ADMIN Network & Security

Domain-Driven Design

ShardingSphere

MS Power Apps

ADMIN Network & Security

ISSUE 78

Domain-Driven Design

Principles for programming a domain model

Event-Driven Ansible

Microsoft Power Apps
Low-code/no-code programming

3 Black Box Monitoring Solutions
Monitoror, Vigil, Statping-ng

Velociraptor Incident Response

Keeping Azure VMs Up to Date

LINUX NEW MEDIA The Pulse of Open Source

Knative
Serverless workloads for Kubernetes

MACsec
Layer 2 link encryption

Visual Studio Code for the Web
Secure remote connectivity

Ripgrep
Accelerated terminal search

WWW.ADMIN-MAGAZINE.COM

DEVELOPERWEEK™ 2024 2024 2024

Feb 21-23 SF Bay Area **Feb 27-29** Live Online

8,000+ Developers at the World's Largest Developer Expo Conference Series

Use code **MP408** for **$100** off all **DeveloperWeek 2024** tickets

Register at **developerweek.com**

250+ Speakers across 15+ technical tracks including:

JavaScript · AI · Containers & Kubernetes · APIs
Microservices · Python · Databases · Dev Tools Dev
Management/Leadership · Blockchain
Product Teams · ML · Serverless · DevOps

OUR NETWORK OF EVENTS

5 events. **4,000+** companies. **15,000+** annual attendees.

DEVELOPERWEEK
February 21–29, 2024

DEVELOPERWEEK EUROPE
April, 2024

DEVELOPERWEEK LATIN AMERICA
June, 2024

DEVELOPERWEEK MANAGEMENT
May, 2024

CLOUDX™
August, 2024

DEVELOPERWEEK ENTERPRISE
November, 2024

Digital Forensics

Consider a new direction in system administration.

In the Welcome column, I write about jobs, careers, trends, and sometimes random but relevant topics. For this issue, I'm discussing a new direction in system administration that you might know as computer forensics, cyberforensics, or digital forensics.

Digital forensics is the discovery, recovery, investigation, and examination of data found in computer systems. Computer systems is a broad category that includes databases, network devices, and mobile devices. It may also include other devices (e.g., supervisory control and data acquisition (SCADA) instruments) that store, process, or use data. Although digital forensics isn't new, it can be a new direction for those who have traditionally held system administration jobs.

You might wonder why I'm discussing a security topic for a column focusing on system administration. I've mentioned before that security is everyone's job, and it's certainly true for system administrators, and digital forensics is an extension of that role. The reality of the system administrator's role is that our job description is "Other duties as assigned" and little else. We do everything, and security is often the least offensive task that we have the pleasure to perform.

To illustrate how the roles overlap, assume that you suspect a system has been compromised. You begin collecting and comparing logs to find out when the breach occurred. Next, you search for compromised or new accounts. You search for open ports and check network data to see if information is being exfiltrated. You isolate systems and run various vulnerability and rootkit scans. You might even enlist the assistance of other digital forensic specialists to help locate backdoors, trojans, scripts, and changed files. You probably changed all your root and administrator passwords. Performing these and similar tasks is digital forensics.

Some sys admins have a special talent for digital forensics, while others will have no interest at all. I was shocked when one of my former colleagues told me to "have fun" doing my investigative work on a suspected breach and let him know when I've "had enough." To his surprise, I solved the issue. I uncovered an internal breach and traced it to the offending person.

In this instance, a set of maintenance scripts used a non-secure protocol to update code from a development system to multiple other staging and production systems. He couldn't be bothered to tunnel or otherwise secure passwords and data traversing the network. It looked like an outside attack from a compromised system because it traversed a firewall, a bastion host, and the DMZ. My colleague had to explain himself to our manager and the security team. He also had to provide extensive documentation and a plan to secure the data and its transfer.

Not all suspected breaches are quite this easy to unravel and resolve. Fortunately, the incident didn't require public disclosure because it only included data and information for our intranet, and no client data or information was involved. The problem required mitigation because the process was a prototype for client production data and information. It would have been much worse in six months when the process was moved to production.

This is what digital forensics is all about. If performing those tasks interests you, several online classes and university options can take your interests to the next level. All system administrators should be required to have digital forensics training. Even if you have not performed any forensics-related tasks, the training will help you protect your systems and assist investigators during the reconnaissance and recovery phases of an incident. If you love to solve puzzles, have an aptitude for detailed work, and enjoy devising strategies against an opponent, digital forensics might be what you're looking for in moving your sys admin career forward.

The job of system administration is fun, but expanding your horizons and exploring something new and different doesn't hurt. You might find yourself on a new path to a great and rewarding career as a full-time digital forensics professional.

Ken Hess • ADMIN Senior Editor

Lead Image © armmypicca, 123RF Free Images

Features

10 **Introduction to DDD**
Domain-driven design comprises principles for team-driven software development, from the design of entire software landscapes to the design of domain models, patterns, and code.

14 **DDD and Agility**
Central components of agility come into play with domain-driven design, which is oriented on business values, a deep understanding of the domain model, and information exchange by business experts and developers.

18 **Domain-Driven Transformation**
Bring out the hidden business treasure in legacy systems by identifying the parts of the source code that contain valuable business knowledge and refurbishing it in increments while mitigating risk.

Tools

22 **Manage AD with PyAD**
Delegate tedious routine tasks when automating Active Directory configuration and management to Python scripts with the PyAD library.

26 **Apache ShardingSphere**
Extend databases by adding a modular abstraction layer to support horizontal sharding and scalability.

32 **Shared and Tiered Storage**
The storage of job output for clusters requires an understanding of resource managers and a discussion of where data "should" or "could" go.

38 **Microsoft Power Apps**
If the IT staff is having trouble keeping up with the demand for custom applications, end users can pitch in with low-code programming tools.

Containers and Virtualization

42 **Knative**
Transfer serverless workloads to Kubernetes, with all the container components you need to build serverless applications and PaaS and FaaS services.

48 **Dockerize a Legacy App**
When you are ready to convert your legacy application to a containerized environment, Docker offers the tools for a smooth and efficient transition.

Security

56 **MACsec**
Encrypt defined links with high performance and secure Layer 2 protocols between client and switch or between two switches.

62 **Visual Studio Code Server**
You can gain a significant advantage for remote development by connecting to remote machines through secure tunnels, without the need for SSH.

70 **Velociraptor**
A powerful query language monitors and queries your IT infrastructure, combining and extending the functionality of GRR Rapid Response and OSQuery when seeking clues to cyberattacks and indicators of compromise.

Service

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

10, 14, 18 | **Domain-Driven Design**

Principles for programming a domain model

Business experts and developers collaborate to define domain models and business patterns that guide software development.

Highlights

48 Dockerize Your Apps
Docker Compose and other tools in the Docker toolset provide a safe, efficient, and versatile approach for migrating your existing applications to a container environment.

56 MACsec
Media Access Control Security, defined by IEEE 802.1X-2010 in combination with 802.1AE, encrypts in Layer 2 for cryptographic point-to-point security on wired networks (e.g., switches) with virtually no loss of speed.

72 Event-Driven Ansible
Agentless automation with a reactive extension that uses events to launch automations transforms Ansible's push-only architecture.

Management

72 Event-Driven Ansible
A powerful rules engine and a modular, open concept enable automation of target systems without the need for an agent.

78 Monitoring Servers and Services
Keep track of internal and external servers and services with black box monitoring by Monitoror, Vigil, and Statping-ng.

Nuts and Bolts

86 Updating Azure VMs
A number of methods allow you to keep the operating system of an Azure VM up to date, including the new Azure Update Management Center.

92 Ripgrep
The best features of tools like Grep, Ack, and Silver Searcher are combined when it comes to the use of search patterns in a terminal window.

On the DVD

Fedora Server 39
Run server workloads on bare metal or virtual machines. Fedora describes its Server Edition as "a platform for developers and system integrators, providing an implementation of the latest server technology."
Fedora Server features:
- Runs on virtual machines and containers
- Enables a variety of server workloads
- The latest open source technologies are curated by the Fedora Community
- The remote server administration tool is ready to use on first boot
- Deployment of servers is quick, with all the tools you need to spin up your workloads

@adminmagazine
@adminmag
ADMIN magazine
@adminmagazine

News for Admins

Tech News

Red Hat Announces Ansible Lightspeed with IBM watsonx Code Assistant

Red Hat has released Ansible Lightspeed with IBM watsonx Code Assistant — a generative AI service for Ansible operators and developers, aimed at helping organizations accelerate IT automation.

Ansible Lightspeed (**https://www.redhat.com/en/technologies/management/ansible/ansible-lightspeed**) "accepts prompts entered by a user and then interacts with IBM watsonx foundation models to produce code recommendations built on Ansible best practices," Red Hat says.

The service also helps keep codebases updated. It "scans existing content and automatically provides update recommendations that are ready to review, test, and apply, making it easier to maintain quality and consistency across the development life cycle," Red Hat says.

Dell APEX Cloud Platform for Red Hat OpenShift Announced

Dell has announced the availability of the Dell APEX Cloud Platform for Red Hat OpenShift (**https://www.dell.com/en-us/dt/apex/cloud-platforms/red-hat-openshift.htm**), which it describes as "the first fully integrated application delivery platform purpose-built for Red Hat OpenShift."

The APEX Cloud Platform, which was jointly engineered with Red Hat, "combines Dell's automation management software, PowerEdge servers, and software-defined storage with Red Hat's container orchestration platform in a single appliance," according to the announcement (**https://www.dell.com/en-us/blog/dell-red-hat-and-the-future-of-containers/**). It offers:

- Simplified multicloud deployment
- Accelerated application delivery
- Optimized workload placement

The collaboration between Dell and Red Hat also "helps ensure that customers have rapid access to new patches, helping to mitigate security vulnerabilities," Dell says.

NSA Offers Best Practices for OSS in Operational Technology Environments

Implementation and patching of open source software (OSS) in operational technology (OT) (**https://en.wikipedia.org/wiki/Operational_technology**) environments "continues to be a challenge due to safety concerns and the potential disruption of critical systems," according to the NSA.

To promote better understanding and highlight best practices, the NSA, along with CISA and other agencies, has released new guidance (**https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/3552309/nsa-and-us-agencies-issue-best-practices-for-open-source-software-in-operationa/**) for securing these systems.

The fact sheet recommends "supporting OSS development and maintenance, patch management, authorization and authentication policies, and establishing common frameworks." The guidance "also encourages the adoption of 'secure-by-design' and 'secure-by-default' principles to decrease cybersecurity risk in OT environments."

Get the latest IT and HPC news in your inbox

Subscribe free to ADMIN Update and HPC Update

bit.ly/HPC-ADMIN-Update

Lead Image © vlastas, 123RF.com

Systems and Infrastructure | HAM Radio and Linux | DevOps | Cloud Native | Developer | Embedded | Home Automation | IoT | Kernel | Low Level Systems | Observability | Artificial Intelligence | Open Government | Kubernetes | Security |

REGISTER NOW!

SCaLE

MARCH 14-17 • Pasadena, CA

21x

Don't miss North America's largest annual community organized Open Source gathering!

50% OFF Use Code **LM50** for an exclusive discount **socallinuxexpo.org**

SCaLE
SOUTHERN CALIFORNIA LINUX EXPO

Want to **Sponsor**, **Exhibit**, or **Volunteer** at SCaLE?
Get in touch with us: **sponsorship@socallinuxexpo.org**

Civil Infrastructure Platform Adds New Super-Long-Term Linux Kernel

The Civil Infrastructure Platform (CIP) (**https://www.cip-project.org/**) has added the 6.1-based Linux kernel series to its super-long-term stable (SLTS) kernel program, which means the project is committed to maintaining the 6.1-cip kernel for a minimum of 10 years after its initial release.

Separate from the Linux kernel project, which recently announced that long-term support (LTS) for Linux kernels would be reduced (**https://www.fosslife.org/linux-long-term-support-being-cut-back**) from six to two years, the CIP clearly has a different mission. The SLTS program is part of CIP's efforts to establish "an open source base layer of industrial grade Linux to enable the use and implementation of software building blocks for civil infrastructure."

Additionally, the announcement notes, "CIP kernels are maintained like regular long-term-stable (LTS) kernels, and developers of the CIP kernel are also involved in LTS kernel review and testing." Other kernels in the program include 4.4-cip, 4.19-cip, and 5.10-cip.

HTTP/2 Protocol Exploited in Largest DDoS Attack Ever

Google, Cloudflare, and Amazon Web Services have revealed a new zero-day vulnerability (**https://nvd.nist.gov/vuln/detail/CVE-2023-44487**) known as "HTTP/2 Rapid Reset."

Attacks exploiting the vulnerability targeted cloud and Internet infrastructure providers and peaked in August. "These attacks were significantly larger than any previously reported Layer 7 attacks, with the largest attack surpassing 398 million requests per second," Google says (**https://cloud.google.com/blog/products/identity-security/google-cloud-mitigated-largest-ddos-attack-peaking-above-398-million-rps**).

The attack used a novel "Rapid Reset" technique leveraging the stream multiplexing feature of the widely implemented HTTP/2 protocol (**https://http2.github.io/**).

See further analysis at Google Cloud (**https://cloud.google.com/blog/products/identity-security/how-it-works-the-novel-http2-rapid-reset-ddos-attack**).

Docker Announces Three New Products for Secure App Delivery

Docker has announced three products aimed at secure app delivery: Docker Scout GA, Docker Build, and Docker Debug.

According to the announcement, "the products combine the responsiveness and convenience of local development with the on-demand resources, connectedness, and collaboration of the cloud."

Docker Scout is now generally available, while the other products are available in public beta:

- Docker Scout — Provides relevant insights and integration to continuously evaluate container images against defined policies, aligned with software supply chain best practices.
- Docker Build — Speeds up builds by as much as 39 times by taking advantage of large, on-demand cloud-based servers and team-wide build caching.
- Docker Debug — Provides a language-independent, integrated toolbox for debugging local and remote containerized apps.

Learn more at Docker (**https://www.docker.com/blog/announcing-docker-scout-ga/**).

CloudBees Updates Jenkins and Offers New DevSecOps Platform

CloudBees has announced major performance and scalability enhancements to its widely used Jenkins CI/CD software, as well as a new DevSecOps solution based on Tekton.

The updates are part of the CloudBees CI (**https://www.cloudbees.com/capabilities/continuous-integration**) enterprise version of Jenkins and include features such as workspace caching, pipeline explorer, and high-availability mode, which aim "to reduce build times, speed up troubleshooting, enhance controller efficiency, and maximize uptime," the announcement says (**https://www.cloudbees.com/blog/biggest-update-for-jenkins-in-over-a-decade-with-cloudbees-ci**).

The new cloud-native DevSecOps platform (**https://www.cloudbees.com/products/saas-platform**) "uses a GitHub Actions style domain-specific language (DSL) and adds feature flagging, security, compliance, pipeline orchestration, analytics and value stream management (VSM) into a fully-managed single-tenant SaaS, multi-tenant SaaS or on-premise virtual private cloud instance," according to the announcement (**https://www.cloudbees.com/newsroom/cloudbees-announces-new-cloud-native-devsecops-platform**).

Additionally, the platform puts the focus on the emerging role of platform engineering, which "brings together multiple roles such as site reliability engineers (SREs), DevOps engineers, security teams, product managers, and operations teams."

Linkerd 2.14 Released with Improved Multi-Cluster Support

The latest release of the Linkerd (**https://linkerd.io/2.14/overview/**) service mesh for Kubernetes features improved support for multi-cluster deployments on shared flat networks, full gateway API conformance, and much more.

Shared flat network architecture is increasingly common in enterprise environments, according to the announcement (**https://www.cncf.io/blog/2023/09/18/announcing-linkerd-2-14-improved-enterprise-multi-cluster-gateway-api-conformance-and-more/**), and it "allows pods in different clusters to establish TCP connections with each other."

"Importantly, this new multi-cluster support retains a critical aspect to Linkerd's design: independence of clusters as a way of isolating security and failure domains. Each cluster runs its own Linkerd control plane, and the failure of a single cluster cannot take down the service mesh on other clusters," the announcement says.

NIST Releases Draft of Cybersecurity Framework v2.0

The National Institute of Standards and Technology (NIST) has released a draft of the Cybersecurity Framework (CSF) 2.0 (**https://csrc.nist.gov/pubs/cswp/29/the-nist-cybersecurity-framework-20/ipd**). The framework was first released in 2014 to help organizations understand cybersecurity risk.

"The NIST Cybersecurity Framework 2.0 provides guidance to industry, government agencies, and other organizations to reduce cybersecurity risks. It offers a taxonomy of high-level cybersecurity outcomes that can be used by any organization — regardless of its size, sector, or maturity — to better understand, assess, prioritize, and communicate its cybersecurity efforts," according to the framework abstract.

NIST is accepting public comment on the draft framework until Nov. 4, 2023, according to the announcement (**https://www.nist.gov/news-events/news/2023/08/nist-drafts-major-update-its-widely-used-cybersecurity-framework**), but does not plan to release another draft. "A workshop planned for the fall will be announced shortly and will serve as another opportunity for the public to provide feedback and comments on the draft. The developers plan to publish the final version of CSF 2.0 in early 2024."

CISA and MITRE Announce Open Source Caldera for OT

The Cybersecurity and Infrastructure Security Agency (CISA) and MITRE have announced Caldera for OT, a cyberattack emulation platform developed specifically for operational technology (OT) networks.

The project is an extension of MITRE Caldera (**https://caldera.mitre.org/**), which is aimed at reducing the amount of time and resources needed for routine cybersecurity testing.

According to the announcement (**https://medium.com/@mitrecaldera/announcing-mitre-caldera-for-ot-47c6f22a676d**), Caldera for OT builds on that functionality, "offering 29 distinct OT abilities to the hundreds of existing enterprise-focused abilities already included with Caldera." These new plugins "enable practitioners to emulate adversary behavior across both enterprise and industrial networks."

The Caldera for OT plugins are free and open source and can be downloaded from the project's GitHub repository (**https://github.com/mitre/caldera-ot**).

The basics of domain-driven design

Bit by Bit

Domain-driven design addresses many aspects of software development, from the design of entire software landscapes and the relationships between (sub)systems to the design of domain models, patterns, and code.

By Stefan Hofer

When Eric Evans wrote his book on domain-driven design (DDD) in 2003 [1], he primarily had enterprise software in mind, wherein project teams develop software for various departments. Since then, a vibrant community has been established around DDD that continues to expand the discipline. As a result, DDD has become widespread in the development of software products.

Domain Models

To build software that solves domain-specific problems, development teams condense their understanding of the domain's structures and rules into a domain model that exists in the minds of developers in the form of diagrams, conversations, text, and code. In DDD, the domain model forms the core of a software system. Of course, this core needs to be surrounded by technology (interfaces, persistence, communication, etc.) for the software to be useful.

Building models was not the brainchild of Evans. He drew on object-oriented analysis and object-oriented design and added two essential concepts that solve problems in the design of large systems. In tactical design (more on this later), Evans describes a pattern language (i.e., coherent patterns) that can be used to design domain models.

The larger the scope of a domain model, the more its inconsistencies become visible. The reasons lie in the domain, because many domain concepts can only be meaningfully defined within a specific context, and forcing them into an enterprise-wide model leads to "god classes" and cyclical dependencies – that is, large, cluttered monoliths that are difficult to evolve. Evans proposed developing context-dependent domain models that unambiguously represent concepts and behavior, which leads to functional modularization of the software, or strategic design.

Strategic DDD

Suppose you had to create software for a movie theater. How would you define a movie ticket? A movie ticket is both a unit of sales (with a price, sales tax, discount options, etc.) and an access token (with validity, validation capability, etc.). To represent all these properties and the complete handling of a move theater ticket in a single model leads to the problems described above, commonly referred to in DDD as the "big ball of mud" [2].

If a domain is too large to be understood and defined as a whole, then it is also too large to model completely in software. In this case, it is better to represent self-contained parts of a domain in separate, smaller models, which allows the software and the development team to grow, limiting the cognitive load for all stakeholders, who no longer need to understand a sprawling overall model, but only manageable models within clearly defined contexts (aka bounded contexts). The goal of strategic design is to design these bounded contexts and their relationships to each other.

To begin, you need to break down the large and complex domain into subdomains. The result of this analysis is the linguistic boundaries within which business solutions can be modeled in a consistent and self-contained manner. In the simplest case, a subdomain becomes a bounded context. Considerations such as team size, technological complexity, strict requirements in terms of user experience, scalability, and more must be taken into account when designing bounded contexts. Designing bounded contexts therefore often means making compromises.

Some subdomains are more critical to business success than others. In

Lead Image © Christos Georghiou, 123RF.com

DDD, three types of subdomains are distinguished:

- Core Subdomains (typically simply known as core domains) distinguish a company and are often technically complex. Software for core domains can be *developed in-house* to provide a competitive advantage.
- Supporting Subdomains do not represent the technical core of a company, but they are necessary for the core domains' tasks. The technical models of supporting domains are typically specific to a company and rarely to an industry. Software for supporting subdomains can be *built by a service provider*.
- Generic Subdomains do not provide a competitive advantage and are not company specific. One classic example is payroll accounting, which is essential for a company, but does not, say, let movie theater operators distinguish themselves from their competitors. As an enterprise, you will want to *use off-the-shelf solutions* for generic subdomains rather than building the software yourself.

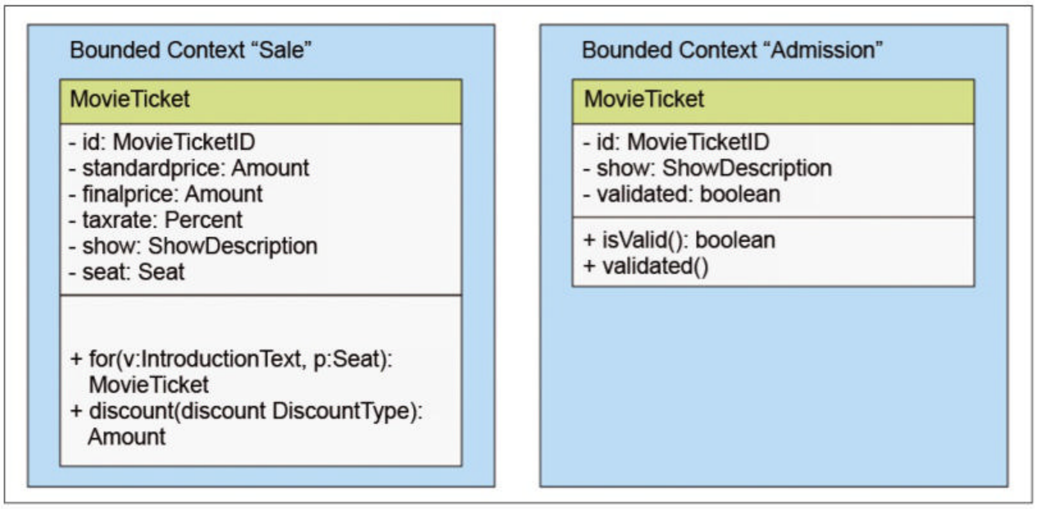


Figure 1: Same name, different model: The MovieTicket class in different contexts of a movie theater.

Bounded contexts can be implemented as modules of a monolith or as (micro-)services. They not only draw boundaries between the models, but also (1) between the teams' responsibilities (a team implements at least one bounded context but can implement more than one); (2) between the requirements (e.g., as a separate backlog for each bounded context); (3) in the source code (e.g., as a package tree or namespace, possibly even in the form of separate code repositories in a version control system), and (4) in the data storage. The last two points in particular are sometimes questioned: Don't they lead to duplicate code and redundant data? In fact, duplication is not a major worry because each bounded context holds only the data relevant to its specific model (Figure 1).

When you draw boundaries, you have to make sure business processes work across bounded contexts. Bounded contexts therefore need to be integrated, and teams need to consult on the required interfaces. In the DDD universe, this process is known as context mapping, and a number of organizational and technical patterns can be used to facilitate cooperation between teams. For example, a customer-supplier relationship defines a directed dependency between two teams: In Figure 2, Team A (supplier) provides the functionality on which Team B (customer) builds. Team B, as the customer, can typically impose requirements on Team A.

Context maps like that in Figure 2 visualize bounded contexts, their dependencies, and their organizational and technical integration.

Bounded contexts need to be able to perform their tasks as independently as possible (i.e., without calling other bounded contexts). They then are very different from the data-centric services of a service-oriented architecture (SOA). Instead of consisting of bounded contexts such as ticket purchase and admission, an SOA consists of a ticket service, a movie service, and so on. In an SOA, multiple services often need to be called to complete a business task.

Therefore, strategic design is about more than software modularization, technical integration, and data flows. The design of bounded contexts also involves model-level dependencies and relationships between the teams developing the bounded contexts. Team organization in particular has been one of the most common reasons companies have looked at DDD

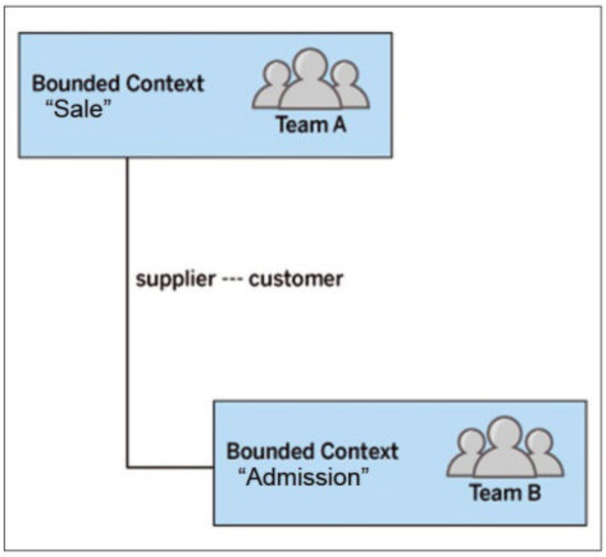


Figure 2: A simple example of a context map for a movie theater.

for a number of years. Strategic DDD also provides ideas for one of the most exciting approaches to organizing development teams: team topologies [3] – a model to describe teams and their interactions.

Tactical DDD

What does the architecture of a bounded context look like? When Evans published his book, layered architecture (including a domain layer) was the state of the art. Since then, other architectural styles have emerged that suit DDD even better, most notably hexagonal architecture with a domain core [4].
DDD supports the design of the domain layer or the domain core with a pattern language that defines the building blocks and permitted usage relationships (Table 1). The pattern language does not have to be used for every bounded context, but the more complex the domain expertise, the more worthwhile it becomes.
How the building blocks are implemented in the code depends on the programming language and programming paradigm. In the early days of DDD, object-oriented programming with Java dominated publications on tactical design. In the meantime, DDD has arrived in many languages. With the advent of languages such as Kotlin and F#, functional programming has gained widespread appeal in DDD. Many a domain model can be elegantly implemented with functional programming.
Technology openness also applies to the persistence concept. The classic approach is to store the state of an aggregate. Event sourcing [5], on the other hand, stores every change to an aggregate in the form of an event. The current state results from the sum of changes, just as the balance of a bank account results from deposits and withdrawals. DDD has been a major influence in the development of event sourcing. Conversely, event sourcing introduced a concept that entered both domain analysis and tactical design as another design pattern.

Table 1: Building Blocks

Module	Meaning
Entity	Domain-oriented concepts that have a life cycle and identity.
Value objects	A domain-oriented concept with value semantics (i.e., without identity); the properties of an entity are expressed as value objects.
Aggregate	Related entities and value objects combined to create a consistent whole.
Repository	A domain-oriented interface that stores and accesses aggregates (i.e., encapsulates persistence).
Domain service	Maps business processes and domain behavior that cannot otherwise be assigned to value objects, entities, and aggregates.

Domain Language

You need a domain language to design and implement domain models. As you know from strategic design, DDD has no such thing as a single definitive model and therefore no single domain language. Like any natural language, a domain language is characterized by dialects; it is ambiguous, context dependent, and subject to constant change.
However, technical models can only be expressed accurately with precise language and codified in software. In DDD, this precise language is known as the ubiquitous language. It is developed for each bounded context and based on a context-dependent subset of the domain language. The terms of the ubiquitous language are defined with sufficient precision (e.g., in a glossary) to describe requirements, to design the domain model of a bounded context, and to name business concepts in the code (e.g., a type of movie ticket).
In fact, this precision makes such a language ubiquitous. It is used in conversations, texts, diagrams, and code; prevents a conceptual break between domain and technical models (Figure 3); and facilitates communications with domain experts.
A ubiquitous language is not simply the result of analysis and is not an upfront design. It emerges from discussions between the subject matter experts and the development teams. It evolves over time and then prompts refactoring in the code.
Ubiquitous language and domain models are not a one-way street from business to IT. Software development can generate new business ideas. To stick to the movie theater example, during the development of an online ticketing service, the notion of a recommendation based on previous purchase behavior can be fed back into the ubiquitous language of the bounded context.

Collaborative Modeling

How do you find good context boundaries and develop ubiquitous languages and domain models? Evans cites interviews, discussing specific scenarios, prototype

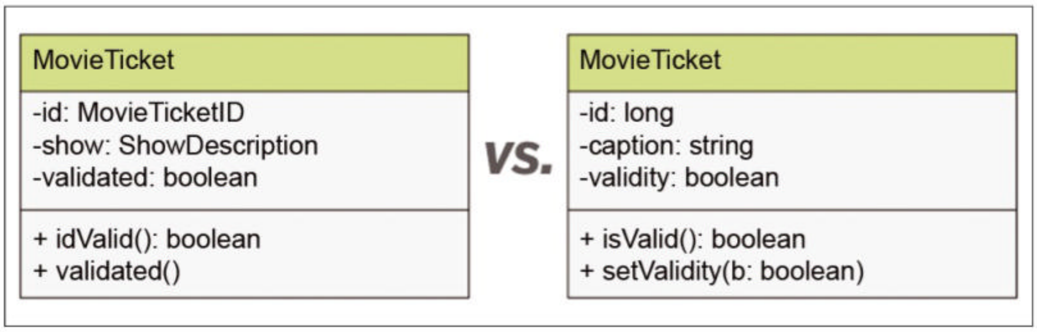


Figure 3: A class diagram with (left) and without (right) taking ubiquitous language into consideration.

implementations, and short feedback cycles. Beyond that, however, he provides aspiring DDD practitioners very few tools of the trade. This methodological gap for the analysis of domains and the design of application-oriented software was filled by others. Popular methods include event storming [6], domain storytelling [7], or example mapping [8]. These methods have in common the bringing together of development teams and subject matter experts who model collaboratively, which is why these workshop formats are grouped together under the term "collaborative modeling." The collaborative modeling toolbox is one of the most important additions to DDD, and the community is developing it intensively. Additionally, collaborative modeling can also be useful independent of DDD (e.g., when discovering requirements).

What DDD Is (Not)

DDD has become far more important in the past 20 years. However, widespread use is also accompanied by misunderstandings and exaggerated expectations, which should be cleared up now. DDD is neither a framework, nor an architectural style, nor a method, because there are no rules on how to apply it. It is also not a waterfall method in which models are designed up front (see the "DDD and Agility" article by Eberhard Wolff in this issue). It is by no means necessary to use the technology throughout from the strategic to the tactical aspects. DDD doesn't force you to build microservices, nor does it conjure away complexity; complex subject matter remains complex even with DDD. What DDD has at its core, on the other hand, is a set of principles [9]. To design software for complex subject matter, development teams (developers, testers, analysts, etc.) need to build a deep, shared understanding of the application domain. In the process, they are guided by subject matter experts. This understanding grows out of the language of the domain, which must be formalized in a joint process in a coordinated and unambiguous manner to create a ubiquitous language. Understanding is expressed in a model shared by business experts and developers that describes the problem space (as opposed to the solution space). The model must explicitly express the essential complexity of the domain. Complex subject matter cannot be efficiently expressed through a single universal model and language, which makes it essential to break the subject matter down into bounded contexts. The model, language, and code need to evolve as the understanding of the domain grows. DDD is not necessarily applied everywhere, in the spirit of pragmatism, but where it will have the greatest effect.
I hope that this article has helped to communicate a clear idea of DDD.
If you want to delve deeper, see the "Recommended Reading" box for my recommendations on the subject. ■

Recommended Reading

Domain-Driven Design Distilled [10] by Vaughn Vernon provides a good overview of the concepts of DDD. For a more comprehensive summary and advice for practical use, see Vlad Khononov's *Learning Domain-Driven Design* [11]. Although a decade has passed since Vaughn Vernon wrote *Implementing Domain-Driven Design* [12], it is still considered *the* reference work for anyone who wants to apply tactical design down to the code level. The author is currently working on a new version in the form of two books: The already published *Strategic Monoliths and Microservices* [13] addresses DDD from a product innovation and architecture perspective, and the complementary book on implementing strategic monoliths and microservices is due to be published in 2024.

Info

[1] Evans, Eric. *Domain-Driven Design: Tackling Complexity in the Heart of Software.* Pearson, 2003: [https://www.pearson.de/domain-driven-design-tackling-complexity-in-the-heart-of-software-9780321125217]
[2] Big ball of mud: [http://www.laputan.org/mud/mud.html]
[3] Team topologies: [https://teamtopologies.com]
[4] Hexagonal architecture: [https://en.wikipedia.org/wiki/Hexagonal_architecture_(software)]
[5] "Event Sourcing" by Martin Fowler: [https://martinfowler.com/eaaDev/EventSourcing.html]
[6] Brandolini, Alberto. *EventStorming.* Leanpub, last updated August 26, 2021: [https://leanpub.com/introducing_event-storming]
[7] Domain Storytelling: [https://domainstorytelling.org/]
[8] "Example Mapping" by Matt Wynne: [https://cucumber.io/blog/bdd/example-mapping-introduction]
[9] "What is Domain-Driven Design (DDD)" by Mathias Verraes: [https://verraes.net/2021/09/what-is-domain-driven-design-ddd]
[10] Vernon, Vaughn. *Domain-Driven Design Distilled.* Addison-Wesley Professional, 2017: [https://www.oreilly.com/library/view/domain-driven-design-distilled/9780134593449/] (video)
[11] Khononov, Vlad. *Learning Domain-Driven Design.* O'Reilly Media, 2021: [https://www.oreilly.com/library/view/learning-domain-driven-design/9781098100124/]
[12] Vernon, Vaughn. *Implementing Domain-Driven Design.* Addison-Wesley Professional, 2013: [https://www.oreilly.com/library/view/implementing-domain-driven-design/9780133039900/]
[13] Vernon, Vaughn, and Tomasz Jaskula. *Strategic Monoliths and Microservices: Driving Innovation Using Purposeful Architecture.* Addison-Wesley Professional, 2021: [https://www.oreilly.com/library/view/strategic-monoliths-and/9780137355600/]

Author

Stefan Hofer, a consultant and trainer, helps customers clarify requirements and apply domain-driven design. He has been working at WPS – Workplace Solutions GmbH (Hamburg and Berlin) since 2005. As one of the minds behind Domain Storytelling [13], he puts the language of subject matter experts at the center of requirements discovery.

Domain-driven design and agile development

Pulling Together

At first glance, the domain-driven design software architecture approach and the agile process model seem to cover different areas of software development. In fact, they do more than generate synergies; in some cases, they even aim for the same targets. By Eberhard Wolff

Domain-driven design (DDD) is a software development approach that develops a domain model – an abstraction of the behavior and data of a system – and refers to the entire design of the software, from the code level to the architecture, and the interaction of the development teams involved. For a consistent domain orientation to work, everybody involved needs to exchange information, especially technical experts and developers.

Ultimately, domain strategies can only form the basis of development if all stakeholders communicate them clearly; otherwise, only the subject matter experts have the required detailed knowledge of the domain – not the developers. This point is exactly where the first tie to agility can be found. One of the 12 principles of the agile software development manifesto includes: "Business people and developers must work together daily throughout the project" [1]. This arrangement is the only way to implement a domain-driven design.

Practice Instead of Theory

Initially, the DDD movement mainly encompassed ideas on structuring code or software systems. Discussing these with subject matter experts is difficult because they often lack the technical background to do so. Concepts like classes and modules are simply not general knowledge. However, it cannot be a prerequisite for joint work to first train subject matter experts in this area – the bar is simply too high. DDD, however, now includes a host of collaborative modeling techniques. The objective is to strengthen the understanding of the domain through joint work on artifacts that are not used directly to structure the software but serve solely to explain the domain in more detail.

Event storming is a well-known example of collaborative modeling, wherein everyone involved in the project uses sticky notes to create complex business processes [2]. Each sticky note contains an event from the domain (e.g., *Order received*) that clarifies the process flow. In practice, the method even works for more complex processes; you just need enough space for the sticky notes.

Because the idea is not particularly complicated at this level, everyone involved should be able to write a couple of sticky notes. Later, more information can be added on additional sticky notes of different colors. Another technique with the same objective – visualizing the flow of a business process – is known as domain storytelling [3]. In this way, DDD adds concrete techniques to the agile idea of collaboration between subject matter experts and developers.

Lead Image © omnimages, 123RF.com

Forced Iterations

DDD aims to establish a model of the domain in executable code. The logic in the code reflects the facts (e.g., calculating the cost of a delivery or its current state). In doing so, DDD assumes that despite all efforts, this model will never be perfect, as Eric Evans wrote in 2003 in one of the first books about DDD [4].

At its core, the teams' role in DDD is to understand the subject matter (knowledge crunching), which is underpinned by a learning process that takes time. Accordingly, the model in the code exclusively represents the developers' current understanding, with all the errors and inaccuracies that entails. The challenge in software development is not programming the code, but understanding what to program.

The model in the code cannot ever be perfect because the subject matter changes. If business processes change, you need to adapt your software. Conversely, introducing software influences the facts. If you engage with and reflect on the facts as you model them, you are likely to identify an optimization potential that could be lost in the daily grind. To take advantage of this potential (e.g., automation), you need to change both the business facts and the software.

If the business facts change during development, you respond to these changes with a new iteration (i.e., a new version) of the software. To ensure that everyone learns, you need to make the software available to users after each iteration. Developers discover new details of the business facts, and subject matter experts discover new optimization potentials. Therefore, one of the core elements of agile software development is developing software iteratively and incrementally. These conditions are required for DDD and knowledge crunching to work.

Prioritization

Relationships between agile values and DDD can be found in other areas, too. For example, the principles of the Agile Manifesto state that the highest priority is "to satisfy the customer through early and continuous delivery of valuable software" [1]. Typical agile processes comply with this principle by having domain experts prioritize features, which allows the principle to be implemented without affecting the structure of the software or the code.

Domain-driven design complements the core domain concept and refers to the area of a system that implements the motivation of the project, which sets it apart from generic subdomains that do not allow for differentiation. A core domain can be a product configurator, which gives the company a unique selling point compared with its competitors. An example of a generic subdomain could be invoicing, which although it is equally important because, without invoices you have no revenue, it does not further the cause of differentiation.

Core domains and generic subdomains break down the value of the software in the structure into more and less valuable parts of the software (Figure 1). The distinction between a core domain and generic subdomain is based on a simple observation. Obviously, not all parts of a software system have the same quality, on which basis you have only two courses of action: Either you leave quality distribution to chance, or you actively control it. If you decide to take control, you need to know which parts need to be particularly good, which is exactly where the core domain comes into play: You accordingly pay attention to the highest quality and create a competitive advantage in the process.

Strictly speaking, quality in this context means maintainability and adaptability. If a company differentiates itself from its competitors through a product configurator, it needs to be easy to use and adapt. Invoicing, on the other hand, requires other qualities, such as accuracy. A quality problem in invoicing would be unpleasant, but in the case of the product configurator, it turns into a genuine competitive disadvantage because of its importance. With concrete patterns like core domains, DDD helps concretize the abstract, agile concept of "valuable" software at the level of the software's structure. Accordingly, software can be described as valuable if it achieves high quality in the core domain.

DDD and Teams

Agile approaches rely on cross-functional teams. In teams, all roles gather to implement a business-relevant feature. The team will include developers with different orientations such as at the front end or the back end. Other roles can also exist such as user experience (UX) experts or operations specialists.

DDD comes to similar conclusions from different directions. It divides a software system into bounded contexts, behind which are business modules that integrate a specific, separate business functionality (e.g., invoicing or taking orders). These modules are significant in terms of analysis, as well, because specific technical terms apply in a bounded context. Accounting includes terms such as "tax" and "debt collection." At first, it seems that bounded contexts simply define the structure of the software. But appearances are deceptive: DDD gives them a role in the project's organization. Ideally, one team will take care of one bounded

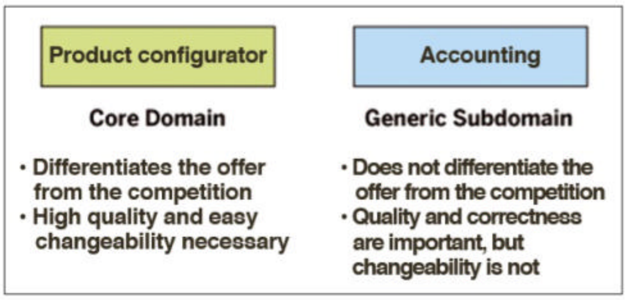

Product configurator
Core Domain
• Differentiates the offer from the competition
• High quality and easy changeability necessary
Accounting
Generic Subdomain
• Does not differentiate the offer from the competition
• Quality and correctness are important, but changeability is not

Figure 1: The product configurator core domain and the generic accounting subdomain are given different priorities.

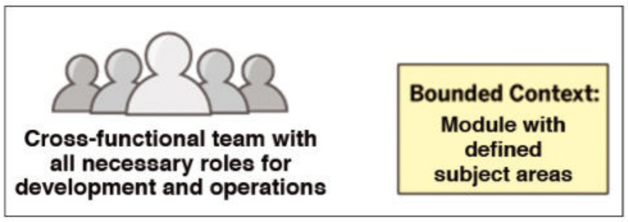

Cross-functional team with all necessary roles for development and operations
Bounded Context: Module with defined subject areas

Figure 2: A cross-functional team works on a bounded context.

context. Because the team is responsible for the specific bounded context and therefore also for the domain-oriented functionality implemented in it, the team can autonomously make certain domain-oriented changes. Cross-functional teams are based on the same idea. You would build a team such that it can implement business features on its own, if possible. Agility focuses on the composition of the team, and ideally it should reflect all roles. DDD, on the other hand, focuses on the division of the software, which needs to proceed such that each team can implement independent functionality. As soon as you combine the two concepts, cross-functional teams emerge that work on a bounded context and cover the business facts in full (Figure 2).

Such teams comprise five to eight people who develop software together. However, systems often turn out to be so large that a single team cannot handle the work involved, so you will end up coordinating several teams. In this case, DDD provides for relationships that result from bounded contexts that you can assign to teams (Figure 3).

When a team needs help from another team to implement features in its own bounded context, it enters into a customer-supplier relationship. As the customer, the team requests functions from the other team, which then assumes the supplier role. Formulated in the context of this example, one team takes care of invoicing and needs support from the team that takes orders. Without an interface that provides information about the order, it is almost impossible to write an invoice. The customer team communicates the necessary features, and the two teams then negotiate the necessary tasks and schedule for delivery.

In the conformist alternative, the team cannot request anything but adapts to what is offered. This situation might occur if the other team is looking after legacy software that is no longer maintainable and therefore simply cannot meet specific requirements. If the order-taking software cannot be changed, the invoicing team will find itself in a conformist role, with the corresponding implications for the development of the software.

Strictly speaking, these relationships exist at the bounded context level, which means that as soon as another team takes responsibility for a bounded context, it also has the associated role (e.g., customer or conformist).

