cinchy
Project: default
Labels:
Status: Healthy Synced
Reposit... https://cinchy.visualstudio.com/DevO...
Target ... cinchy_internal_non_prod
Path: environment_kustomizations/cinchy_...
Destina... in-cluster
Names... cs-aurora-1

cs-aurora-1-connections
Project: default
Labels: app.kubernetes.io/instance=cs-auror...
Status: Healthy Synced
Reposit... https://cinchy.visualstudio.com/DevO...
Target ... cinchy_nonprod
Path: environment_kustomizations/cinchy_...
Destina... in-cluster
Names... cs-aurora-1

cs-aurora-1-event-listener
Project: default
Labels: app.kubernetes.io/instance=cs-auror...
Status: Healthy Synced
Reposit... https://cinchy.visualstudio.com/DevO...
Target ... cinchy_nonprod
Path: environment_kustomizations/cinchy_...
Destina... in-cluster
Names... cs-aurora-1

cs-aurora-1-idp
Project: default
Labels: app.kubernetes.io/instance=cs-auror...
Status: Healthy Synced
Reposit... https://cinchy.visualstudio.com/DevO...
Target ... cinchy_nonprod
Path: environment_kustomizations/cinchy_...
Destina... in-cluster
Names... cs-aurora-1

cs-aurora-1-maintenance-cli
Project: default
Labels: app.kubernetes.io/instance=cs-auror...
Status: Healthy Synced
Reposit... https://cinchy.visualstudio.com/DevO...
Target ... cinchy_nonprod
Path: environment_kustomizations/cinchy_...
Destina... in-cluster
Names... cs-aurora-1

cs-aurora-1-meta-forms
Project: default
Labels: app.kubernetes.io/instance=cs-auror...
Status: Degraded Synced
Reposit... https://cinchy.visualstudio.com/DevO...
Target ... cinchy_nonprod
Path: environment_kustomizations/cinchy_...
Destina... in-cluster
Names... cs-aurora-1

cs-aurora-1-web
Project: default
Labels: app.kubernetes.io/instance=cs-auror...
Status: Healthy Synced
Reposit... https://cinchy.visualstudio.com/DevO...
Target ... cinchy_nonprod
Path: environment_kustomizations/cinchy_...
Destina... in-cluster
Names... cs-aurora-1

cs-aurora-1-worker
Project: default
Labels: app.kubernetes.io/instance=cs-auror...
Status: Healthy Synced
Reposit... https://cinchy.visualstudio.com/DevO...
Target ... cinchy_nonprod
Path: environment_kustomizations/cinchy_...
Destina... in-cluster
Names... cs-aurora-1

Figure 4: Existing projects on the market, such as Argo CD, promise similar features to Spacelift but require deployment by Kubernetes. Spacelift can also roll out "real" IaaS with the large hyperscalers and offers more flexibility. © Cinchy Project

CD can also be used to create an existing setup from scratch in a relatively short time.

The bells and whistles, then, are an integral part of the Spacelift product, and the Spacelift developers are doing a great deal to set themselves apart from the field with these additional functions, starting with compliance. Larger companies with several development teams, in particular, need binding specifications to enforce their internal standards when rolling out applications in the cloud. In addition to system configuration, these specifications also include monitoring mechanisms. For example, for user authentication standards to take effect, you first need to implement them correctly in your stack or blueprint, although it does not automatically mean they will work. Several features that Spacelift provides in terms of security and compliance come into play.

The solution is primarily concerned with compliance within your own field of responsibility. Spacelift offers an API for this purpose, which you can access from either the graphical user interface provided for all platforms or a command-line client for Linux and macOS. In many companies, logging staff actions is a prerequisite so that changes can be tracked retrospectively and any errors in the configuration and workflow can be identified and rectified.