Once again, DDD and agility complement each other. Autonomous teams that can meet technical requirements as independently as possible are supported by patterns because they can now request help from other teams. These patterns rely on a decentralized organization. After defining a customer-supplier relationship, teams can interact in line with that relationship. Central coordination, as envisaged by some approaches to scaling agile methods, is no longer needed in this case.

The team topology [5] describes further relationships between teams. It extends these ideas to include constructs such as platform teams, which are not responsible for a technical part of the system such as a bounded context, but instead offer a platform that provides technical features on which other teams can build.

Humans, Not Robots

All of these patterns and practices in both DDD and agile development have an effect on the formal organization. Accordingly, they would be reflected in an organizational chart, but people don't always follow organizational charts. Just because it says somewhere that one team should help another team doesn't mean that employees will actually behave that way. They all know ways to get around reorganization or instructions from management.

When this situation transpires, an important idea from the Agile Manifesto takes hold. Individuals and interactions play a more important role than processes and tools. This statement can be interpreted as meaning that formal collaborations, such as those established by DDD or team topologies, do not matter as much as the actual interactions of individuals. The work of many agile coaches or scrum masters focuses on improving these interactions and interpersonal relationships because both are critical to the success of agile implementation.

The term "sociotechnical system" has become established in the field of DDD and means not only considering the software but also the organization that develops the software system. Experience shows that

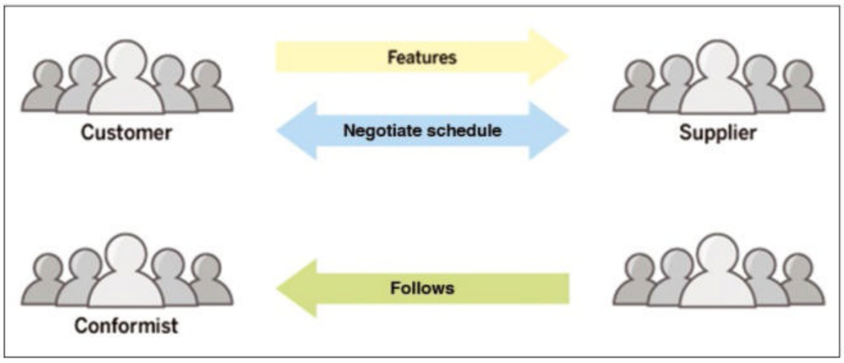


Figure 3: The two types of relationships between teams in DDD – customer-supplier and conformist – follow different rules.

problems in projects mainly occur in the interaction between the organization and the software. Additionally, organizations and people react very differently to measures such as reorganization, whereas software is essentially deterministic and can be implemented by an engineering approach.

Fact and Fiction

Therefore, DDD and agile development face a very similar problem: Ideas only appear to be realized. The term "cargo cult" is used in this context. Cargo cult manifested with the inhabitants of Pacific Islanders who watched American planes carrying goods to the islands during World War II. However, they did not understand the context, so after the Americans left the islands and more goods failed to arrive, the islanders built runways or headphones to mimic the Americans' behavior – in the hope of receiving deliveries.

In the case of cargo cult agility, you can hold all scrum meetings without following the Agile Manifesto and the values defined therein, including, among other things, generating business value, close cooperation between subject matter experts and developers, and process optimization. Thanks to DDD, you can use the techniques (i.e., decompose a system into bounded contexts) without aligning the architecture with the business value or genuinely understanding and optimally supporting the business meaning of the system.

The risk of implementing DDD or agile development in this way exists because these superficial activities are easy to review and easy to implement compared with agile values. Working on the orientation of the project, the architecture, or the values requires far more effort. Additionally, you will find it much more difficult to identify whether these values or the desired alignment has been achieved.

Collaborative modeling is a practical technique from DDD and a good example of the different levels. At first glance, it is simply a matter of creating an artifact; for example, event storming yields an overview of all the events in a given business context. As a concrete, tangible effect, the end result is an artifact that possesses a specific quality.

At the same time, when people interact with each other in the event storming process, the tensions or alliances that exist in the organization are revealed, as well as which people have particularly detailed knowledge in which areas. Moreover, participants can practice collaborating on a problem during an event-storming session.

Conclusions: Only Together

To see how close the relationship between agility and DDD is, it's worth asking whether you can even realize DDD without agility. DDD requires that the project is oriented on business values and a precise understanding of what the project is really trying to achieve within the business. However, this process only works through close collaboration, and that requires agility. Similarly, as already mentioned, adapting the model implemented in the software as the basis of DDD knowledge crunching is something that can only be achieved through iterative and incremental development. Accordingly, you need to implement the basic cornerstones of agility to operate DDD successfully.

Now you are thinking that maybe you can implement agility without DDD. Ultimately, DDD seems to be more technical at its core and more focused on architecture. In fact, it is also possible to develop highly technical systems in an agile manner (e.g., a machine control system). DDD makes little sense in such an area because it focuses on systems that support business processes. For such systems, however, an agile process will likely lead to the use of at least basic elements of DDD.

DDD and agility are two essential mechanisms of modern software development. The core of agility is values from the Agile Manifesto and an approach defined in iterations. DDD is about mapping the domain and the subject matter particularly well and therefore operates more at the level of development and architecture, whereas agility tends to describe a process model. However, the two are more or less mutually dependent if you want to implement them effectively. People are ultimately at the center of both strategies and the question is how they can develop in an agile and domain-oriented manner. In areas such as sociotechnical systems, the two approaches come together to solve this central problem of software development.

If you want to delve further into the subject, you can check out a video online [6] to learn about possible challenges when implementing DDD. ■

Info

[1] Agile Manifesto: [https://agilemanifesto.org/principles.html]

[2] Event storming: [https://www.eventstorming.com]

[3] Domain storytelling: [https://domainstorytelling.org]

[4] Evans, Eric. *Domain-Driven Design: Tackling Complexity in the Heart of Software*, O'Reilly, 2003: [https://www.oreilly.com/library/view/domain-driven-design-tackling/0321125215/]

[5] Team topologies: [https://teamtopologies.com]

[6] DDD Europe 2019: [https://www.youtube.com/watch?v=eKIMpCF-cqU]

[7] Eberhard Wolff homepage: [https://ewolff.com]

Author

Eberhard Wolff is Head of Architecture at SWAGLab GmbH (Hamburg, Germany) and has been working as a software architect and consultant for more than 15 years. He is the author of numerous articles and books, is a keynote speaker at international conferences, and offers a weekly live-stream on the topic of software architecture [7]. His technological focus is on modern architecture and development approaches such as continuous delivery, DevOps, and microservices.

Revamp your software architectures with Domain-Driven Transformation

Panacea

Domain-Driven transformation can refurbish a legacy system in increments while mitigating risk. By Carola Lilienthal and Henning Schwentner

When greenfield teams start a software project, it's all fun and easy. The developers respond to new requirements at lightning speed, and the users are thrilled. Development proceeds in leaps and bounds. However, this picture changes over the lifetime of the system as the complexity of the software inevitably increases, making the system more prone to error, slowing progress and affecting maintainability.

When worst comes to worst, even the smallest change can take months to reach production. What was initially a flourishing green meadow has turned into a brownfield. "Legacy system," "old software," "big ball of mud," and "monolith" are the unflattering names teams use for these kinds of systems, but don't give up hope: You can bring flexibility, error resilience, and development speed back to aging systems. The core task is to control and break up the complexity.

Software systems suffer from different "diseases," and you need a variety of medicines to cure them. Four malaises in various permutations are observed in organizations and their legacy systems, whether monoliths or microservices.

Over time, a legacy system grows into a "big ball of mud," wherein unmanaged dependencies lead to everything being interrelated with everything else. Additionally, business cases become entangled in a large domain model whose parts don't genuinely fit together – or that even get in each other's way. In the third disease, the domain and technical source code intermingle. In such cases, replacing obsolete technology or expanding the business case have mutated into Herculean tasks. To make things worse, the stakeholders are bogged down in a team structure that does not lend itself to, or in fact prevents, fast progress.

Within the last 20 years, work with Domain-Driven Design (DDD) and legacy software has identified some cures for these diseases: refactoring, domain storytelling, event storming, team topologies, and the modularity maturity index (MMI). These cures can be combined in a kind of therapeutic plan referred to as Domain-Driven Transformation.

When addressing the treatment of a project, you should ensure that the development team has a positive outlook that in turn boosts motivation. The further the healing process progresses, the more satisfied the users, project leads, and managers, as the clumsy and expensive legacy software becomes more stable,

Lead Image © faithie, 123RF.com

adapts more quickly, and ultimately even opens up to innovative, forward-looking extensions.

The Choices

The first question is whether to replace the legacy system completely or keep it as a foundation and transform or refactor it. The replacement solution has two variants: a big bang or a step-by-step replacement. Big bang means building a completely new system on a greenfield site while the legacy system remains in use. At a certain point, you flip the switch from the legacy to the new system.
The gradual replacement of a legacy system allows you to develop a new system one slice at a time and use each slice in production at the earliest possible stage. At the same time, you disable the corresponding functionality in the legacy system. If you decide to walk down the reshaping road, you need to refactor the big ball of mud because it remains operational all the time.
To begin, you should look at the advantages and disadvantages of the different types of transformation so you can make the decisions that are right for you. Figure 1 visualizes the first replacement variant, the big bang. This strategy is known as the "big bang" because in the end you are in a whole new world. Unfortunately, the name also hits the mark because, inevitably, something is going to blow up in your face. What sounds good in theory (i.e., establishing the new system next to the old one) does not work in practice for various reasons. For example, a lot of unknown knowledge is hidden in the legacy system that often falls by the wayside during reconstruction. The legacy system often turns out to be so large that it cannot be rebuilt in a short period of time. Therefore, the legacy system cannot be "frozen," so it becomes a moving target during refactoring.
You can probably tell from these comments that we are not fans of a big bang rollout, and we are not the only ones who see the difficulties of replacing legacy systems in this way. The collective experience of the IT industry shows that the second approach – replacing the legacy system step-by-step with a new system that grows by increments (Figure 2) – works better than the big bang.
In the process, you gradually cut out parts of the legacy system, revamp the design, and set the new parts up alongside the legacy software. After a short time, users are directly confronted with the new system and can work with it in production. As soon as the desired functionality is available in the new system, you can switch off the old one (arrow 4).
Nearly two decades ago, Martin Fowler gave this approach the name "strangler fig application" [1]. He chose this name because the old system is entangled and eventually overcome by the new system, like a host tree is by a strangler fig.
This kind of modernization lends itself to the agile approach favored in IT, wherein you learn from iteration to iteration. You can continually evaluate and adjust the path forward according to the latest findings.
Organizations assume a great deal of intelligence resides in their legacy systems; understandably, they want to keep this know-how for the future. In this case, it's not a good idea to replace the legacy system. Instead, just continue to operate it, rebuilding it in increments through refactoring.

Strategic Transformation

To disassemble legacy software technically, you need to capture and think through the current work processes with all their workarounds. In other words, the key to success lies in (re) understanding the domain. It is not enough to look at the solution (i.e., the existing source code) alone. Instead, you have to start with the problem – the domain – to unravel and simplify a software system that has become entangled over the years through additions and changes. When doing so, you need to consider two things.
To begin, you mentally have to put aside the monolith and the structure that it contains and look at the basics of the business case or, more specifically, break down the domain into subdomains. Interestingly, at least from an analytics point of view, you start from scratch, even though you already have a system. From this fresh understanding of the domain, you can identify the redundant or misunderstood parts in the software solution and remove and rebuild them.

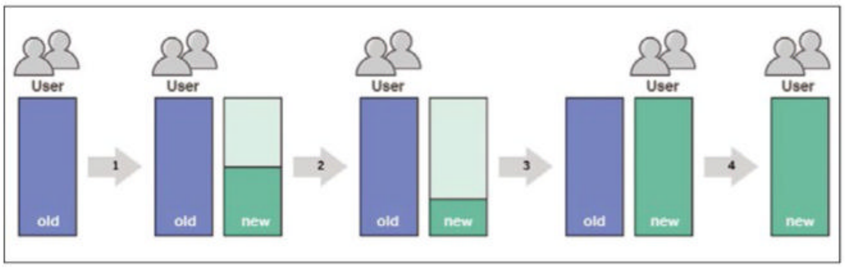


Figure 1: The old system remains in service (arrows 1 and 2), but only until the new system is complete. At this point, you switch over in one fell swoop (arrow 3).

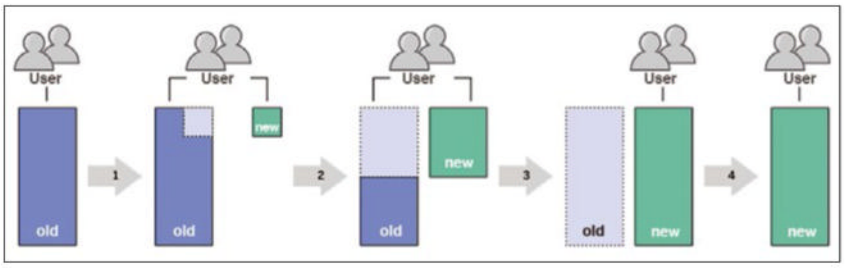


Figure 2: In a step-by-step replacement, you slowly replace individual components until the legacy system has completely disappeared.

Therefore, for both the development of the new software system and the decomposition of the monolith, the critical activities do not simply consist of writing code but, first and foremost, include stakeholders learning from and understanding each other. Developers need to learn what exactly the business experts do in their daily work. On the other hand, the business experts need to understand the technical reasons for the constraints and prioritization in the solutions.

Strategic transformation can be described by four substeps that include rediscovering the domain, modeling the domain-oriented target architecture, comparing the architecture with the target architecture, and prioritizing and implementing the restructuring measures.

Sociotechnical Transformation

To transform a legacy system into something better, you need to consider more than just the technical and business dimensions of domain-driven transformation. It often makes sense to rethink team structures and procedures at the same time.

Fortunately, the agile approach is well established in many organizations. If this is not yet the case in your organization, you need to consider taking the plunge because agile transition and domain-driven transformation often go hand in hand. Moreover, organizational domain-driven transformation often means evolving horizontally organized teams (Figure 3) into vertically organized workgroups (Figure 4).

In the following discussion, the term "refactoring" is used in a very broad sense. Alternative terms would be "reorganization" or "sociotechnical transformation." To give teams some idea of how the planned changes should proceed, you need to look at some types of sociotechnical refactoring that include the cross-layer refactoring [2] method, wherein an interdisciplinary team is establish from the members of the various specialist groups (user interface, business logic, database, etc.).

In partly layered refactoring [3], you establish a second interdisciplinary team from members of additional functional groups and the first cross-functional team. During second-team refactoring [4], you take a similar approach, the difference being that the existing interdisciplinary teams continue to work without interruption because none of their members need to join the new working group. At the same time, you can create several new interdisciplinary teams.

Tactical Transformation

Tactical transformation is about strengthening the expertise in your legacy software that is often inconsistent and hidden under a heap of technology. A number of measures help to expose the business case [5].

Start by separating the business and technical source codes from each other. Then, to increase cohesion, you enrich your modeling from a technical perspective, use value objects [6], introduce technical prerequisites, and make the technical identity of entities explicit. At the same time, you reduce the coupling by introducing ID references at unit boundaries, reducing inheritance in domain-oriented source code, and using domain events. Finally, it's essential to document the architecture in code and test it.

Conclusions

Whether legacy systems are simply in poor condition or already burning wrecks, business treasure is hidden in them all. In most cases, the goodies are valuable enough to bring out of hiding and back into the light, rather than being replaced with completely new development. With new development, you can expect so many unknowns that the overhead often exceeds the estimates many times over. Like greenfield development, refurbishing a legacy system does

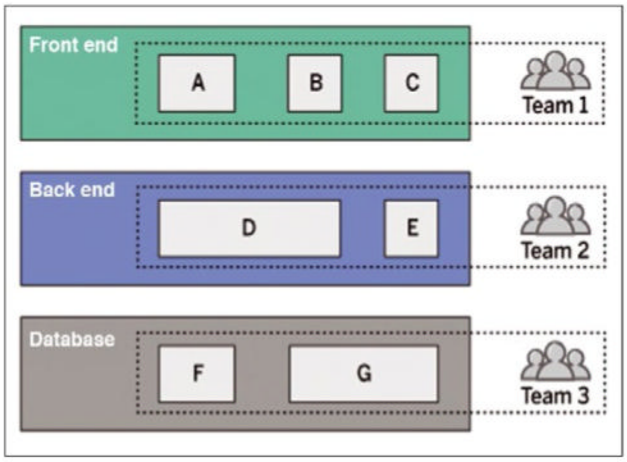


Figure 3: The way to dependencies: layers with horizontal teams.

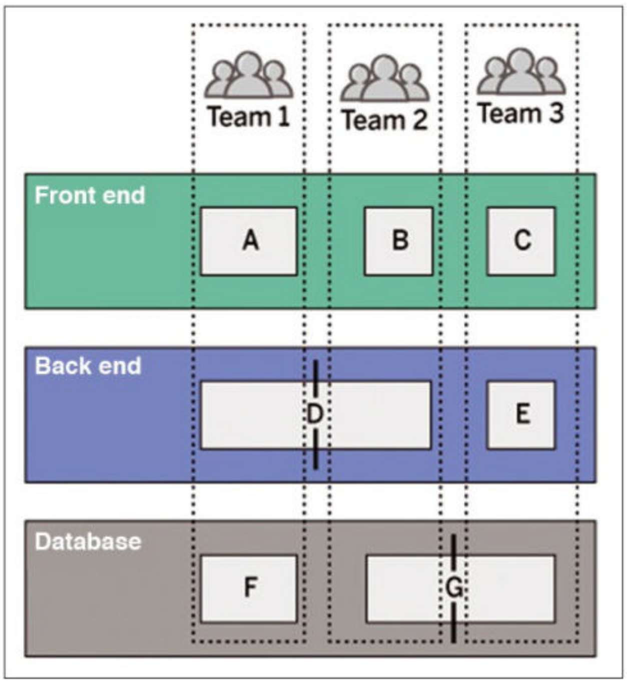


Figure 4: The possibility of independence: layers with cross-functional teams.

consume serious chunks of time. Domain-driven transformation helps implement this undertaking in small increments, mitigating risk at the same time. To unearth the domain-oriented treasure in a legacy system, you need to identify the parts of the source code that contain the valuable business knowledge.

Of course, this article can only give you a rough overview of the massive field of software architecture. For more on the topic, see our book, *Domain-driven Transformation* [7]. ■

Info

[1] "StranglerFigApplication" by Martin Fowler, June 2004: [https://martinfowler.com/bliki/StranglerFigApplication.html]

[2] Cross-layer refactoring: [https://hschwentner.io/domain-driven-refactorings/socio-technical/form-cross-functional-team-out-of-layer-team-members]

[3] Partly layered refactoring: [https://hschwentner.io/domain-driven-refactorings/socio-technical/form-second-team-out-of-partly-layer-team-and-first-team-members]

[4] Second-team refactoring: [https://hschwentner.io/domain-driven-refactorings/socio-technical/form-second-team-out-of-layer-team-only]

[5] Domain-driven refactoring: [https://hschwentner.io/domain-driven-refactorings]

[6] Value object: [https://martinfowler.com/bliki/ValueObject.html]

[7] Lilienthal, Carola, and Henning Schwentner. *Domain-Driven Transformation: Monolithen und Microservices zukunftsfähig machen.* dpunkt, 2023, [https://dpunkt.de/produkt/domain-driven-transformation/] (in German) *Domain-Driven Transformation.* Addison-Wesley, upcoming. (in English)

[8] Hofer, Stefan, and Henning Schwentner. *Domain Storytelling: Gemeinschaftlich, visuell und agil zu fachlich wertvoller Software.* dpunkt, 2023, [https://dpunkt.de/produkt/domain-storytelling/] (in German); *Domain Storytelling: A Collaborative, Visual, and Agile Way to Build Domain-Driven Software.* Addison-Wesley, 2021 (in English)

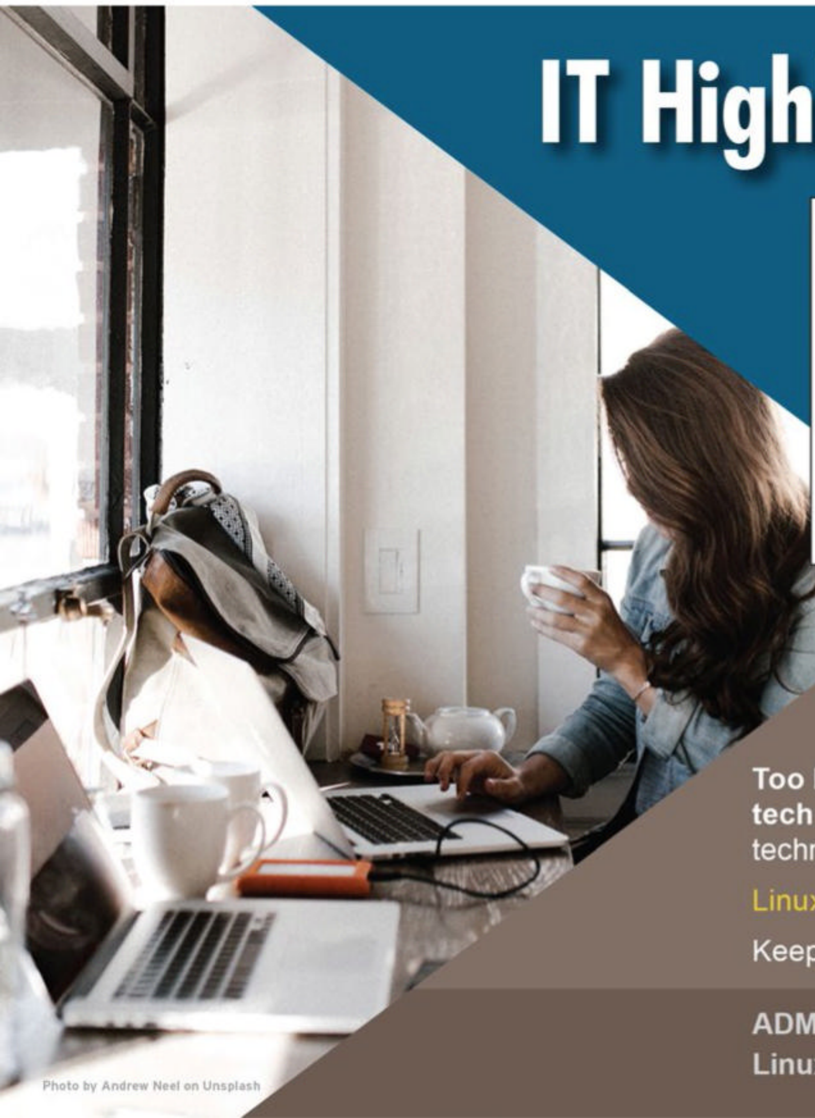
[9] ComoCamp: [https://comocamp.org]

Authors

Dr. Carola Lilienthal is a software architect and managing director at WPS – Workplace Solutions GmbH (Hamburg and Berlin). Since 2003 she has been analyzing the future viability of software architectures, writing books and articles on the subject, and delivering keynotes at conferences.

Henning Schwentner lives out his passion for high-quality programming as a coder, coach, and consultant at WPS – Workplace Solutions GmbH. He is a public speaker on domain-driven design, the author of *Domain Storytelling* [8], and a co-founder of ComoCamp [9].

IT Highlights at a Glance



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

Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

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

Automate Active Directory management with the Python PyAD library

Snappy Python

Windows admins can use the Python PyAD library to automate Active Directory configuration and management. Deployment on a Windows server is a snap and paves the way for delegating tedious routine tasks to Python scripts. By Tam Hanna

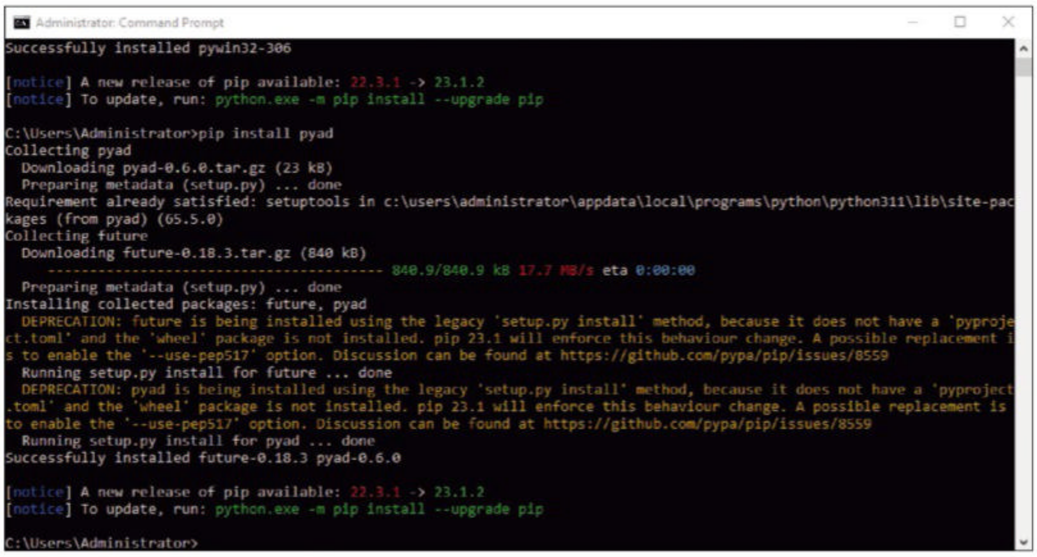
PyAD [1] is a useful tool for Active Directory (AD) automation with Python in many environments. However, about two years ago, the developer decided to stop maintaining the open source product. The lack of pull requests in the Git repository shows that the library generally works without problems.

In practice, however, the library needs to be assessed as a potential risk; after all, it does affect Active Directory. Administrators who are not forced to use Python will want to evaluate their alternatives. Moreover, conscientious testing is strongly recommended before rolling out a new Windows Server version.

Getting Set Up

Unix operating systems come with a Python runtime in place out of the box. However, Windows Server 2022, which is used in the following examples, does not have Python support. The Python you need is available from the Python website [2]. For this project, I used the 3.11.3 version current at the time of writing and the Windows Installer for 64-bit architectures. For easier handling, it makes sense to let the installation wizard include the Python EXE file in the server's path. Otherwise, you can work with the *Install Now* option: You do not need to adjust the default settings of the installation wizard. At the end of the installation, you will want to disable the path length limitations and then restart the server to complete the installation.

The Python development team takes care to make the integration of extensions easy. All of the current crop of Python runtimes come with a package manager that is based on the



```
Successfully installed pywin32-306

[notice] A new release of pip available: 22.3.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Administrator>pip install pyad
Collecting pyad
  Downloading pyad-0.6.0.tar.gz (23 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: setuptools in c:\users\administrator\appdata\local\programs\python\python311\lib\site-pac
kages (from pyad) (65.5.0)
Collecting future
  Downloading future-0.18.3.tar.gz (840 kB)
     ---------------------------------------- 840.9/840.9 kB 17.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Installing collected packages: future, pyad
  DEPRECATION: future is being installed using the legacy 'setup.py install' method, because it does not have a 'pyproje
ct.toml' and the 'wheel' package is not installed. pip 23.1 will enforce this behaviour change. A possible replacement i
s to enable the '--use-pep517' option. Discussion can be found at https://github.com/pypa/pip/issues/8559
  Running setup.py install for future ... done
  DEPRECATION: pyad is being installed using the legacy 'setup.py install' method, because it does not have a 'pyproject
.toml' and the 'wheel' package is not installed. pip 23.1 will enforce this behaviour change. A possible replacement is
to enable the '--use-pep517' option. Discussion can be found at https://github.com/pypa/pip/issues/8559
  Running setup.py install for pyad ... done
Successfully installed future-0.18.3 pyad-0.6.0

[notice] A new release of pip available: 22.3.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Administrator>
```

Figure 1: The deprecation warning in Pip is relevant for PyWin32: Avoid the update.

Lead Image © Eric Issele, 123RF.com

structure of the defaults known from Linux. The `pip` command can be used in the following, but watch out: As part of the initial execution, Pip offers to update from version 22.3.1 to 23.1.2 (Figure 1). Do not agree to this request under any circumstances if you want to use PyAD, because the library relies on an outdated installation command that no longer works in the new version.

The PyWin32 extension is required to use PyAD. You can install the extension and the PyAD library with

```
pip install pywin32
pip install PyAD
```

Previously, it was necessary to download PyWin32 manually because the PyAD development team was a little lacking in the dependencies department. The repository does not have a dependency on PyWin32, so Pip does not install the support library when deploying the Active Directory access module.

First Steps

After restarting the server, you need a work file in PY format. For an initial test, I used the sample program:

```
import pyad
from pyad import aduser
user = pyad.aduser.ADUser.from_cn(➘
      "Administrator");
print (user);
```

`ADUser` is an AD user object. The method assigned to `user` fetches the user object from the domain controller (DC) associated with the currently active user account.

The `PyAD.set_defaults` method can be used to adjust the default settings for connecting to AD instances. It supports configurations that the library implicitly accepts on calls and uses them to establish the connection:

```
import pyad
from pyad import *
pyad.set_defaults(➘
  ldap_server="<DC address>", ➘
  username="<Account>", ➘
  password="<Password>")
user = ➘
  pyad.aduser.ADUser.from_cn ("<user>")
```

You can also specify settings within the connection setup as follows:

```
import pyad
from pyad import aduser
user = aduser.ADUser.from_cn (➘
  "<user account>", options=dict (➘
    ldap_server=➘
    "<domain controller address>"))
```

The library supports the `ldap_server`, `gc_server`, `ldap_ port`, `gc_port`, `username`, `password`, and `ssl` parameters. Note that the data transfer between the Python runtime and AD server is typically encrypted. If you do not want encryption, manually change the `SSL` value to `False`.

Managing Users in AD

The returned `ADUser` object provides insights into the settings applicable to the user account. The class contains methods for editing the account. One good example is determining the age of the password. The `user.get_ password_last_set()` method returns a date-time object, so now you can send the return value directly to the command line with `print`. The following two-liner reveals the age of the admin password:

```
user = pyad.aduser.ADUser.from_cn (➘
      "Administrator");
print (user.get_password_ last_set());
```

You can find more options in the documentation [3]. The `set_ expiration(dt)` method is interesting. It lets you set the password expiration date for users.

Because you are using the PyWin library, standard Python `datetime` objects are incompatible. Calls therefore fail and output an error message stating *TypeError: must be a pywintypes time object (got datetime.date)*. However, converting Python data types to PyWin time types is a task that has already been solved [4].

In practice, however, it can be more convenient to create a direct instance of a date-time object instead. The object can be transferred directly afterward:

```
import time
import pywintypes
from pyad import aduser

now_seconds = time.time ()
now_pytime = pywintypes.Time (➘
  now_seconds+60*60*24*2)

user = pyad.aduser.ADUser.from_cn (➘
  "Administrator");
print (user.set_expiration( now_pytime));
```

In the lab, when I tried to set an expiration time for the administrator account, I occasionally saw an error stating *A device attached to the system is not functioning.*\r\n. The recommendation is to use scripts with exception handling to field any errors caused by PyAD.

PyAD also lets you use new users to AD. In the simplest case, it looks like this:

```
from pyad import aduser
from pyad import *

ou = pyad.adcontainer.ADContainer.from_dn (➘
  "test.tamoggemon.com");
new_user = pyad.aduser.ADUser.create (➘
  "<New User>", ou, password="<Password>")
print (new_user)
```

The method for `new_user` requires an AD organizational unit (OU) in addition to a username and password. The easiest way to meet these requirements is by typing

```
PyAD.adcontainer.ADContainer.from_dn
```

The parameter passed is the distinguished name of the OU container you need to address.

The most important obstacle when running the above script is that the password criteria stored on the Windows server also apply to users entered via PyAD. If the supplied password does not meet these conditions, Active Directory refuses to accept it and outputs an error message. Assuming everything works, you will find the accounts in Active Directory Users and Computers (Figure 2).

Searching for Resources

PyAD is not limited to generating new content. The library can also extract information from Active Directory and make it accessible by the various PyAD editing functions. As before, the resources returned by PyAD depend on the permissions with which the script is running. The method referred to earlier for setting connection parameters lets you make some adjustments in this area. Note, however, that storing credentials in script files is not the preferred approach in terms of security.

The simplest way to find resources is by their globally unique identifiers (GUIDs). Active Directory needs them for its objects, but they are also used during the installation of programs for unique identification. A GUID is a 16-byte (128-bit) number that contains a large amount of information (e.g., the MAC address

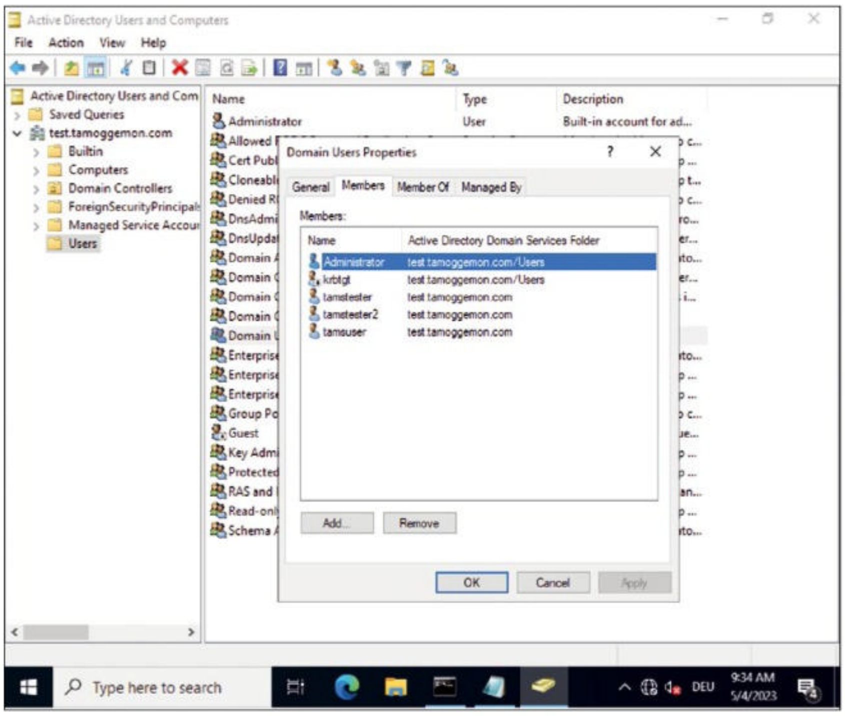

Active Directory Users and Computers
File Action View Help
Active Directory Users and Com
Saved Queries
test.tamoggemon.com
Builtin
Computers
Domain Controllers
ForeignSecurityPrincipals
Managed Service Accounts
Users
Name
Type
Description
Administrator
User
Built-in account for ad...
Domain Users Properties
General
Members
Member Of
Managed By
Members:
Name
Active Directory Domain Services Folder
Administrator test.tamoggemon.com/Users
krbtgt test.tamoggemon.com/Users
tamstester test.tamoggemon.com
tamstester2 test.tamoggemon.com
tamsuser test.tamoggemon.com
Add...
Remove
OK
Cancel
Apply
Type here to search
DEU
9:34 AM
5/4/2023

Figure 2: Sufficiently complex passwords lead to the display of user accounts created in Python.

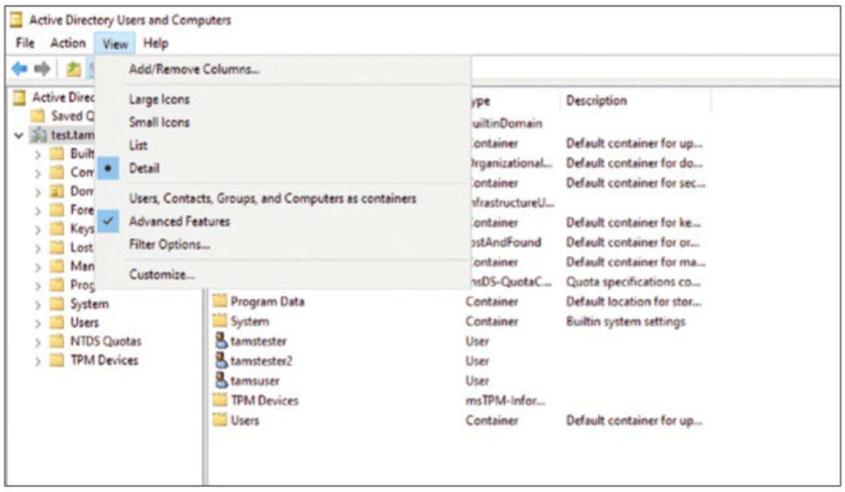


Figure 3: GUIDs appear in the Active Directory Users and Computers window if you enable the *Advanced Features* option.

of the network card to ensure that the GUID is globally unique). Make sure you enable the option shown in Figure 3 in Active Directory Users and Computers. You can then get a GUID using:

```
user = aduser.ADUser.from_guid (➘
  "<XXX-XXX-XXX>")
```

Because the GUID editor included in Windows Server does not directly support exporting to the clipboard, it is more convenient in many cases to use distinguished names to determine immutable Active Directory elements:

```
testuser = aduser.ADUser.from_dn (➘
  "CN=tamstester, DC=test, ➘
   DC=tamoggemon, DC=com");
print (testuser)
```

At the command line, these lines output the properties belonging to the AD user object. Alternatively, the methods discussed previously are available to adapt the user account's status to something closer to the desired configuration.

In practice, however, the distinguished name or GUID are rarely available. Searching AD is difficult in PyAD because the search process comprises two steps. ADQuery objects use a query language similar to SQL that records the query you want to run against the AD server. The following parameters are useful for an initial attempt:

```
import pyad.adquery
q = pyad.adquery.ADQuery()
q.execute_query(attributes = [➘
  "<Distinguished-Name>", ➘
  "<Description>"], ➘
  where_clause = "objectClass = '*'",➘
  base_dn = "CN=<users>, DC=test, ➘
             DC=tamoggemon, DC=com"
)
```

What is important here is to adapt the string passed in `q.execute_query` to match the current domain. The return value is a data field whose values are pushed to the command line with a `for` iterator:

```
for row in q.get_results():
  print (row["<Distinguished-Name>"])
```

For example, you can determine AD groups as shown in Figure 4.
The next step is a more advanced analysis. You need to break down the AD group objects into their members with another iterator:

```
for row in q.get_results():
  group = pyad.adgroup.ADGroup.from_dn (↩
    row["<DistinguishedName>"])
  print (group)
  for item in group.get_members():
    print (item)
```

You will also want to protect the `get_results()` functions with exception handling. The underlying library throws an exception of the type *(0, 'Active Directory', 'There is no such object on the server.\r\n', None, 0, -2147217865)* if it encounters empty AD groups, and this error usually terminates the program run.
Note that Active Directory implements rate limiting in the query functions used by PyAD [5] [6], which is significant when a query returns more than 1,500 results.

Moving Objects in AD

PyAD does not limit you to generating and querying AD objects. The library can also generate various elements.
In general, a method that follows this pattern is used:

```
ou = ADContainer.from_dn (↩
  "ou=<workstations>, ↩
   dc=<domain>, dc=com")
```

The `ou` variable must find an AD container instance, which determines the container in which the changes requested by the Python code will be executed. AD container objects also support you at this point with a number of `create` methods. For example, the following lines add a new computer to the created AD container group:

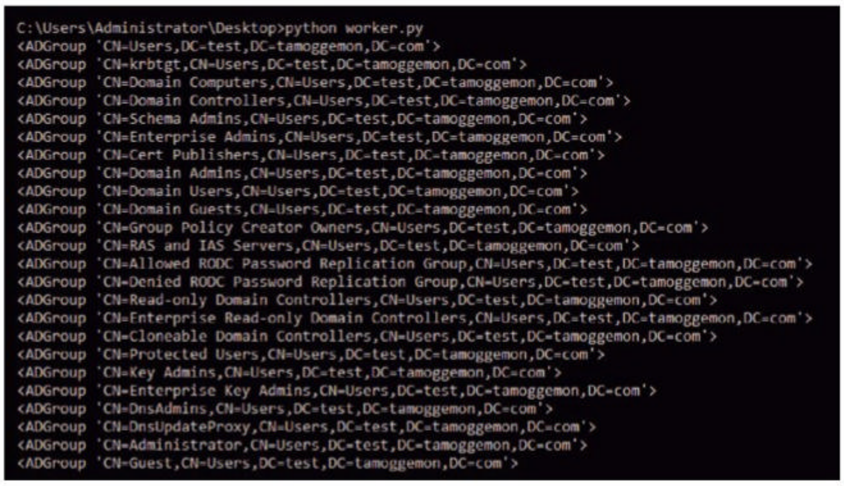
```
new_computer = ↩
  ADComputer.create ("WS-489", ou)
new_group = ↩
  ADGroup.create ("<IT Department>", ↩
    security_enabled=True, ↩
    scope='UNIVERSAL', ↩
    optional_attributes = {"description":↩
    "<All IT employeees in the company>"})
```

If your access authorizations are sufficient, you can rename objects or change their locations in Active Directory:

```
computer = ou.create_computer ("WS-490")
computer.move(ADContainer.from_dn (↩
  "ou=<Workstations>, ou=HR, ↩
   dc=<Company>, dc=com"))
computer.rename("WS-501")
```

In this case, you are customizing the name of a workstation. The next command deletes Active Directory objects with Python:

```
ADComputer.from_cn ("WS-500").delete()
```



```
C:\Users\Administrator\Desktop>python worker.py
<ADGroup 'CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=krbtgt,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Domain Computers,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Domain Controllers,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Schema Admins,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Enterprise Admins,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Cert Publishers,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Domain Admins,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Domain Users,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Domain Guests,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Group Policy Creator Owners,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=RAS and IAS Servers,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Allowed RODC Password Replication Group,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Denied RODC Password Replication Group,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Read-only Domain Controllers,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Enterprise Read-only Domain Controllers,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Cloneable Domain Controllers,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Protected Users,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Key Admins,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Enterprise Key Admins,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=DnsAdmins,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=DnsUpdateProxy,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Administrator,CN=Users,DC=test,DC=tamoggemon,DC=com'>
<ADGroup 'CN=Guest,CN=Users,DC=test,DC=tamoggemon,DC=com'>
```

Figure 4: PyAD finds a list of Active Directory groups.

in this case, a workstation.