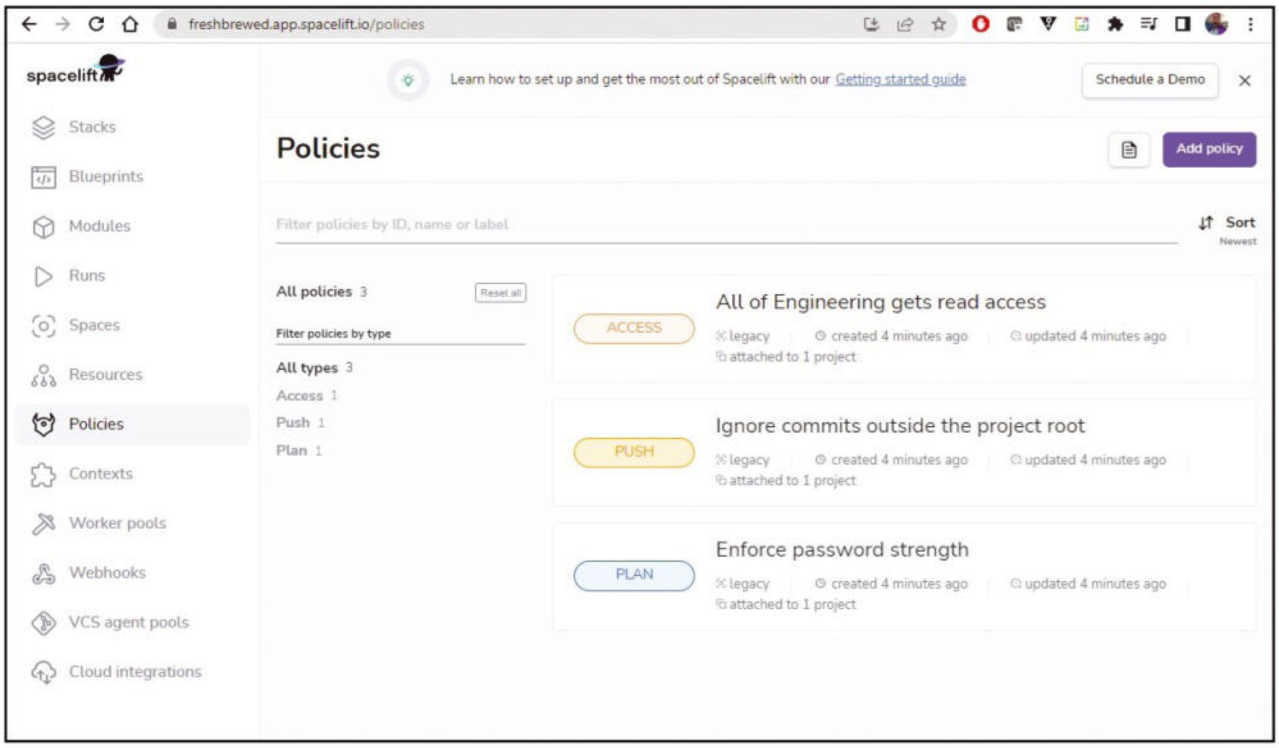
Linking to existing user directories is not a problem for Spacelift. Security assertion markup language (SAML) and OAuth are available as authentication mechanisms and therefore offer connection to the identity and access management services of the major providers, to a self-operated Keycloak with LDAP in the background, or to a local Active Directory. However, Spacelift also offers the logging function for audit trails. Depending on the configuration, the service then logs everything done by individual users, including the complete payload of an API call. To ensure that Spacelift itself does not run out of space, the tool can also use various standard protocols such as Amazon Simple Storage Service to save the audit logfiles. Spacelift also monitors the upload, preventing audit logs from disappearing down the drain because of an unnoticed misconfiguration.

So that the boundary between the platform and the workload does not become an insurmountable hurdle for Spacelift, the service can also automatically roll out the OPA as part of a deployment (Figure 5). If these are part of a stack's configuration, Spacelift enforces them completely automatically at the administrator's request.

Communication and Monitoring

Spacelift is also extremely communicative. To meet the modern requirements of ChatOps collaboration models, the tool offers native interfaces to Microsoft Teams and Slack. For example, certain actions in Spacelift can be configured to trigger automatic messages on previously configured Teams or Slack channels.

Spacelift also enables extensive remote control of external services. It can use webhooks to send commands to external Git directories when certain events occur. Additionally, Spacelift exposes an interface for you to execute webhooks, ensuring that a commit in the remote Git directory



Learn how to set up and get the most out of Spacelift with our Getting started guide

Policies

Filter policies by ID, name or label

All policies 3

Filter policies by type

All types 3
Access 1
Push 1
Plan 1

ACCESS — All of Engineering gets read access — legacy — created 4 minutes ago — updated 4 minutes ago — attached to 1 project

PUSH — Ignore commits outside the project root — legacy — created 4 minutes ago — updated 4 minutes ago — attached to 1 project

PLAN — Enforce password strength — legacy — created 4 minutes ago — updated 4 minutes ago — attached to 1 project

Figure 5: Policies are a special feature of Spacelift: Compliance and security rules can be enforced within rolled-out Spacelift stacks, to the delight of security officers. © Spacelift

automatically triggers an adjustment of the stack in Spacelift, including any security precautions. You can build entire command sequences at this point: A Git commit then initially leads to a proposed run taking place. A run completing without error in turn triggers an update of the tracked run.

In combination, these features are much more powerful than anything that, say, Argo CD can achieve, and this is all the more true because Spacelift is quite open in terms of the internal communication of its individual services. Stacks, for example, can now be set in relation to each other with the use of dependencies. In this way, something will only happen in one stack once another task has successfully completed in another stack, although you need to be aware that this kind of setup will significantly increase complexity in Spacelift.

Spacelift aims to mitigate errors caused by complexity by offering a number of monitoring options. Datadog and Prometheus are part of the standard repertoire, including the respective metrics collectors, but you need at least to define the setup in the stack or in blueprint configurations. Logically, if an application has a native interface to Datadog, the interface also needs to be enabled in the configuration.

By the way, Spacelift does not view the tools as peers. Prometheus is primarily intended for monitoring Spacelift itself, whereas Datadog can also monitor services running in stacks within Spacelift.

Pricing

Small teams that do not need many of the features described and do not need too many workloads can use the free version of Spacelift, although it only supports two users and one project. That said, most of the features are already integrated. If you need the described dependencies on stacks or deeper integration with cloud and authentication services, you can opt for the cloud option – which costs $250 per month and includes five users, with each additional user costing $10 – and multitenant capability.

The Enterprise package includes the described audit trail feature and single sign-on with SAML or OpenID Connect; it also supports private Git directories. Unfortunately, I cannot give you a price for this option. If you want to order the Enterprise product including commercial support, you have to call Spacelift beforehand, which is extremely annoying. However, it can be assumed that there will be a significant surcharge for the Enterprise version compared with the cloud option.

Two things annoyed me. Most companies probably need many of the functions reserved for the expensive Enterprise version, and Spacelift is not cheap. The manufacturer's response to this criticism is that its CI/CD and IaC solution can save the average corporation several full-time equivalents (FTEs) over the years, which has to be factored into the price. In most companies, however, procurement is unlikely to wave Spacelift through quickly, because the prices are too stiff.

Conclusions

All told, Spacelift is as a state-of-the-art deployment tool. In addition to the regularly required functions, it also includes special features, such as the audit trail and support for automated monitoring.

Refreshing and not often found today is the fact that Spacelift not only offers standard functions tailored to Kubernetes but also supports other IaC tools such as Ansible and Pulumi, making Spacelift a valid alternative to tools such as Foreman and Jenkins, not to mention DIY tools. Compared with DIY, Spacelift definitely produces presentable results very quickly, provided you do the work on the application side. Of course, you need to do this work for other tools as well.

The negative aspect is that Spacelift cannot be used without non-European cloud services. If you are based in Europe and value digital sovereignty, you are left out in the cold.