Conclusions

The PyAD Python tool facilitates interaction with Active Directory. With the appropriate know-how, you can integrate AD configurations in Python workflows. However, because PyAD is no longer under active development, you need to consider carefully whether you want to start using it. If you have not used Python scripts thus far, and if you exclusively want to use Python for AD automation, it makes sense to check out the alternatives. However, for experienced Python script writers, PyAD is a useful, customized administration tool.
Because of space limitations, I couldn't cover the PyAD library completely. If you want to learn more, the documentation [7] is a big help, and the source code is available on GitHub [1]. The Stack Overflow [8] website is also very helpful: A search for the "PyAD" tag returns dozens of questions on the topic, along with the answers. ■

Info

[1] PyAD: [https://github.com/zakird/pyad]

[2] Python for Windows: [https://www.python.org/downloads/windows/]

[3] ADUser in PyAD: [https://zakird.github.io/pyad/objects.html#aduser]

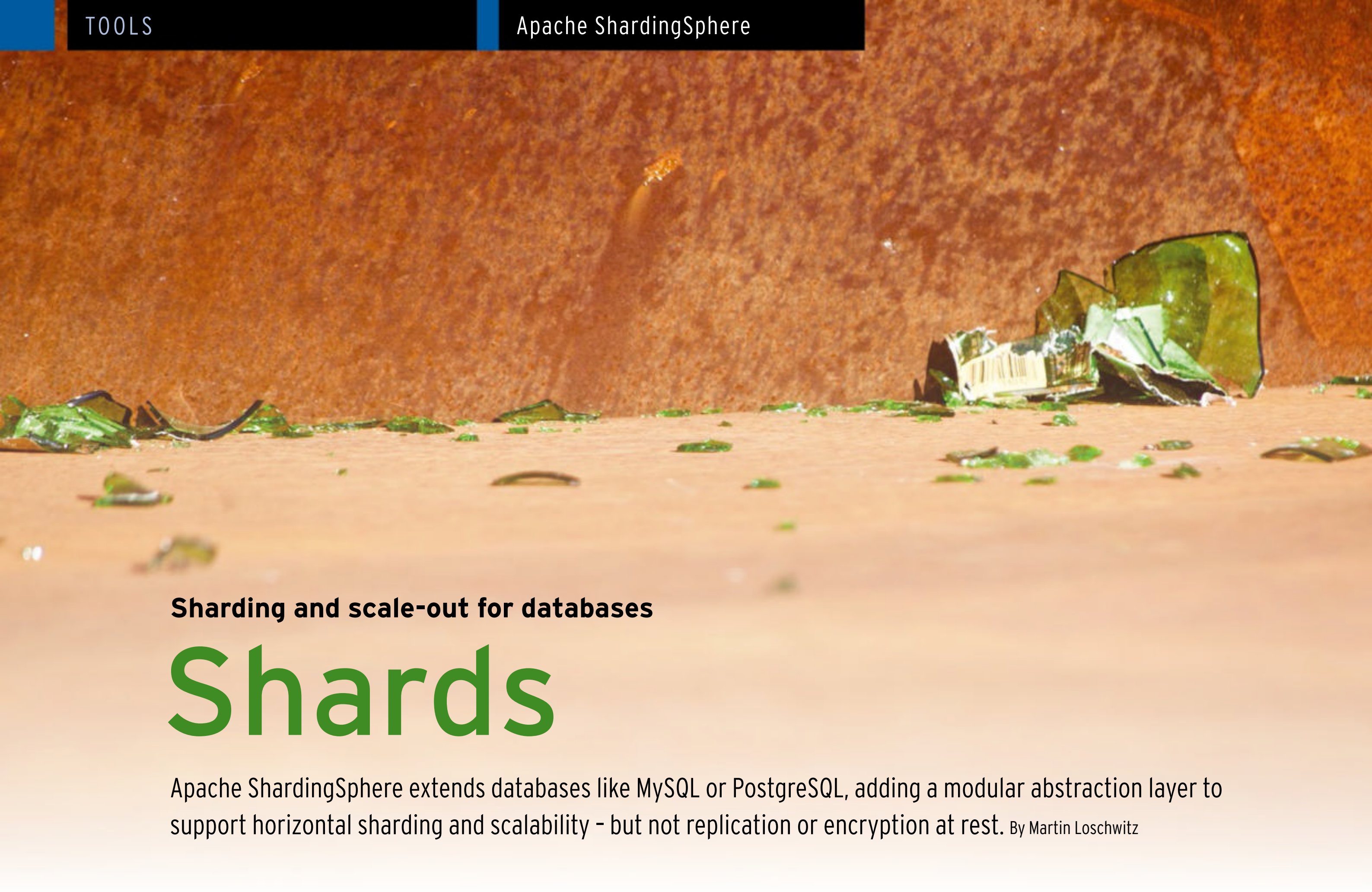
[4] Convert Python PyWin time types: [https://stackoverflow.com/questions/39028290/python-convert-pywintypes-datetime-to-datetime-datetime]

[5] Maximum number of return values: [https://stackoverflow.com/questions/56854376/python-active-directory-module-PyAD-max-vreturned-group-members]

[6] Code for return values: [https://stackoverflow.com/questions/60153136/python-3-PyAD-get-members-how-to-list-more-than-1500-members]

[7] PyAD documentation: [https://zakird.github.io/pyad/]

[8] Stack Overflow: [https://stackoverflow.com]



Sharding and scale-out for databases

Shards

Apache ShardingSphere extends databases like MySQL or PostgreSQL, adding a modular abstraction layer to support horizontal sharding and scalability – but not replication or encryption at rest. By Martin Loschwitz

Scalable databases have mushroomed in recent years and, in some cases, rely on completely new architectural approaches. For example, YugabyteDB is a key-value store, but it offers a MySQL compatibility layer. Vitess, on the other hand, uses native MySQL databases in the background but inserts an abstraction layer between the user and the database management system (DBMS) that takes care of sharding and, in turn, horizontal scalability. The Apache project takes a similar approach with its ShardingSphere [1] variants but promises easy handling, seamless extensibility with plugins, and support for most common databases.

Redundancy and Scalability

Databases are a central component of most complex setups. True to the motto "a special tool for every task," a long-established practice is to let databases take care of data management because they solve the task best and most efficiently. The changes that have occurred since the advent of cloud computing – containers and the cloud-ready principle – also lead to tougher requirements for databases. In a scalable environment, the database also needs to be able to scale; single instances of MySQL or PostgreSQL are no longer sufficient.
Out-of-the-box DBMSs come with quite a few limitations that make them unsuitable for operation in cloud environments. Two problems are immediately apparent: First, modern environments assume that the services running in them are implicitly redundant. Typically – and this is especially true in microarchitecture services – any number of instances of a service are available for each task, monitoring each other and seamlessly taking over the tasks of a failed instance in an emergency. Anyone who has ever faced the challenge of achieving high availability for MySQL will be aware that, out of the box, MySQL does not come with any functions for this task. The same is true for PostgreSQL.
The second problem arises from the need for scalability. Monolithic databases of the past, such as MySQL or PostgreSQL, are not designed for seamless horizontal scaling. Instead, the principle is that exactly one central instance of the service is present, where the capacity of the single server can be expanded, at most.
In the cloud context, this arrangement is a problem primarily because it is diametrically opposed to the requirements described for cloud-ready applications.

Sharding

Before discovering how the Apache project addresses these problems with ShardingSphere, a review of terminology is needed, especially with regard to the concept of "sharding." You might be familiar with sharding from the early days of IT, although it referred to a completely different technical issue back then.
The journey goes way back into the past, to a time when the capacity of hard disks was still measured in gigabytes, not terabytes. Even then, the operators of large mail servers regularly had the problem of running out of local space for messages. At that time, the term "sharding" primarily meant the distribution of user mailboxes to different machines, which continued to appear to the outside world as one logical mail server. In principle, sharding in ShardingSphere is no different, except it involves databases, not email.
In the database context, sharding means breaking down a database namespace into its logical elements and distributing them to different databases in the background. DBMS

Photo by Dan Dennis on Unsplash

sharding is therefore ultimately just an abstraction layer that provides a uniform view to the outside world and, when access occurs, knows to which of the available instances it needs to forward a client's request. Sharding supports additional features in this way, such as replicating individual parts of the namespace (or shards) between different database instances in the background. Deduplication, parallel read-only or read-write access, and encryption during data transfer (on the fly) and when storing data (at rest) can also be implemented in the abstraction layer.

Architecture

Saying that ShardingSphere is a solution is not completely correct. It comprises two components with different feature sets that can be extended with plugins. The reason is historical: ShardingSphere was initially developed as a Java Database Connectivity (JDBC) module (Figure 1).

JDBC describes a driver environment for accessing databases with Java and offers the huge advantages of modularity and the ability to stack and combine modules within the JDBC environment by connecting them in series. The motivation for ShardingSphere was to create a flexible intermediate layer for the dominant databases on the market at the time (e.g., MySQL, PostgreSQL) for sharding, encryption, redundancy, and availability without having to make major modifications to the database itself. Instead, it was enough to define a pool of database back ends and let ShardingSphere do the rest. This scenario is probably where ShardingSphere is used most frequently.

Shortly after its inception in 2016, ShardingSphere-JDBC caused quite a stir. Soon people were looking for a way to use its functionality outside of JDBC, which saw the birth of the second ShardingSphere variant, simply known today as ShardingSphere-Proxy (Figure 2). It does essentially the same as the JDBC variant but comes as a standalone service. In the background, ShardingSphere-Proxy manages a pool of connections to database back ends, much like JDBC, while maintaining protocol compatibility with MySQL and PostgreSQL. Client-side databases connect to the Proxy instead of directly to the database.

The advantage of ShardingSphere is that by managing and controlling the connection between client and server, it can implement all kinds of practical features without requiring a special configuration client-side or on the server. One of the main architectural principles of ShardingSphere is that database clients must always be able to talk to a DBMS in its SQL dialect through the service without errors and without any special customization. Therefore, ShardingSphere always remains transparent from both the client's and the server's point of view.

The functional range of ShardingSphere is quite impressive. The linchpin of all features is, as described, the ability to break down a database into small logical segments – or shards, if you like. The developers emphasize that they can scale both the storage of data and any compute tasks horizontally: The problem with a database often is not that the individual instance does not have enough local space, but rather that it collapses under the load of incoming requests because resources such as CPU and RAM are finite.

Sharding, as implemented by ShardingSphere, avoids this problem, because the back ends that are currently holding the respective shards are still responsible for processing the queries

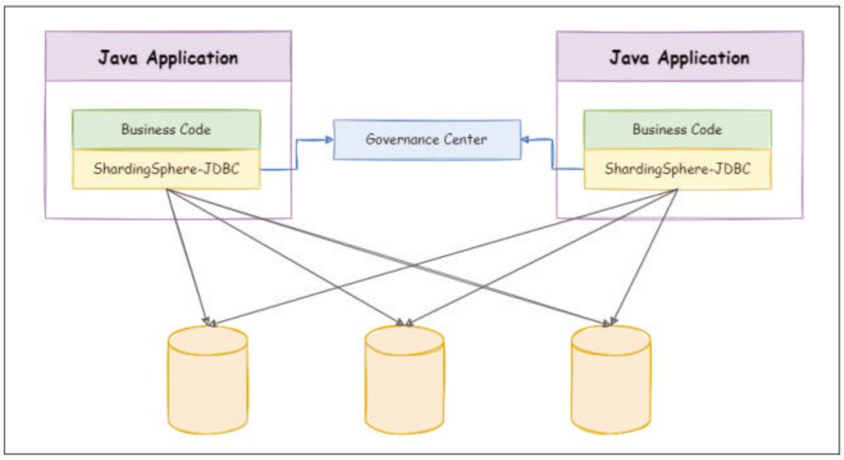


Figure 1: By design, the ShardingSphere-JDBC application is designed for use in Java environments. It is transparent for applications and clients. © ShardingSphere

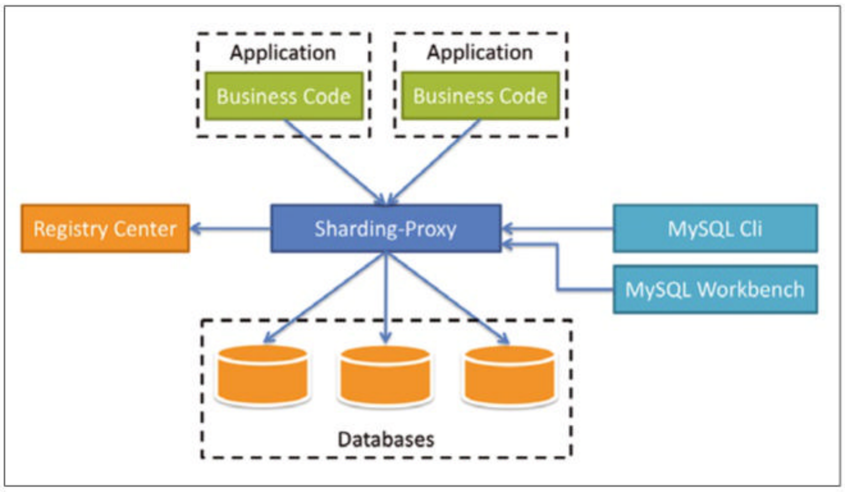


Figure 2: ShardingSphere-Proxy provides functions similar to the JDBC implementation but does not require the Java interface and can therefore be used more generically. © ShardingSphere

relating to the individual database shards. If two large queries access different data, they are handled by different back ends and do not affect the entire database.

Encryption and More

ShardingSphere does not exhibit any weaknesses in any other functional area. The project claims that it is significantly faster than comparable competitor solutions, which the developers have proven with benchmarks [2]. The ShardingSphere team outlined how it achieved near-native performance of the underlying databases on the basis of its solution. ShardingSphere's performance overhead is minimal.

Even with ShardingSphere-Proxy, which according to the developers is significantly slower than the JDBC option, the performance overhead is now 25 percent less (Figure 3). This performance may well be considered a special feature: The Vitess developers, for example, state that performance can drop by up to 50 percent compared to a native MySQL database with their software. However, this is partly to do with the issue of replication, which the article will go into in detail later.

ShardingSphere contains an encryption layer that can be dynamically activated as a plugin. It encrypts the data in transit. If you need encryption at rest, you have to take care of this yourself with one of the solutions available for MySQL or PostgreSQL, according to the terse note from the developers.

No Replication

If you only want to create a distributed database, ShardingSphere leaves out a central aspect that comparable solutions handle: replication. Databases usually take care of this feature when scaling horizontally: It is a requirement that the market quite simply imposes on cloud-native and cloud-ready applications. Adding a high-availability layer is obvious when implementing sharding, because then, from a database point of view, it is possible to copy a shard to multiple back ends so that any one of them can step in if another fails.

At this point, the description sounds more trivial than its technical implementation. After all, if you replicate a database, you need to offer guarantees such as atomicity, consistency, isolation, and durability (ACID). Moreover, replication is not very popular precisely because compliance with consistency guarantees usually forces synchronous replication, which in turn consumes performance. It is quite possible that the people at ShardingSphere would be unable to maintain their otherwise fantastic performance values if replication were activated.

The answer to the replication issue is a bit surprising, because ShardingSphere has decided without further ado to ignore the issue almost completely. Instead, the documentation succinctly states that you will want to implement replication at the level of the database back end. Although this solution gives you high-availability (HA) functionality, ShardingSphere is not responsible for compliance. At the end of the day, you end up building countless HA clusters from MySQL or PostgreSQL pairs and then feeding these to ShardingSphere as back ends. However, if you have ever manually tried to make MySQL highly available, you will inevitably have dealt with tools such as Pacemaker or distributed replicated block devices (DRBDs). Pacemaker in particular is not only extremely complex to use but also is definitely not highly available.

The ShardingSphere developers understand that their user story has a hole in it in terms of redundancy. A lot of information online offers guidance about what a valid HA setup can look like with ShardingSphere. However, some guides refer to plugins for ShardingSphere that no longer exist or that rely on hosted solutions, such as database as a service (DBaaS) from AWS to offload the management of the database instances to the provider.

Of course, this is only a valid user story in environments where suitable as-a-service offerings exist at all, which is precisely what private cloud environments, such as those based on OpenStack, often do not offer. The fact that ShardingSphere today has a granular integration with Kubernetes and that all ShardingSphere services can be run as a Kubernetes service does not help, because integration for the underlying application is virtually nonexistent.

The Proxy introduces another peculiarity in the ShardingSphere context:

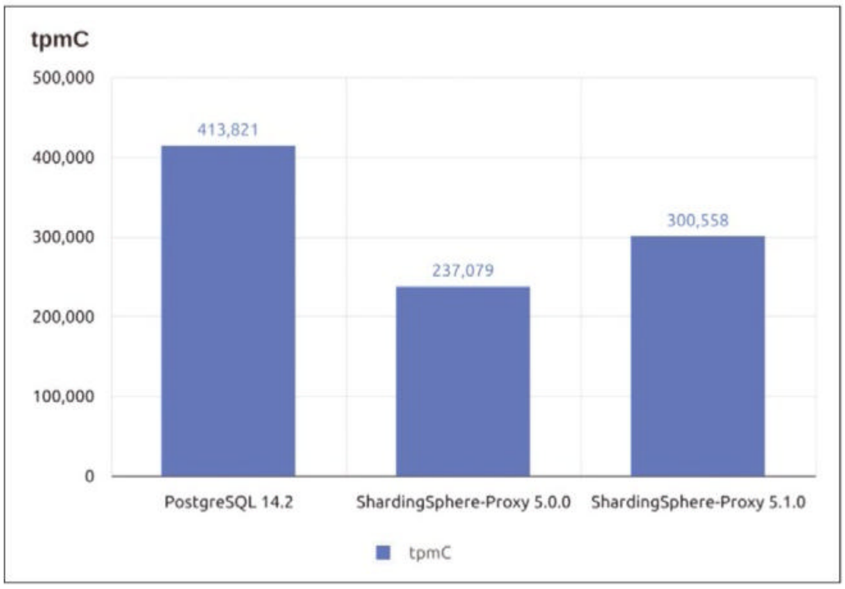


Figure 3: Even with ShardingSphere-Proxy, the slower of the two variants, the performance overhead for PostgreSQL is still well under 25 percent (tpmC = number of transactions that can be fully processed per minute). © ShardingSphere

In the documentation, the developers regularly refer to a cluster mode, but you should not be misled by this reference. Cluster mode in ShardingSphere is simply an operating mode that groups the various database instances in the background. Most administrators probably associate the term "cluster" with high availability, but it is completely missing in ShardingSphere.

Eye Candy

Not to be left unmentioned at this point is the ShardingSphere user interface (UI), which reveals almost in passing elementary weaknesses of the product documentation and the developer community of ShardingSphere. On paper, the ShardingSphere UI is a Vue.js-based graphical interface for the database distributor (Figure 4). On the one hand, it enables administrative tasks such as setting the ShardingSphere configuration parameters. On the other hand, it is intended to help you understand the structure of the data currently stored in ShardingSphere, to identify the distribution and to analyze the back ends currently in use.

It's a pity that the ShardingSphere documentation for the UI says virtually nothing, and although the Git repository contains the code, it only offers a few lines of advice for the install. The rest is left to you to figure out on your own. This task is nontrivial, especially with respect to connecting the UI to a running ShardingSphere instance or cluster. If you don't have any experience with the solution, in the worst case you won't understand the configuration syntax and will spend hours experimenting. Fatally, the UI documentation is not the only missing part. Time and time again you come across passages in the documentation that you can only understand if you have already worked with the JDBC or gained previous experience with the Proxy. Despite several quickstart guides, life is anything but easy for ShardingSphere newcomers, because the guides are often only links to GitHub directories with sample code, distributed over a multitude of files. How you get from bare metal to a running ShardingSphere instance is not revealed by these documents.

What About Competition?

As I mentioned earlier, ShardingSphere is a comprehensive solution for dragging databases into the present day and adding cloud-ready capabilities and scalability. How does the tool fare compared with its competitors? The question is not so easy to answer, precisely because there are now a large number of complex solutions in the market segment that ShardingSphere addresses. However, their underpinnings differ from those of ShardingSphere – in some cases considerably.

The closest thing to ShardingSphere is Vitess [3], which implements its own MySQL-specific sharding with a similar component structure. Again, it does not implement its own storage but accesses MySQL instances in the back end and then uses them to distribute its own logical database namespace. Unlike ShardingSphere, however, Vitess specializes in MySQL and completely lacks support for PostgreSQL. Other solutions such as YugabyteDB [4] do offer this support. YugabyteDB operates as a classic key-value store under the hood but exposes its structures to the outside world with a PostgreSQL compatibility layer created specifically for this purpose. In the worst case, this setup will cause an application to fail if it tries to use a PostgreSQL feature that the YugabyteDB replica cannot handle. ShardingSphere takes a smarter approach because it uses actual PostgreSQL or MySQL databases in the background. Anyone looking for an add-on solution for distributed databases will want to include ShardingSphere in their evaluation.

However, ShardingSphere does not immediately cover the issue of implicit high availability. The competitors are far more advanced in this case: Vitess, for example, can replicate the individual shards of a node to other nodes and seamlessly exchange each of these during operation. The same is true for the key-value database at the heart of YugabyteDB.

Future Outlook

The ShardingSphere developers consider their work far from complete and are already working on a third flavor of their product: ShardingSphere-Sidecar (Figure 5). Anyone who works in the container and Kubernetes environment will already have some idea of where this product is headed. Sidecar is said to offer ShardingSphere capabilities while coming across as a cloud-native service and integrating seamlessly with container fleets.

Sidecar works in close cooperation with the Proxy, which is essential in a Sidecar setup that uses ShardingSphere. Thanks to an integrated mesh

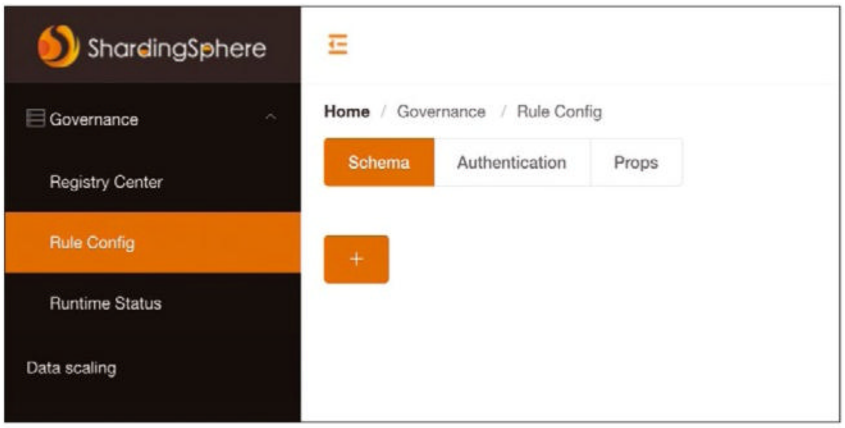


Figure 4: The ShardingSphere UI is intended to make it far easier to understand the stored data and cluster functionality, but it is noticeably under-documented. © ShardingSphere

functionality, the individual applications no longer communicate directly with the Sharding Proxy, but with a local instance of a Sharding Mesh Sidecar. The active Mesh Sidecars, in turn, forward the data to ShardingSphere's Sharding Sidecars, which ultimately communicate with the database back ends.

At the time of writing, ShardingSphere-Sidecar was not released for production; you will have to make do with alternatives for now, which could mean, for example, combining the ShardingSphere-Proxy server with a mesh (e.g., the Istio service mesh).

Conclusions

ShardingSphere is a comprehensive and powerful tool for upgrading conventional-style databases for today's scalable IT world. Unlike its competitor YugabyteDB, for example, it does not try to reinvent the wheel: ShardingSphere lets applications talk to a real MySQL database instead of a translation layer.

However, the individual ShardingSphere components noticeably are not fully feature-compatible with each other. That the JDBC implementation was the first is noticeable by the fact that it still offers the most features. However, this does not demote ShardingSphere-Proxy to a second-class component. From the developer's or administrator's point of view, it is important to choose carefully between the variants. If JDBC is available within the scope of a project anyway, this option is a good choice.

Either way, if you want to look beyond ready-made boxed solutions for scalable databases, ShardingSphere should be on your list of solutions to evaluate, as should Vitess, which takes a similar approach for MySQL. ■

Info

[1] Apache ShardingSphere: [https://shardingsphere.apache.org]

[2] Li, R., L. Zhang, J. Pan, J. Liu, P. Wang, N. Sun, S. Wang, C. Chen, F. Gu, and S. Guo. Apache ShardingSphere: A Holistic and Pluggable Platform for Data Sharding: VIII. Evaluations. *In: Proceedings of 2022 IEEE 38th International Conference on Data Engineering (ICDE)* (IEEE, May 2022), pp. 2468-2480: [http://www.kangry.net/paper/ICDE2022_SS.pdf]

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

[4] YugabyteDB: [https://www.yugabyte.com]

The Author

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

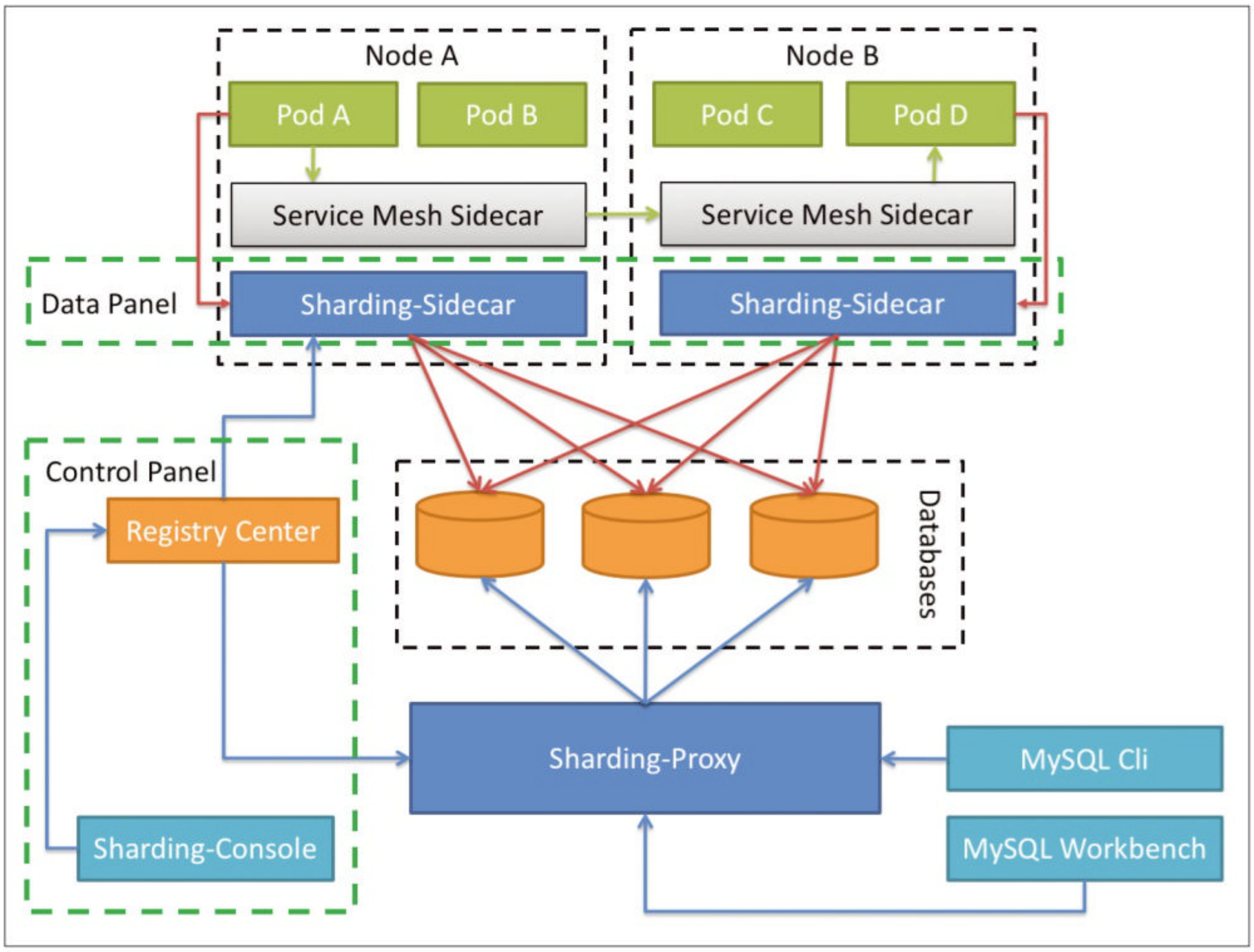


Figure 5: The JDBC driver and Proxy will soon be joined by ShardingSphere-Sidecar, which is optimized for operating the solution in clouds. However, a production-ready version was not available when this magazine went to press. © ShardingSphere

Sponsored by

IEEE SA OPEN · arm · avanade · GitHub · Google
ODI · FerretDB · Microsoft · PERCONA · SUSE · CATAPULT Digital

Table Holders

APACHE · arm · avanade · CHAOSS · CODING BLACK FEMALES · controlplane · CATAPULT Digital · ECLIPSE FOUNDATION · eBPF · EDB · FerretDB · FINOS · flox · FreeBSD FOUNDATION
GitHub · GitLab · GNOME · IEEE SA OPEN · KDE · matrix · mautic · Microsoft · ODI · OS
OpenTofu · OpenUK · PERCONA · Foundation for Public Code · SIGNIFICANT GRAVITAS · SUSE · Red Hat · Ubuntu · weaveworks

Where does job output go?

Data Depot

Where does your job data go? The answer is fairly straightforward, but I add some color by throwing in a little high-level background about what resource managers are doing and evolve the question to include a discussion of where data "should" or "could" go. By Jeff Layton

The second question I need to answer from the top three storage questions [1] a friend sent me is "How do you know where data is located after a job is finished?" This is an excellent question that HPC users who use a resource manager (job scheduler) should contemplate. The question is straightforward to answer, but the question also opens a broader, perhaps philosophical question: Where "should" or "could" your data be located when running a job (application)?

To answer the question with a little background, I'll start with the idea of a "job." Assume you run a job with a resource manager such as Slurm. You create a script that runs your job – this script is generically referred to as a "job script" – and submit the job script to the resource manager with a simple command, creating a "job." The job is then added to the job queue controlled by the resource manager. Your job script can define the resources you need; set up the environment; execute commands, including defining environment variables; execute the application(s); and so on. When the job finishes or the time allowed is exceeded, the job stops and releases the resources.

As resources change in the system (e.g., nodes become available), the resource manager checks the resource requirements of the job, along with any internal rules that have been defined about job priorities, and determines which job to run next. Many times, the rule is simply to run the jobs in the order they were submitted – first in, first out (FIFO).

When you submit your job script to the "queue," creating the job, the resource manager holds the details of the job script and a few other items, such as details of the submit command that was used. After creating the job, you don't have to stay logged in to the system. The resource manager runs the job (job script) on your behalf when the resources you requested are available and it's your turn to run a job. In general, these are jobs that don't require any user interaction; they are simply run and the resources are released when it's done. However, you can create interactive jobs in which the job script sets up an interactive environment and returns a prompt to a node from the resource manager. When this happens, you can execute whatever commands or scripts or binaries you want. Today, one common interactive application is a Jupyter Notebook. When you exit the node or your allocated time runs out, the job is done and the resources are returned to the resource manager. If you choose to run interactive jobs, be sure to save your data often; otherwise, when the resource manager takes back the resources, you could lose some work.

Because the job, regardless of its type, is executed for you, it is important that you tell the resource manager exactly what resources you need. For example, how many nodes do you need? How many cores per node do you want? Do you need any GPUs or special resources? How much time do you think it will take to complete the job? Equally important is explaining in your job script any special options pertaining to how the job script should be run and the location of the input and output data that the resource manager creates.

Photo by Charles Forerunner on Unsplash

If your job script or your application doesn't specify data locations, then Slurm will accept all input data to be in the directory where you created the job, along with all the output. You will see this referred to as `pwd` (present working directory), which refers to the directory in which you submitted the job script. (Slurm captures the `pwd` information.) Of course, the application can specify specific paths to the input data and the paths where it will create the output data. These paths can be hard coded in the application; many times, the application will read environment variables that tell it where to perform the I/O.

In general, if the job script or the application doesn't specify the location of the input and output, then all the data will appear in the `pwd`.

Almost always, the resource manager captures the stdout and stderr output and puts in into a resource-manager-defined file. For example, in Slurm this would be something like `slurm-.out`. But the resource managers let you change the name of this output file in the job script or on the command line when you submit the job script.

I don't want to dive into the writing of job scripts, but because the resource manager creates an output file for you, or lets you define the name of the file, you can include all kinds of detail about the job in this output file, including dates and times, paths, nodes used, the states of nodes (e.g., how much memory is available), and so on. This information can help document the job details so you can scan through the files to look for specific details. For example, when I first started using a resource manager, one of the engineers on the team created a common definition that we included in the job script to capture the details on the problem examined. It made life so much easier when scanning for details about the simulations.

Overall, that's it. All data will be read and written to your `pwd` when you submit your job script to the resource manager. The application or job script can specify other directories with needed data, if you like. If you don't specify where the input data resides or the name of the output, it will show up as `slurm-.out` in the `pwd` where you submitted your job script.

Unless told otherwise, users will almost always submit their jobs from their `/home` directory, meaning all the input and output data will be located there. As users are added, as projects grow, and as the applications require more input data, this arrangement might be untenable because of the capacity and performance limits of `/home`. The cluster admin can impose quotas that help control the growth of data and suggest compressing data files to save space. In the end, capacity is finite.

Additionally, storage has a fixed performance limit, and the more users sharing that resource, the more the effective performance that each user can get goes down, possibly reducing application performance. At some point cluster admins and users start wondering whether there isn't a better way to organize their data, application data, and so on.

Where to Run Jobs

The obvious answer to needing more storage capacity and possibly performance is simply to buy more. Your storage solution can add capacity and performance easily, right? However, notice that this is a single tier of storage; that is, all the storage has the same performance for all the data.

If you need more performance, you just buy a faster storage solution or add more storage hardware (e.g., more storage servers). Sometimes this can be done without adding much (any?) additional capacity and almost always means that the cost per gigabyte increases, so the overall cost for the same amount of space goes up. I've never seen storage solution costs go down with faster performance.

At some point, you will have to balance the almost insatiable appetite for more storage capacity and the need for faster storage to run applications effectively. What HPC data centers have done to help address this problem is to examine what is happening with the data.

Usually, the results of the examination are twofold:

- The "age" distribution of the user data skews older; that is, a significant portion of the data is older and hasn't been touched or used for a long time, compared with other data that is actively used.
- Many applications don't need extreme storage performance to run effectively. They can run using slower performance storage with little effect on applications.

From these observations, questions start popping up. Does the old data that hasn't been used in a long time need access to storage with very high performance? Why not put this data on slower storage that has a much greater capacity and has a much lower cost per gigabyte? It seems logical to create different pools of storage that each have very different characteristics.

This approach is called "tiering" storage, in which a small amount of really fast storage with a large cost per gigabyte is purchased. The savings come because you don't need to buy a huge amount. Instead, you buy a great deal more storage that has much less performance but also has a much smaller cost per gigabyte, allowing for much more capacity. This slower storage has the tongue-in-cheek title of "cheap and deep" (low-cost and lots of it).

Compared with the original single tier of storage where you had to balance capacity and cost so that neither requirement was really satisfied, this new approach of tiered storage – with a much smaller amount of storage that has much greater performance than before, along with huge amounts of "cheap and deep" storage that is much less expensive and has massive capacity – can produce a much better overall solution satisfying, to some extent, the two extremes: performance and capacity.

Of course, tiering introduces complications. You have to worry about copying data from one of the slower tiers to the fastest tier, run the applications, and copy the data back to the slower tier; that is, you save the capacity of the fastest tier for jobs that are actively running or soon will run applications. To perform this data migration, users must learn about the commands and tools needed for the manual intervention, which, by virtue of being manual, is prone to mistakes.

Combining the storage tiers into one seemingly unified storage space or using "automatic" tools to migrate data has not had much success because the processes that users develop to perform their work are so diverse that no single tool is sufficient. Creating a new type of filesystem with tiers is a very nontrivial task. Therefore, most sites stick with manual data migration by users, but they really focus on training and developing any sort of tool to make the task easier.

Another complication that arises with manual data migration is that the admin needs to police the fast-tier storage, so users don't forget about any data sitting there that isn't being actively used.

You can have as many tiers as you like. Many larger HPC systems have perhaps three tiers (Figure 1). The top tier has the fastest performance with the smallest capacity (greatest $/GB). The second tier holds the `/home` directories for users, with less performance than the top tier but a much larger capacity (smaller $/GB than the top tier). The third tier has very little performance but has massive capacity (smallest $/GB).

Storage Tiering with Local Node Storage

When HPC clusters first became popular, each node used a single disk to hold the operating system (OS) that was just large enough for the OS and anything else needed locally on the node. These drives could be a few gigabytes in capacity, perhaps 4GB, or about double that capacity. Linux distributions, especially those used in HPC, did not require much space. Consequently, the unused capacity in these drives was quite large.

This leftover space could be used for local I/O on the node, or you could use this extra space in each node to create a distributed filesystem (e.g., a parallel virtual filesystem, PVFS [2], which is now known as OrangeFS [3]) that is part of the kernel. Applications could then use this shared space to, perhaps, access better performing storage to improve performance.

Quite a few distributed applications, primarily the message passing interface (MPI) [4], only had one process – the rank 0 process – perform I/O. Consequently, a shared filesystem wasn't needed, and I/O could happen locally on one node.

For many applications, local node storage provided enough I/O performance that the overall application runtime was not really affected, allowing applications that were sensitive to I/O performance to use the presumably faster shared storage. Of course, when using the local node storage, the user had to be cognizant of having to move the files onto and off of the local storage to something more permanent.

Over time, drives kept getting larger and the cost per gigabyte dropped rapidly. Naturally, experiments were tried that put several drives in the local node and used a hardware-based RAID controller to combine them as needed. This approach

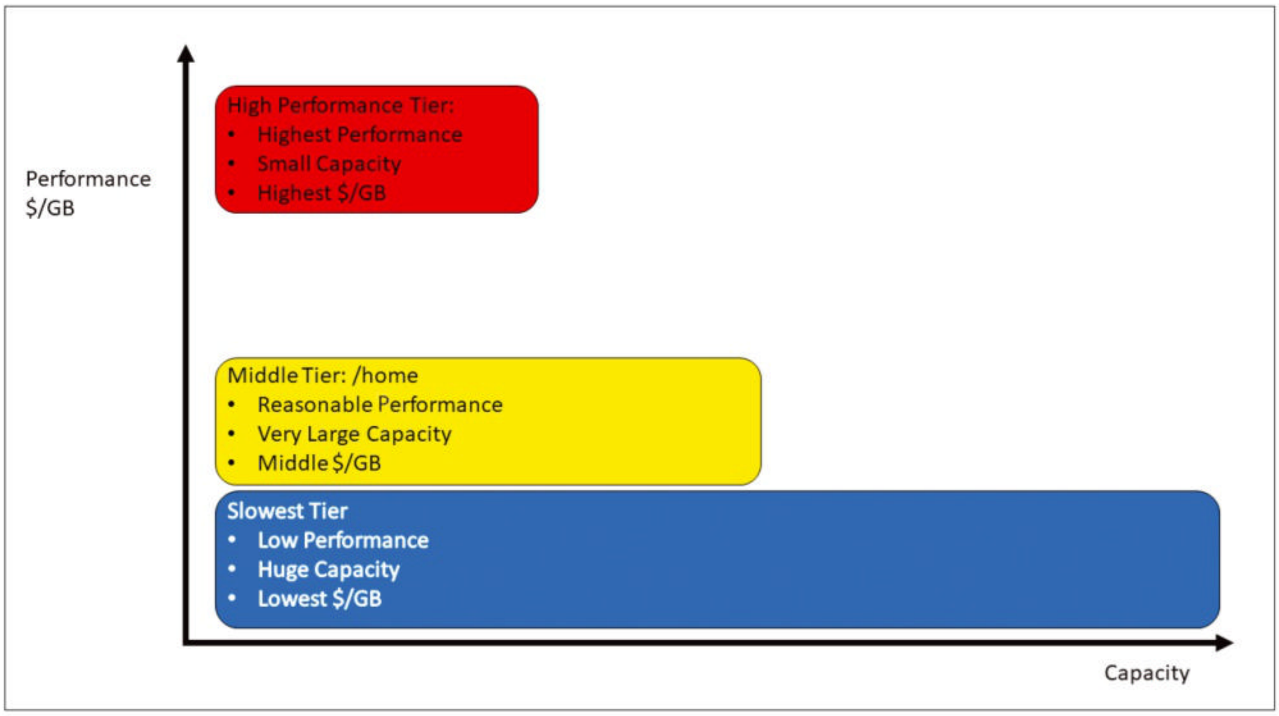


Figure 1: Three tiers of storage are approximately placed in the chart according to their performance. The width of the block is relative to the capacity of that tier. Travel up the y-axis for increased performance and cost per gigabyte. Go right along the x-axis for increased capacity.

allowed applications with somewhat demanding I/O performance requirements to continue to use local storage and worked well, but for a few drawbacks:

- The cost of the hardware RAID card and extra drives could notably add to the node cost.
- The performance of the expansion slot that held the RAID card could limit storage performance, or the controller's capability could limit performance.
- Hard drives failed more often than desired, forcing the need for keeping spares on hand and thus adding to the cost.

As a result, many times only a small subset of nodes had RAID-enabled local storage.

As time progressed, I noticed three trends. The first is that CPUs gained more and more cores and more performance, allowing some of the cores to be used for RAID computations (so-called software RAID) so that hardware-based RAID cards were not needed, saving money and improving performance.

The second trend was systems with much more memory than ever before could be used in a variety of ways (e.g., for caching filesystem data, although at some risk). In the extreme, you could even create a temporary RAM disk – system memory made to look like a storage device – that could be used for storage.

The third trend is that the world now had solid-state drives (SSDs) that were inexpensive enough to put into compute nodes. These drives had much greater performance than hard drives, making them very appealing. Initially the drives were fairly expensive, but over time costs dropped. For example, today you can get consumer-grade 1-2TB SSD drives for well under $100, and I've seen some 1TB SSDs for under $30 (August 2023). Don't forget that SSDs can range from thin 2.5-inch form factors to little NVMe drives that are only a few millimeters thick and very short. The M.2 SSDs are approximately 22mm wide and 60-80mm long. It became obvious that putting a fair number of SSDs that are very fast, very small, and really inexpensive in every node is an easily achievable goal, but what about the economics of this configuration?

With SSD prices dropping and capacities increasing, the cost per gigabyte has dropped rapidly, as well. At the same time, the cost of CPUs and accelerators, such as GPUs, has been increasing rapidly. These computational components of a node now account for the vast majority of the total node cost. Adding a few more SSDs to a node does not appreciably affect the total cost of a node, so why not put in a few more SSDs and use software RAID?

Today you see compute nodes with 4-6 (or more) SSD drives that are combined by software RAID with anything from RAID 0 to RAID 5 or RAID 6. These nodes now have substantial performance in the tens of gigabytes per second range and a capacity pushing 20TB or more. Many (most?) MPI applications still do I/O with the rank 0 process, and this amount of local storage with fantastic performance just begs for users to run their code on local node storage.