If this is not a problem, Spacelift is definitely worth a closer look. ■

Info

[1] Spacelift: [https://spacelift.io]

The Author

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

Go testing frameworks

Going Faster

Explore the Go language performance framework and thrash the cache in style. By Federico Lucifredi

Traditionally the best language to implement command-line interface (CLI) shell utilities in Linux was Perl [1] – primarily because of its large collection of pre-existing modules, a resource unparalleled at the time called CPAN [2]. Things have since changed, with Perl's popularity steadily declining in the eyes of developers and other languages rising to produce comparably large module libraries, as in Python [3] or Node [4]. Recent books have been published about writing shell commands in Rust [5], Python [6], Node.js [7], and even Go [8], and it is into this last language's interesting performance testing facilities I shall delve deeper. Go famously includes a testing framework, but perhaps less well known is its equally impressive built-in performance testing facility.

To start, the directories should be set up as presented in Figure 1, with the project `row.go` detailed in Listing 1.

Thrashing Caches

In 2019, I analyzed in detail what happens when an algorithm ignores access locality, unnecessarily looking up data all over physical memory [9]. Because RAM is several orders of magnitude slower than the CPU caches, this (usually inadvertent) waltzing all over the memory space needlessly reduces performance. Just as I did back then, I compare the performance of Listings 1 and 2, their sole difference being the order of the array lookup in line 11 (switching the order of the `i` and `j` variables) in a fashion known as row-major and column-major.

A quick comparison on a Core i5 processor produces the results in Figure 2: hardly any difference here. Bear in mind, I set up a comparison between 10x10 arrays, 100 elements defined as integers on a 64-bit processor, totaling 8,000 bytes. Even on the dated Core i5 processor used for this benchmark, L2 cache sizes are 256KB/core, easily accommodating the entire array and making the access pattern irrelevant.

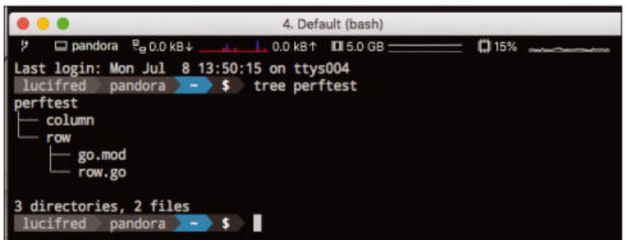
In the next experiment, I'll update the `size` variable to 10,000 as the sole change, to obtain Listings 3 and 4. The array now spans 100 million elements, taking up 800MB of RAM – not at all an unusual size in any kind of numerical computing. This system is equipped with 8GB of RAM, so allocating the array itself is no trouble, but each row is now stretching to 80KB of length. The cache can hold only the merest fraction of the array's total data set.

Listing 1: row.go

```
01 package main
02
03 func main() {
04
05     const size = 10
06
07     var array = [size][size]int {{0},{0},}
08
09     for i := 0; i < size; i++ {
10         for j := 0; j < size; j++ {
11             array[i][j]++
12         }
13     }
14
15 }
```

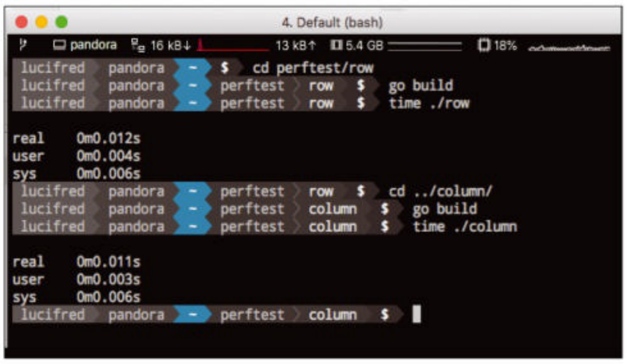
Listing 2: column.go

```
01 package main
02
03 func main() {
04
05     const size = 10
06
07     var array = [size][size]int {{0},{0},}
08
09     for i := 0; i < size; i++ {
10         for j := 0; j < size; j++ {
11             array[j][i]++
12         }
13     }
14
15 }
```



```
Last login: Mon Jul  8 13:50:15 on ttys004
 lucifred  pandora  ~  $  tree perftest
perftest
├── column
└── row
    ├── go.mod
    └── row.go

3 directories, 2 files
 lucifred  pandora  ~  $
```

Figure 1: The project starts off simple, but directory structure matters.



```
 lucifred  pandora  ~  $  cd perftest/row
 lucifred  pandora  ~  perftest  row  $  go build
 lucifred  pandora  ~  perftest  row  $  time ./row

real    0m0.012s
user    0m0.004s
sys     0m0.006s
 lucifred  pandora  ~  perftest  row  $  cd ../column/
 lucifred  pandora  ~  perftest  column  $  go build
 lucifred  pandora  ~  perftest  column  $  time ./column

real    0m0.011s
user    0m0.003s
sys     0m0.006s
 lucifred  pandora  ~  perftest  column  $
```

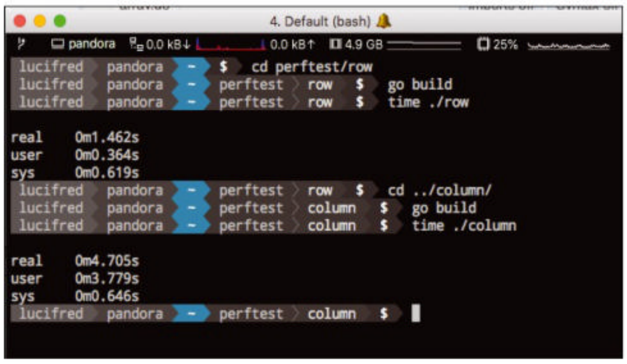
Figure 2: No real difference between the row- and column-major access patterns, but only with a small array.

Running the same test produces the results in Figure 3 – there indeed is a difference. The column-major program is now running more than 10 times slower than the row-major code. The single size change turned the array lookup into a decidedly cache-unfriendly algorithm, because (as it turns out) Go folds arrays in memory in row-major fashion. Looking up the next value in a computation 80KB away from the last is not local enough for any L2 cache, effectively forcing the algorithm to execute at RAM speeds through constant cache misses. In my previous article mentioned earlier, I studied the cache misses with Cachegrind [10], but this time I am looking at automating the testing for accuracy and convenience.

Automation Road

For a better fit in the Go testing framework, I modified the code at the cost of some additional complexity. I could run independent tests on entirely separate programs, but it is easier in this case to compare results coming from the same binary, so I moved the row-major and column-major implementations, each into its own function, and introduced a command-line parameter to select between these (and possibly more) algorithms. I want the 800MB variable to remain in the package scope so that it gets compiled into the program's data segment. A wrapper `run()` function is added to encapsulate calls for the testing framework's convenience. This new (but functionally identical) program is found in Listing 5 and is seen in action in Figure 4.

Go is big on naming conventions, and the testing framework is no exception. Filenames ending in `_test` are expected to contain test functions. Function names declared therein starting with `Benchmark` are just that, benchmarks. Listing 6 introduces the testing framework, the loop variable `N` allowing for repeated testing and the `ResetTimer()` annotation a useful reminder to exclude test setup overheads not present in these tests.



```
lucifred  pandora  ~  $  cd perftest/row
lucifred  pandora  ~  perftest  row  $  go build
lucifred  pandora  ~  perftest  row  $  time ./row

real    0m1.462s
user    0m0.364s
sys     0m0.619s
lucifred  pandora  ~  perftest  row  $  cd ../column/
lucifred  pandora  ~  perftest  column  $  go build
lucifred  pandora  ~  perftest  column  $  time ./column

real    0m4.705s
user    0m3.779s
sys     0m0.646s
lucifred  pandora  ~  perftest  column  $
```

Figure 3: With a significantly larger array, the difference becomes evident as cache thrashing dominates.