Local storage can now be considered an alternative or, at the very least, a partner to the very fast, expensive, top-tier storage solution. Parallel filesystems can be faster than local node storage with SSDs, but the cost, particularly the cost per gigabyte, can be large enough to restrict the total capacity significantly. If applications

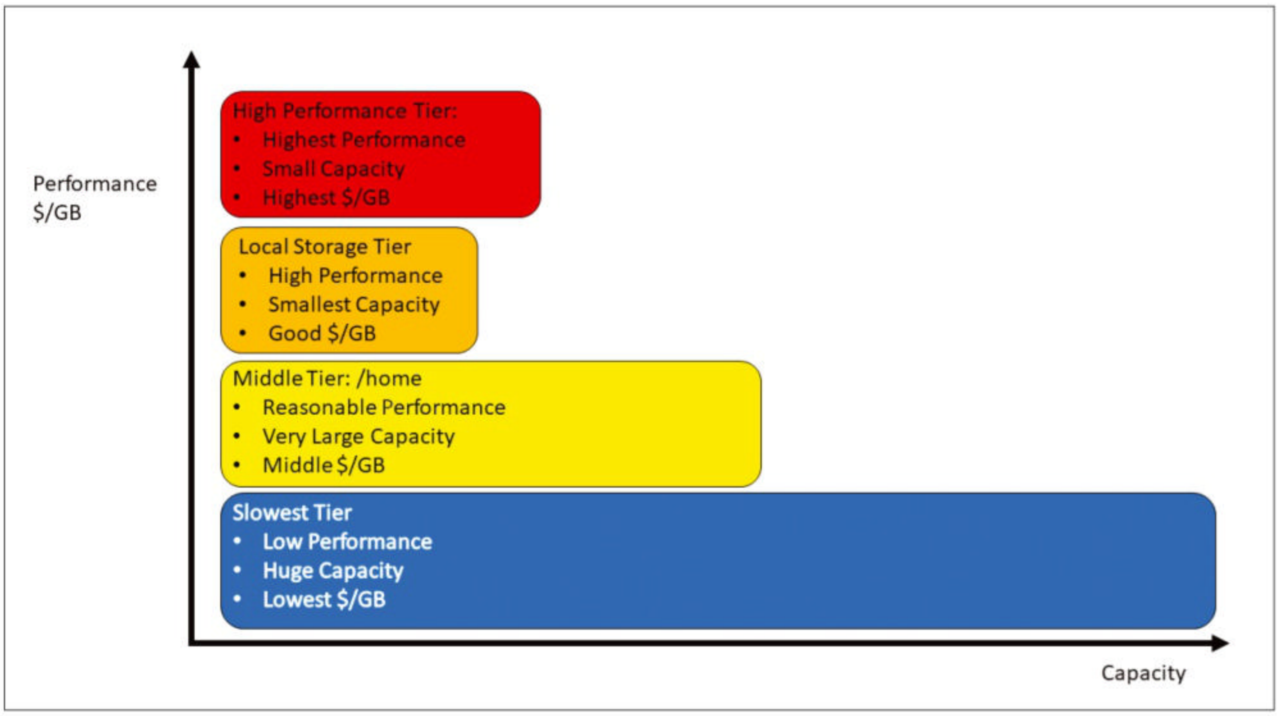


Figure 2: Storage tiering with local storage added.

don't need superfast storage performance, local storage offers really good performance that is not shared with other users and is less expensive. As before, you have freed the shared really fast storage for applications that need performance better than the local node.

Figure 2 is the same chart as in Figure 1, but local storage has been added. Note that local storage isn't quite as fast as the high-performance tier, but it is faster than the middle tier. The local storage tier has a better (lower) cost per gigabyte than the highest performance tier, but it is a bit higher than the middle tier. Overall the local storage tier has less capacity than the other tiers, but that is really the storage per node. Nonetheless, applications can't access the local storage in other nodes.

Users still have to contend with moving their data to and from the compute nodes' local storage. (Note that you will always have a user who forgets to copy their data to and copy their data back from the compute node. This is the definition of humanity, but you can't let this stop the use of this very fast local storage.) Users have had to contend with storage tiers, so adding an option that operationally is the same as before will only result in a little disturbance.

The crux of this section? Buy as much SSD capacity and performance for each compute node as you can, and run your applications on local storage, unless the application requires distributed I/O or massive performance that local nodes cannot provide. The phrase "as you can" generally means to add local storage until the price per node noticeably increases. The point where this occurs depends on your system and application I/O needs.

Summary

The discussion of where the data should or could go started with a little history of shared storage for clusters and then sprang into a discussion of storage tiering. Storage tiers are all the rage in the cluster world. They are very effective at reducing costs while still providing great performance for those applications that need it and high capacities for storing data that isn't used very often but can't be erased (or the user doesn't want to erase it).

With the rise of computational accelerators and their corresponding applications, using storage that is local to the compute node has become popular. Putting several SSDs into an accelerated server with software RAID does not appreciably increase the node cost. Moreover, this local storage has some pretty amazing performance that fits into a niche between the highest and middle performance tiers. Many applications can use this storage to their advantage, perhaps reducing the capacity and performance requirements for the highest performance tier. ■

Info

[1] Storage Questions: [https://www.admin-magazine.com/HPC/Articles/Getting-Data-Into-and-Out-of-the-Cluster]

[2] PVFS: [https://en.wikipedia.org/wiki/Parallel_Virtual_File_System]

[3] OrangeFS: [https://www.orangefs.org/]

[4] MPI: [https://en.wikipedia.org/wiki/Message_Passing_Interface]

The Author

Jeff Layton has been in the HPC business for over 30 years (starting when he was 4 years old). When he's not grappling with a stubborn systemd script, he's looking for deals for his home cluster. His twitter handle is @JeffdotLayton.

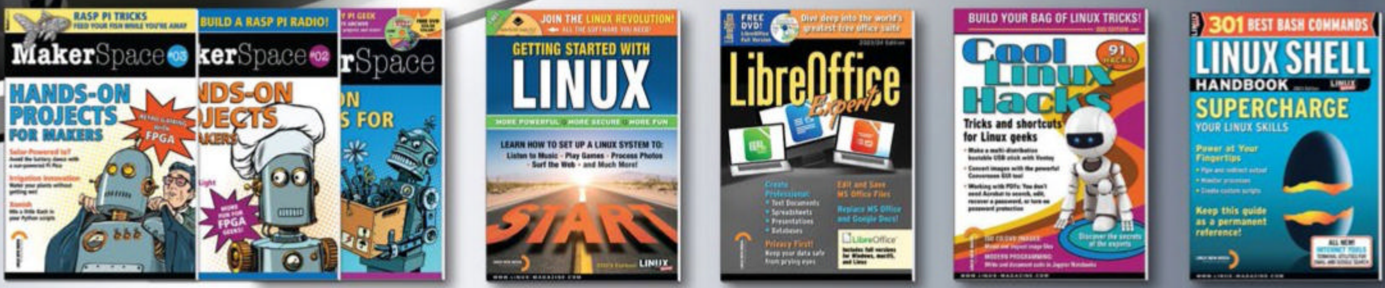
■

Hone Your Skills with Special Issues!

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

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

Available in print or digital format



Check out the full library!

shop.linuxnewmedia.com

Lead Image © cobalt, 123RF.com

Low-code development with Microsoft Power Apps

Special Delivery

If the IT staff is having trouble keeping up with the demand for custom applications, end users can pitch in with low-code programming tools like Microsoft Power Apps. By Joydip Kanjilal

Low-code and no-code methodologies allow users to build applications with no or minimal coding skills. These approaches leverage declarative programming, abstracting the challenges of conventional programming. The biggest difference between low-code and no-code platforms is that low-code provides the option to add code manually, if necessary, whereas no-code platforms abstract the code entirely. Low-code tools typically offer visual interfaces, pre-built components, and access to code libraries, allowing developers to create applications by easily dragging and dropping elements and connecting data sources. Some popular low-code platforms include Appian, Mendix, OutSystems, and Microsoft Power Apps [1].

The goal of no-code platforms is to allow the non-technical user to create basic applications or automate processes without support from IT or development staff. No-code tools make extensive use of visual modeling and pre-built templates. Examples of no-code platforms are: Airtable, Bubble, Glide, Webflow, and Zapier.

Skillful use of low-code and no-code programming can improve time to market, increase agility, enhance collaboration, and reduce costs. However, as you might expect, low-code and no-code techniques are not right for every project. In some scenarios, low-code and no-code can limit flexibility, reduce scalability, inhibit integration with legacy systems, and promote vendor lock-in. Typical uses for low-code methods include handling complex processes, system integration, and custom logic. No-code platforms are often used for prototyping, automating repetitive tasks, and building small applications.

Table 1: Low-Code vs. No-Code

Feature	Low-Code	No-Code
Primary user	Developers	Business users
Support for end-to-end development?	Yes	No
Purpose	Provides support for Rapid Application Development (RAD)	Provides the necessary interfaces and tools to design and build apps without the need for coding skills
Support for customization	Pre-built templates to build custom applications	Support for complete customization
Application complexity	Can be used to build complex applications	Can be used to build simple applications

Lead Image © Peapop, Fotolia.com

Table 1 shows some of the key differences between low-code and no-code platforms.

Building a Low-Code Application

Microsoft Power Apps (Figure 1) is a low-code development platform for building web and mobile applications. Microsoft calls Power Apps "a suite of apps, services, and connectors, as well as a data platform, that provides a rapid development environment to build custom apps for your business needs." Power Apps is a good example of a low-code platform that lets you roll out a custom application in just a few steps.

You can build an app for free by signing in to the Power Apps website. You'll need a license to use the apps you create. Microsoft offers a 30-day trial, and pricing thereafter starts at $20 for a single user, with packages for larger groups and bundles containing additional services. Business professionals often use Power Apps to deploy applications for employee onboarding, supply chain management, and customer relationship management.

One powerful feature of Power Apps is its support for application templates, which you can use to build an app in only a few steps. For instance, to create a simple expense report application, click *Start with an app template* in the Power Apps home screen

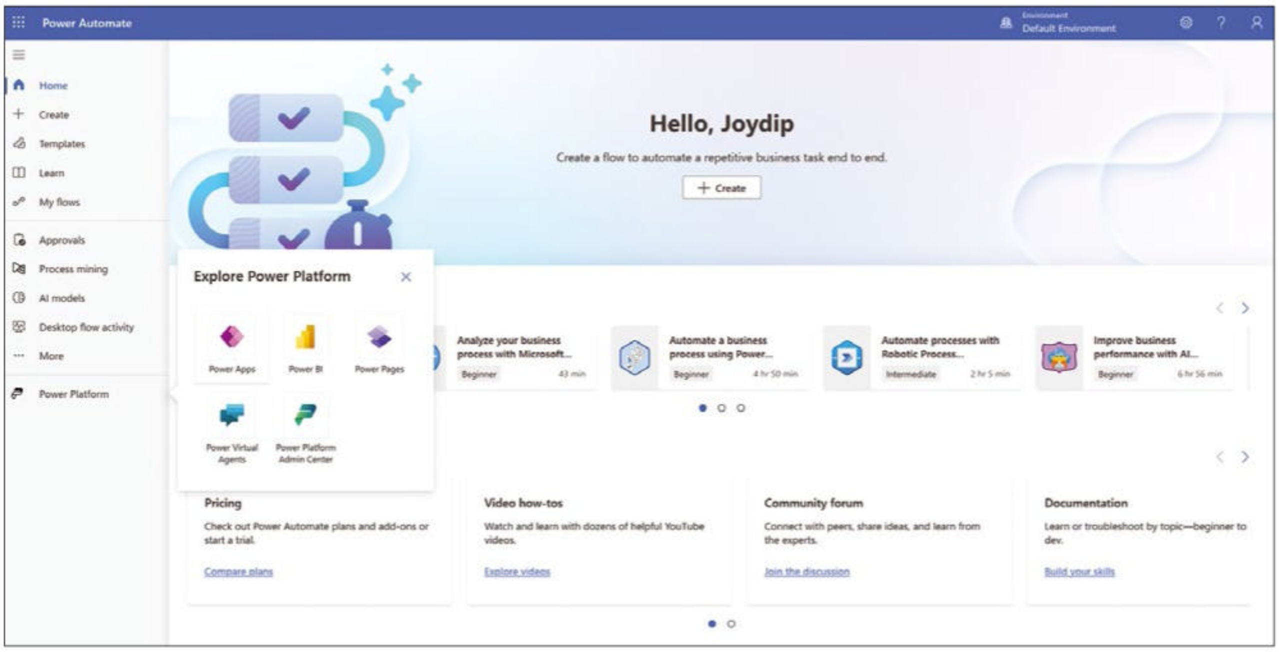


Figure 1: Microsoft Power Automate in action. Power Apps helps build applications that connect to several data sources, such as Office 365, SharePoint, SQL Server, Microsoft Azure, JIRA, OneDrive, and Power BI.

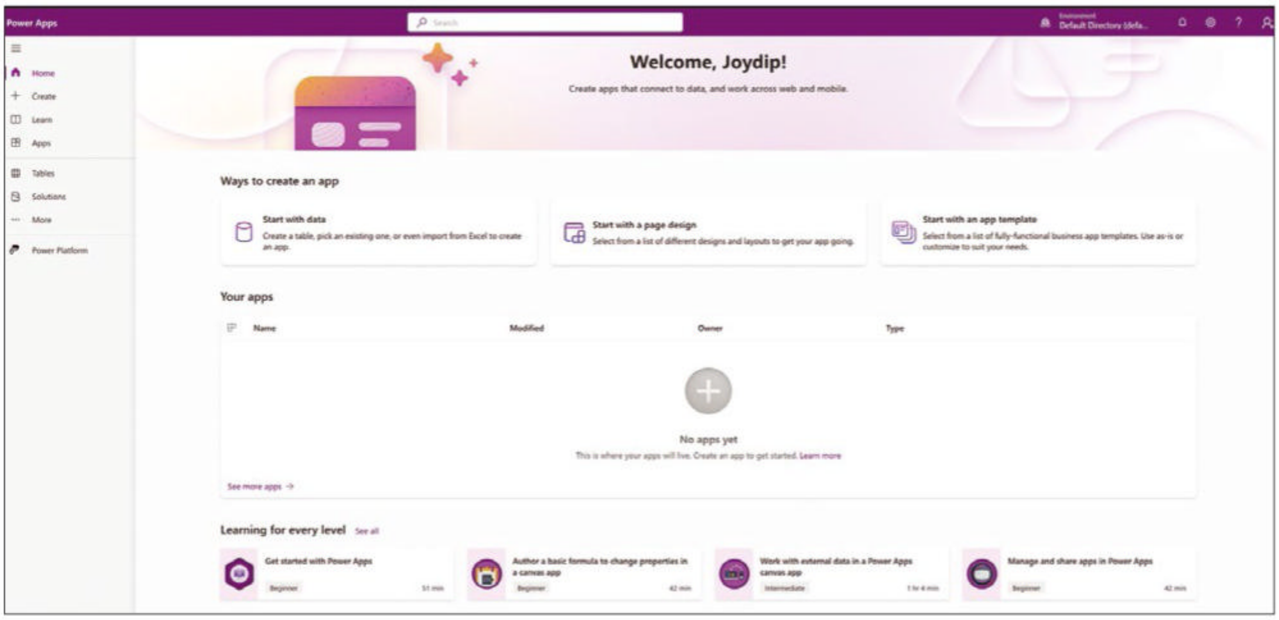


Figure 2: The PowerApps home screen.

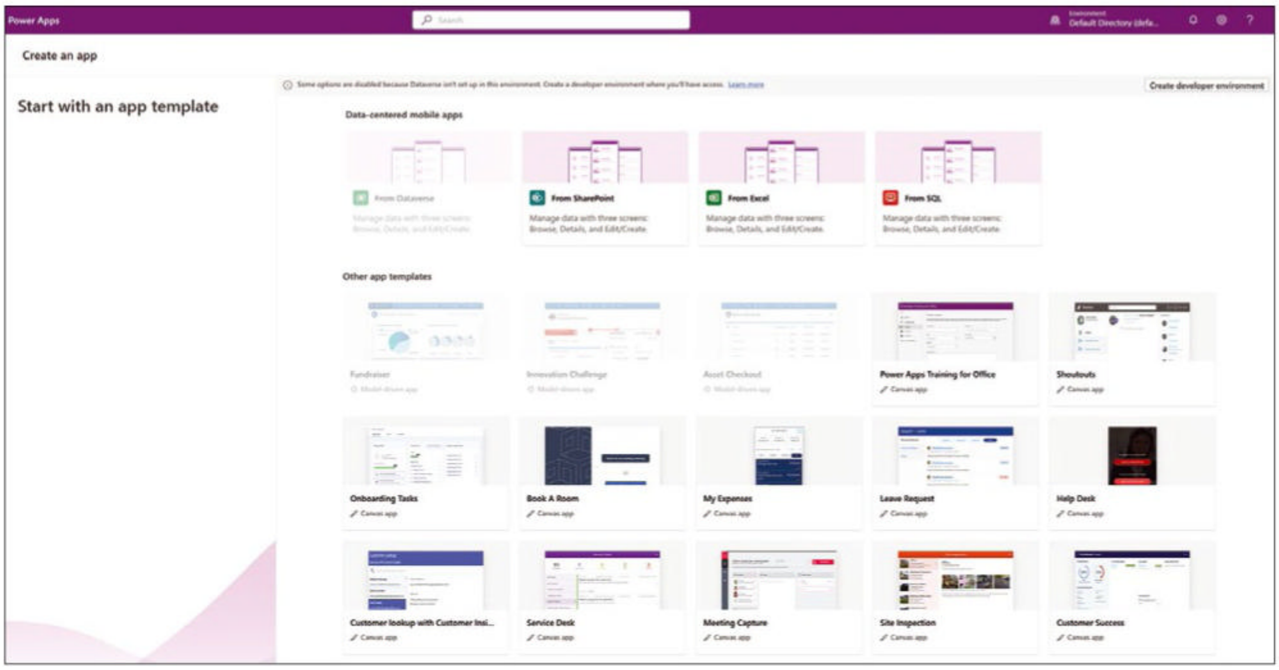

Power Apps
Search
Create an app
Start with an app template
Some options are disabled because Dataverse isn't set up in this environment. Create a developer environment where you'll have access. Learn more
Create developer environment
Data-centered mobile apps
From Dataverse
From SharePoint
From Excel
From SQL
Manage data with three screens: Browse, Details, and Edit/Create
Other app templates
Fundraiser
Innovation Challenge
Asset Checkout
Power Apps Training for Office
Shoutouts
Onboarding Tasks
Book A Room
My Expenses
Leave Request
Help Desk
Customer lookup with Customer Insi...
Service Desk
Meeting Capture
Site Inspection
Customer Success
Canvas app
Model-driven app

Figure 3: The My Expenses template.

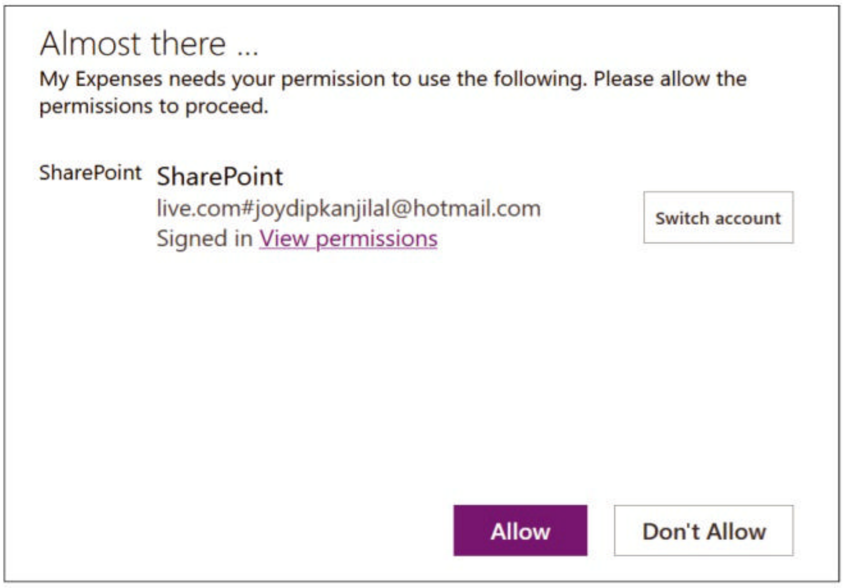
(**Figure 2**) and select the *My Expenses* app (**Figure 3**). Next, choose a name for the app and click *Next*.

When prompted for permission to use SharePoint, click *Allow* (**Figure 4**); then, click on the tree view in the left pane of the Power App window (**Figure 5**). As you can see in the figure, the tree view easily lets you select options for building the expense report. You can create new screens and add components to the report.

Now click on *NewExpenseCreateButton* to create a button for the interface (**Figure 6**). Select the *Advanced* tab in the Properties window (on the right) to view the code, change the properties for the button control, or add some custom code of your own.

Conclusion

Low-code and no-code technologies make it easier for non-programmers to create their own applications, allowing professional IT personnel to focus on matters that still require their expertise. For many businesses, adopting a low-code or no-code approach can save effort, time, and money. Microsoft Power Apps is a low-code platform that makes extensive use of templates, allowing users to build easy business apps that would take much longer to create by conventional techniques. ■


Almost there ...

My Expenses needs your permission to use the following. Please allow the permissions to proceed.

SharePoint SharePoint
live.com#joydipkanjilal@hotmail.com
Signed in View permissions

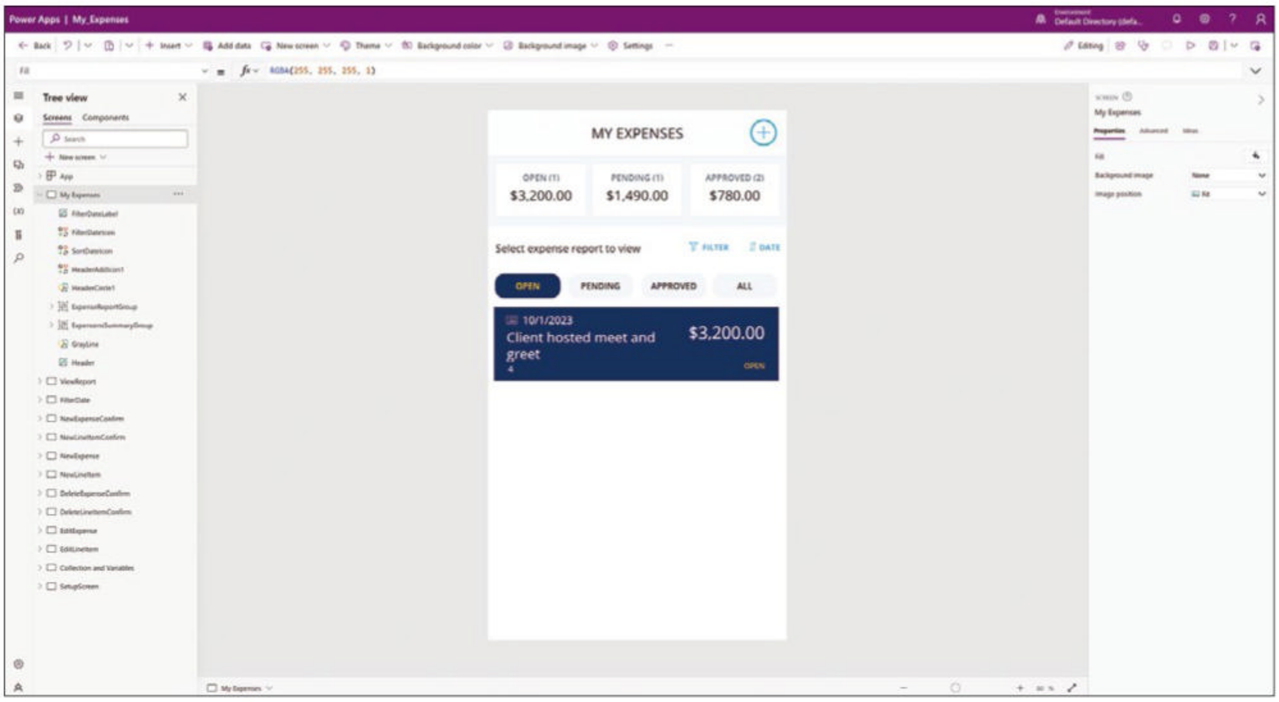
Figure 4: Prompt for permission to use SharePoint.

Info

[1] Microsoft Power Apps: [https://powerapps.microsoft.com/en-us/]

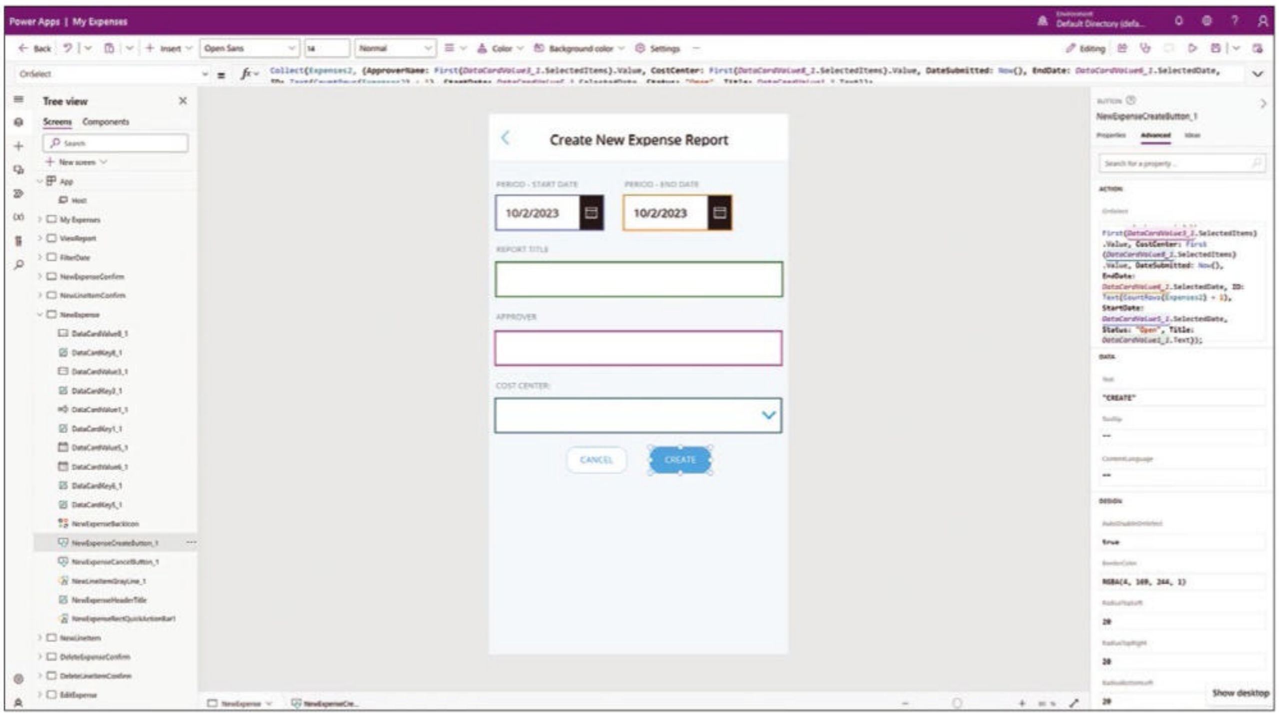
Author

Joydip Kanjilal ([https://joydipkanjilal.com/]) is a Microsoft Most Valuable Professional in ASP.NET (2007-2012), speaker, and author of several books and articles. He has more than 25 years of experience in IT with more than 20 years in Microsoft .NET and its related technologies. He has been a community credit winner at [http://www.community-credit.com] several times. His technical strengths include: C#, Microsoft .NET, ASP.NET Core, ASP.NET Core MVC, Azure, AWS, Microservices, Serverless Architecture, Kubernetes, Kafka, RabbitMQ, REST, SOA, Design Patterns, SQL Server, Oracle, Machine Learning, and Data Science.



```
RGBA(255, 255, 255, 1)
```

Figure 5: The completed Expense Report application.



```
Collect(Expenses2, {ApproverName: First(DataCardValue3_2.SelectedItems).Value, CostCenter: First(DataCardValue8_2.SelectedItems).Value, DateSubmitted: Now(), EndDate: DataCardValue6_2.SelectedDate, ID: Text(CountRows(Expenses2) + 1), StartDate: DataCardValue5_2.SelectedDate, Status: "Open", Title: DataCardValue1_2.Text});
```

Figure 6: The properties of the Expense Report application.

Native serverless computing in Kubernetes

Go Native

Knative transfers serverless workloads to Kubernetes and provides all the container components you need to build serverless applications and PaaS and FaaS services. By Martin Loschwitz

If you check out the undisputed star of the container scene and its serverless capabilities, you might be somewhat disappointed to find that Kubernetes (K8s) really does not shine out of the box. Ultimately, Kubernetes sees itself as an orchestrator for container workloads across the boundaries of individual compute nodes. The keyword is "orchestrator." Kubernetes looks to manage existing containers and sees its strengths precisely there. Creating matching workloads is something it tends not to consider its job. At times, this perspective has devastating consequences from the user's point of view.

When you look at the K8s landscape from the Cloud Native Computing Foundation (CNCF), you will find countless solutions from which you are expected somehow to put together your own setup, including many continuous integration/continuous delivery (CI/CD) tools that let admins create serverless workloads for Kubernetes. That said, managing the external resources for serverless operation that are somehow grafted onto Kubernetes often turns out to be a bumpy road. Knative is the name of the project and the tool that aims to change just that.

The "K" in Knative stands – you guessed it – for Kubernetes, so it's about native integration of workloads into Kubernetes. Like many components from the K8s universe, Knative is not particularly open or flexible for newcomers, but people in the function-as-a-service (FaaS) or platform-as-a-service (PaaS) universes will benefit from the advantages of running serverless workloads with Knative. In this article, I introduce you to Knative and explain the most important terms and features.

The Beginning

If you haven't had much to do with serverless computing or Knative, you're likely to find yourself sitting in front of the Knative documentation fairly confused. Some of it reads like it was written by a marketing department – serverless here, cloud native there. The buzz will more likely worry than instill confidence in experienced administrators of classic workloads.

What you need are a few clearly defined terms before setting off into the Knative adventure. Like other solutions, Knative suffers greatly from terms like "serverless" or "cloud native" being thrown around everywhere and all the time with either no definition or everyone simply ignoring it. At the beginning of this article, I already defined what serverless is all about: being able to access services such as databases or load balancers without having to go down the classic route of a virtual instance with the installed service and matching configuration.

What sounds like nitpicking from the professional admin's point of view is hugely important for developers. The advantage to developers is huge if they can click together a database with a standardized API and not have to worry about its maintenance afterward. The main idea behind serverless computing is to hide factors such as hardware and operating systems,

Photo by Adam Bixby on Unsplash

as well as their maintenance, from an environment's users. Instead, these tasks are handled by the platform operators, who build their own tools for this purpose and maintain specific processes for the tasks.

In the Knative context, though, a few more technical terms need clarification. These terms are particularly interesting for administrators who do not currently work with Kubernetes on a daily basis. They are directly related to how Kubernetes manages resources and how Knative is integrated. One of these terms is the custom resource definition (CRD). Kubernetes itself is known to be controlled by an administrator or developer through a REST API. Commands to Kubernetes always go from the client to the server and use the HTTP format.

Kubernetes handles objects internally, and resources are an object type. In Kubernetes, you have containers, pods, virtual IP addresses, and other facilities out of the box. Custom resource definitions are a bridge to external applications that let developers and admins create their own resource types in Kubernetes, which they can then use and manage later just like resources that are included in Kubernetes out of the box. Knative makes almost excessive use of CRDs, extending an existing K8s cluster to include a large number of CRDs that establish Knative-specific resources at the K8s cluster level.

Knative uses CRDs as a point to dock onto the container orchestrator, which is absolutely essential: It is the only way that Knative can deliver on its own promise that running serverless applications is running first-class citizens in Kubernetes. In the cloud native world, "first-class citizens" refers to resources that can be fully controlled by the existing API of the orchestrator or fleet manager.

Two Features

If you search online with the Knative keyword, you will regularly come across older documentation and presentations that describe three of Knative's core components: Serving, Eventing, and Building. Some caution is advised here. Today, the Building component is no longer a part of Knative. Instead, it is being continued as an independent project named Tekton Pipelines [1], partly with contributions from other developers. Knative has returned to its roots, so to speak, and now just comprises Serving and Eventing. Tekton is discussed in more detail later. For the moment, however, the focus is on Knative; for starters, understanding it is complicated enough. To look under the Knative hood, you should visualize the basic tasks that Knative is designed to handle from the developer's point of view.

At the top of all considerations is, as usual, the concept of the app. If you are operating serverless applications in Knative, the ultimate aim is to roll out and operate a specific application automatically, without external intervention. However, an app can comprise multiple services, which is why Knative uses the term "services" internally, although it means a function rather than a specific service. A service is a central object in a K8s cluster extended by Knative. All other resources to be created in Kubernetes depend on this object to ensure that a service can reliably handle the task expected of it. In MariaDB, for example, the service would be the active `mysqld` instance – not because it is a single process, but because this service provides the central function of the database. Several processes might well be required to perform a function as part of an overall structure.

Several resources are usually attached to a service, typically including at least one route and a service-specific configuration. The route is a reference to the element of a virtual network through which the actual function can be reached. In addition to services, routes, and configurations, Knative has a fourth resource type, revision, which denotes a specific configuration consisting of a service, a route, and a configuration and can occur any number of times per serverless service. A revision is a kind of static mapping to the combination of the three previously mentioned factors.

If you think about the four aspects that form the Serving component (Figure 1), a coherent picture quickly emerges. The term "services" describes logical Knative functions in Kubernetes that take care of managing the entire lifecycle of an application. In other words, the services component creates all the services required for a virtual setup, provides them with a valid route and a valid configuration, and creates a revision for them in its own database. How the parts needed for a service come together is decided by the administrator or developer as part of the configuration. You can use Knative to store configurations and define routes or, instead, tell Knative to find the best possible route to access a network element itself.

Here, Knative also indirectly competes with other solutions such as Istio and the like, which use Kubernetes-native resources but then smuggle some of the tasks past Kubernetes.

Events Count, Too

The other major Knative subcomponent is less about what and how and far more about when, and once again goes by a technical term – event-driven architecture – which in information technology refers to an architectural approach in software development that allows an event A to be followed by a response B. Put simply, an event-based architecture in Kubernetes is based on the principle that changes to resources automatically trigger other actions. These actions are recognized as events to which a higher level instance responds. In the example here, the parent instance is Knative Eventing.

A separate standard named CloudEvents, under the auspices of the CNCF, governs the way events are sent in the cloud native context.

A functional Eventing architecture requires a sender as well as

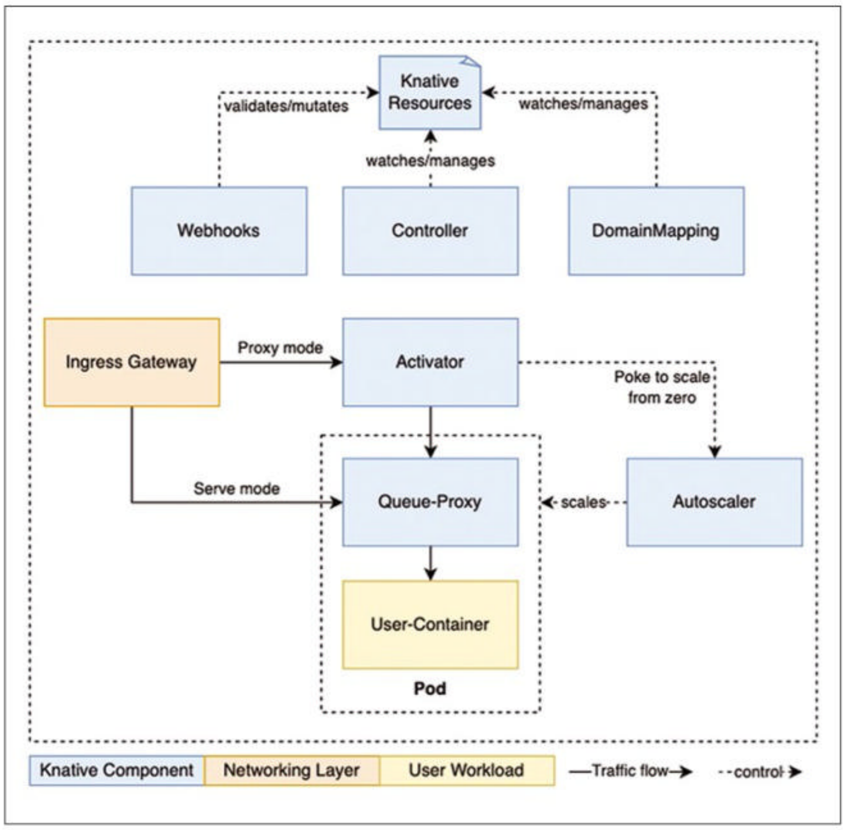


Figure 1: Knative services take a prebuilt container and roll it out along with all the required services. It also has its own infrastructure services, such as an ingress gateway. © Knative

a receiver for the event. In simple terms, the running setup needs to send a notification if its state changes somehow, and another component needs to receive the event and take action on it. In the context of event-based architectures, the entity that outputs events is also referred to as an agent, whereas the entity that receives the events and responds to them is also known as a sink. Again, Knative Eventing exclusively acts inside of Kubernetes and does not require external resources. Like Knative Serving, Eventing extends the K8s API to include several CRDs that can be used from within running programs.

Hands-On

If you are not at home in the cloud native world, you might not fully understand the central concepts of the solution. A few practical examples should help provide a better understanding of the ideas behind Knative as a whole. Conveniently, the Knative developers provide quite a few examples of various functions [2], so you do not have to search for them. One of these examples is a classic from the developer world: a web server that outputs *Hello World!* [3].

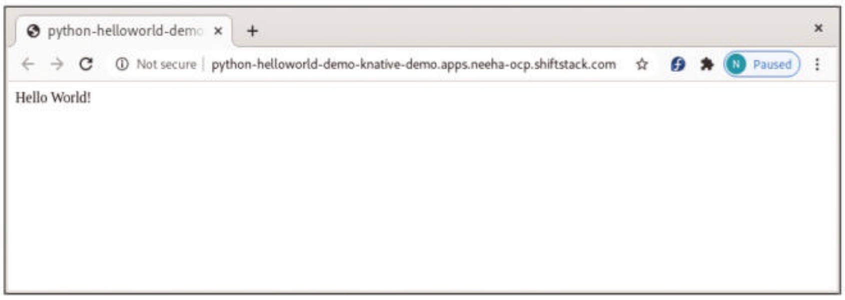
The example comprises two components and is written in Go. The `helloworld.go` file contains the service, which listens on port 80 and prints out *Hello World!* in an HTTP-compatible format (Figure 2) as soon as someone contacts the service. Far more important is the `service.yaml` service definition for Knative, because it is what makes the application palatable for Knative in the first place (Listing 1).

Anyone who has at least looked at code snippets for Kubernetes will quickly understand: Initially, the code sample uses the Knative `serving.knative.dev` resource to create a service named `helloworld-go` in the default namespace. The service specification states that the `helloworld-go` image from `docker.io` (i.e., from Docker Hub) runs the service. The `TARGET` environment variable containing `Go Sample v1` is also set up.

If you apply this file to Kubernetes with Knative, Knative launches the container in a pod through Kubernetes and makes it accessible from the outside. This is where it also becomes clear why Knative sees itself as a solution for operating serverless architectures: Many of the parameters you would have to specify in Kubernetes for a normal container without Knative support

Listing 1: service.yaml for Hello World!

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
  namespace: default
spec:
  template:
    spec:
      containers:
      - image: docker.io/{username}/helloworld-go
        env:
        - name: TARGET
          value: "Go Sample v1"
```


python-helloworld-demo
Not secure
python-helloworld-demo-knative-demo.apps.neeha-ocp.shiftstack.com
Paused

Hello World!

Figure 2: The *Hello World!* example only scratches the surface of what Knative can do. But the pod definition for the service is already far easier than a manual approach in Kubernetes. © Neeharika Kompala/GitConnected

are autonomously implemented by Knative without user intervention. You can then focus on your applications without having to worry about the details of Kubernetes or running containers in it, and it also applies to operational tasks. Out of the box, for example, Knative scales the pod up or down as a function of the incoming load so that an appropriate number of instances of the respective pod are running – or none at all. The scaling behavior can be completely predefined.
A second example shows how events can be intercepted and processed in Knative (Figure 3). The entire wealth of Knative functions is available, such as automatic horizontal scaling or specific routing between Kubernetes and a service, right through to the use of external routing services.
The standard example is slightly simpler. It is again based on Go and implements an HTTP service that opens a port and then waits on it for incoming messages. The implementation is for purely academic reasons and therefore is outside the box. If from this example you send the running service a name in the body of an HTTP request, you get back a *Hello <Name>!*, but – and this is the important bit – not because the service itself uses a function for the response. Instead, a request to the service triggers an event in the Knative API that passes the tool to a specially defined sink. In the example, the running application itself serves as the sink. It then responds on the code side by sending back a *Hello* with the matching HTTP body when a Knative event is received.
The key point here is the integration of the event agent and event sink into Kubernetes itself, which lacks the entire infrastructure for managing events – Knative provides this instead. The example from Listing 2 shows the complete implementation of the sample service.
Admittedly, these two examples do not come close to demonstrating the power offered by the Knative feature set. The project lists quite a few more examples in its documentation, including code for integration into Kubernetes and the application code. These examples give developers a better idea of the solution's capabilities. Specific examples of autoscaling applications can also be found.
Moreover, it is quite remarkable that Knative implements all the required functions itself on the basis of built-in K8s resources. Autoscaling, for example, exclusively relies on basic

Listing 2: service.yaml for Services

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: cloudevents-go
  namespace: default
spec:
  template:
    spec:
      containers:
      - image: ko://github.com/knative/docs/code-samples/
               serving/cloudevents/cloudevents-go
        - name: K_SINK
          value: http://default-broker.default.svc.
                 cluster.local
```

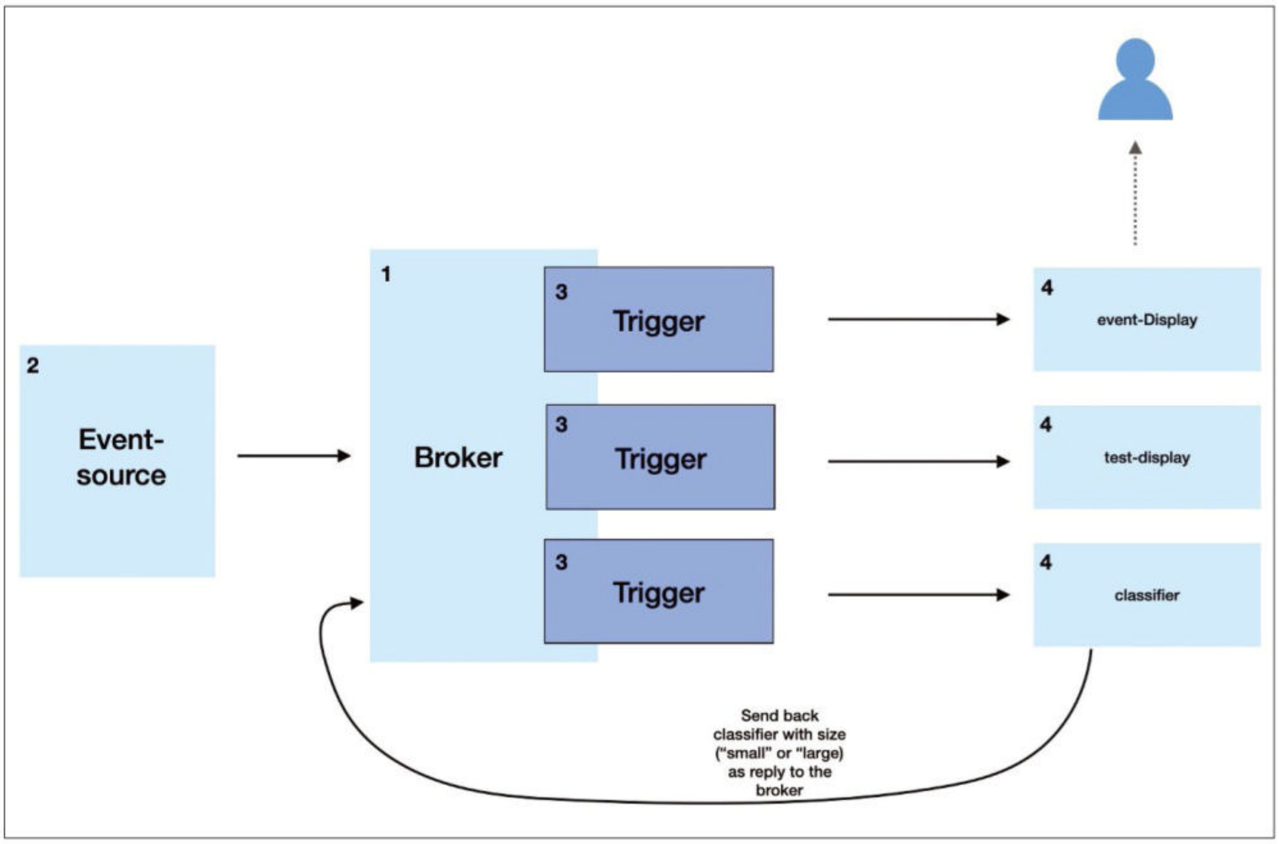


Figure 3: Knative Serving provides a broker service for events that receives incoming events as a sink and forwards them to defined targets, which means that developers can build a system of events and responses into their services. © Knative

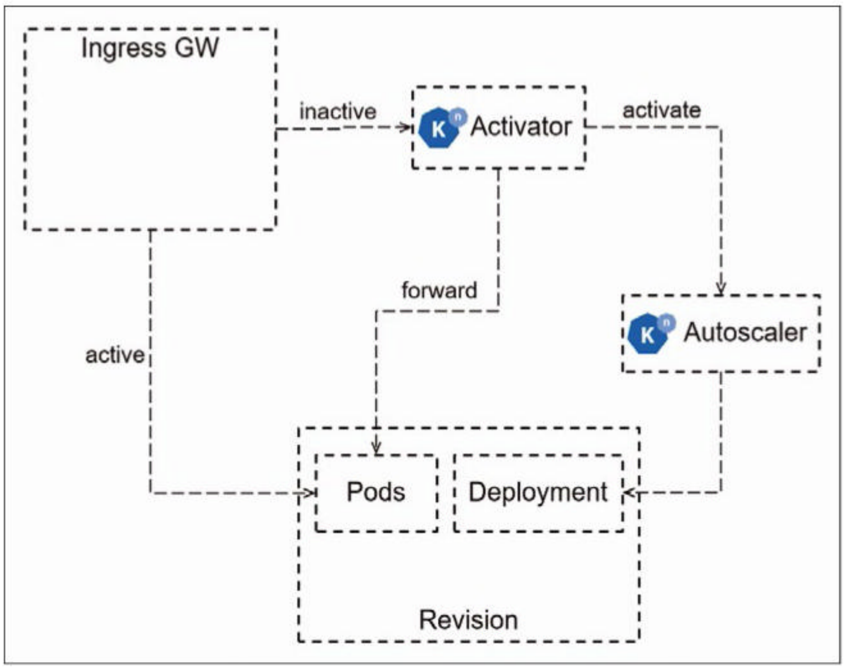


Figure 4: **The Activator plays a crucial role in Knative. It fields most of the incoming commands and processes them or forwards them to the appropriate Knative services.** © Knative

features that Kubernetes includes anyway. If you want to use Knative with external extensions (e.g., Istio) because you are already familiar with them, you will find instructions online that describe the steps. Knative is extremely powerful, but also very sociable with both internal and external solutions.

At this point, also remember that Knative launches quite a few services for its own operations in an active K8s cluster. The Activator plays the central role, receiving most of the requests from Knative CRDs and acting something like a central switchboard (Figure 4).

The Renegade Son

As mentioned at the beginning, Knative originally comprised three components: Serving, Eventing, and Building. The Building component, however, was something of an outsider from the very beginning. The Knative developers have always distinguished between operating applications and the build process, which is understandable from a logical point of view and completely correct if you think things out. Although the Serving and Eventing layers of Knative cannot be used without one another, the path and approach used to create the artifacts to be operated are basically irrelevant for Knative itself. Some time ago, the company got down to business and outsourced the Building component to a separate project; it has been operating as Tekton ever since.

However, to this day Tekton can't quite conceal its family ties to Knative. Under the hood, the component's architecture is similar to Knative's Serving and Eventing. Like Knative, Tekton extends an existing K8s cluster to include a number of CRDs – for creating and building applications in this case. Not to be outdone by its competitors in terms of marketing, Tekton developers now describe this integration as a CI/CD pipeline for cloud native environments.

The focus is still on serverless applications. Meanwhile, Tekton is also great to use to build other applications within Kubernetes. To do this, it relies on pipelines that it creates and configures in Kubernetes (Figure 5). Argo CD, for example, is a separate CI/CD system and does not have much to do with Tekton. However, Argo CD and Tekton can be teamed up to integrate artifacts created in Argo CD directly into Kubernetes.

If you then add Knative, you can perform party tricks, such as building an image on demand after a commit to a Git directory, with the image then being automatically rolled out to the production environment. The key factor that sets Tekton apart from quite a few competing products is the ability to build a container with a command to the K8s API. Unlike Argo CD, Tekton provisions all resources and infrastructure for upcoming build tasks independently and autonomously in Kubernetes. It also cleans up afterward, if desired.

Tekton closes a gap in this respect. When reading the paragraphs on Serving and Eventing, many experienced K8s admins may have been bothered that a ready-made Docker

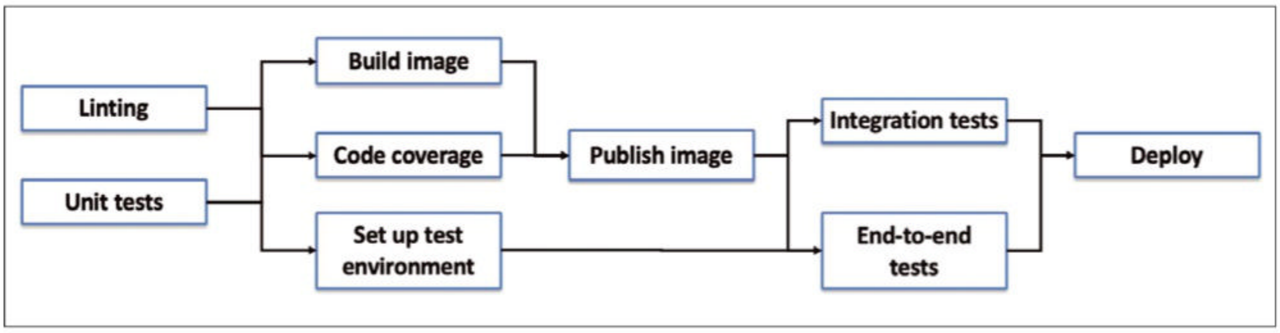


Figure 5: **Tekton implements pipelines and infrastructure at the K8s API level to build application images to run in Kubernetes.** © IBM

container and application must be available, which Knative then launches as an instance in the running cluster. Although it has little to do with CI/CD, it is precisely the CI/CD factor that plays a significant role in Kubernetes. Tekton builds the running container from the sources of a Docker image, which Serving and Eventing then process.

Conclusions

Knative proves to be a powerful tool and really turns up the heat when paired with Tekton. At the moment, this combination is the only way to achieve true CI/CD directly in Kubernetes. Other solutions might ultimately give you the same results, but it means operating the infrastructure outside of Kubernetes without the ability to control it with the Kubernets API. If you value a solution from a single source, Knative and Tekton are the right choice.

Boundless euphoria is not the order of the day, however. Many external tools offer functions that Knative cannot implement within Kubernetes because of the system. At this point, admins and developers need to test the available alternatives and choose the tool that best fits their own requirements. In many cases, this is likely to be Knative, but there are exceptions. ■

Info

[1] Tekton Pipelines: [https://tekton.dev/docs/pipelines/pipelines/]

[2] Knative examples: [https://knative.dev/docs/samples/serving/]

[3] "Hello World!" with Knative: [https://github.com/knative/docs/tree/main/code-samples/serving/hello-world/helloworld-go]

The Author

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

shop.linuxnewmedia.com

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

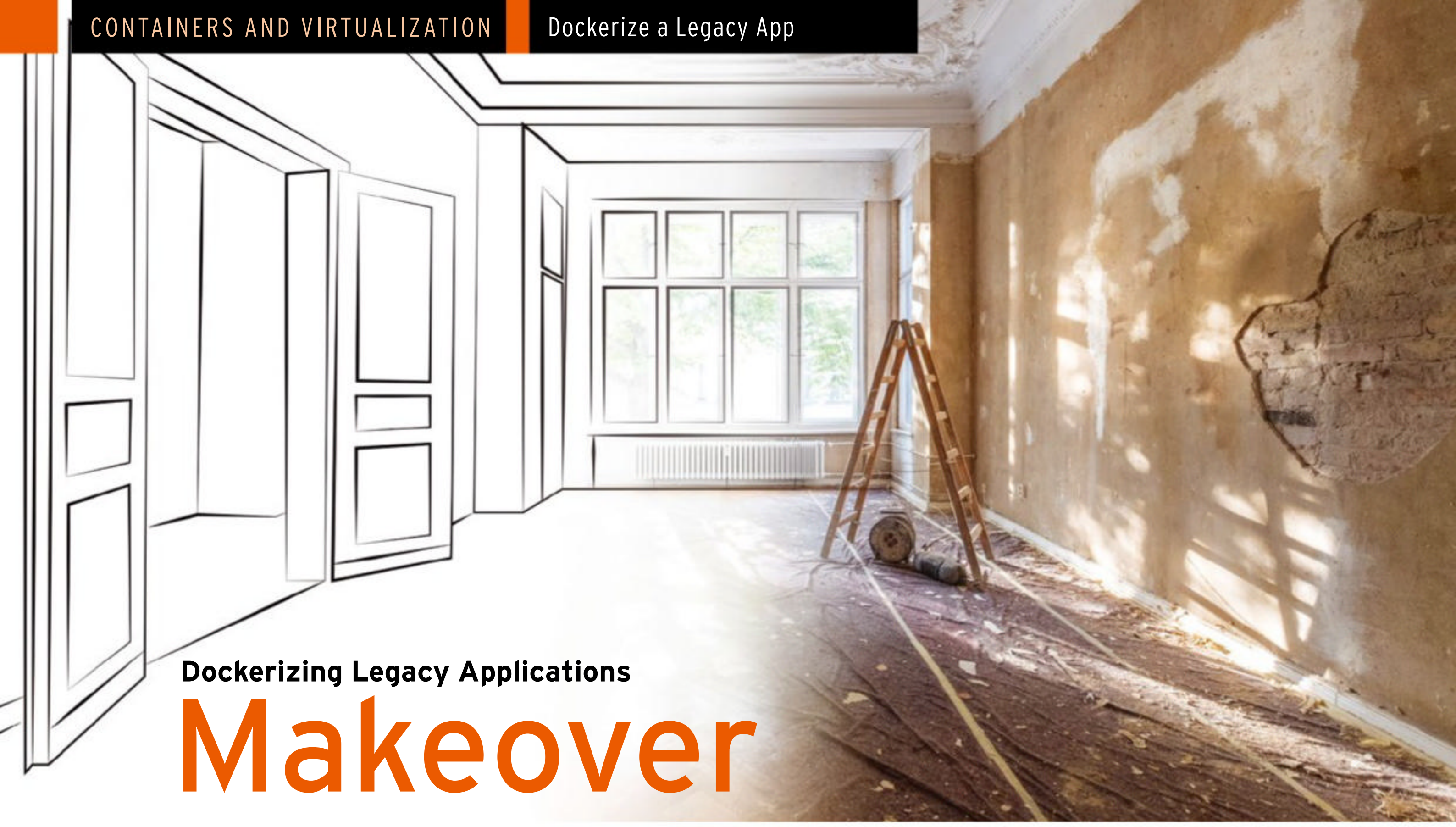
Want to subscribe?

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

shop.linuxnewmedia.com

DIGITAL & PRINT SUBSCRIPTIONS

SPECIAL EDITIONS



Dockerizing Legacy Applications

Makeover

Sooner or later, you'll want to convert your legacy application to a containerized environment. Docker offers the tools for a smooth and efficient transition. By Artur Skura

In the past, we ran applications on physical machines. We cared about every system on our network, and we even spent time discussing a proper naming scheme for our servers (RFC 1178 [1]). Then virtual machines came along, and the number of servers we needed to manage increased dramatically. We would spin them up and shut them down as necessary. Then containers took this idea even further: It typically took several seconds or longer to start a virtual machine, but you could start and stop a container in almost no time.

In essence, a container is a well-isolated process, sharing the same kernel as all other processes on the same machine. Although several container technologies exist, the most popular is Docker. Docker's genius was to create a product that is so smooth and easy to use that suddenly everybody started using it. Docker managed to hide the underlying complexity of spinning up a container and to make common operations as simple as possible.

Containerizing Legacy Apps

Although most modern apps are created with containerization in mind, many legacy applications based on older architectures are still in use. If your legacy application is running fine in a legacy context, you might be wondering why you would want to go to the trouble to containerize.

The first advantage of containers is the uniformity of environments: Containerization ensures that the application runs consistently across multiple environments by packaging the app and its dependencies together. This means that the development environment on the developer's laptop is fundamentally the same as the testing and production environments. This uniformity can lead to significant savings with testing and troubleshooting future releases. Another benefit is that containers can be horizontally scaled; in other words, you can scale the application by increasing (and decreasing) the number of containers. Adding a container orchestration tool like Kubernetes means you can optimize resource allocation and better use the machines you have – whether physical or virtual. The power of container orchestration makes it easy to scale the app with the load. Because containers start faster than virtual machines, you can scale much more efficiently, which is crucial for applications that have to deal with sudden load spikes. The fact that you can start and terminate containers quickly has several other consequences. You can deploy your applications much faster – and roll them back equally quickly if you experience problems.

Getting Started

To work with Docker, you need to set up a development environment. First, you'll need to install Docker itself. Installation steps vary, depending on your operating system [2]. Once Docker is installed, open a terminal and execute the following command to confirm Docker is correctly installed:

```
docker --version
```

Now that you have Docker installed, you'll also need Docker Compose, a tool for defining and running multi-container Docker applications [3]. If you have Docker Desktop installed, you won't need to install Docker Compose separately because the Compose plugin is already included.

For a simple example to illustrate the fundamentals of Docker, consider a Python application running Flask, a web framework that operates on a specific version of Python and relies on a few third-party packages.

Lead Image © hanohiki, 123RF.com

Listing 1 shows a snippet of a typical Python application using Flask.

To dockerize this application, you would write a Dockerfile – a script containing a sequence of instructions to build a Docker image. Each instruction in the Dockerfile generates a new layer in the resulting image, allowing for efficient caching and reusability. By constructing a Dockerfile, you essentially describe the environment your application needs to run optimally, irrespective of the host system.

Start by creating a file named `Dockerfile` (no file extension) in your project directory. The basic structure involves specifying a base image, setting environment variables, copying files, and defining the default command for the application. Listing 2 shows a simple Dockerfile for the application in Listing 1.

In this Dockerfile, I specify that I'm using Python 3.11, set the working directory in the container to `/app`, copy the required files, and install the necessary packages, as defined in a `requirements.txt` file. Finally, I specify that the application should start by running `app.py`.

To build this Docker image, you would navigate to the directory containing the Dockerfile and execute the following commands to build and run the app:

```
docker build -t my-legacy-app .
docker run -p 5000:5000 ↩
  my-legacy-app
```

With these steps, you have containerized the Flask application using Docker. The application now runs isolated from the host system, making it more portable and easier to deploy on any environment that supports Docker.

Networking in Docker

Networking is one of Docker's core features, enabling isolated containers to communicate amongst themselves and with external networks. The most straightforward networking scenario involves a single container that needs to be accessible from the host machine or the outside world. To support network connections, you'll need to expose ports.

When running a container, the `-p` flag maps a host port to a container port:

```
docker run -d -p 8080:80 ↩
  --name web-server nginx
```

In this case, NGINX is running inside the container on port 80. The `-p 8080:80` maps port 8080 on the host to port 80 on the container. Now, accessing `http://localhost:8080` on the host machine directs traffic to the NGINX server running in the container.

For inter-container communication, Docker offers several options. The simplest approach involves using container names as DNS names, made possible by the default `bridge` network. First, run a database container:

```
docker run -d --name ↩
  my-database mongo
```

Now, if you want to link a web application to this database, you can reference the database container by its name:

```
docker run -d --link my-database:db ↩
  my-web-app
```

In this setup, `my-web-app` can connect to the MongoDB server by using `db` as the hostname.

Although useful, the `--link` flag is considered legacy and is deprecated. A more flexible approach is to create custom bridge networks. A custom network facilitates automatic DNS resolution for container names, and it also allows for network isolation.

For example, you can create a custom network as follows:

```
docker network create my-network
```

Now, run containers in this custom network with:

```
docker run -d --network=my-network ↩
  --name my-database mongo ↩
  --network-alias=db
docker run -d --network=my-network ↩
  my-web-app
```

Here, `my-web-app` can still reach `my-database` using its name or a DNS alias, but now both containers are isolated in a custom network, offering more control and security.

For applications requiring more complex networking setups, you can use Docker Compose and define multiple services, networks, and even volumes in a single `docker-compose.yml` file (Listing 3).

When you run `docker-compose up`, both services will be instantiated, linked, and isolated in a custom network, as defined.

As you can see, effective networking in Docker involves understanding and combining these elements: port mapping for external access, inter-container communication via custom bridge networks, and orchestration (managed here by Docker Compose).

Volumes and Persistent Data

Managing persistent data within Docker involves understanding and leveraging volumes. Unlike a container, a volume exists independently

Listing 1: Simple Flask App

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
  return 'Hello, World!'

if __name__ == '__main__':
  app.run(host='0.0.0.0', port=5000)
```

Listing 2: Dockerfile for Flask App (Listing 1)

```
# Use an official Python runtime as a base image
FROM python:3.11-slim

# Set the working directory in the container
WORKDIR /app

# Copy the requirements.txt file into the container
COPY requirements.txt /app/

# Install the dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the current directory contents into the container
COPY . /app/

# Run app.py when the container launches
CMD ["python", "app.py"]
```

and retains data even when a container is terminated. This characteristic is crucial for stateful applications, like databases, that require data to persist across container life cycles.
For simple use cases, you can create anonymous volumes at container runtime. When you run a container with an anonymous volume, Docker generates a random name for the volume. The following command starts a MongoDB container and attaches an anonymous volume to the /data/db directory, where MongoDB stores its data:

```
docker run -d --name my-mongodb ➘
  -v /data/db mongo
```

Whereas anonymous volumes are suitable for quick tasks, named volumes provide more control and are easier to manage. If you use `docker run` and specify a named volume, Docker will auto-create it if needed. You can also create a named volume explicitly with:

```
docker volume create my-mongo-data
```

Now you can start the MongoDB container and explicitly attach this named volume:

```
docker run -d --name my-mongodb ➘
  -v my-mongo-data:/data/db mongo
```

You can use named volumes to share data between containers. If you need to share data between the container and the host system, host volumes are the choice. This feature mounts a specific directory from the host into the container:

```
docker run -d --name my-mongodb ➘
  -v /path/on/host:/data/db mongo
```

Here, `/path/on/host` corresponds to the host system directory you want to mount.
With Docker Compose, volume specification becomes streamlined and readable, especially when dealing with multi-container, stateful legacy applications. Listing 4 shows how you could define a service in `docker-compose.yml` with a named volume.
When you run `docker-compose up`, it will instantiate the service with the specified volume.
Data persistence isn't confined to just storing data; backups are equally vital. Use `docker cp` to copy files or directories between a container and the local filesystem. To back up data from a MongoDB container, enter:

```
docker cp my-mongodb:/data/db ➘
  /path/on/host
```

Here, data from `/data/db` inside the `my-mongodb` container is copied to `/path/on/host` on the host system.

Dockerizing a Legacy Web Server

Containerizing a legacy web server involves several phases: assessment, dependency analysis, containerization, and testing. For this example, I'll focus on how to containerize an Apache HTTP Server. The process generally involves creating a Dockerfile, bundling configuration files, and possibly incorporating existing databases.
The first step is to create a new directory to hold your Dockerfile and configuration files. This directory acts as the build context for the Docker image:

```
mkdir dockerized-apache
cd dockerized-apache
```

Start by creating a Dockerfile that specifies the base image and installation steps. Imagine you're using an Ubuntu-based image for compatibility with your legacy application (Listing 5).
In Listing 5, the `RUN` instruction installs Apache, and the `COPY` instruction transfers your existing Apache configuration file (`my-httpd.conf`) into the image. The `CMD` instruction specifies that Apache should run in the foreground when the container starts.
Place your existing Apache configuration file in the same directory as the Dockerfile. This configuration should be a working setup for your legacy web server. Build the Docker image from within the `dockerized-apache` directory:

```
docker build -t dockerized-apache .
```

Run a container from this image, mapping port 80 inside the container to port 8080 on the host:

```
docker run -d -p 8080:80 --name ➘
  my-apache-container ➘
  dockerized-apache
```

The legacy Apache server should now be accessible via `http://localhost:8080`.

If your legacy web server interacts with a database, you'll likely need to

Listing 3: Sample docker-compose.yml File

```
services:
  web:
    image: nginx
    networks:
      - my-network
  database:
    image: mongo
    networks:
      - my-network
networks:
  my-network:
    driver: bridge
```

Listing 4: Sample Named Volume

```
services:
  database:
    image: mongo
    volumes:
      - my-mongo-data:/data/db
volumes:
  my-mongo-data:
```

Listing 5: A sample Dockerfile for Apache web server

```
# Use an official Ubuntu as a parent image
FROM ubuntu:latest

# Install Apache HTTP Server
RUN apt-get update && apt-get install -y apache2

# Copy local configuration files into the container
COPY ./my-httpd.conf /etc/apache2/apache2.conf

# Expose port 80 for the web server
EXPOSE 80

# Start Apache when the container runs
CMD ["apachectl", "-D", "FOREGROUND"]
```

dockerize that component as well or ensure the web server can reach the existing database. For instance, if you have a MySQL database, you can run a MySQL container and link it to your Apache container. A tool like Docker Compose can simplify the orchestration of multi-container setups.
For debugging, you can view the logs using the following command:

```
docker logs my-apache-container
```

This example containerized a legacy Apache HTTP Server, but you can use this general framework with other web servers and applications as well. The key is to identify all dependencies, configurations, and runtime parameters to ensure a seamless transition from a traditional setup to a containerized environment.

What About a Database?

Containers are by nature stateless, whereas data is inherently stateful. Therefore databases require a more nuanced approach. In the past, running databases in containers was usually not recommended, but nowadays you can do it perfectly well – you just need to make sure the data is treated properly.
Or, you can decide not to containerize your databases at all. In this scenario, your containers connect to a dedicated database, such as an RDS instance managed by Amazon Web Services (AWS), which makes sense if your containers are running on AWS. Amazon then takes care of provisioning, replication, backup, and so on. This safe and clean solution lets you concentrate on other tasks while AWS is doing the chores. One common scenario is to use a containerized database in local development (so it's easy to spin up/tear down), but then swap out for a managed database service in production. At the end of the day, your app is using the database's communication protocol, regardless of where and how the database is running.
Dockerizing an existing database like MySQL or Oracle Database is a nontrivial task that demands meticulous planning and execution. The procedure involves containerizing the database, managing persistent storage, transferring existing data, and ensuring security measures are in place. One area where containerizing a traditional SQL database is extremely useful is in development and testing (see the "Testing" box).
If you choose to dockerize a database, the first step is to choose a base image. For MySQL, you could use the official Docker image available on Docker Hub. You will also find official images for Oracle Database.
The following is a basic example of how to launch a MySQL container:

```
docker run --name my-existing-mysql ⏎
  -e MYSQL_ROOT_PASSWORD=⏎
  my-secret-pw -d mysql:8.0
```

In this example, the environment variable `MYSQL_ROOT_PASSWORD` is set to your desired root password. The `-d` flag runs the container in detached mode, meaning it runs in the background.
This quick setup works for new databases, but keep in mind that existing databases require you to import existing data. You can use a Docker volume to import a MySQL dump file into the container and then import it into the MySQL instance within the Docker container.

Configurations and Environment Variables

Legacy applications often rely on complex configurations and environment variables. When dockerizing such applications, it's crucial to manage these configurations efficiently, without compromising security or functionality. Docker provides multiple ways to inject configurations and environment variables into containers: via Dockerfile instructions, command-line options, environment files, and Docker Compose. Each method serves a particular use case.
Dockerfile-based configurations are suitable for immutable settings that don't change across different environments. For instance, setting the `JAVA_HOME` variable for a Java application can be done in the Dockerfile:

```
FROM openjdk:11
ENV JAVA_HOME ⏎
  /usr/lib/jvm/java-11-openjdk-amd64
```

However, hard-coding sensitive or environment-specific information in the Dockerfile is not recommended, because it compromises security and reduces flexibility.

Testing

You can quickly spin up a container with your database, usually containing test data, and then immediately verify if your app works properly with the database. And you can do it all without asking the Ops team to provision a database host for you.
Good examples of this usage are services in GitLab CI/CD pipelines, such as PostgreSQL [4] or MySQL [5]. In Listing 6, I use a Docker image containing Docker and Docker Compose and the usual variables defining the database, its user, and password. I also define the so-called service, based on the image **`postgres:16`** and aliased as **`postgres`**. In the **`test`** job, I install the PostgreSQL command-line client and execute a sample SQL query connecting to the **`postgres`** service defined earlier, having exported the password.
The **`postgres`** service is simply a Docker container with the chosen version of PostgreSQL conveniently started in the same network as the main container so that you can connect to it directly from your pipeline.

Listing 6: PostgreSQL in a GitLab CI Pipeline

```
image: my-image-with-docker-and-docker-compose

variables:
  POSTGRES_DB: my-db
  POSTGRES_USER: ${USER_NAME}
  POSTGRES_PASSWORD: ${USER_PASSWORD}

services:
  - name: postgres:16
    alias: postgres

test:
  script:
    - apt-get update && apt-get install -y
      postgresql-client
    - PGPASSWORD=$POSTGRES_PASSWORD psql -h postgres
      -U $POSTGRES_USER -d $POSTGRES_DB -c 'SELECT 1;'
```

For more dynamic configurations, use the `-e` option with `docker run` to set environment variables:

```
docker run -e "DB_HOST=➘
  database.local" -e "DB_PORT=➘
  3306" my-application
```

While convenient for a few variables, this approach becomes unwieldy with a growing list. As a more scalable alternative, Docker allows you to specify an environment file:

```
# .env file
DB_HOST=database.local
DB_PORT=3306
```

Then, run the container as follows:

```
docker run --env-file .env ➘
  my-application
```

This method keeps configurations organized, is easy to manage with version control systems, and separates the configurations from application code. However, exercise caution; ensure the `.env` files, especially those containing sensitive information, are adequately secured and not accidentally committed to public repositories.
In multi-container setups orchestrated with Docker Compose, you can define environment variables in the `docker-compose.yml` file:

```
services:
  my-application:
    image: my-application:latest
    environment:
      DB_HOST: database.local
      DB_PORT: 3306
```

For variable data across different environments (development, staging, production), Docker Compose supports variable substitution:

```
services:
  my-application:
    image: my-application:➘
      ${TAG-latest}
    environment:
      DB_HOST: ${DB_HOST}
      DB_PORT: ${DB_PORT}
```

Run it with environment variables sourced from a `.env` file or directly from the shell:

```
DB_HOST=database.local ➘
  DB_PORT=3306 docker-compose up
```

Configuration files necessary for your application can be managed using Docker volumes. Place the configuration files on the host system and mount them into the container:

```
docker run -v ➘
  /path/to/config/on/host:➘
  /path/to/config/in/container ➘
  my-application
```

In Docker Compose, use:

```
services:
  my-application:
    image: my-application:latest
    volumes:
      - /path/to/config/on/host:➘
        /path/to/config/in/container
```

This approach provides a live link between host and container, enabling real-time configuration adjustments without requiring container restarts.

Docker and Security Concerns

Securing Docker containers requires checking every layer: the host system, the Docker daemon, images, containers, and networking. Mistakes in any of these layers can expose your application to a variety of threats, including unauthorized data access, denial of service, code execution attacks, and many others.
Start by securing the host system running the Docker daemon. Limit access to the Docker Unix socket, typically `/var/run/docker.sock`. This socket allows communication with the Docker daemon and, if compromised, grants full control over Docker. Use Unix permissions to restrict access to authorized users.
Always fetch Docker images from trusted sources. Scan images for vulnerabilities using a tool like Docker Scout [6] or Clair [7].
Implement least privilege principles for containers. For instance, don't run containers as the root user. Specify a non-root user in the Dockerfile:

```
FROM ubuntu:latest
RUN useradd -ms /bin/bash myuser
USER myuser
```

Containers also should not run with least privileges. The following example:

```
docker run --cap-drop=all -cap-add=➘
  net_bind_service my-application
```

starts a container with all capabilities dropped and then adds back only the `net_bind_service` capability required to bind to ports lower than 1024.
Use read-only mounts for sensitive files or directories to prevent tampering:

```
docker run -v /my-secure-data:➘
  /data:ro my-application
```

If the container needs to write to a filesystem, consider using Docker volumes and restricting read/write permissions appropriately.
It is also important to implement logging and monitoring to detect abnormal container behavior, such as unexpected outgoing traffic or resource utilization spikes.

Dockerizing a Legacy CRM System

To dockerize a legacy Customer Relationship Management (CRM) system effectively, you need to first understand its current architecture. The hypothetical legacy CRM I'll dockerize consists of an Apache web server, a PHP back end, and a MySQL database. The application currently runs on a single, aging physical server, handling functions from customer data storage to sales analytics.
The CRM's monolithic architecture means that the web server, PHP back end, and database are tightly integrated, all residing on the same machine. The web server listens on port 80 and communicates directly with

the PHP back end, which in turn talks to the MySQL database on port 3306. Clients interact with the CRM through a web interface served by the Apache server.

The reasons for migrating the CRM to a container environment are as follows:

- Scalability: The system's monolithic nature makes it hard to scale individual components.
- Maintainability: Patching or updating one part of the applications often requires taking the entire system offline.
- Deployment: New feature rollouts are time-consuming and prone to errors.
- Resource utilization: The aging hardware is underutilized but can't be decommissioned due to the monolithic architecture.

To containerize the CRM, you need to take the following steps.

Step 1: Initial Isolation of Components and Dependencies

Before you dive into dockerization, it is important to isolate the individual components of the legacy CRM system: the Apache web server, PHP back end, and MySQL database. This step will lay the groundwork for creating containerized versions of these components. However, the tightly integrated monolithic architecture presents challenges in isolation, specifically in ensuring that dependencies are correctly mapped and that no features break in the process.

Start by decoupling the Apache web server from the rest of the system. One approach is to create a reverse proxy that routes incoming HTTP requests to a separate machine or container where Apache is installed. You can achieve this using NGINX:

```
# nginx.conf
server {
  listen 80;
  location / {
    proxy_pass ↩
    http://web:80;
  }
}
```

Next, move the PHP back end to its own environment. Use PHP-FPM to manage PHP processes separately. Update Apache's `httpd.conf` to route PHP requests to the PHP-FPM service:

```
# httpd.conf
ProxyPassMatch ^/(.*\.php(/.*)?)$↩
  fcgi://php:9000/path/to/app/$1
```

For the MySQL database, configure a new MySQL instance on a separate machine. Update the PHP back end to connect to this new database by altering the database connection string in the configuration:

```
<?php
$db = new PDO('mysql:host=db;dbname=↩
  your_db', 'user', 'password');
?>
```

During this isolation, you might find that some components have shared libraries or dependencies that are stored locally, such as PHP extensions or Apache modules. These should be identified and installed in the respective isolated environments. Missing out on these dependencies can cause runtime errors or functional issues.

While moving the MySQL database, ensuring data consistency can be a challenge. Use tools like `mysqldump` [8] for data migration and validate the consistency (Listing 7).

If user sessions were previously managed by storing session data locally, you'll need to migrate this functionality to a distributed session management system like Redis.

Step 2: Creating Dockerfiles and Basic Containers

Once components and dependencies are isolated, the next step is crafting Dockerfiles for each element: the Apache web server, PHP back end, and MySQL database. For Apache, the Dockerfile starts from a base Apache image and copies the necessary HTML and configuration files. A simplified Dockerfile appears in Listing 8.

Build the Apache image with:

```
docker build -t my-apache-image .
```

Then, run the container:

```
docker run --name ↩
  my-apache-container ↩
  -d my-apache-image
```

For PHP, start with a base PHP image and then install needed extensions. Add your PHP code afterwards (Listing 9).

Build and run the PHP image similarly to Apache:

```
docker build -t my-php-image .
docker run --name my-php-container ↩
  -d my-php-image
```

MySQL Dockerfiles are less common because the official MySQL Docker images are configurable via environment variables. However, if you have SQL scripts to run at startup, you can include them (Listing 10).

Run the MySQL container with environment variables to set up the database name, user, and password:

```
docker run --name my-mysql-container ↩
  -e MYSQL_ROOT_PASSWORD=↩
  my-secret -d my-mysql-image
```

Listing 7: MySQL Data

```
# Data export from old MySQL
mysqldump -u username -p database_name > data-dump.sql

# Data import to new MySQL
mysql -u username -p new_database_name < data-dump.sql
```

Listing 8: Dockerfile for Apache

```
# Use an official Apache runtime as base image
FROM httpd:2.4

# Copy configuration and web files
COPY ./my-httpd.conf /usr/local/apache2/conf/httpd.conf
COPY ./html/ /usr/local/apache2/htdocs/
```

Listing 9: Dockerfile for PHP

```
# Use an official PHP runtime as base image
FROM php:8.2-fpm

# Install PHP extensions
RUN docker-php-ext-install pdo pdo_mysql

# Copy PHP files
COPY ./php/ /var/www/html/
```

For production, you'll need to optimize these Dockerfiles and runtime commands with critical settings, such as specifying non-root users to run services in containers, fine-tuning Apache and PHP settings for performance, and enabling secure connections to MySQL.

Step 3: Networking and Data Management

At this point, the decoupled components – Apache, PHP, and MySQL – each reside in a separate container.

Listing 10: Dockerfile for MySQL Startup Scripts

```
# Use the official MySQL image
FROM mysql:8.0

# Initialize database schema
COPY ./sql-scripts/ /docker-entrypoint-initdb.d/
```

Listing 11: Network Setup

```
docker run --network crm-network --name
  my-apache-container -d my-apache-image
docker run --network crm-network --name my-php-container
  -d my-php-image
docker run --network crm-network --name
  my-mysql-container -e MYSQL_ROOT_PASSWORD=my-secret -d
  my-mysql-image
```

Listing 12: docker-compose.yml Network Setup

```
services:
  web:
    image: my-apache-image
    networks:
      - crm-network

  php:
    image: my-php-image
    networks:
      - crm-network

  db:
    image: my-mysql-image
    environment:
      MYSQL_ROOT_PASSWORD: my-secret
    volumes:
      - mysql-data:/var/lib/mysql
    networks:
      - crm-network

networks:
  crm-network:
    driver: bridge

volumes:
  mysql-data:
```

For these containers to function cohesively as your legacy CRM system, appropriate networking and data management strategies are vital. Containers should communicate over a user-defined bridge network rather than Docker's default bridge to enable hostname-based communication. Create a user-defined network:

```
docker network create crm-network
```

Then attach each container to this network (Listing 11).

Now, each container can reach another using an alias or the service name as the hostname. For instance, in your PHP database connection string, you can replace the hostname with `my-mysql-container`.

Data in Docker containers is ephemeral. For a database system, losing data upon container termination is unacceptable. You can use Docker volumes to make certain data persistent and manageable:

```
docker volume create mysql-data
```

Bind this volume to the MySQL container:

```
docker run --network crm-network ⮯
  --name my-mysql-container ⮯
  -e MYSQL_ROOT_PASSWORD=⮯
  my-secret -v mysql-data:⮯
  /var/lib/mysql -d my-mysql-image
```

For the Apache web server and PHP back end, you should map any writable directories (e.g., for logs or uploads) to Docker volumes.

Docker Compose facilitates running multi-container applications. Create a `docker-compose.yml` file as shown in Listing 12.

Execute `docker-compose up`, and all your services will start on the defined network with the appropriate volumes for data persistence. Note that user-defined bridge networks incur a small overhead. Although this overhead is negligible for most applications, high-throughput systems might require `host` or `macvlan` networks.

If you decide to run your app in Kubernetes, for example, you will not need to worry about Docker networking, because Kubernetes has its own networking plugins.

Step 4: Configuration Management and Environment Variables

Configuration management and environment variables form the backbone of a flexible, maintainable dockerized application. They allow you to parametrize your containers so that the same image can be used in multiple contexts, such as development, testing, and production, without alteration. These parameters might include database credentials, API keys, or feature flags.

You can pass environment variables to a container at runtime via the `-e` flag:

```
docker run --name my-php-container ⮯
  -e API_KEY=my-api-key ⮯
  -d my-php-image
```

In your PHP code, the `API_KEY` variable can be accessed as `$_ENV['API_KEY']` or `getenv('API_KEY')`. For a more comprehensive approach, Docker Compose allows you to specify environment variables for each service in the `docker-compose.yml` file:

```
services:
  db:
    image: my-mysql-image
    environment:
      MYSQL_ROOT_PASSWORD: my-secret
```

Alternatively, you can use a `.env` file in the same directory as your `docker-compose.yml`. Place your environment variables in the `.env` file:

```
API_KEY=my-api-key
MYSQL_ROOT_PASSWORD=my-secret
```

Reference these in `docker-compose.yml`:

```
services:
  db:
    image: my-mysql-image
    environment:
      MYSQL_ROOT_PASSWORD: ⮯
        ${MYSQL_ROOT_PASSWORD}
```

Running `docker-compose up` will load these environment variables automatically. Never commit sensitive information like passwords or API keys in your Dockerfiles or code. Configuration files for Apache, PHP, or MySQL should never be hard-coded into the image. Instead, mount them as volumes at runtime. If you're using Docker Compose, you can specify a volume using the `volumes` directive:

```
services:
  web:
    image: my-apache-image
    volumes:
- ./my-httpd.conf:/usr/local/➘
  apache2/conf/httpd.conf
```

Some configurations might differ between environments (e.g., development and production). Use templates for your configuration files where variables can be replaced at runtime by environment variables. Tools like `envsubst` can assist in this substitution before the service starts:

```
envsubst < my-httpd-template.conf > ➘
  /usr/local/apache2/conf/httpd.conf
```

Strive for immutable configurations and idempotent operations to ensure your system's consistency. Once a container is running, changing its configuration should not require manual intervention. If a change is needed, deploy a new container with the updated configuration.

While this approach is flexible, it introduces complexity into the system, requiring well-documented procedures for setting environment variables and mounting configurations. Remember that incorrect handling of secrets and environment variables can lead to security vulnerabilities.

Step 5: Testing and Validation

Testing and validation are nonnegotiables in the transition from a legacy system to a dockerized architecture. Ignoring or cutting corners in this phase jeopardizes the integrity of the system, often culminating in performance bottlenecks, functional inconsistencies, or security vulnerabilities. The CRM system, being business-critical, demands meticulous validation.

The most basic level of validation is functional testing to ensure feature parity with the legacy system. Automated tools like Selenium [9] for web UI testing or Postman [10] for API testing offer this capability. Running a test suite against both the legacy and dockerized environments verifies consistent behavior. For example, to run Selenium tests in a Docker container, you would type a command similar to the following:

```
docker run --net=host selenium/➘
  standalone-chrome python ➘
  my_test_script.py
```

Once functionality is confirmed, performance metrics such as latency, throughput, and resource utilization must be gauged using tools like Apache JMeter, Gatling, or custom scripts. You should also simulate extreme conditions to validate the system's reliability under strain.

Static application security testing (SAST) and dynamic application security testing (DAST) should also be employed. Tools like OWASP ZAP can be dockerized and incorporated into the testing pipeline for dynamic testing. While testing, activate monitoring solutions like Prometheus and Grafana or ELK stack for real-time metrics and logs. These tools will identify potential bottlenecks or security vulnerabilities dynamically.

Despite rigorous testing, unforeseen issues might surface post-deployment. Therefore, formulate a rollback strategy beforehand. Container orchestration systems, such as Kubernetes and Swarm, provide the ability to easily rollout changes and rollback when issues occur.

Step 6: Deployment

Deployment into a production environment is the final phase of dockerizing a legacy CRM system. The delivery method will depend on the application and your role as a developer. Many containerized applications reside today in application repositories, including Docker's own Docker Hub container image library. If you are deploying the application within your own infrastructure, you will likely opt for a container orchestration solution already in use, such as Kubernetes.

Conclusion

Containerization offers many technical benefits, including uniformity, security, and better scaling. In addition, containerizing your apps can save you money with more efficient testing and rollout, and a container strategy can minimize the need for continual customization to adapt to new hardware and software settings. Docker Compose and other tools in the Docker toolset provide a safe, efficient, and versatile approach for migrating your existing applications to a container environment. ■

*This article was made possible by support from Docker through Linux New Media's Topic Subsidy Program (***https://www.linuxnewmedia.com/Topic_Subsidy***).*

Info

[1] RFC 1178: Choosing a Name for your Computer: [https://datatracker.ietf.org/doc/html/rfc1178]
[2] Install Docker Engine: [https://docs.docker.com/engine/install/]
[3] Docker Compose: [https://docs.docker.com/compose/]
[4] GitLab: Using PostgreSQL: [https://docs.gitlab.com/ee/ci/services/postgres.html]
[5] GitLab: Using MySQL: [https://docs.gitlab.com/ee/ci/services/mysql.html]
[6] Docker Scout: [https://docs.docker.com/scout/]
[7] Clair: [https://github.com/quay/clair]
[8] mysqldump: [https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html]
[9] Selenium: [https://www.selenium.dev/]
[10] Postman: [https://www.postman.com/automated-testing/]

Author

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

Link Encryption with MACsec

Under Seal

MACsec encrypts defined links with high performance and secures Layer 2 protocols between client and switch or between two switches. By Benjamin Pfister

Networks are exposed to more than external attacks. Appropriate defenses need to be implemented at the entry point to the internal network or, if third parties have physical access, to access points on the network. Initial authentication during access to the local area network (LAN) without downstream verification of the transmitted packets, as with classic network access control (NAC) systems, is no longer sufficient. One approach is Media Access Control Security, (MACsec), which encrypts in Layer 2, with virtually no loss of speed.

The MACsec [1] Layer 2 security protocol is used for cryptographic point-to-point security on wired networks (e.g., on switches). Network access controls compliant with IEEE 802.1X-2004 (i.e., port-based network access control) only provide authentication by the Extended Authentication Protocol (EAP) framework – in the best case combined with periodic re-authentication. However, without an integrity check, confidentiality cannot be guaranteed at this level of the communication relationship, unless you apply a later version, IEEE 802.1X-2010, in combination with 802.1AE (MACsec).

The standard offers better performance and is less complex to implement than classic Internet Protocol Security (IPsec)-based encryption. If required, however, a combination with other security protocols such as IPsec and Transport Layer Security (TLS) is also possible. At the same time, Layer 2 protocols such as Link Layer Discovery Protocol (LLDP), Cisco Discovery Protocol (CDP), and Link Aggregation Control Protocol (LACP), as well as Address Resolution Protocol (ARP), can be transmitted transparently. MACsec also is compatible with IPv4 and IPv6 because it resides one layer below in the OSI reference model.

Because MACsec is implemented at a low level close to the hardware, it demonstrates high performance up to the full line rate (i.e., the maximum possible data rate of the link). Attacks such as session spoofing, replay attacks, or man-in-the-middle attacks are thwarted. However, MACsec does not ensure end-to-end encryption.

Securing Switch to Terminal Device

In scenario 1, MACsec secures the link from a terminal device to the switch. The objective is encrypted transmission between the two devices after successful authentication and authorization. In the IEEE 802.1X terminology, the end device in this case is referred to as the supplicant, the switch is referred to as the authenticator, and the RADIUS server is the authentication server.

The first step after link building is to authenticate the supplicant with EAP over LAN (EAPoL) frames between the supplicant and the authenticator. To generate a RADIUS request for authentication from the authenticator to the authentication server on the basis of this type of EAP frame, the authenticator packages the requests into RADIUS request messages. The supplicant's corresponding EAP method must be enabled on the RADIUS server – for example, with the Protected Extensible Authentication Protocol and Microsoft Challenge-Handshake Authentication Protocol version 2 (PEAP-MSCHAPv2) for TLS-protected transfer of the username and password after validation of the RADIUS server's authentication certificate, or with EAP-TLS if you

Lead Image © Oleksandr Omelchenko, 123RF.com

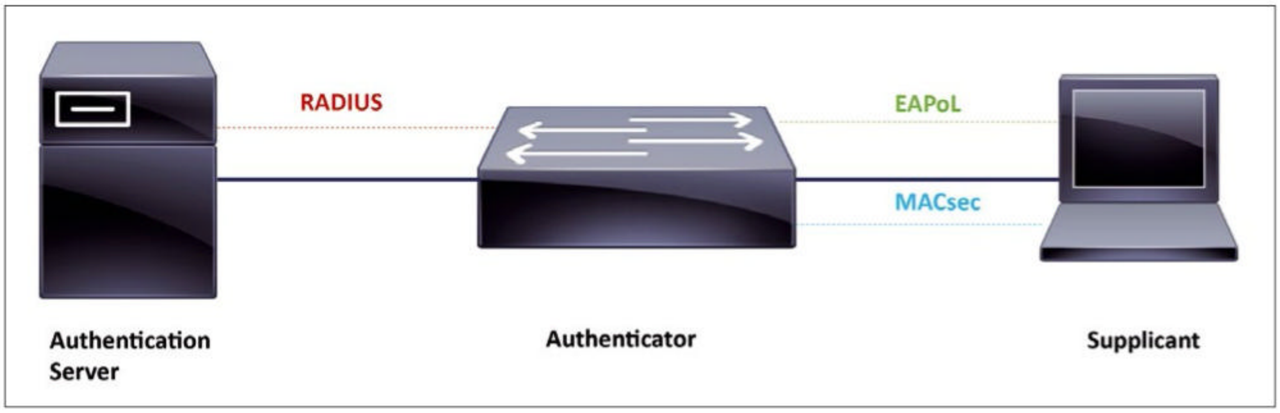


Figure 1: Communication between the authentication server, authenticator, and supplicant.

want two-way TLS authentication between the supplicant and the authentication server.

The main difference between authentication in the older variant by IEEE 802.1X-2004 and IEEE 802.1X-2010 in conjunction with 802.1AE is that authorization with additional attributes occurs on the basis of successful authentication. For this to happen, the authentication server returns a *RADIUS Access-Accept* with the MACsec policy to the authenticator. In this way, the authenticator recognizes that it needs to use MACsec (Figure 1).

The matching encryption algorithms, such as the Advanced Encryption Standard in Galois Counter Mode at 128 or 256 bits (AES128-GCM or AES256-GCM), must be configured identically on the supplicant and authenticator. Additionally, the master session key is generated.

This master session key is then used as the basis for a key exchange with a MACsec Key Agreement (MKA; see Table 1 for MACsec terminology). After the MKA process, the authenticator and supplicant have the key material required to handle encrypted data transmission.

Securing Switch to Switch

In scenario 2, MACsec encryption is used for the connection between two switches. If third parties have physical access to the cabling, MACsec transmission is recommended if you have a corresponding need for protection and if no other adequate encryption methods are used. Examples of this scenario include Layer 2 connections by providers, as used in metropolitan area Ethernet environments. However, the same also applies to dark fiber connections operated by third parties or if your own fiber optic connections run through a third party.

Without encryption on these kinds of routes, someone could route data out by couplers or network TAPs for data analysis downstream. The payload of the MAC frame is encrypted if MACsec is used and therefore cannot be easily evaluated. For provider connections, however, you need to clarify whether you can transmit MACsec frames with an Ethertype of

Table 1: MACsec Terminology

Abbreviation	Meaning	Function
CA	Connectivity association	Secured control plane connection between MACsec peers.
CAK	CA key	Control plane key from which the session key is derived.
CKN	CAK name	Frame for the CAK transmitted by the peers to each other for validation in plain text.
ICK	ICV key	Integrity check of each MKA protocol data unit (MKPDU) sent between two CAs.
ICV	Integrity check value	Provides secure connectivity associations with the AES-GCM Cipher Suites at 128/192/256 bits.
KEK	Key encryption key	Transmits the generated SAKs to the peer through the CA.
MKA	MACsec key agreement	A protocol to locate MACsec peers and to generate, renew, and exchange keys.
SA	Security association	Connection between two MACsec peers that guarantees an encrypted connection on the basis of the SAK.
SAK	SA key	Session key derived from the CAK and used for encryption between two MACsec peers.
SC	Secure channel	Logical channel on which encrypted transmission takes place.
SCI	SC identifier	Unique identifier of the channel for encrypted transmission consisting of the MAC address and port designation.
Authenticator		Provides authentication by EAPoL with the supplicant by the RADIUS protocol to the authentication server. The RADIUS server denies or grants access by authorization.
Authentication Server		RADIUS server for authentication, authorization, and accounting according to IEEE 802.1X.
Supplicant		Client component for authentication according to IEEE 802.1X.

0x88e5 on the provider's network. Because MACsec works on Layer 2, it must be individually enabled for each interface.

Encryption Method

MACsec uses AES-GCM as the encryption algorithm, encrypting hop-by-hop, which has both advantages and disadvantages. Key lengths of 128 and 256 bits are available. With the 802.1AEbw standard, IEEE introduced extended packet numbering (GCM-AES-XPN) for the current requirements for high data rates. As a result, MACsec can transmit 232 frames within a security association key (SAK). This extension enables MACsec to encrypt securely at data rates above 100Gbps. The algorithm used must match the configuration of the switch and the end device or match both switches.

The MACsec header in IEEE 802.1AE is also referred to as the security tag (SecTAG). Its length is 16 bytes. The same applies to the integrity check value (ICV) that MACsec appends to the frame. The security tag comprises five fields: EtherType, TAG control information/association number (TCI/AN), short length (SL), packet number (PN), and SCI. The MACsec EtherType contains the value 0x88E5. The TCI/AN defines a version number for MACsec without the need for a new EtherType. The SL field states the length of the encrypted data. The SCI is based on an identifier of the respective port on the component and the MAC address.

The entire MACsec frame is similar to an Ethernet frame. It also contains an ICV to ensure that the frame has not been manipulated en route. To this end, the MACsec peers decrypt incoming frames and calculate the expected ICV with the session key from the MKA. If this does not match the transmitted ICV, the receiving peer discards the frame.

Static and Dynamic Key Distribution

Each MACsec session is built on top of a Connectivity Association (CA), which describes a logical session between peers. The MACsec key agreement (MKA) protocol establishes this connection (Figure 2). However, before encrypted transmission can take place, the required key material must be distributed. MACsec can use both static and dynamic key distribution.

As the name suggests, the static variant uses preconfigured connectivity association keys (CAKs), which must match the peers involved. For this purpose, symmetric pre-shared keys (the CAKs) and a connectivity association key name (CKN) must be stored on the MACsec peers. This arrangement acts as a framework for the CAK, with peers exchanging CKNs in plain text. They also need to match on both sides of the connection.

However, the CAK is initially only used to secure the control data (control plane) and not to encrypt the user data (data plane). For this function, you need a SAK. The CAK and CKN must match to generate this key. If this is the case, the MKA goes into action. It first discovers neighboring peers and then determines the key server among them. MKAs with lower numerical priority values take priority over those with higher numerical values.

The key server then generates the symmetrical SAKs and distributes them between the opposing switches. Downstream, encrypted data transfer can take place on the data plane. The MKA then periodically generates new SAKs and distributes them in a process known as key rollover. This method is mostly used when coupling two switches with MACsec.

Dynamic Key Distribution

For the dynamic method, MACsec builds on the EAP framework from IEEE 802.1X. After successful authentication and authorization of the

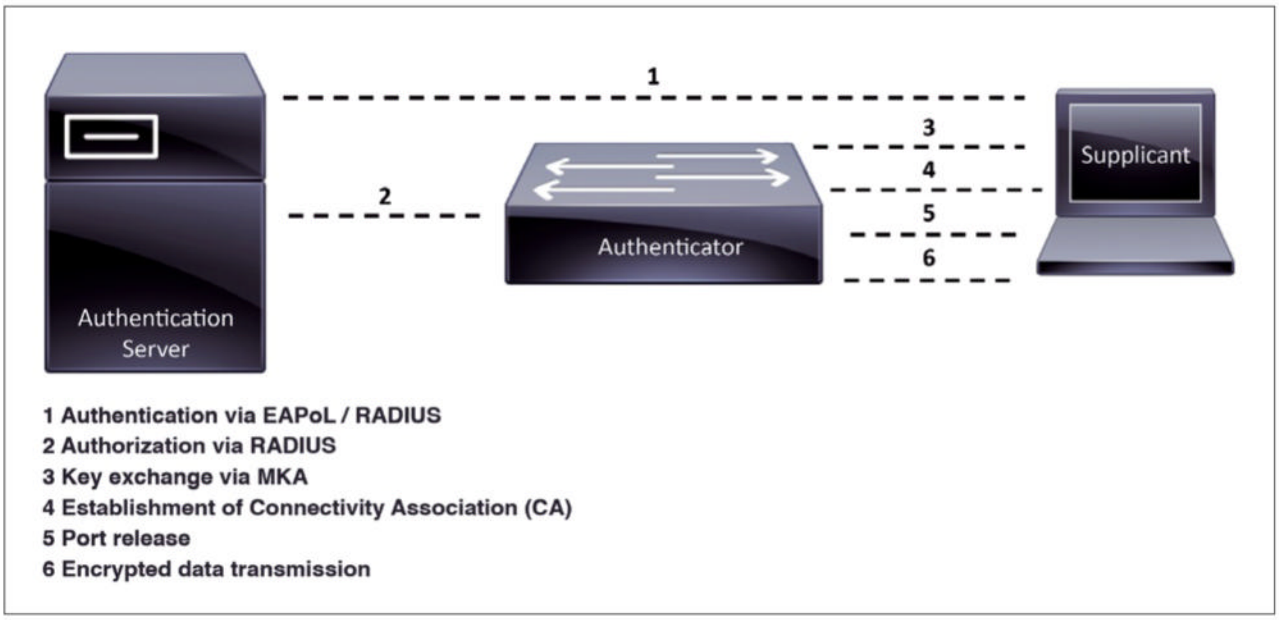

Authentication Server
Authenticator
Supplicant
1
2
3
4
5
6
1 Authentication via EAPoL / RADIUS
2 Authorization via RADIUS
3 Key exchange via MKA
4 Establishment of Connectivity Association (CA)
5 Port release
6 Encrypted data transmission

Figure 2: A MACsec session proceeds in a defined order up to the encrypted data transmission.

supplicant and authenticator, the two exchange the MKA data with a special EAPoL type. As in static key distribution, the MKA first discovers the MACsec peers. With a successful EAP authentication, a RADIUS server distributes a master key from which the CAK is derived. As with the static variant, CAKs also have an associated CKN.

Further keys are then derived from the CAK: The ICK (ICV key) and the KEK (key encryption key). The key server determined by the MKA uses the KEK to transmit the generated SAKs to the peer via the CA. The AES Key Wrap algorithm secures this transfer. These SAKs then protect the user data transmission on the data plane. The SAK has unique identifiers: The key identifier (KI, 128 bits) and the key number (KN, 32 bits), with the peers transmitting the KI in plain text in all MACsec frames.

Practical Configuration Examples

A couple of examples will illustrate the various MACsec encryption scenarios on a switch. The examples are based on Cisco Catalyst 9300 switches [2] [3] with IOS XE version 17.6 as the authenticator, a Windows endpoint with Cisco Secure Client as the supplicant, and a Cisco Identity Services Engine as the authentication server. These are just examples and do not claim to be complete. The configurations are limited to the portion specific to MACsec and 802.1X. Other manufacturers' hardware, hardware models, and software versions may differ.

Switch to End Device with Dynamic Key Distribution

The supplicant needs a MACsec profile. You can create this in the associated profile editor for the Cisco Secure Client by selecting MKA as the key management approach in the *Networks/Security* area and the encryption algorithm for MACsec. This setting must be compatible with the switch and be configured symmetrically. In this example, I choose AES128-GCM. Next, import the profile on the client. The authenticator (switch) requires a little more manual work: First, you need to define the RADIUS servers with IP addresses, ports, and a shared secret. Second, reference them in the settings for authentication, authorization, and accounting (AAA) and enable IEEE 802.1X globally if you have not already done so.

Additionally, you need a MACsec-specific MKA policy. The key server priority is set in the policy. Also, you need to specify the encryption algorithm to match that of the supplicant (i.e., AES128-GCM for the

Shop the Shop shop.linuxnewmedia.com

Missed an issue?

You're in luck.

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

shop.linuxnewmedia.com

GET IT NOW!

SAVE TIME ON DELIVERY WITH OUR ALTERNATIVE PDF EDITIONS

example in Listing 1) and set the Confidentiality Offset to 0 bytes in the MKA policy. This setting tells MACsec to encrypt the whole frame. Where switches are coupled, a certain portion of the frame could remain unencrypted (e.g., to allow virtual local area network (VLAN) tags to be evaluated for transit components) (Figure 3). Replay protection defines the extent to which the frame order can deviate from the regular order.

Listing 1: MACsec Downlink to Terminal Device

```
radius server macsec1
  address ipv4 192.0.2.1 auth-port 1812 acct-port 1813
key t0ps3cr3t
!
radius server macsec2
  address ipv4 192.0.2.2 auth-port 1812 acct-port 1813
key t0ps3cr3t
!
aaa group server radius macsec
  server name macsec1
  server name macsec2
!
aaa new-model
aaa authentication dot1x default group macsec
aaa authorization network default group macsec
aaa accounting dot1x default start-stop group macsec
!
dot1x system-auth-control
!
mka policy ITA_MKA
  key-server priority 100
  macsec-cipher-suite gcm -aes-128
  confidentiality-offset 0
  replay-protection window-size 10
!
interface GigabitEthernet2/0/1
  description ITA_MACsec_Client
  switchport mode access
  switchport access vlan 10
  macsec
  authentication host-mode multi-auth
  authentication order dot1x
  authentication port-control auto
  dot1x pae authenticator
  authentication linksec policy must-secure
  mka policy ITA_MKA
  spanning-tree portfast
```

Finally, set up the physical port, which is used to define both the legacy 802.1X settings and a reference to the MKA policy; MACsec is enabled, and a `linksec policy` with `must-secure` enforces encryption.

On the authentication server, you first need to define an authorization profile with the `must-secure` policy, which you then reference in an authorization policy for the desired supplicants on the basis of their properties from the authentication.

Switch to Switch in Static CAK Mode

The setup for MACsec between two switches is somewhat simpler than dynamic key distribution in the case of a switch to end device. The configuration is the same on both switches except for the key server priority. First, define a keychain for MACsec with a key number, which acts as the CKN. Inside the key number, the CKN creates the algorithm for the key and the key string, which acts as the CAK. Second, define the MKA policy (which I will not explain again here). Finally, bind the MKA policy and the keychain as a pre-shared key and enable MACsec on the physical interface (Listing 2).

Conclusions

MACsec cannot replace end-to-end encryption; however, when it comes to encrypting a defined link with high performance, MACsec is a very good choice. The same applies to securing Layer 2 protocols between client and switch or between two switches. Despite the latest revision in 2018, supplicant availability is still fairly low. ■

Info

[1] MACsec IEEE 802-1AE: [https://1.ieee802.org/security/802-1ae/]

[2] Cisco terminology for MACsec: [https://community.cisco.com/t5/networking-knowledge-base/macsec-history-amp-terminology/ta-p/4436094]

[3] MACsec configuration on Cisco Catalyst 9300: [https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst9300/software/release/17-6/configuration_guide/sec/b_176_sec_9300_cg/macsec_encryption.html]

Listing 2: MACsec Uplink to Switch

```
key chain ITA macsec
  key 1000
  cryptographic-algorithm aes-256-cmac
  key-string 12345678911234567890123456789012
!
mka policy ITA_MKA_Switch
  key-server priority 100
  macsec-cipher-suite gcm-aes-256
  confidentiality-offset 30
!
interface TenGigabitEthernet1/0/1
  description ITA_MACsec_Client
  switchport mode trunk
  macsec network-link
  mka policy ITA_MKA_Switch
  mka pre-shared-key key-chain ITA
```

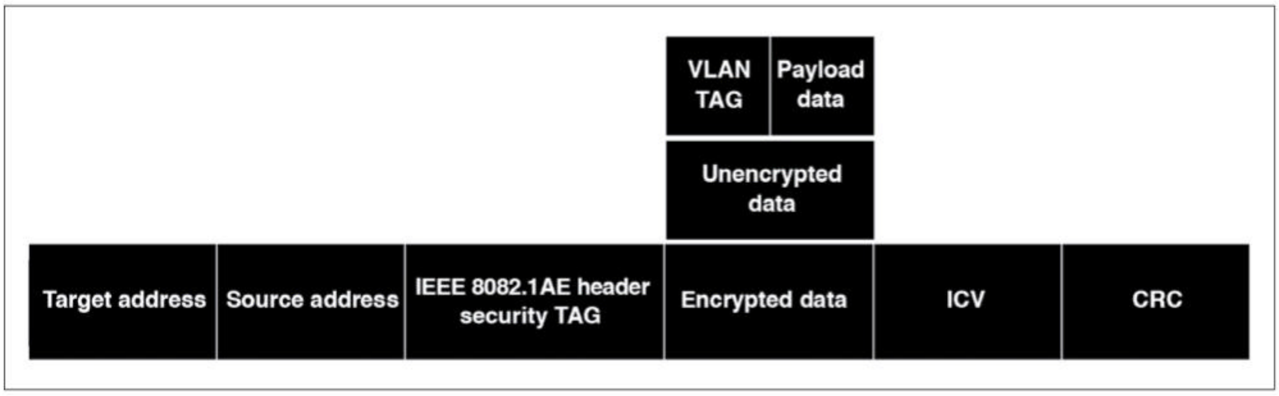


Figure 3: MACsec frame setup with encrypted VLAN tag and payload. In addition to classic Ethernet headers, the 802.1AE header (security tag) and an ICV are inserted.

Linux Magazine Subscription

Print and digital options
12 issues per year

LINUX MAGAZINE
Maker Tricks: Build an electronic scoreboard
FREE DVD
Podcasting
Support your passion and reach out to your online audience
NetBox: Keep network knowledge at your finge
Fair Play: This cool Go program divides soccer into teams of equal skill
Fyne GUI Toolkit: Code for multiple platforms
AI on the Pi: Add intelligence to your maker creations
The Best of Small Distros
Tipi Home Server Manager 100 apps at your command
Pi-hole Ad blocker for everyone
SystemRescue
10 PRACTICAL FOSS GEMS
Compromising WordPress Read this if you spend time on the web!
Think like an Intruder
Know your enemy with these real-world attack techniques
Lemmy: Free alternative to Reddit
Gesture Control: Follow a recipe without getting batter on the keys
DietPi: Lean and fast distro for the Rasp Pi
CardStock: Add a GUI to your Python app
Kopia: Stay safe with regular backups
10 TANTALIZING FREE TOOLS!
WWW.LINUX-MAGAZINE.COM

SUBSCRIBE
shop.linuxnewmedia.com

Expand your Linux skills:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

Go farther and do more with Linux, subscribe today and never miss another issue!

Follow us

@linux_pro
Linux Magazine
@linuxpromagazine
@linuxmagazine

Need more Linux?

Subscribe free to Linux Update

Our free Linux Update newsletter delivers insightful articles and tech tips to your inbox every week.

bit.ly/Linux-Update

LINUX UPDATE

No. 329 • October 5, 2023

Creating Custom ISO Images

If you are looking to customize your Linux distribution, we show you three graphical front ends for creating bootable ISO images.

In the News

- The GNU Project Celebrates Its 40th Birthday
- elementary OS 7.1 Now Available for Download

In Case You Missed It

- Simpler Content Management
  Not everyone needs a conventional database content management system. For beginners and smaller websites, a flat-file content management system offers ease of use, while eliminating attack vectors.
- Cloning a Debian System with apt-clone
  In the right circumstances, apt-clone can be a simple option for cloning your Debian system.

Did you miss last week's Linux Update? Read it here.

From the Vault

Command Line: Taking Your Hardware's Temperature

Secure remote connectivity with VS Code for the Web

Tunnel Tech

Connect to remote machines with Visual Studio Code for the Web through secure tunnels – no SSH needed. By Kevin Wittmer

The Visual Studio Code ecosystem continues to thrive and expand. With thousands of extensions available for download from the marketplace, developers have a bewildering selection to enrich their daily coding experience. The Remote-Tunnels extension [1] [2] is an example of powerfully expanding a developer's remote capabilities. This extension integrates with vscode.dev hosting Visual Studio online services, allowing GitHub users to access a lightweight Visual Studio (VS) Code experience directly through a web browser. Pairing the remote tunneling capabilities of VS Code Server with GitHub integrated cloud services powers this secure remoting capability. As a result, developers can establish remote connectivity to Linux servers hosted in private networks, further expanding the possibilities of their development workflows. In this article, I unpack the details.

Remote SSH Connectivity Revisited

Before diving into the developer experience of a web-based VS Code environment, I'll briefly revisit the established Remote-SSH VS Code extension that provides SSH connectivity capabilities to virtual machines (VMs) or containers and seamlessly integrates with the standard functionalities of a local SSH client.

If you don't already have the Remote-SSH extension installed, use the `--install-extension` argument on the command-line interface (CLI):

```
code --install-extension ⮐
  ms-vscode-remote.remote-ssh
```

Of course, you can also access the Extensions view in the VS Code user interface (UI) to search and install. This command-line action will result in the installation of three extensions. To verify that these extensions were installed successfully, enter:

```
code --list-extensions
```

The output dumped to the console will be all the extensions currently installed, including the three newly installed extensions:

```
ms-vscode-remote.remote-ssh
ms-vscode-remote.remote-ssh-edit
ms-vscode.remote-explorer
```

The extension `ms-vscode-remote.remote-ssh` does the heavy lifting of integrating with the local SSH client for SSH host selection and connectivity. The `.remote-ssh-edit` extension has a secondary purpose of providing the ability to edit SSH config files. Finally, `.remote-explorer` shows a list of available remote machines to which you can connect.

It's worth noting that the `.remote-ssh` extension uses the standard facilities of a locally available SSH client. Therefore, an SSH client must be installed and available in the environment runtime path. The conventional steps of SSH key generation and distribution apply here. Please see the "SSH Key Generation Refresher" boxout for a quick refresher. By convention, assume that the `.ssh` subdirectory exists in the user's home directory. The SSH config file residing in this subdirectory references the private key data file associated with the target SSH host as such:

```
IdentityFile ⮐
  ~/.ssh/my_id_ed25519-remote-server
```

The file path separators in the `IdentityFile` directive assume a Linux shell context.

The `.remote-explorer` extension UI will render the SSH hosts defined in the config file. Any SSH hosts with

Lead Image © fckncg, 123RF.com

a valid definition and up-to-date private key are available for connection. Select an SSH host entry and hit Enter to initiate a connection. What happens behind the scenes when you successfully authenticate via SSH, in a nutshell, is that VS Code will copy and install the VS Code Server component into the target Linux environment. This footprint, well beyond 100MB, represents the runtime software for Visual Code Server. From the SSH remoting scenario, you can typically locate the binary distribution in the hidden subdirectory `.vscode-server` (on the target SSH host for the specified users). Key entities that land in the `.vscode-server` subdirectory include:

```
.vscode-server
├── bin
| └── <40-char-hash>
| | ├── bin, extensions,
          node, node_modules, ..
| ├── bin
| | ├── helpers
| | └── remote-cli
| ├── extensions
| | ├── ...
```

The remote SSH extension will populate the Remote user interface view on the basis of the config file contents maintained in the `.ssh` subdirectory. You must refresh or relaunch VS Code for the target services to appear in the list view.

VS Code in Your Browser

SSH connectivity is a dependable and secure approach to enabling remote connectivity for VS Code developers. However, a non-SSH alternative is now available through the VS Code Web experience (hosted vscode.dev). This offering allows access to a development environment over a secure, non-SSH tunnel.

Figure 1 shows the integrated components and cloud services, including the Remote-Tunnels extension, the VS Code CLI, and VS Code Server. From the illustration, it's evident that this remote access model necessitates a GitHub user account, which most folks already have; if not, creating a new GitHub user account is simple.

Getting Started

To begin with a streamlined server-side approach, download the VS Code CLI. For the scope of this article, I zero in on Linux-based servers. This CLI initializes the VS Code server through a download and basic setup tailored for remote connectivity (similar to the SSH scenario). Manual setup and registration steps are required within the server environment to associate with an active GitHub web user session. Behind the scenes, the VS Code Server establishes a secure tunnel with AES encryption. I'll detail the commands to implement this remote setup in the subsequent sections.

Remote Tunneling Step-by-Step

To get started, first identify in which Linux user and user subdirectory to locate the VS Code Server CLI binaries. Next, log in and `cd` into the target destination subdirectory (possibly creating a new subdirectory as you go). Download the VS Code CLI from the VS Code download page. You can also use a tool like `wget` to download the binaries directly. For example, to download the Insiders edition of VS Code, enter:

```
wget -O vscode_cli_alpine_x64_cli.tar.gz ➜
  'https://code.visualstudio.com/sha/➜
   download?build=insider&os=cli-alpine-x64'
```

SSH Key Generation Refresher

Below is a quick refresher for getting SSH keys and interacting with SSH configuration:

1. Generate SSH key-data values (public and private):

```
ssh-keygen -t ed25519 -a 100 -f
ubuntu-sre-id_ed25519 -q -N
```

2. Copy the SSH public key data on the target server to the `authorized_keys` file in the `$HOME/.ssh` directory. You can use the secure copy step to complete this.
3. Copy the SSH private key data to the local `system/env`, where the SSH client is also installed.
4. Update the config file containing the SSH session server details. The file follows the SSH format and structure:

```
Host Ubuntu-SRE_Penguin
User penguin
HostName 127.0.0.1
Port 3092
IdentityFile "/Users/penguin/.ssh/
                 ubuntu-sre-id_ed25519"
```

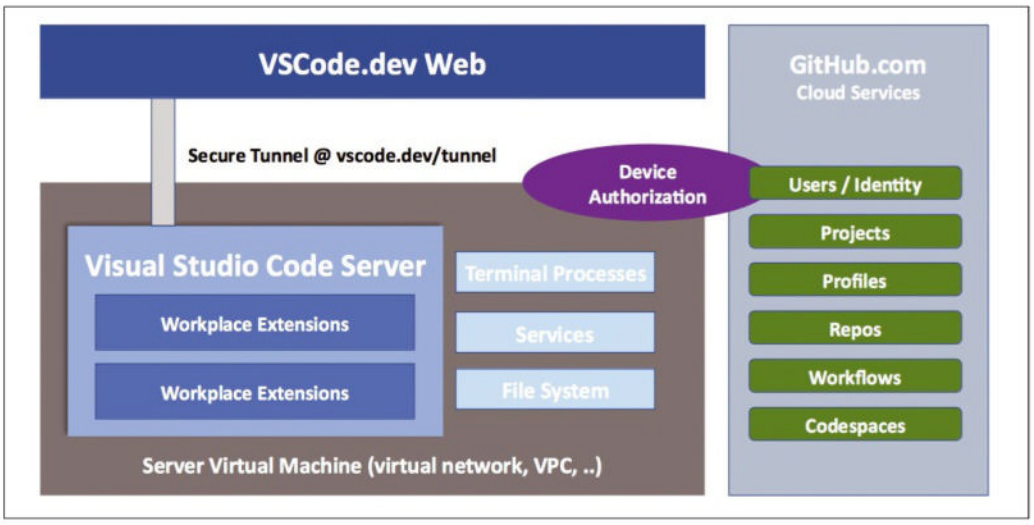


Figure 1: Screen capture of the SSH extension view from VS Code.

Note that if you want the stable edition of VS Code, then change the `build` query string to `stable`.
Next, unpack the compressed TAR with the command:

```
tar xzvf vscode_cli_alpine_x64_cli.tar.gz
```

If you want to make the binary available at the system level, one option is to copy the `code` executable to `/usr/local/bin`. Once you have copied the binary to a preferred location, verify the binary state and confirm the version of VS Code:

```
code --version
```

This command prints the `code` version and the final build commit.

As a side note, you can also source the standard version of VS Code from your local package manager. For example, you can install VS Code with `snap`:

```
sudo snap install code --classic
```

The next step is to set up the tunnel. First, inspect the tunneling command-line configuration options (Table 1):

```
code tunnel --help
```

Now register and establish the tunnel while accepting first-time licensing services. Be sure to name the tunnel, and also remember the name! It will become important when you transition into a browser context. In my example, I named my tunnel *remote384*:

```
code tunnel --name remote384 ↩
            --accept-server-license-terms
```

I suggest a naming convention, something like *remoteXYZ*. Avoid using a naming pattern with *tunnel* because the VS Code remote extension already inserts the string `tunnel` as a fixed part of the URI base path. Monitor the log output from the startup phase of the VS Code Server process and scan for the device code:

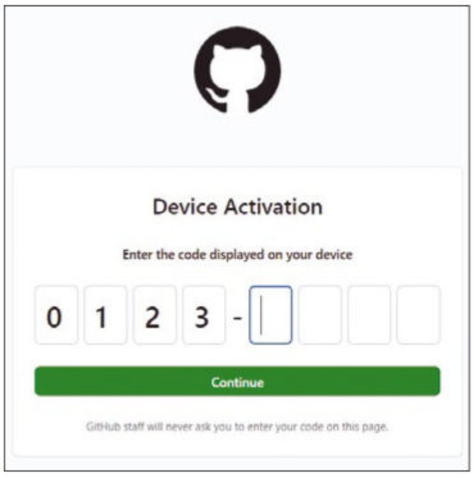
```
*
* Visual Studio Code Server
* ..
* To grant access to the server, please ↩
  log into https://github.com/login/device ↩
  and use code 0123-9A8B
```

Table 1: Tunnel Commands and Arguments

Command or Argument	Function
Most Useful Commands	
`code tunnel --help`	Dump tunnel command-line help contents to standard output.
`code tunnel --accept-server-license-terms --name remote384`	Create a tunnel that is accessible to vscode.dev (integrates with GitHub.com). Accept the license terms and respond to the browser UI prompting to enter the code from initial CLI interaction. It is highly recommended to name your tunnel; otherwise, a host name will often be taken. You can also specify the argument `--random-name` to randomly name a tunnel for the port forwarding service.
`code tunnel status`	Show the status of the tunnel, including the connection status.
`code tunnel restart`	Restart all tunnels running locally.
Useful to Clean Up, Uninstall	
`code tunnel kill`	Stop all tunnels running locally.
`code tunnel unregister`	Remove this machine's association with the port forwarding service.
`code tunnel prune`	Delete all VS Code servers that are NOT currently running.
`code tunnel rename`	Rename the specific tunnel.
Useful Arguments (e.g., to inspect verbose comments and debug info)	
`--cli-data-dir`	Directory where CLI metadata should be stored [`env: VSCODE_CLI_DATA_DIR=`]. The default is normally the `.vscode-cli` home subdirectory. Use this CLI argument to change to override and set an alternative path. Default home show/appears: `/home/kevin/.vscode-cli/server-stable/`.
`--log <level>`	Log level to use (possible values: `trace`, `debug`, `info`, `warn`, `error`, `critical`, `off`).
`--verbose`	Print verbose output during code tunnel execution.
Linux Background Service Management	
`code tunnel service install`	Install the tunnel service on this local machine.
`code tunnel service log`	Dump service log contents to the console.
`code tunnel service uninstall`	Uninstall the tunnel service on this local machine.
... with `systemctl`	
`systemctl --user restart code-tunnel.service`	systemctl to stop, start, restart, and get status.
`systemctl --user status code-tunnel.service`	Manage at the user level.
`systemctl --user stop code-tunnel.service`	Note differences between user and system.
`systemctl --user --state=running | grep code`	Search for running Visual Studio Code server instances.
`sudo loginctl enable-linger $USER`	Ensure the service stays running after you disconnect.

Once the device code is printed to the console, copy it to your text buffer and switch back to the active GitHub.com session in your web browser.

If you haven't already, log in to your GitHub account from the web browser where you plan to manage your remote session; then, input the authorization code to the GitHub device login page [3] (Figure 2).


Device Activation

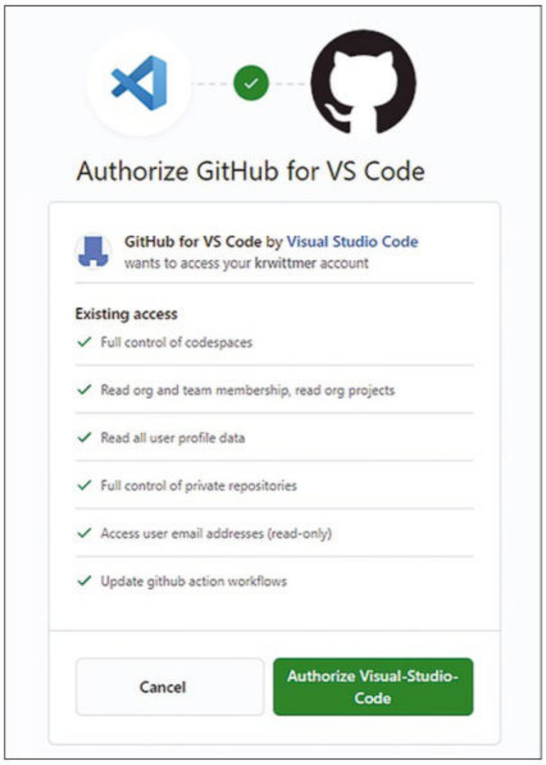
Enter the code displayed on your device

0 1 2 3 -

GitHub staff will never ask you to enter your code on this page.

Figure 2: GitHub.com prompts for authorization.

Once you verify the device code, the system will ask you to confirm GitHub authorizations for VS Code (Figure 3). The default authorizations grant full control over namespaces and access to read org projects, view user profile data, manage private repos fully, and update GitHub Actions workflows.


Authorize GitHub for VS Code

GitHub for VS Code by Visual Studio Code wants to access your krwittmer account

Existing access
- Full control of codespaces
- Read org and team membership, read org projects
- Read all user profile data
- Full control of private repositories
- Access user email addresses (read-only)
- Update github action workflows

Figure 3: Authorize GitHub for VS Code access.

If you forget to specify the argument `-accept-server-license-terms`, you will be prompted to accept (y/n) the conditions. Similarly, if you don't specify the name argument, the system will prompt you to name the tunnel (aka machine name).

Once acknowledged, VS Code for Web indicates the "device" is now connected and that you are all set. For example, a console message will print the VS Code for Web URL (*https://vscode.dev/tunnel/remote384*).

You will be prompted to allow GitHub account access as part of the sign-in process. Depending on your browser user state, you could also see a dialog box confirming authorization. Opening the remote VS Code for Web session URL in a web browser will initiate the VS Code Server download to the remote environment (Figure 4). As the messaging illustrates (Listing 1), studying the log output can also confirm this behavior.

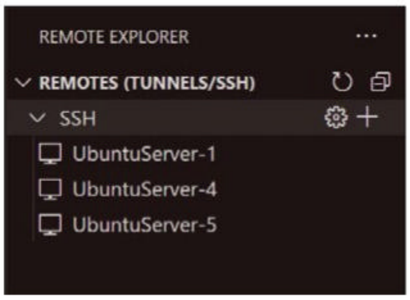


Figure 4: Feedback showing the VS Code Server download.

Part of session initialization will also activate several extensions, including `ms-vscode.remote-server`, in the local server-side environment. To view the list of extensions installed as part of the initial setup, including the setup of the remote tunnel, switch back to the VS Code for the Web user interface, then browse to the running extensions view (Figure 5).

VS Code Through a Browser

The basic layout of the VS Code for the Web user interface hosted at vscode.dev differs from its desktop sibling. Instead of a fixed top-level menu with File, Edit, Selection, and so forth, the launching point to access the menu navigation hierarchy is the icon with three vertical bars (hamburger menu) anchored to the top of the Activity Bar (Figure 6).

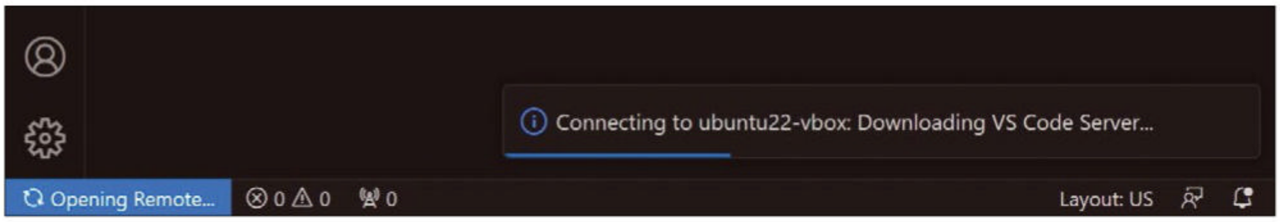
To open a project folder, click the hamburger menu, navigate to the *File | Open Folder* (Figure 7). The Integrated Terminal is also available in the *Terminal* submenu item. You can open multiple terminal windows from the browser, which display as tabs near the bottom-right corner.

VS Code supports comprehensive keyboard mappings to shortcut virtually all functions available in VS Code. With few exceptions, VS Code for the Web supports these keyboard mappings.

Typically classified as a code editor, VS Code provides enhanced debugging capabilities through language and tool-specific extensions. Figure 8 illustrates a Python debugging session in a VS

Listing 1: VS Code Download Log

```
[rpc.0] Checking /home/kevin/.vscode/cli/servers/Stable-f1b07bd25dfad64b0167beb15359ae573aecd2cc/log.
   txt and /home/kevin/.vscode/cli/servers/Stable-f1b07bd25dfad64b0167beb15359ae573aecd2cc/pid.txt for
   a running server...
[rpc.0] Downloading Visual Studio Code server -> /tmp/.tmp9ANcrP/vscode-server-linux-x64.tar.gz
[rpc.0] Starting server...
[rpc.0] Server started
.. ..
```



Connecting to ubuntu22-vbox: Downloading VS Code Server...

Figure 5: **Viewing running extensions (in particular, extensions installed by default).**

Code for the Web context. Considering mainstream tasks and capabilities, VS Code for the Web is on par with its desktop sibling. (See also the "Managing Long-Running Tasks" boxout.)

VS Code Server in the Background

The previous command series demonstrated how to interact manually with VS Code for the Web running in a foreground context. To make this a robust configuration, place VS Code Server in the background, configure it to auto-start from a user context, and verify that VS Code is available in the shell search path:

```
which code
```

To configure VS Code to run in the background, install the back-end program as a service:

```
code tunnel service install ⮐
  --accept-server-license-terms ⮐
  --name remote32
```

Similar to the earlier example, the arguments `--accept-server-license-terms` and `--name <NAME>` are available for first-time setup. Omitting either command-line argument will result in interactive prompting before installation.

The output of this service-oriented installation step will generate

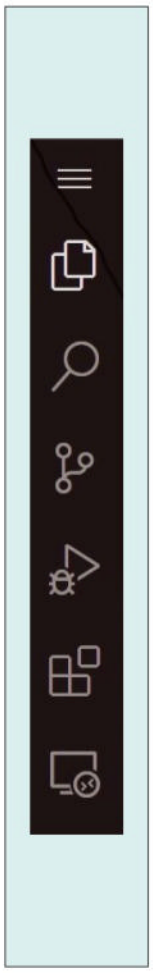


Figure 6: **Access the menu hierarchy from the hamburger menu.**

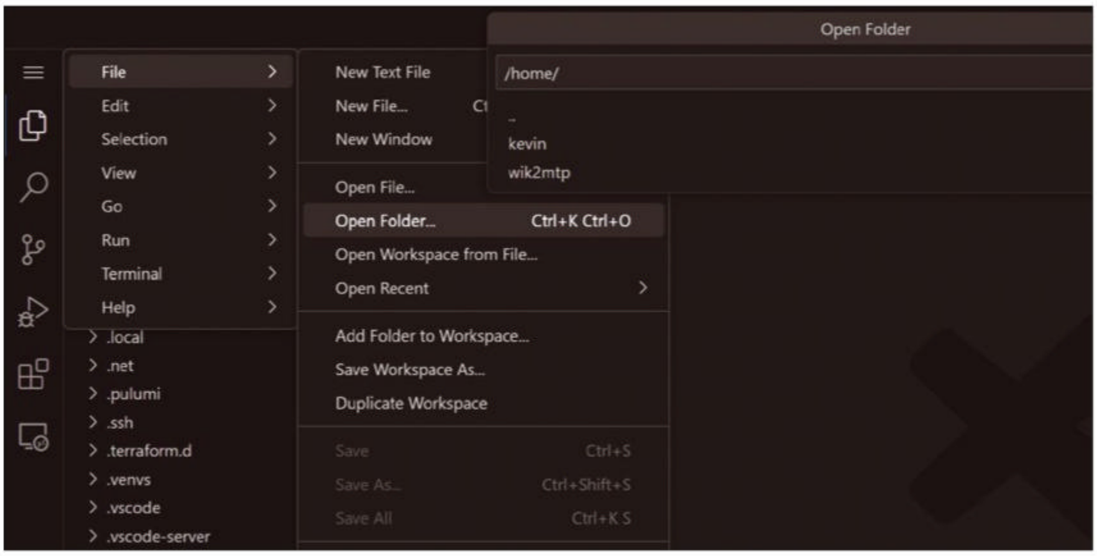
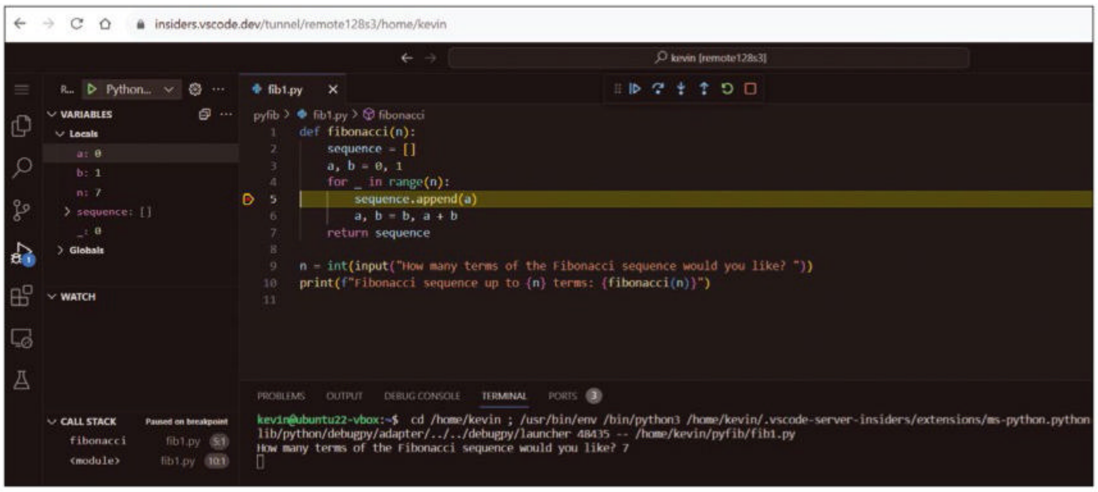


Figure 7: **Cascading *Open Folder* submenus.**



insiders.vscode.dev/tunnel/remote128s3/home/kevin