Listing 3: row.go with Large Array

```
package main

func main() {

    const size = 10000

    var array = [size][size]int {{0},{0},}

    for i := 0; i < size; i++ {
        for j := 0; j < size; j++ {
            array[i][j]++
        }
    }

}
```

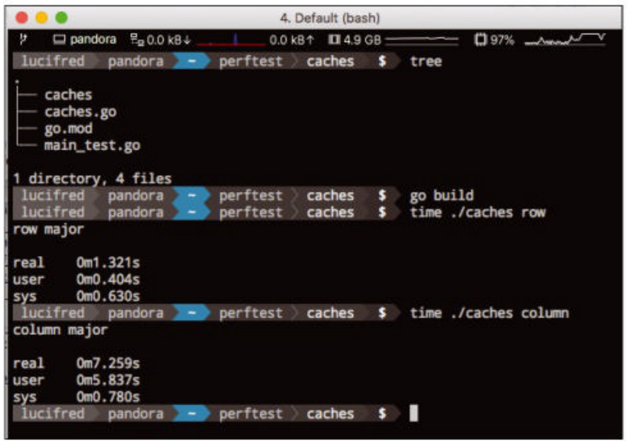
Listing 4: column.go with Large Array

```
package main

func main() {

    const size = 10000

    var array = [size][size]int {{0},{0},}

    for i := 0; i < size; i++ {
        for j := 0; j < size; j++ {
            array[j][i]++
        }
    }

}
```

Figure 5 shows the framework in action. The parameter `-v` initiates verbose mode, and `-bench` limits tests to functions matching the regular expression that follows (here, I selected the `Row` test). The second run executes all tests, with the tool independently deciding to execute the `Row` test three times, a behavior that can be forced with the `-benchtime` parameter.

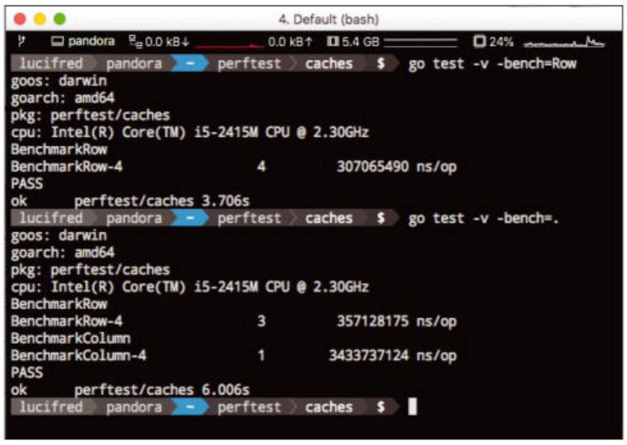
Listing 5: caches.go

```
package main

import (
        "os"
        "fmt"
)

const size = 10000
var array = [size][size]int {{0},{0},}

func main() {
    if len(os.Args) < 2 {
        fmt.Println("expected an algorithm selection")
        os.Exit(1)
    }

   run(os.Args[1])
}

func row() {
    for i := 0; i < size; i++ {
        for j := 0; j < size; j++ {
            array[i][j]++
        }
    }
}

func column() {
    for i := 0; i < size; i++ {
        for j := 0; j < size; j++ {
            array[j][i]++
        }
    }
}

func run(algo string) {
   switch algo {
      case "row":
            fmt.Println("row major")
            row()
      case "column":
            fmt.Println("column major")
            column()
    default:
       fmt.Println("expected 'row' or 'column'")
      os.Exit(1)
     }
 }
```



```
4. Default (bash)
pandora  0.0 kB↓  0.0 kB↑  4.9 GB  97%
lucifred  pandora  ~  perftest  caches  $  tree
.
├── caches
├── caches.go
├── go.mod
└── main_test.go

1 directory, 4 files
lucifred  pandora  ~  perftest  caches  $  go build
lucifred  pandora  ~  perftest  caches  $  time ./caches row
row major

real    0m1.321s
user    0m0.404s
sys     0m0.630s
lucifred  pandora  ~  perftest  caches  $  time ./caches column
column major

real    0m7.259s
user    0m5.837s
sys     0m0.780s
lucifred  pandora  ~  perftest  caches  $
```

Figure 4: The single refactored program takes a parameter to select the access pattern.



```
4. Default (bash)
pandora  0.0 kB↓  0.0 kB↑  5.4 GB  24%
lucifred  pandora  ~  perftest  caches  $  go test -v -bench=Row
goos: darwin
goarch: amd64
pkg: perftest/caches
cpu: Intel(R) Core(TM) i5-2415M CPU @ 2.30GHz
BenchmarkRow
BenchmarkRow-4                 4         307065490 ns/op
PASS
ok      perftest/caches 3.706s
lucifred  pandora  ~  perftest  caches  $  go test -v -bench=.
goos: darwin
goarch: amd64
pkg: perftest/caches
cpu: Intel(R) Core(TM) i5-2415M CPU @ 2.30GHz
BenchmarkRow
BenchmarkRow-4                 3         357128175 ns/op
BenchmarkColumn
BenchmarkColumn-4              1        3433737124 ns/op
PASS
ok      perftest/caches 6.006s
lucifred  pandora  ~  perftest  caches  $
```

Figure 5: The testing framework independently chose to repeat the `BenchmarkRow` test to achieve statistical significance in the results.

Repeated tests are a best practice, particularly for CPU-bound tests. Saving the results of past benchmarks is also advisable and can be easily done with `tee` [11]:

```
go test -bench=Row ↩
        -benchmem ↩
        -benchtime=10x ↩
         | tee testresults/bench00.txt
```

Also of interest is `-benchmem`, which is used to measure total memory allocation of the tested function; yet, it is not useful in this particular case because I am not allocating any memory in the function itself.

Listing 6: Testing Framework

```
package main

import "testing"

func BenchmarkRow(b *testing.B) {
    //b.ResetTimer()
    for i := 0; i < b.N; i++ {
        run("row")
        }
}

func BenchmarkColumn(b *testing.B) {
    //b.ResetTimer()
    for i := 0; i < b.N; i++ {
        run("column")
        }
}
```

Laziness Is a Virtue

During development, optimizing a program requires repeating tests hundreds, if not thousands, of times. Go's testing frameworks assist not just with unit tests, but also in the all-too-common scenarios in performance tuning where different approaches have to be repeatedly compared and contrasted. As with the `row.go` and `column.go` examples, having many alternative implementations makes it really conveniently manageable to compare them again and again, as may be necessary to determine the final implementation strategy. ■

Info

[1] "Why Perl Is Still Relevant in 2022" by Girish Venkatachalam, July 2022: [https://stackoverflow.blog/2022/07/06/why-perl-is-still-relevant-in-2022/]

[2] CPAN: [https://www.cpan.org]

[3] PyPi: [https://pypi.org]

[4] npm registry: [https://www.npmjs.com]

[5] Youens-Clark, Ken. *Command-Line Rust*. O'Reilly, 2022

[6] Gift, Noah, and Alfredo Deza. *Python Command Line Tools*. Pragmatic AI Labs, 2020

[7] Kowalski, Robert. *The CLI Book*. Apress, 2017

[8] Gerardi, Ricardo. *Powerful Command-Line Applications in Go*. Pragmatic Bookshelf, 2021

[9] "Thrashing the Data Cache for Fun and Profit" by Federico Lucifredi, *ADMIN*, issue 54, 2019, pg. 95: [https://www.admin-magazine.com/Archive/2019/54/Thrashing-the-data-cache-for-fun-and-profit/]

[10] Cachegrind manual: [http://valgrind.org/docs/manual/cg-manual.html]

[11] tee (1) man page: [https://manpages.ubuntu.com/manpages/noble/en/man1/tee.1.html]

The Author

Federico Lucifredi (@0xf2) is the Product Management Director for Ceph Storage at IBM and Red Hat, formerly the Ubuntu Server Product Manager at Canonical, and the Linux "Systems Management Czar" at SUSE. He enjoys arcane hardware issues and shell-scripting mysteries, and takes his McFlurry shaken, not stirred. You can read more from him in the new O'Reilly title *AWS System Administration*. ■

ADMIN Network & Security

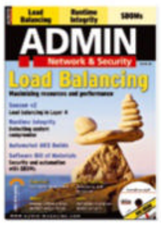
NEWSSTAND

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.

#81 – May/June 2024

Load Balancing

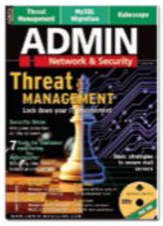


Load balancing on heavily frequented networks improves performance, availability, security, scalability, and the ability to handle peak loads.

On the DVD: SystemRescue 11.01

#80 – March/April 2024

Threat Management

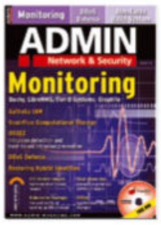


Digital infrastructures are vulnerable to all kinds of attacks. You need strategies and tools to detect and defend.

On the DVD: openSUSE Leap 15.5

#79 – January/February 2024

Monitoring

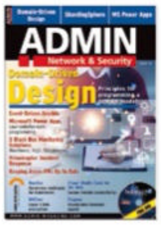


This issue takes a deep dive into monitoring solutions for your IT infrastructure, including Dashy, LibreNMS, Tier 0 systems, and Graphite.

On the DVD: FreeBSD 14.0

#78 – November/December 2023

Domain-Driven Design

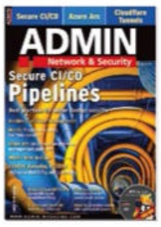


Business experts and developers collaborate to define domain models and business patterns that guide software development.

On the DVD: Fedora Server 39

#77 – September/October 2023

Secure CI/CD Pipelines

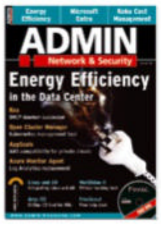


DevSecOps blends security into every step of the software development cycle.

On the DVD: IPFire 2.27

#76 – July/August 2023

Energy Efficiency



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

On the DVD: Finnix 125 (Live boot

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

Contact Info

Editor in Chief
Joe Casad, jcasad@linuxnewmedia.com

Managing Editors
Rita L Sooby, rsooby@linuxnewmedia.com
Lori White, lwhite@linuxnewmedia.com

Senior Editor
Ken Hess

Localization & Translation
Ian Travis

News Editor
Amber Ankerholz

Copy Editors
Amy Pettle, Aubrey Vaughn

Layout
Dena Friesen, Lori White

Cover Design
Lori White, Illustration based on graphics by adam121, 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

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. July/August 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.

REGISTER ON OR BEFORE OCTOBER 15 AND SAVE!

Create the future of HPC with thousands of researchers, engineers, technologists, educators, and students during a week of exhilarating programs and events.

SC24.SUPERCOMPUTING.ORG

SC24

NOV 17–22

The INTERNATIONAL CONFERENCE for HIGH PERFORMANCE COMPUTING, NETWORKING, STORAGE, & ANALYSIS

ATLANTA, GA

SPONSORED BY IEEE COMPUTER SOCIETY TCHPC acm Association for Computing Machinery sighpc

HETZNER

HIGH POWERED FOR SMALL BUDGETS

EFFICIENCY CHAMPION

AX42 WITH AMD RYZEN™ PRO 8700GE

with 8 cores and 64 GB DDR5 ECC RAM

DEDICATED SERVER AX42

- ✓ AMD Ryzen™ 7 8700GE
  Octa-Core, Phoenix 1 (Zen4)
- ✓ 2 x 32 GB DDR5 ECC UDIMM
- ✓ 2 x 512 GB NVMe SSD
- ✓ Unlimited traffic
- ✓ Location Germany & Finland
- ✓ No minimum contract
- ✓ Setup fee € 39.00

monthly from **€46.00**
$ 49.41

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.

The ideal ratio of price, performance and power consumption.

Get your server here:
htznr.li/admin/AX42

OR SCAN THE CODE

www.hetzner.com