```
a: 0
b: 1
n: 7
> sequence: []
_: 0
```

```
def fibonacci(n):
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence

n = int(input("How many terms of the Fibonacci sequence would you like? "))
print(f"Fibonacci sequence up to {n} terms: {fibonacci(n)}")
```

```
kevin@ubuntu22-vbox:~$ cd /home/kevin ; /usr/bin/env /bin/python3 /home/kevin/.vscode-server-insiders/extensions/ms-python.python
lib/python/debugpy/adapter/../../debugpy/launcher 48435 -- /home/kevin/pyfib/fib1.py
How many terms of the Fibonacci sequence would you like? 7
```

Figure 8: **The debugger facilities rendered in VS Code for the Web.**

Managing Long-Running Tasks

By nature, remote tunnels are susceptible to disconnection events, especially if you are working in a dynamic environment where you might need to close your notebook. Consider the `screen` terminal multiplexer to manage such long-running tasks. Figure 9 illustrates the Redis project open in a VS Code for the Web session with the long-running `make` process executing in an integrated terminal. In this case, the build process is running for the Redis project, which is an extensive open source software system with many C source files and several build steps. In this scenario of using the `screen` utility, disconnection events will not cause the build process to bail out. Instead, the process can continue in the background, and the user or developer can later recall the screen as needed.

additional details regarding systemd unit files (Listing 2). Additionally, please pay attention to the `loginctl` tip it provided. Specifying the `enable-linger` command will enable this user-oriented background service to pre-initialize before any user login activity. In other words, the tunnel will be available in the background after a complete system startup, after a reboot, or after the user (or developer) that initiated this tunnel service logs out.

To keep track of user sessions commenced on a particular Linux host, use this:

```
sudo loginctl list-sessions
```

Also, note that you can monitor and observe the most recent log contents of the VS Code server with the command:

```
code tunnel service log
```

Listing 3 is example log output from the VS Code Tunnel service. You can check the status of the tunnel with the `status` argument, which is available to both the `code tunnel` and `code tunnel service` commands. This example uses the `tunnel service` context:

```
code tunnel service status
```

Listing 2: Back-End Installation Output

```
*
* Visual Studio Code Server
* ..
*
[2023-10-07 22:40:26] info Successfully registered service...
[2023-10-07 22:40:26] info Successfully enabled unit files...
[2023-10-07 22:40:26] info Tunnel service successfully started
[2023-10-07 22:40:26] info Tip: run `sudo loginctl enable-linger $USER` to
   ensure the service stays running after you disconnect.
```

The tunnel status output (Insiders edition of VS Code) is shown in Listing 4. You will see an error status if no tunnel process is running.

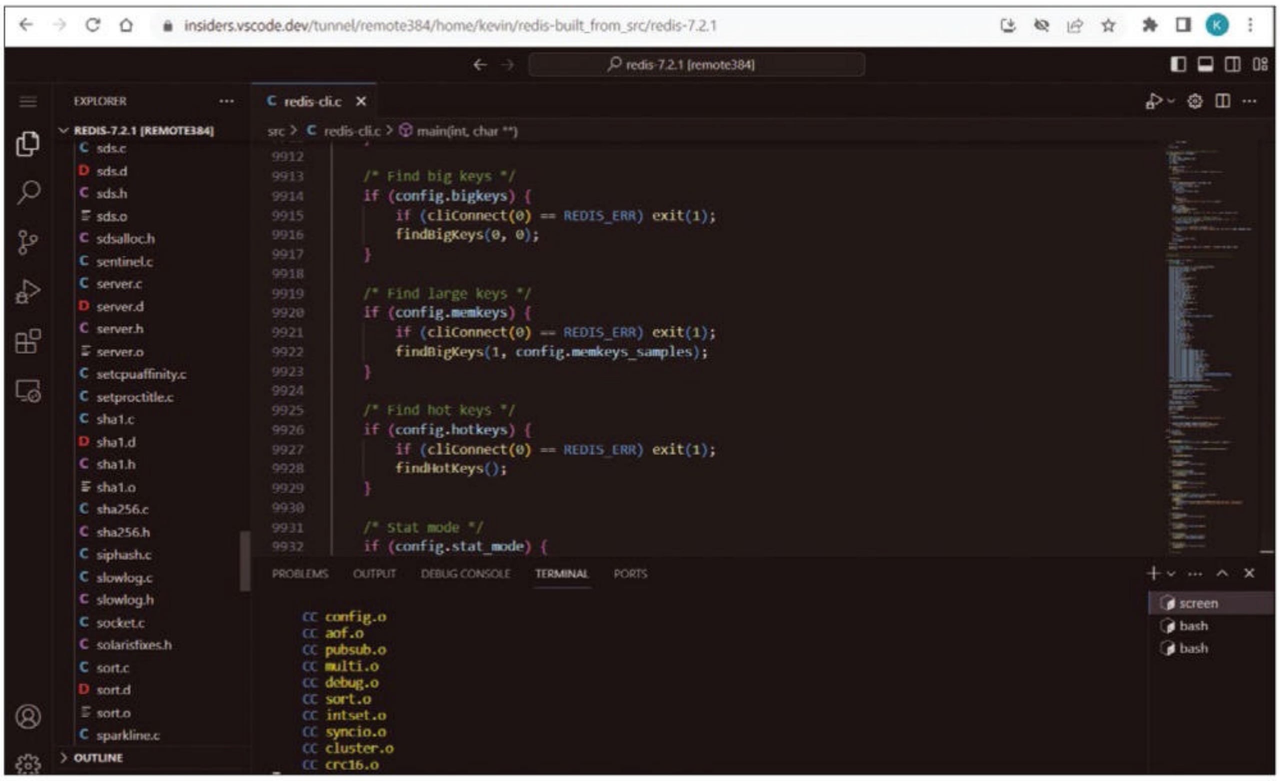
You can also check the status with `systemctl` by specifying the `--user` argument:

```
systemctl --user status code-tunnel.service
```

Note that this status command's output is identical to the previous tunnel log command.

Bouncing VS Code Server

In some instances you might need to restart the VS Code Server process. By bouncing the tunnel and monitoring the logs, you will observe that



```
    /* Find big keys */
    if (config.bigkeys) {
        if (cliConnect(0) == REDIS_ERR) exit(1);
        findBigKeys(0, 0);
    }

    /* Find large keys */
    if (config.memkeys) {
        if (cliConnect(0) == REDIS_ERR) exit(1);
        findBigKeys(1, config.memkeys_samples);
    }

    /* Find hot keys */
    if (config.hotkeys) {
        if (cliConnect(0) == REDIS_ERR) exit(1);
        findHotKeys();
    }

    /* Stat mode */
    if (config.stat_mode) {
```

```
    CC config.o
    CC aof.o
    CC pubsub.o
    CC multi.o
    CC debug.o
    CC sort.o
    CC intset.o
    CC syncio.o
    CC cluster.o
    CC crc16.o
```

Figure 9: Building the Redis open source project in VS Code for the Web.

Listing 3: VS Code Tunnel Log Output

```
code-tunnel.service - VS Code Tunnel
Loaded: loaded (/home/kevin/.config/systemd/user/code-tunnel.service; enabled; vendor preset: enabled)
Active: active (running) since .. ..
CGroup: /user.slice/user-1000.slice/user@1000.service/app.slice/code-tunnel.service
|-- 33011 /usr/local/bin/code --verbose --cli-data-dir /home/kevin/.vscode/cli tunnel service internal-run
|-- 35321 sh /home/kevin/.vscode/cli/servers/Stable-f1b07bd25dfad64b0167beb15359ae573aecd2cc/server/bin/code-server --connection-tok>
...
Oct 14 15:20:43 ubuntu22-vbox code[33011]: [2023-10-14 15:20:43] debug [tunnels::connections::ws] sent liveness ping
Oct 14 15:20:43 ubuntu22-vbox code[33011]: [2023-10-14 15:20:43] debug [tunnels::connections::ws] received liveness pong
```

this action initiates a new connection to vscode.dev. This process will also produce log messages that display the vscode.dev URL.

As previously mentioned, you can use `systemctl` commands to manage the service. To restart the service and check the service status, extracting the specific URL of the web-based tunnel, use:

```
systemctl --user restart code-tunnel.service
systemctl ↩
  --user status code-tunnel.service | ↩
    grep vscode.dev
```

On restarting the service, the console will display the link, typically formatted as *https://vscode.dev/tunnel*.

VS Code Server Hidden Parts

Like SSH remote tunneling, web-based tunneling methods also use hidden subdirectories for VS Code Server installation and runtime hosting. Initializing VS Code Server for the first time creates hidden subdirectories in the filesystem named `.vscode-server` and `.vscode`. In this context, the main subdirectory path that houses the server components is:

```
.vscode/cli/servers/Stable-<40-char-hash>
```

Listing 4: Tunnel Status Output

```
{"tunnel":
{"name":"remote32",
"started_at":"2023-10-14T03:49:25.040529254Z",
"tunnel":"Connected",
"last_connected_at":"2023-10-14T15:11:02.730231871Z",
"last_disconnected_at":null,
"last_fail_reason":null},
"service_installed":true}
```

If you have opted for an Insiders edition of Visual Code (i.e., the nightly build), the child server subdirectory will begin with the Insiders prefix.

A breakdown of the directory structure is:

```
.vscode
└── cli
└── servers
└── Stable--<40-char-hash>
└── server
├── bin
├── extensions
├── node_modules
└── out
```

Additionally, a hidden subdirectory called `.vscode-server` contains folders for data and extensions. Extensions added by the user will land in the extensions subdirectory:

```
.vscode-server
├── data
│ ├── CachedExtensionVSIXs
│ ├── CachedProfilesData
│ ├── logs
│ │ ...
│ ├── Machine
│ └── User
│ ├── globalStorage
│ ├── History
│ └── workspaceStorage
└── extensions
├── ms-python.python-x.y.z
└── ...
```

Installing Extensions from the CLI

Although many users opt to install extensions in the Extensions pane of the VS Code for the Web UI, performing this task from the command line is also possible. To do so, start by determining the subdirectory location of the VS Code Server instance relevant to the user context:

```
which code-insiders
/home/kevin/.vscode-insiders/cli/↩
  servers/Insiders-↩
  c72447e8d8aaa7497c9a4bd68bc4301584b92beb↩
  /server/bin/remote-cli/code-insiders
```

The output provides the path, which includes a unique 40-character hash, indicating the location of the VS Code Server. Navigate to the directory as shown in the output:

```
cd /<path-to-directory>/code-insiders
```

Use the `code-insiders` command to install your desired extension. For instance, to install the Python extension for this specific user instance, use:

```
code-insiders ↩
  --install-extension ms-python.python
```

Note that the installation of the `ms-python.python` extension will trigger the installation of Pylance. These extensions will land in the `.vscode-server/extensions` subdirectory.

Clean Up VS Code Server

The tunnel provides command-line capabilities to uninstall and delete VS

Code Server. From the command line, uninstall the service and remove the server-side bits:

```
code tunnel service uninstall
code tunnel unregister
code tunnel prune
```

The last two commands unregister the machine and environment (note that this requires a connection to the network and Internet) and remove the (local) VS Code Server instances. The hidden subdirectory will remain. Deleting the `vscode`-related hidden directories will purge the last remaining bits:

```
sudo rm -fr .vscode/
sudo rm -fr .vscode-server/
```

VS Code Server and its remote extensions provide significant advantages for remote development, but as the saying goes, with greater power comes greater responsibility. Integration with external vscode.dev cloud services are particularly beneficial for isolated, transient cloud server environments, making it ideal for individual developers or small teams in fast-paced development or testing settings. Such scenarios might be its most suitable application, keeping it distant from the critical systems of enterprise IT. ■

Info

[1] VS Code Marketplace: Remote – Tunnels: [https://marketplace.visualstudio.com/items?itemName=ms-vscode.remote-server]

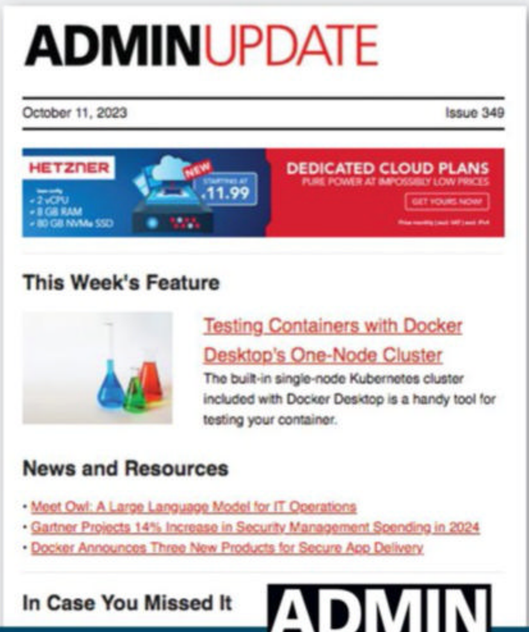
[2] Developing with Remote Tunnels: [https://code.visualstudio.com/docs/remote/tunnels]

[3] GitHub device login page: [https://github.com/login/device]

Author

Kevin Wittmer, a chief IT specialist at Bosch Group, has a strong affinity for all aspects of Visual Studio Code.

Too Swamped to Surf?



ADMINUPDATE

October 11, 2023 Issue 349

HETZNER DEDICATED CLOUD PLANS PURE POWER AT IMPOSSIBLY LOW PRICES €11.99

This Week's Feature

Testing Containers with Docker Desktop's One-Node Cluster

The built-in single-node Kubernetes cluster included with Docker Desktop is a handy tool for testing your container.

News and Resources

- Meet Owl: A Large Language Model for IT Operations
- Gartner Projects 14% Increase in Security Management Spending in 2024
- Docker Announces Three New Products for Secure App Delivery

In Case You Missed It

Our ADMIN Update newsletter delivers news and technical articles to your inbox every week.

Subscribe free, and you'll discover:

- practical articles on security, cloud computing, DevOps, and more
- IT industry news
- job listings for IT professionals
- IT industry events
- and more!

ADMIN Network & Security

bit.ly/HPC-ADMIN-Update

Incident response with Velociraptor

The Hunter

The software incarnation of the feared predator in the *Jurassic Park* movies has been on the hunt for clues to cyberattacks and indicators of compromise. We show you how to tame the beast and use it for your own purposes. By Matthias Wübbeling

From the IT department's point of view, it always makes sense to have an overview of your company's IT infrastructure – or at least be able to create one in a timely manner. In the immediate aftermath of an IT security incident, you need information quickly about which systems an attacker may have accessed and which systems are still operational. Department staff can then look specifically for indicators of compromise (IoCs) with the help of Velociraptor [1].

The developers cite two well-known tools as the basic idea for their own software: the GRR Rapid Response (GRR) [2] incident response tool and the OSQuery [3] monitoring tool. GRR lets you hunt for IoCs and run them over a period of time on all clients connected to your network. The reports are sent to a centralized server where they are available to admins. OSQuery, on the other hand, lets you query information from your clients in a language similar to SQL. The tool provides information in more than 275 tables – from CPU data to network settings (e.g, DNS or static routes) to installed Chrome extensions – you can find out pretty much everything about your systems.

Velociraptor now aims to combine the capabilities of GRR and OSQuery into one tool, while being faster, smaller, more scalable, and easier to install. Like GRR and OSQuery, the software works independent of the selected operating system and comes with virtually no dependencies. Beyond the functionality of GRR and OSQuery, it is possible for defined events to trigger queries and to use the Velociraptor Query Language (VQL), both to execute queries in the sense of OSQuery and to transfer files, modify systems and settings, and control the entire client-server infrastructure.

Quick Install

The architecture of a Velociraptor installation is simple: A centralized server maintains a permanent command and control connection to all the devices (clients) in your IT infrastructure. The entire installation is controlled over a web interface, which is where you configure settings, define and start hunts, and document and edit incidents.

To test Velociraptor, first install the server in a Docker container with the Compose tool. Prepared files are available online [4]. Clone the repository and adapt the ENV file to your environment. The web interface in the Docker container is accessible over port 8889. After opening it in the browser, you can accept the certificate self-signed by Velociraptor and enter the credentials stored in the ENV file for authentication. The default combination is *admin/admin*. Of course, you will want to change these credentials for production systems.

To install, simply use the appropriate binary for your operating system from the container. The `./velociraptor/clients` folder is included in the

Lead Image © Valentyn Ihnatkin, 123RF.com

container to help, and binaries with the corresponding certificates and configuration are created and stored at startup. For Microsoft systems, an MSI file lets you distribute to your clients by Active Directory.

For Linux systems, copy the `./velociraptor/clients/linuxvelociraptor_client` and `./velociraptor/client.config.yaml` files into a standard directory (e.g., `/tmp`). For macOS or Windows, look for the appropriate binary. Next, look into the configuration file and check the specified server URL and the settings that start with `writeback_`. Your local user must have write permissions to the directory configured there. If necessary, adjust the path (e.g., to `/tmp/etc/velociraptor.writeback.yaml` on Linux) and then create the `/tmp/etc/` folder with appropriate permissions.

Now start the client with the command

```
./velociraptor_client ⮐
  --config client.config.yaml client -v
```

which passes the configuration file with `--config` and activates the detailed output in the terminal with `-v`. Watch the client start up and then check the web browser to see whether the client is connected. If you cannot see the client directly in the display, use the search function at the top of the page. Just click on the magnifying glass without entering a search string, and you should get a list of connected clients. After clicking on the ID of the client, you are taken to the detailed view with further information, such as when and with which IP address the client logged in. You will also see the operating system, hostname, and architecture of the system. If you click *>_Shell* in the upper right corner, you can execute commands on the system if it is connected at the time. Try the commands

```
uname -a
id
```

and look at the return values by clicking on the eye icon. Clicking on the *Logs* link highlighted in green will take you to detailed information of the runtime.

Velociraptor Query Language

VQL is based on SQL and is used to control the entire environment. You can use it to query information from the clients, control monitoring and automated response technologies on the clients, and control the Velociraptor server itself. Because of space limitations, I only allow myself to request simple information. As before, you could send queries to a client in *>_Shell*; instead, select *VQL* from the drop-down list next to the input field and execute the query

```
SELECT * FROM info()
```

In the resulting table, you will see information for your client system. Instead of an asterisk (*), you can specify single fields or multiple fields separated by commas, as in SQL. Unlike SQL, however, you will not be using tables; rather, plugins present the information to you as a table. In this example, the query uses the `info` plugin. If you first enter a *?* in the input line instead of a plugin name, you will see a list of available plugins from which to choose. You can specify arguments in the parentheses of `info()` if the plugin requires additional information.

As with SQL, you can use a filter expression with `WHERE` to further narrow the results. VQL also supports aliases or subqueries, as well as constructs such as `if-then-else` or `foreach`. In this way, even complex queries can be displayed in a simple, structured manner.

Gone Hunting

The *Hunt Manager* in the sidebar is where you create hunts in a guided dialog; you might already be familiar with this procedure from GRR. After entering a short description and setting the selection criteria for the clients, select one or more artifacts, configure parameters for the hunt, and specify any runtime constraints, such as a share of processor time or a maximum runtime on individual clients. After you have checked your JSON-formatted search once again, launch it by clicking *Launch Hunt* and *Start*.

To search for the preselected artifacts and leverage the power of VQL at the same time, select *Generic.Client.VQL* as the artifact. You can enter arbitrary queries in the *Configure Parameters* item by clicking on the wrench icon. In the hunt overview you can then monitor the progress of your hunts and view the results.

If you're working on a recent incident, you will want to do more than run individual queries; in fact, you will probably want to document them systematically. Velociraptor offers notebooks for this purpose. Select the appropriate item in the left menu and create a new notebook by clicking the plus symbol (+). Notebooks consist of various cells, such as Markdown cells for documentation and VQL cells for query definition. The queries are executed directly. When combined with the information from the Markdown cells, you can create complete reports for your investigation in next to no time – and always with up-to-date results. Notebooks can also be shared with multiple users.

Conclusions

Velociraptor provides many useful features and a powerful query language to monitor and query almost all aspects of your IT infrastructure. The tool combines the functionality of GRR Rapid Response and OSQuery and extends both in a very useful way. ■

Info

[1] Velociraptor: [https://github.com/Velocidex/velociraptor]

[2] GRR: [https://github.com/google/grr]

[3] OSQuery: [https://github.com/osquery/osquery]

[4] Velociraptor files: [https://github.com/Velocidex/velociraptor/releases]

Agentless automation with Event-Driven Ansible

On Call

Event-Driven Ansible is a reactive extension that uses events to launch automations. We explain the ruleset and present examples that show how to monitor logs and call other tools. By Andreas Stolzenberger

The Ansible automation platform does not require an agent on the systems to be managed, which is both a curse and a blessing. Without a client, Ansible can control any system directly by SSH and Python, or even Windows Remote Management (WinRM) and PowerShell, whereas other platforms like Puppet or Salt need their agents. On the other hand, tools with agents can react quickly to incidents on the managed system because the client is in constant communication with the automation tool. These applications also take a "reactive" approach, which Ansible has been unable to do thus far because of its push-only architecture. The community's new Event-Driven Ansible (EDA) [1] project now aims to change this situation, while still doing without an agent.

Modular Architecture

Red Hat first introduced the concept of EDA as a Developer Preview at the AnsibleFest user and developer conference in Fall 2022. EDA adapts the "state machine" concept as used by ManageIQ, among others, wherein the state machine is triggered by an event (e.g., a log message with *error*). It then checks one or more conditions (e.g., `if alerting.service=dns`) and then initiates appropriate actions (e.g., `restart named.service`).

EDA listens for events from one or more sources. Like regular Ansible, EDA is modular and can handle different sources. As of today, very few of these source plugins are around; then again, EDA is still in its infancy. The EDA website documents how you can program your own source plugins [2]. Therefore, the number of available sources should increase steadily now that EDA has been officially released [3]. Spoiler: This article is based on a developer prerelease version of EDA. Some of the procedures described here may have changed in the official release.

The plugins already available include, for example, `ansible.eda.webhook`, which lets Ansible tap into web applications such as GitHub. A Git commit in a web repository can therefore trigger a playbook in the data center. The powerful `ansible.eda.kafka` source plugin lets EDA tap into the Apache Kafka message bus. I use this plugin in one of the examples here. Additionally, some initial plugins from commercial manufacturers have already been made available, such as `ansible.eda.dynatrace`, `ansible.eda.aws_cloudrail`, and `ansible.eda.azure_service_bus`.

Others under development as of May 2023 were not in the beta build I used when I wrote this article, such as `ansible.eda.journald` for listening to systemd log messages. However, by the publication date of this article, this plugin should be usable.

Ruleset

EDA introduces a new variant of Ansible's YAML-based programming

Lead Image © Valery Kachaev, 123RF.com

language for playbooks in the form of rulebooks, in which you describe the sources to which you want EDA to listen and how it will react to events. They are executed by a program that was also newly developed: `ansible-rulebook` [4]. Rulebooks are not launched like playbooks. A rule runs permanently, like a daemon, because it permanently monitors the specified source.

In practice, users are likely to use rulebooks in containers. In addition to the rulebook runtime, the containers also have `ansible-runner` to execute the defined action playbooks in the active container. Alternatively, the rulebook addresses the API of an Ansible controller (AWX instance) to launch a job template there. As of writing, EDA still worked separately from these web user interface (UI) instances. Integration was planned to take place after the official release.

Numerous Dependencies

The regular Ansible playbook gets by with Python plus some Python libraries, but the current pre-build of EDA took me into entirely different realms of code and runtime dependencies. Because Python (from version 3.9) alone is not sufficient for EDA, it uses the free Drools [5] business rules management engine. Besides Python, the EDA container also required version 17 of the Java runtime for Drools.

For the Python part of EDA to be able to communicate with its Java part, the Java-to-Python bridge `jpy` [6] is also required. To compile `jpy` in the container, you then need the Maven software project manager, whose RPM package in Enterprise Linux 8 (EL8) requires the old Java 1.8 runtime instead of 17. In other words, the EDA Developer container came with Python 3.9 including the GCC development environment, `make`, and two Java runtimes. This was also to change before the official release.

Fortunately, at the time of writing, a ready-made container image for EDA testing was already on `quay.io/ansible/ansible-rulebook:main`. The `main` tag is important here; the image with the `latest` tag used an even older version.

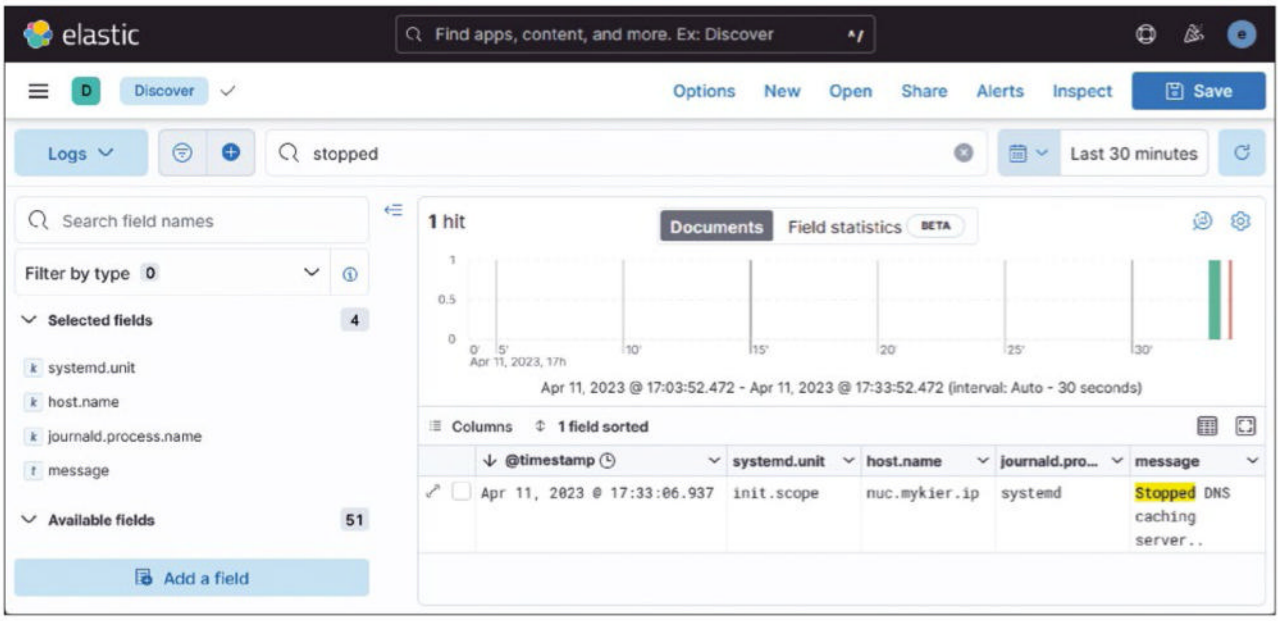
Log Monitoring by Message Bus

For this practical example, EDA monitors the system logs of multiple servers and responds to a wide variety of log entries. To access the logs, EDA could use the `ansible.eda.journald` source plugin; however, it would then have to keep active connections to all monitored servers open. In many places, administrators collect all logfiles in a centralized database anyway.

To ensure that the log data is not only available to a single service such as Logstash, you can deploy a message bus in the middle – in the best scale-out style: Apache Kafka. All log information passes through the message broker and target systems such as the log collection database, and EDA also connects to the message bus as a "subscriber."

If you are not yet familiar with this message bus technology, just think of Kafka as WhatsApp for applications. The logging systems join a chat group (topic) and post all their log entries in this group. Now "subscribers" such as EDA can also join the group and view and evaluate all the messages. In parallel, a log aggregator such as Logstash can retrieve all the messages in the group and forward them to a database such as Elasticsearch for archiving.

In this example, the Filebeat [7] tool first runs as a log shipper on the servers to be monitored. It takes care of turning unstructured log entries of system services into semi-structured messages such as `Service=sshd` or `Message=Failed Login` before Filebeat forwards them. The tool also caches all messages if the connection to the message bus goes down so that nothing is lost.



stopped

1 hit

Apr 11, 2023 @ 17:03:52.472 - Apr 11, 2023 @ 17:33:52.472 (interval: Auto - 30 seconds)

@timestamp	systemd.unit	host.name	journald.pro...	message
Apr 11, 2023 @ 17:33:06.937	init.scope	nuc.mykier.ip	systemd	Stopped DNS caching server..

Figure 1: Logstash can retrieve logs from the Kafka message bus and back them up to Elasticsearch.

In many installations, Filebeat sends the data directly to Logstash, so it is not available to other services. However, switching to the message bus as a middleman does not involve too much overhead (Figure 1). It makes sense to do this in distributed, hybrid environments with multiple clouds and data center setups because it reduces the connections needed between the separate networks. Filebeat-Logstash turns into Filebeat-Kafka-Logstash, with the option for other services to listen in on Kafka.

From Filebeat to Kafka

Although large installations run Kafka as a cluster, a single container is fine for the test setup. You can optionally log the raw messages of the bus to a persistent volume. Messages usually end up with a log aggregator such as Logstash anyway, which is why the Kafka container works well in a stateless setup (i.e., without permanent storage).
Follow the instructions on the page of the image [8] and use the setup from the "Using the command line" section, which will set up kafka with "KRaft" as quorum instead of Zookeeper. Unlike the example, enter the external IP address of your server where you run the Kafka container with Docker or Podman. Add the an additional configuration line to advertise the external IP of your Kafka-Server.

```
KAFKA_CFG_ADVERTISED_LISTENERS=↩
  PLAINTEXT://192.168.2.12:9092
```

(if your Docker/Podman server is running on 192.168.2.12). In a production environment, of course, the Kafka setup would not run in plaintext and would only allow authorized and encrypted connections. Once the Kafka container is running, configure the Filebeat clients on the servers to send their log information to Kafka. The `/etc/filebeat/filebeat.yml` file then looks something like:

```
filebeat.inputs:
- type: journald
  id: everything

output.kafka:
  hosts: ["192.168.2.12:9092"]
  topic: 'journals'
  partition.round_robin:
    reachable_only: false
```

Filebeat sends all the log messages to the message broker *journals* topic. Alternatively, Filebeat could send messages to different topics depending on the content, such as with the optional entry:

```
topics:
  - topic: "critical"
    when.contains:
      message: "CRITICAL"
```

As soon as Filebeat sends log data you can check – on the host with the Kafka container – whether the information is reaching the message bus correctly. To do this, simply run the console consumer in the active Kafka container, `kafka`:

```
podman exec ↩
  -it kafka kafka-console-consumer.sh ↩
  --bootstrap-server localhost:9092 ↩
  --topic journals --from-beginning
```

Now you should see the JSON-formatted log entries of your systems with the Filebeat setup. If you optionally want to collect all the log information with Logstash, the lines:

```
input {
  kafka{
    codec => json
    bootstrap_servers => "192.168.2.12:9092"
    topics => ["journals"]
  }
}
```

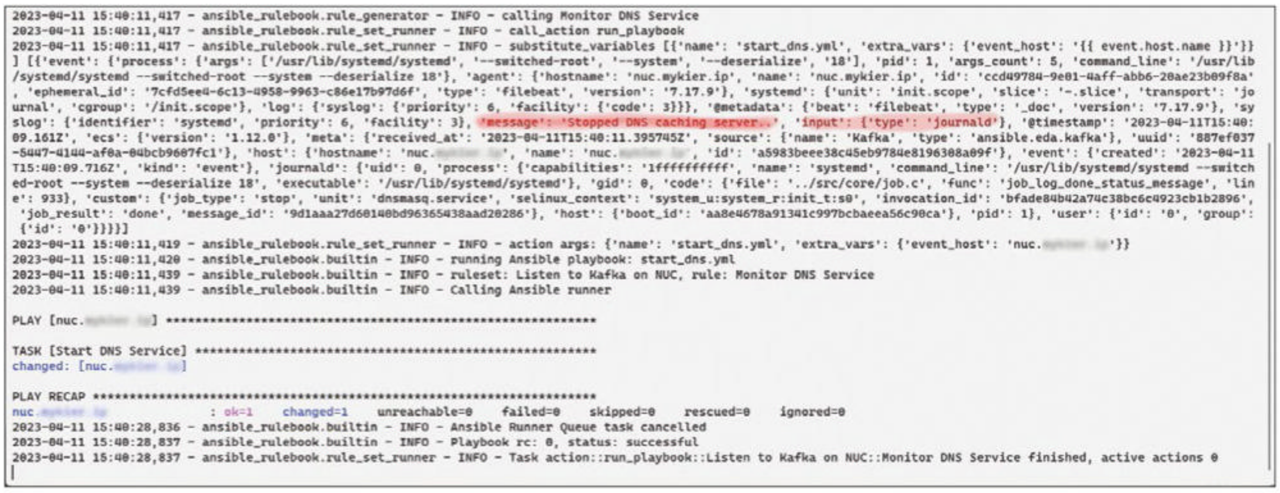
in the Logstash configuration are all it takes.

Creating an Inventory

In a terminal, you can start the EDA container interactively with:

```
podman run -it --name eda ↩
  --volume /home/user/rules:rules:Z ↩
  quay.io/ansible/ansible-rulebook:main ↩
  /bin/bash
```

Now you can edit the rulebooks and playbooks on the host machine and test them in the interactive container.



```
2023-04-11 15:40:11,417 - ansible_rulebook.rule_generator - INFO - calling Monitor DNS Service
2023-04-11 15:40:11,417 - ansible_rulebook.rule_set_runner - INFO - call_action run_playbook
2023-04-11 15:40:11,417 - ansible_rulebook.rule_set_runner - INFO - substitute_variables [{'name': 'start_dns.yml', 'extra_vars': {'event_host': '{{ event.host.name }}'}}
] [{'event': {'process': {'args': ['/usr/lib/systemd/systemd', '--switched-root', '--system', '--deserialize', '18'], 'pid': 1, 'args_count': 5, 'command_line': '/usr/lib
/systemd/systemd --switched-root --system --deserialize 18'}, 'agent': {'hostname': 'nuc.mykier.ip', 'name': 'nuc.mykier.ip', 'id': 'ccd49784-9e01-4aff-abb6-20ae23b09f8a'
, 'ephemeral_id': '7cfd5ee4-6c13-4958-9963-c86e17b97d6f', 'type': 'filebeat', 'version': '7.17.9'}, 'systemd': {'unit': 'init.scope', 'slice': '-.slice', 'transport': 'jo
urnal', 'cgroup': '/init.scope'}, 'log': {'syslog': {'priority': 6, 'facility': {'code': 3}}}, '@metadata': {'beat': 'filebeat', 'type': '_doc', 'version': '7.17.9'}, 'sy
slog': {'identifier': 'systemd', 'priority': 6, 'facility': 3}, 'message': 'Stopped DNS caching server..', 'input': {'type': 'journald'}, '@timestamp': '2023-04-11T15:40:
09.161Z', 'ecs': {'version': '1.12.0'}, 'meta': {'received_at': '2023-04-11T15:40:11.395745Z', 'source': {'name': 'Kafka', 'type': 'ansible.eda.kafka'}, 'uuid': '887ef037
-5447-4144-af0a-04bcb9607fc1'}, 'host': {'hostname': 'nuc.[redacted]', 'name': 'nuc.[redacted]', 'id': 'a5983beee38c45eb9784e8196308a09f'}, 'event': {'created': '2023-04-11
T15:40:09.716Z', 'kind': 'event'}, 'journald': {'uid': 0, 'process': {'capabilities': '1ffffffffff', 'name': 'systemd', 'command_line': '/usr/lib/systemd/systemd --switch
ed-root --system --deserialize 18', 'executable': '/usr/lib/systemd/systemd'}, 'gid': 0, 'code': {'file': '../src/core/job.c', 'func': 'job_log_done_status_message', 'lin
e': 933}, 'custom': {'job_type': 'stop', 'unit': 'dnsmasq.service', 'selinux_context': 'system_u:system_r:init_t:s0', 'invocation_id': 'bfade84b42a74c38bc6c4923cb1b2896',
 'job_result': 'done', 'message_id': '9d1aaa27d60140bd96365438aad20286'}, 'host': {'boot_id': 'aa8e4678a91341c997bcbaeea56c90ca'}, 'pid': 1}, 'user': {'id': '0', 'group':
 {'id': '0'}}}}]
2023-04-11 15:40:11,419 - ansible_rulebook.rule_set_runner - INFO - action args: {'name': 'start_dns.yml', 'extra_vars': {'event_host': 'nuc.[redacted]'}}
2023-04-11 15:40:11,420 - ansible_rulebook.builtin - INFO - running Ansible playbook: start_dns.yml
2023-04-11 15:40:11,439 - ansible_rulebook.builtin - INFO - ruleset: Listen to Kafka on NUC, rule: Monitor DNS Service
2023-04-11 15:40:11,439 - ansible_rulebook.builtin - INFO - Calling Ansible runner

PLAY [nuc.[redacted]] **************************************************************

TASK [Start DNS Service] ***********************************************************
changed: [nuc.[redacted]]

PLAY RECAP *************************************************************************
nuc.[redacted]              : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
2023-04-11 15:40:28,836 - ansible_rulebook.builtin - INFO - Ansible Runner Queue task cancelled
2023-04-11 15:40:28,837 - ansible_rulebook.builtin - INFO - Playbook rc: 0, status: successful
2023-04-11 15:40:28,837 - ansible_rulebook.rule_set_runner - INFO - Task action::run_playbook::Listen to Kafka on NUC::Monitor DNS Service finished, active actions 0
|
```

Figure 2: The Kafka messaging bus's *Stopped DNS* event triggers the Ansible rule. The triggered playbook restarts the failed service.

By the way, this even works on Windows if you use a WSL distribution with Podman.

To begin, create an inventory in `/home/user/rules`. EDA currently only supports the YAML format for inventories and not the old INI format. In the test, two systems provide log information to Kafka, so the `inventory.yml` file looks like:

```
all:
  children:
    servers:
    hosts:
      srv1.local.ip:
        ansible_host: 192.168.2.1
      srv2.local.ip:
        ansible_host: 192.168.2.2
  vars:
    ansible_user: root
```

For the tests, Ansible logs in to the remote systems as the root user. In a production environment, there would be a `user` directory with sudo privileges and the playbook would use `become`.

I list the hosts with their fully qualified domain names (FQDNs) in the inventory because that is how they are listed in the log entries on the Kafka bus and can therefore be assigned to the inventory. However, you need to specify the IP address for the Ansible connection. The example here shows how EDA can react in case of a DNS failure; in that case, it has to contact the host by the IP address, of course.

Setting Rules

The test setup can basically react to any log entry and run a matching Ansible playbook. This example restarts the `dnsmasq` DNS/DHCP server if it fails for any reason. To do this, EDA monitors the log messages from systemd. The `rule_dns.yml` rulebook starts with the Kafka source:

```
- name: Kafka Monitor
  hosts: all
  sources:
    - name: Kafka
      ansible.eda.kafka:
        host: 192.168.2.12
        port: 9092
        topic: journals
```

With this rulebook, `ansible-rulebook` later taps into the message bus and forwards incoming messages in the hierarchical variable `events`. Sources also have `filters` that can change the content or structure of the variables:

```
filters:
  - json_filter:
    exclude_keys: ['user']
```

For example, it would remove all `event.user` values from the source event. An important filter is `insert_hosts_to_meta`, which takes one or more host names from the source

A Webzine for High-Performance Computing Specialists

HPC UPDATE

September 12, 2023 Issue 176

HPC ILLUMINATIONS PAVILION
A New Opportunity for Underrepresented Groups at SC23
LEARN MORE & APPLY

This Month's Feature

Where Does Job Output Go?
By Jeff Layton
Where does your job data go? The answer is fairly straightforward, but Jeff evolves the question to include a discussion of where data "should" or "could" go.

News and Resources
• RIKEN Brings AI to Quantum Error Correction
• Submissions Are Open for ISC 2024
• Math Magic with MathLex

ADMIN Network & Security

If you work with high-performance clusters, or if you're ready to expand your skill set with how-to articles, news, and technical reports on HPC technology.

admin-magazine.com/HPC

message and then uses them as limits when executing a playbook. An Ansible playbook started by EDA will therefore only contact those hosts previously transferred to the `event.meta.host` variable by `insert_hosts_to_meta`. This example does not use the filter, though, simply because it was not yet included in the test build of `ansible-rulebook` (0.11) used.
The test rule responds to the message that the DNS server has been stopped:

```
rules:
  - name: Monitor DNS Service
    condition: event.message is search(⮐
      "Stopped DNS", ignorecase=true)
```

If a message on the message bus now contains *Stopped DNS* (**Figure 2**), EDA steps in and runs an action:

```
action:
  run_playbook:
    name: start_dns.yml
    extra_vars:
      event_host:
        "{{ event.host.name }}"
```

In the example, EDA only starts one action. The rulebook could alternatively use the `Actions` keyword and then perform several actions in sequence. Because the complete event variable of the rulebook is not automatically available for the playbook, you have to use `extra_vars` to pass information from the rulebook variable into the playbook.
The referenced playbook starts the DNS server but needs to know on which host to start the service. As mentioned before, because the `limit` filter was not working when this example was created, I used a simple hack in the playbook instead, which explains why the `start_dns.yml` playbook looks like it does:

```
- hosts: "{{ event_host }}"
  gather_facts: no

  tasks:
  - name: Start DNS Service
    ansible.builtin.service:
      name: dnsmasq
      state: started
```

It simply takes the value from the event as the host variable and only performs the actions on the host that triggered the event.

Simple with Huge Potential

The example given is quite simple, but it shows the great potential of EDA. The rulebook used is, of course, not limited to one rule. You can react with further rules to combinations of source events and link the conditions to several sources (AND/OR) with something like:

```
condition:
  all:
    - event.host.name == "srv1.local.ip"
      - event.journald.process.name == ⮐
        "systemd"
      - event.systemd.unit == ⮐
        "dnsmasq.service"
      - event.message is search(⮐
        "Stopped DNS", ignorecase=true)
```

The event only triggers if all of the specified values match (AND). The `any` keyword starts the action if one of the specified conditions is true (OR). Actions also have more options. While developing your own rulebooks, you will often use `print_event` to view the complete event variable or parts of it and adjust your rules and playbooks accordingly.
The `start_playbook` action uses Ansible Runner to run a playbook in the EDA container. In the future, however, it will be far more interesting to launch an existing job template on an Ansible controller or an AWX setup, for which you can turn to the `run_job_template` action; it connects to an existing controller or AWX system with a URL and a token. Sooner or later, EDA is likely to find its way into the Controller and AWX web UIs and become a part of those tools.

Conclusions

Event-Driven Ansible architecture continues the idea of automating target systems without an agent. However, any kind of practical implementation is still complicated at the current early stage. Drools is a powerful rules engine and was definitely undertasked with EDA's previous simple capabilities. The question inevitably arises as to whether EDA really needs the complex setup with Python, Java, and the JPY bridge.
On the other hand, EDA is likely to add massive functionality in future versions, precisely because it is based on such a powerful rules engine. Thanks to this modular, open concept, it will be possible to use EDA in many scenarios with many different sources in the future – as long as users and developers continue to develop the tool and provide additional source plugins. ■

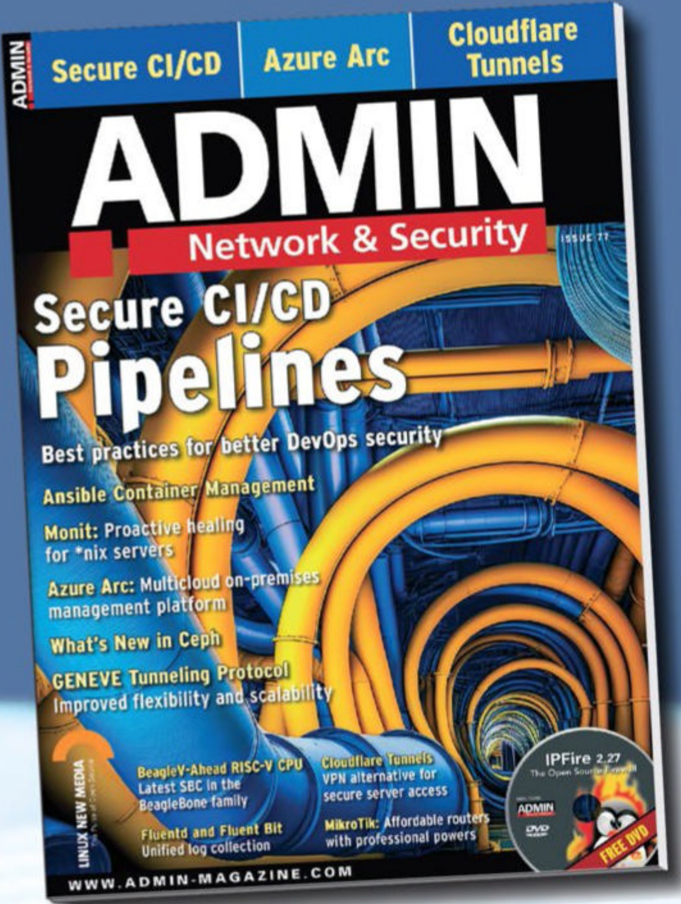
Info

[1] Event-Driven Ansible: [https://github.com/ansible/event-driven-ansible]
[2] Event source plugins: [https://ansible.readthedocs.io/projects/rulebook/en/stable/sources.html]
[3] EDA release: [https://www.ansible.com/blog/event-driven-ansible-is-here]
[4] Ansible rulebook docs: [https://ansible.readthedocs.io/projects/rulebook/en/stable/index.html]
[5] Drools: [https://www.drools.org]
[6] jpy: [https://pypi.org/project/jpy/]
[7] Filebeat: [https://www.elastic.co/beats/filebeat]
[8] Apache Kafka: [https://github.com/bitnami/containers/blob/main/bitnami/kafka/README.md]

The Author

Andreas Stolzenberger worked as an IT magazine editor for 17 years. He was the deputy editor in chief of the german *Network Computing* Magazine from 2000 to 2010. After that, he worked as a solution engineer at Dell and Vmware. In 2012 Andreas moved to Red Hat. There, he currently works as principal solution architect in the Technical Partner Development department.

REAL SOLUTIONS FOR REAL NETWORKS



ADMIN is your source for technical solutions to real-world problems.

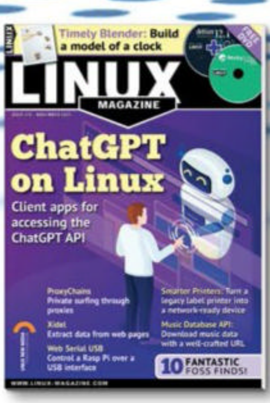
Improve your admin skills with practical articles on:

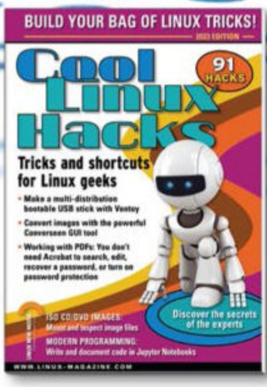
- Security
- Cloud computing
- DevOps
- HPC
- Storage and more!

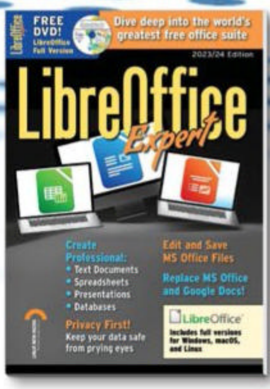
SUBSCRIBE NOW!

shop.linuxnewmedia.com

@adminmagazine @adminmag ADMIN magazine @adminmagazine







CHECK OUT OUR FULL CATALOG: SHOP.LINUXNEWMEDIA.COM

What's your status (page)?

Custodian

Businesses with modern IT infrastructures can keep track of internal and external servers and services with black box monitoring by Monitoror, Vigil, and Statping-ng. By Ankur Kumar

Keeping the lights on round the clock in a modern IT infrastructure is pretty complicated. The usual procedure consists of running both internal and external servers and services to deliver end products and services. Keeping a close watch on each and every element of the infrastructure is a necessity for any technology-driven business.

The modern monitoring solutions are devised to address the critical need to be proactive, rather than reactive, and spot problems before failures can impair a business. Having a status page for internal and external servers and services that provides a quick overview of what's failing where can keep IT teams on top of their infrastructure.

Most enterprise monitoring solutions are overkill and way too expensive for many companies, especially small to medium-sized businesses. In this article, I look at some amazing, free and open source solutions that set up various kinds of status pages performing black box monitoring. The only requirement to test these solutions on your IT infrastructure is a running Docker engine, which is pretty common nowadays.

Monitoror Wallboard

The first free open source status page solution I will talk about is Monitoror [1]. It is know as a monitoring wallboard because it is a single-page app comprising different colored rectangular tiles. Monitoror is mainly concerned with three kinds of general-purpose monitoring checks: ping, port, and HTTP.

The ping check verifies connectivity to a configured host, the port check verifies port listening on a configured endpoint, and the HTTP checks GET requests to a URL. It also has special built-in checks for Azure DevOps, GitHub, GitLab, Jenkins, Pingdom, and Travis CI (continuous integration).

The wallboard highlights the configured tiles either in green or red according to a respective check passing or failing. To see Monitoror yourself in action, use a terminal command to create a Docker network for test container(s):

```
docker network create statuspage-demo
```

Next, create the `monitoror_stack.yml` and `config.json` files (Listings 1 and 2) [2] to launch a Monitoror stack and supply its configuration, respectively.

The Monitoror configuration file defines an arrangement of desired monitoring tiles in a given number of columns. If the columns are fewer than the number of tiles, then the screen is filled vertically, too. An array of tiles is defined to contain various monitoring checks on the wallboard. The `PING`, `PORT`, and `HTTP-STATUS` tiles are self-explanatory. A tile of type `GROUP` is defined that shows a single rectangular area to represent multiple checks. This

Listing 1: `monitoror_stack.yml`

```
version: '3.5'
services:
  monitoror:
    image: monitoror/monitoror:${MTRRTAG:-latest}
    ports:
      - "38080:8080"
    environment:
      - "MO_CONFIG=/etc/config.json"
    restart: unless-stopped

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

Lead Image © Tatiana Venkova, 123RF.com

tile will be red when single or multiple checks in the group fail. This kind of tile makes use of the limited page area, so you can pack more checks into the wallboard.

The Monitoror documentation has complete information about tiles that can be monitored, along with their respective parameters, the information displayed, and so on. Even if you need to run Monitoror natively, just grab the appropriate Golang static binary from its GitHub releases page and run the binary after making it executable. To launch the Monitoror container and supply the required Monitoror configuration to demonstrate the tiles monitoring localhost in its container and itself running on port 8080, execute the two commands in Listing 3.

Now when you access the Monitoror wallboard page in your browser at *localhost:38080*, you should see a page with monitoring tiles (Figure 1).

I intentionally provided false IP addresses in the demo config file to show how failing tiles look on the wallboard. Monitoror picks up config changes during its periodic checks. To see how the tiles change, with changing input, correct the invalid IP addresses and introduce a typo in the `HTTP-STATUS` tile by modifying the `config.json` as in Listing 4.

When you provide Monitoror the new configuration (second line of Listing 3), the wallboard should reflect the new configuration (Figure 2). Correcting the typo in the HTTP tile URL and copying the new configuration should turn all tiles on the wallboard green.

The second demo arrays the Monitoror wallboard with tiles monitoring some popular modern cloud servers. To launch single monitoring instances for OpenSearch, Kafka, and Redis, modify the `monitoror_stack.yml` file as in Listing 5 and the `config.json` file as in Listing 6. Use the commands in Listing 3 to launch and copy the Monitoror config. You should see the wallboard with the new server tiles (Figure 3). You should now be feeling at home with the elegant capabilities of Monitoror that let you get a monitoring wallboard up and running. The command

```
docker run -it --rm -v /var/run/docker.sock:/var/run/↩
  docker.sock:ro -v ./monitoror_stack.yml:/etc/compose/↩
  monitoror_stack.yml:ro docker docker compose ↩
  -f /etc/compose/monitoror_stack.yml down
```

Listing 2: config.json

```
{
  "version": "2.0",
  "columns": 2,
  "tiles": [
  { "type": "PING", "params": {"hostname": "127.0.0.1"}},
  { "type": "PORT", "params": {"hostname": "129.0.0.1", "port": 8080}},
  { "type": "HTTP-STATUS", "params": {"url": "https://google.com"}},
  {
    "type": "GROUP",
    "label": "localhost PING/PORT/HTTP Tests",
    "tiles": [
    {
      "type": "PING",
      "params": {
        "hostname": "128.0.0.1"
      }
    },
    {
      "type": "PORT",
      "params": {
        "hostname": "127.0.0.1",
        "port": 8080
      }
    },{
  "version": "2.0",
  "columns": 2,
  "tiles": [
  { "type": "PING", "params": {"hostname": "127.0.0.1"}},
  { "type": "PORT", "params": {"hostname": "129.0.0.1", "port": 8080}},
  { "type": "HTTP-STATUS", "params": {"url": "https://google.com"}},
  {
    "type": "GROUP",
    "label": "localhost PING/PORT/HTTP Tests",
    "tiles": [
    {
      "type": "PING",
      "params": {
        "hostname": "128.0.0.1"
      }
    },
    {
      "type": "PORT",
      "params": {
        "hostname": "127.0.0.1",
        "port": 8080
      }
    },
    {
      "type": "HTTP-STATUS",
      "params": {
        "url": "http://localhost:8080"
      }
    }
  ]
  }
  ]
}
```

Listing 3: Monitoror Container and Config

```
docker run -it --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v
  ./monitoror_stack.yml:/etc/compose/monitoror_stack.yml:ro docker docker
  compose -f /etc/compose/monitoror_stack.yml up -d
docker run -it --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v
  ./config.json:/etc/monitoror/config.json:ro -v ./monitoror_stack.yml:/
  etc/compose/monitoror_stack.yml:ro docker docker compose -f /etc/compose/
  monitoror_stack.yml cp /etc/monitoror/config.json monitoror:/etc/config.json
```

Listing 4: Diff of config.json

```
5,6c6,7
<   { "type": "PORT", "params": {"hostname": "129.0.0.1", "port": 8080}},
<   { "type": "HTTP-STATUS", "params": {"url": "https://google.com"}},
---
>   { "type": "PORT", "params": {"hostname": "127.0.0.1", "port": 8080}},
>   { "type": "HTTP-STATUS", "params": {"url": "https://gogle.com"}},
15c15
<           "hostname": "128.0.0.1"
---
>           "hostname": "127.0.0.1"
```

cleans up the running stack once you're done playing with Monitoror.

Vigil Status Page

The Monitoror single-page wallboard is a quick and nice solution but has limited capabilities and is suited to a relatively smaller number of servers and services. The next free open source status page solution, Vigil [3], is more mature and capable of handling large numbers of servers and services with additional capabilities, including branding, alerting, announcements, and other options. To bring up Vigil quickly to see it in action, create the YML and CFG files shown in Listings 7 and 8.

This Vigil configuration with minimal necessary settings is self-explanatory. The `[server]` section controls on which IP and port

Listing 5: Diff for `monitoror_stack.yml`

```
8a10,27
>       - "MO_MONITORABLE_HTTP_SSLVERIFY=false"
>     restart: unless-stopped
>
>   opensearch:
>     image: opensearchproject/opensearch:${OSRHTAG:-latest}
>     environment:
>       - "discovery.type=single-node"
>     restart: unless-stopped
>
>   kafka:
>     image: bitnami/kafka:${KFKATAG:-3.2.3}
>     environment:
>       - "ALLOW_PLAINTEXT_LISTENER=yes"
>       - "KAFKA_CFG_LISTENERS=PLAINTEXT://0.0.0.0:9092,CONTROLLER://:9093"
>       - "KAFKA_ENABLE_KRAFT=yes"
>     restart: unless-stopped
>
>   redis:
>     image: redis:${RDSSTAG:-latest}
>     command: "redis-server --save 60 1 --loglevel warning"
```

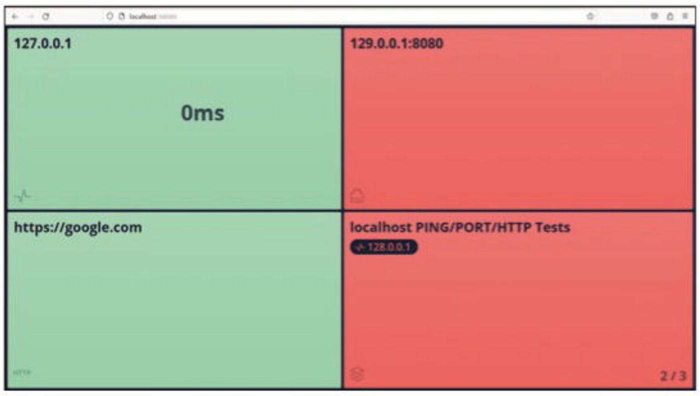


Figure 1: Monitoror wallboard page.

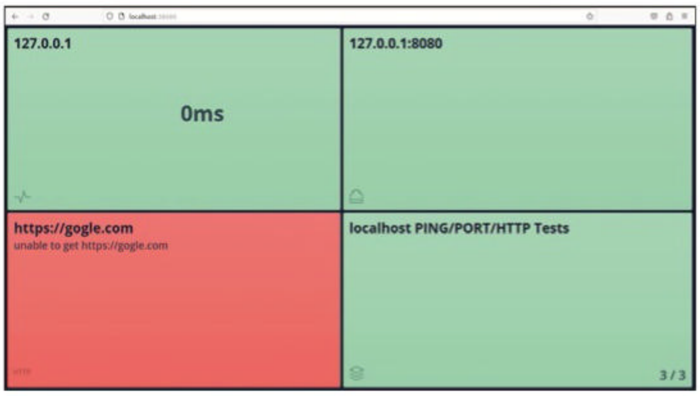

127.0.0.1
0ms
127.0.0.1:8080
https://gogle.com
unable to get https://gogle.com
localhost PING/PORT/HTTP Tests
3/3

Figure 2: Monitoror wallboard page with corrected IPs.

Listing 6: Diff for `config.json`

```
<   "columns": 2,
---
>   "columns": 3,
5,7d4
<   { "type": "PING", "params": {"hostname": "127.0.0.1"}},
<   { "type": "PORT", "params": {"hostname": "127.0.0.1", "port": 8080}},
<   { "type": "HTTP-STATUS", "params": {"url": "https://google.com"}},
10c7
<     "label": "localhost PING/PORT/HTTP Tests",
---
>     "label": "opensearch PING/PORT/HTTP Tests",
12,30c9,28
<       {
<         "type": "PING",
<         "params": {
<           "hostname": "129.0.0.1"
<         }
<       },
<       {
<         "type": "PORT",
<         "params": {
<           "hostname": "127.0.0.1",
<           "port": 8080
<         }
<       },
<       {
<         "type": "HTTP-STATUS",
<         "params": {
<           "url": "http://localhost:8080"
<         }
<       }
---
>       {"type": "PING", "params": {"hostname": "opensearch"}},
>       {"type": "PORT", "params": {"hostname": "opensearch", "port": 9200}},
>       {"type": "PORT", "params": {"hostname": "opensearch", "port": 9600}},
>       {"type": "HTTP-STATUS", "params": {"url": "https://admin:admin@
              opensearch:9200"}}
>     ]
>   },
>   {
>     "type": "GROUP",
>     "label": "kafka PING/PORT Tests",
>     "tiles": [
>     {"type": "PING", "params": {"hostname": "kafka"}},
>     {"type": "PORT", "params": {"hostname": "kafka", "port": 9092}}
>     ]
>   },
>   {
>     "type": "GROUP",
>     "label": "redis PING/PORT Tests",
>     "tiles": [
>     {"type": "PING", "params": {"hostname": "redis"}},
>     {"type": "PORT", "params": {"hostname": "redis", "port": 6379}}
```

Vigil is running with a defined number of parallel workers. The `[branding]` section contains various settings for the status page header (e.g., company name, logo, website). The `[metrics]` section defines various polling parameters for the Vigil probes.
Vigil notifies you of the different monitoring events emitted in different ways (e.g., email, Twilio, Slack, Telegram, XMPP, Webex). The test configuration uses a random webhook (it will be different for you) generated through the random URL and email address generator *Webhook.site*, so you can see some events generated by Vigil during testing.
The Vigil GitHub project provides a complete configuration file [4], so you to move quickly through all the settings it provides. The probe section has various subsections to group and define your various ICMP, TCP, and HTTP probes against various hosts and endpoints provided in the replica array. Vigil provides a script probe as well to cover monitoring checks not served by other probes. The Vigil GitHub project page provides a detailed description of all the configuration settings.

Listing 7: `vigil_stack.yml`

```
version: '3.5'
services:

  vigil:
    image: valeriansaliou/vigil:${VGILTAG:-v1.26.0}
    ports:
      - "48080:8080"
    restart: unless-stopped

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

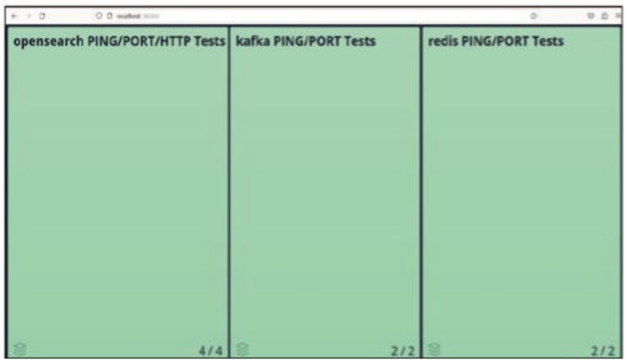


Figure 3: Monitoror wallboard page with server tiles.

Listing 8: `config.cfg`

```
[server]
log_level = "debug"
inet = "0.0.0.0:8080"
workers = 4
manager_token = "REPLACE_THIS_WITH_A_VERY_SECRET_KEY"
reporter_token = "REPLACE_THIS_WITH_A_SECRET_KEY"

[assets]
path = "./res/assets/"

[branding]
page_title = "Vigil Localhost Test Status Page"
page_url = "https://teststatus.page/status"
company_name = "RNG"
icon_color = "#1972F5"
icon_url = "https://avatars.githubusercontent.com/u/226598?v=4"
logo_color = "#1972F5"
logo_url = "https://avatars.githubusercontent.com/u/226598?v=4"
website_url = "https://teststatus.page/"
support_url = "mailto:help@teststatus.page"
custom_html = ""

[metrics]
poll_interval = 60
poll_retry = 2
poll_http_status_healthy_above = 200
poll_http_status_healthy_below = 400
poll_delay_dead = 30
poll_delay_sick = 10
push_delay_dead = 20
push_system_cpu_sick_above = 0.90
push_system_ram_sick_above = 0.90
script_interval = 300
local_delay_dead = 40

[notify]
startup_notification = true
reminder_interval = 300

[notify.webhook]
hook_url = "https://webhook.site/4406e2a4-13cd-4c99-975c-d3456
            a148b26"

[probe]
[[probe.service]]
id = "ping"
label = "PING"
[[probe.service.node]]
id = "invalidiping"
label = "Invalid IP Ping"
mode = "poll"
replicas = ["icmp://129.0.0.1"]

[[probe.service]]
id = "port"
label = "PORT"
[[probe.service.node]]
id = "localhostport"
label = "Localhost Port 8080 Probe"
mode = "poll"
replicas = ["tcp://localhost:8080"]

[[probe.service]]
id = "http"
label = "HTTP"
[[probe.service.node]]
id = "googlehttp"
label = "Google Http Probe"
mode = "poll"
replicas = ["https://google.com"]
```

The commands in Listing 9 bring up the container, provide the required configuration, and restart Vigil. When you open *localhost:48080* in your browser, you will see the Vigil status page (Figure 4).

To add more external servers in a second test setup, as for Monitoror, change the YML file as in Listing 10. Also, change the previous configuration file to include probes for the OpenSearch, Kafka, and Redis containers (Listing 11).

Execute the commands in Listing 9 to launch the containers, copy the updated config, and restart Vigil, respectively. Now refresh your browser and you should see an updated page (Figure 5).

You can see for yourself that the status page is user friendly and interactive, instantly helping you figure out where the probes are passing or failing so you can dig in further. If you add

```
reveal_replica_name = true
```

in every [[probe.service.node]] subsection, tool tips will show replica details on mouseover. The Vigil status page enables you to add a large number of probe targets because of its vertical layout. Please note that the OpenSearch HTTP probe is failing here because Vigil has no way to turn off the SSL certificate through the config file. However, you could solve this issue with the script probe provided by

Listing 9: Launch and Configure Vigil Container

```
docker run -it --rm -v /var/run/docker.sock:/var/run/docker.sock:ro
  -v ./vigil_stack.yml:/etc/compose/vigil_stack.yml:ro docker docker
  compose -f /etc/compose/vigil_stack.yml up -d
docker run -it --rm -v /var/run/docker.sock:/var/run/docker.sock:ro -v
  ./vigil_stack.yml:/etc/compose/vigil_stack.yml:ro -v ./config.cfg:/
  etc/vigil.cfg:ro docker docker compose -f /etc/compose/vigil_stack.yml
  cp /etc/vigil.cfg vigil:/etc/vigil.cfg
docker run -it --rm -v /var/run/docker.sock:/var/run/docker.sock:ro
  -v ./vigil_stack.yml:/etc/compose/vigil_stack.yml:ro docker docker
  compose -f /etc/compose/vigil_stack.yml restart vigil
```

Listing 10: Diff for vigil_stack.yml

```
11a12,30
>   opensearch:
>     image: opensearchproject/opensearch:${OSRHTAG:-latest}
>     environment:
>       - "discovery.type=single-node"
>     restart: unless-stopped
>
>   kafka:
>     image: bitnami/kafka:${KFKATAG:-3.2.3}
>     environment:
>       - "ALLOW_PLAINTEXT_LISTENER=yes"
>       - "KAFKA_CFG_LISTENERS=PLAINTEXT://0.0.0.0:9092,CONTROLLER://:9093"
>       - "KAFKA_ENABLE_KRAFT=yes"
>     restart: unless-stopped
>
>   redis:
>     image: redis:${RDSSTAG:-latest}
>     command: "redis-server --save 60 1 --loglevel warning"
>     restart: unless-stopped
>
```

Listing 11: Diff for config.cfg

```
30,34d29
< push_delay_dead = 20
< push_system_cpu_sick_above = 0.90
< push_system_ram_sick_above = 0.90
< script_interval = 300
< local_delay_dead = 40
45,46c40,41
< id = "ping"
< label = "PING"
---
> id = "kafka"
> label = "KAFKA"
48,49c43,44
< id = "invalidiping"
< label = "Invalid IP Ping"
---
> id = "kafkaping"
> label = "Kafka Ping"
51c46,53
< replicas = ["icmp://129.0.0.1"]
---
> replicas = ["icmp://kafka"]
> reveal_replica_name = true
> [[probe.service.node]]
> id = "kafkaport9092"
> label = "Kafka Port 9092"
> mode = "poll"
> reveal_replica_name = true
> replicas = ["tcp://kafka:9092"]
54,55c56,75
< id = "port"
< label = "PORT"
---
> id = "opensearch"
> label = "OPENSEARCH"
> [[probe.service.node]]
> id = "opensearchping"
> label = "Opensearch Ping"
> mode = "poll"
> reveal_replica_name = true
> replicas = ["icmp://opensearch"]
> [[probe.service.node]]
> id = "opensearchport9200"
> label = "Opensearch Port 9200"
> mode = "poll"
> reveal_replica_name = true
> replicas = ["tcp://opensearch:9200"]
> [[probe.service.node]]
> id = "opensearchport9600"
> label = "Opensearch Port 9600"
> mode = "poll"
> reveal_replica_name = true
> replicas = ["tcp://opensearch:9600"]
57,58c77,78
< id = "localhostport"
< label = "Localhost Port 8080 Probe"
---
> id = "opensearchttp9200"
> label = "Opensearch Http 9200"
60c80,81
< replicas = ["tcp://localhost:8080"]
---
> reveal_replica_name = true
> replicas = ["https://admin:admin@
              opensearch:9200"]
63,64c84,91
< id = "http"
< label = "HTTP"
---
> id = "redis"
> label = "REDIS"
> [[probe.service.node]]
> id = "redisping"
> label = "Redis Ping"
> mode = "poll"
> reveal_replica_name = true
> replicas = ["icmp://redis"]
66,67c93,94
< id = "googlehttp"
< label = "Google Http Probe"
---
> id = "redisport6379"
> label = "Redis Port 6379"
69c96,97
< replicas = ["https://google.com"]
---
> reveal_replica_name = true
> replicas = ["tcp://redis:6379"]
```

Vigil by creating an inline script that makes use of `curl` with a flag to skip SSL certificate checks. To obtain a new image for Vigil, modify the YML and CFG files, as shown in Listings 12 and 13, create the `Dockerfile_VigilSSLCertIgnore` file in the current working directory with the lines,

```
FROM  valeriansaliou/vigil:v1.26.0
RUN apk --no-cache add curl
```

and run the command

```
docker build ↩
  -f  Dockerfile_VigilSSLCertIgnore . ↩
  -t vigilsci:v1.26.0
```

Now execute the Docker commands used previously to launch the Vigil service, copy the new Vigil config, and restart the Vigil service. Voilà, the script probe fixes the limitation of the HTTP probe and all the tiles are now green. You also can administer Vigil through its APIs to publish public announcements, manually report node metrics, and so on. The Vigil GitHub project page has relevant information for you to make use of the Manager HTTP and Reporter HTTP APIs. A related optional component known as Vigil Local can be used to add the health of local services on the status page. Last but not the least, Vigil Reporter libraries are provided for various programming languages to submit health information to Vigil from your apps. All of this information should be enough for you to make full use of the Vigil capabilities to craft a powerful black box monitoring status page.

Statping-ng Status Page and Monitoring Server

Finally, I look at a black box monitoring status page solution full of features known as Statping-ng [5]. To explore its vast array of functionalities, create a `statpingng_stack.yml` file in your current directory (Listing 14), and execute the command

```
docker run -it --rm ↩
  -v /var/run/docker.sock:/var/run/↩
     docker.sock:ro ↩
  -v ./statpingng_stack.yml:/etc/compose/↩
     statpingng_stack.yml:ro ↩
     docker docker compose ↩
  -f /etc/compose/statpingng_stack.yml up -d
```

in the current directory to launch the Statping-ng Docker container. Accessing *localhost:58080* in your web browser should present you with the Statping-ng setup (Figure 6). Just fill in the necessary details and hit *Save Settings*. It should now proceed to another page with your entered Name and Description and be populated with multiple kinds of demo probes supported by Statping-ng (Figure 7).

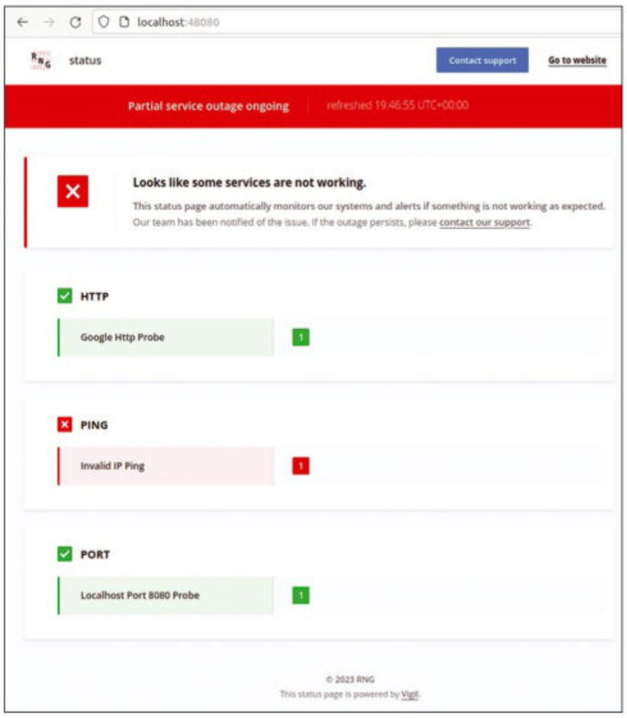
It's pretty cool that the crisp-looking status page not only provides demo probes to familiarize yourself with the solution right away, but on scrolling down, you'll find monitoring graphs for

Listing 12: Diff for `vigil_stack.yml`

```
<     image: valeriansaliou/vigil:${VGILTAG:-v1.26.0}
---
>     image: vigilsci:${VGILTAG:-v1.26.0}
```

Listing 13: Diff for `config.cfg`

```
79c79
< mode = "poll"
---
> mode = "script"
81c81,86
< replicas = ["https://admin:admin@opensearch:9200"]
---
> scripts = [
> '''
> /usr/bin/curl -k https://admin:admin@opensearch:9200
> return $?
> '''
> ]
```


localhost:48080

status

Contact support | Go to website

Partial service outage ongoing | refreshed 19:46:55 UTC+00:00

Looks like some services are not working.

This status page automatically monitors our systems and alerts if something is not working as expected. Our team has been notified of the issue. If the outage persists, please contact our support.

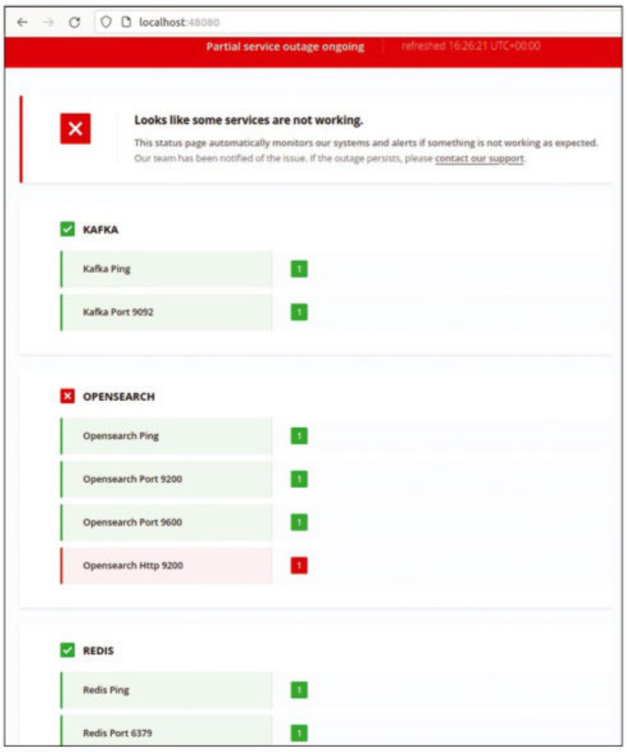
✓ HTTP
- Google Http Probe 1

✗ PING
- Invalid IP Ping 1

✓ PORT
- Localhost Port 8080 Probe 1

© 2023 RNG
This status page is powered by Vigil.

Figure 4: Vigil status page.


localhost:48080

Partial service outage ongoing | refreshed 16:26:21 UTC+00:00

Looks like some services are not working.

This status page automatically monitors our systems and alerts if something is not working as expected. Our team has been notified of the issue. If the outage persists, please contact our support.

✓ KAFKA
- Kafka Ping 1
- Kafka Port 9092 1

✗ OPENSEARCH
- Opensearch Ping 1
- Opensearch Port 9200 1
- Opensearch Port 9600 1
- Opensearch Http 9200 1

✓ REDIS
- Redis Ping 1
- Redis Port 6379 1

Figure 5: Updated Vigil status page with servers' probes.

these demo services (Figure 8). It's just the tip of the iceberg; you can dig into the fine details about the monitored endpoint by clicking on the *View* buttons located on the respective graphs. A *Dashboard* link at the top of the status page takes you to another page (after entering the admin credentials you already set in the Statping-ng setup page), presenting each and every possible setting provided for the Statping-ng configuration.

In the *Services* tab you can see and modify the demo services. You don't need to learn anything else to make use of Statping-ng because every operation is driven by its user-friendly tab pages. In the *Services* tab try adding and removing some of the probes by selecting the appropriate drop-down items for HTTP, TCP, UDP, gRPC, and Static services; ICMP Ping; and various other applicable settings. You could also set up various *Notifiers* to receiving online and offline alerts, post *Announcements* for respective services, browse through the Statping-ng logs, add different kinds of users, and so on.

An important feature of Statping-ng is the ability to back up and restore current Statping services, groups, notifiers, and other settings to and from a JSON file, respectively. The Statping-ng wiki [6] provides more detailed info about its various aspects.

Finally, shift gears to start and configure Statping-ng programmatically through its configuration settings – but without involving any manual steps: Create the `Dockerfile_MyStatpingNG` file with the lines

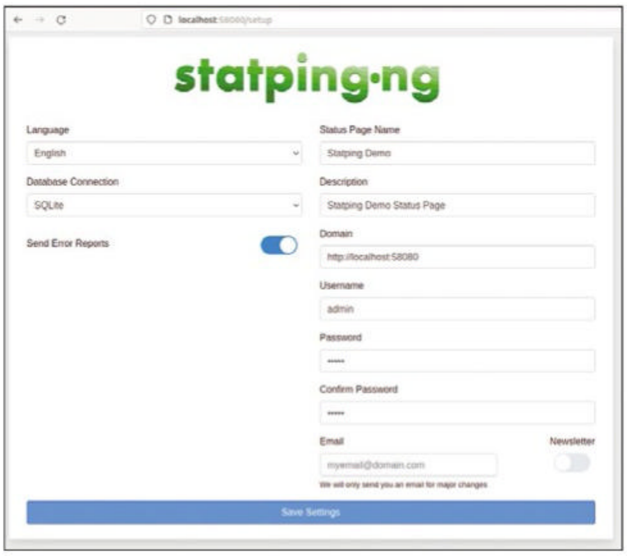
```
FROM adamboutcher/statping-ng

CMD statping --port $PORT -c /app/config.yml
```

and create a new Docker image with the command:

```
docker build -f  Dockerfile_MyStatpingNG . ↩
             -t mystatpingng
```

Now modify `statpingng_stack.yml` as shown in Listing 15 to include the servers to be monitored, and then create the required bind mount directory for Statping-ng with

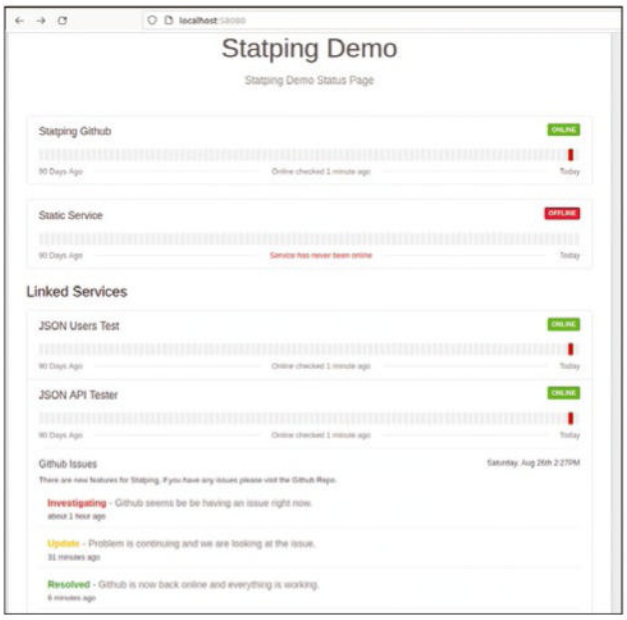
```
mkdir config
```


localhost:58080/setup

statping·ng

Language: English
Database Connection: SQLite
Send Error Reports
Status Page Name: Statping Demo
Description: Statping Demo Status Page
Domain: http://localhost:58080
Username: admin
Password: ••••••
Confirm Password: ••••••
Email: myemail@domain.com
Newsletter
We will only send you an email for major changes
Save Settings

Figure 6: Statping-NG setup page.


localhost:58080

Statping Demo

Statping Demo Status Page

Statping Github — ONLINE
90 Days Ago — Online checked 1 minute ago — Today

Static Service — OFFLINE
90 Days Ago — Service has never been online — Today

Linked Services

JSON Users Test — ONLINE
90 Days Ago — Online checked 1 minute ago — Today

JSON API Tester — ONLINE
90 Days Ago — Online checked 1 minute ago — Today

Github Issues — Saturday, Aug 26th 2:27PM
There are new features for Statping. If you have any issues please visit the Github Repo.

Investigating - Github seems be be having an issue right now.
about 1 hour ago

Update - Problem is continuing and we are looking at the issue.
31 minutes ago

Resolved - Github is now back online and everything is working.
6 minutes ago

Figure 7: Statping-NG status page with demo services.

Listing 14: `statpingng_stack.yml`

```
version: '3.5'
services:

  statping:
    container_name: statpingng
    image: adamboutcher/statping-ng:${SPNGTAG:-latest}
    ports:
      - 58080:8080
    restart: unless-stopped

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

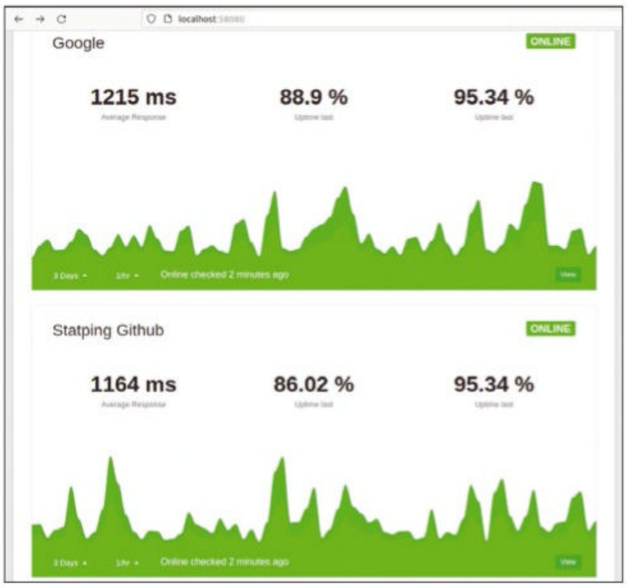


Figure 8: Statping-NG demo services graphs.

and create a `services.yml` file (Listing 16) in the `config` directory. Finally, set up the correct owner for the bind mount directory with the command

```
chown -R root:root config
```

and bring up the new Statping-ng containers with the command used earlier. When you refresh the Statping-ng status page, you should see the new servers being monitored.

You should feel confident now to start using Statping-ng for your production-level status pages. The server provides many additional features, including a choice to use Postgres or MySQL for the production back end, a server configuration with more environmental variables, a full-fledged API to access data on your Statping server programmatically, a Lets Encrypt-enabled automatic SSL certificate, exporting your status page to a static HTML file, and so on. The Statping-ng wiki provides relevant documentation for most of these features.

Conclusion

The problem of segregating decisions on a status page with black box monitoring is a logical first step when running modern complicated IT setups. Monitoror is a quick solution to set up for a few tens of servers and services to make binary decisions about what and where things are failing. Vigil adds more functionality over Monitoror to probe a large number of infrastructure endpoints, and Statping-ng goes even further and provides more user friendliness and professional black box monitoring, presenting tough competition even to expensive enterprise solutions. ■

Listing 15: Diff for `statpingng_stack.yml`

```
6c6
<     image: adamboutcher/statping-ng:${SPNGTAG:-latest}
---
>     image: mystatpingng:${SPNGTAG:-latest}
8a9,36
>     volumes:
>       - ./config:/app
>     environment:
>       - "DB_CONN=sqlite"
>       - "STATPING_DIR=/app"
>       - "SAMPLE_DATA=false"
>       - "GO_ENV=test"
>       - "NAME=StatpingNG Probes Demo"
>       - "DESCRIPTION=StatpingNG Probes Configuration Demo"
>     restart: unless-stopped
>
>   opensearch:
>     image: opensearchproject/opensearch:${OSRHTAG:-latest}
>     environment:
>       - "discovery.type=single-node"
>     restart: unless-stopped
>
>   kafka:
>     image: bitnami/kafka:${KFKATAG:-3.2.3}
>     environment:
>       - "ALLOW_PLAINTEXT_LISTENER=yes"
>       - "KAFKA_CFG_LISTENERS=PLAINTEXT://0.0.0.0:9092,
          CONTROLLER://:9093"
>       - "KAFKA_ENABLE_KRAFT=yes"
>     restart: unless-stopped
>
>   redis:
>     image: redis:${RDSSTAG:-latest}
>     command: "redis-server --save 60 1 --loglevel warning"
```

Info

[1] Monitoror: [https://github.com/monitoror/monitoror]

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

[3] Vigil: [https://github.com/valeriansaliou/vigil]

[4] Vigil sample config file: [https://github.com/valeriansaliou/vigil/blob/master/config.cfg]

[5] Statping-ng: [https://github.com/statping-ng/statping-ng]

[6] Statping-ng wiki: [https://github.com/statping-ng/statping-ng/wiki]

Author

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

Listing 16: `services.yml`

```
x-tcpservice: &tcpservice
  type: tcp
  check_interval: 60
  timeout: 15
  allow_notifications: true
  notify_after: 0
  notify_all_changes: true
  public: true
  redirect: true

x-httpservice: &httpservice
  type: http
  method: GET
  check_interval: 45
  timeout: 10
  expected_status: 200
  allow_notifications: true
  notify_after: 2
  notify_all_changes: true
  public: true
  redirect: true

x-icmping: &icmping
  type: icmp
  check_interval: 60
  timeout: 15
  allow_notifications: true
  notify_after: 0
  notify_all_changes: true
  public: true

services:
  - name: ICMP Kafka
    domain: kafka
    <<: *icmping

  - name: TCP Kafka 9092
    domain: kafka
    port: 9092
    <<: *tcpservice

  - name: ICMP opensearch
    domain: opensearch
    <<: *icmping

  - name: TCP opensearch 9200
    domain: opensearch
    port: 9200
    <<: *tcpservice

  - name: TCP opensearch 9600
    domain: opensearch
    port: 9600
    <<: *tcpservice

  - name: HTTP opensearch
    domain: https://admin:admin@opensearch:9200
    <<: *httpservice

  - name: ICMP redis
    domain: redis
    <<: *icmping

  - name: TCP redis 6379
    domain: redis
    port: 6379
    <<: *tcpservice
```

Keeping Azure VMs up to date

New Models

The operating system of an Azure virtual machine can be kept up to date by a number of methods; we provide an overview and look in detail at Azure Automation Update Management, the Azure Update Management Center, automation options, and other related topics. By Thomas Joos

Both Windows and Linux can be virtualized in different orientations by Microsoft Azure with the use of various prebuilt images. Azure virtual machines (VMs) can be virtualized not only in the cloud, but also directly in an on-premises data center with Azure Stack hyperconverged infrastructure (HCI), while retaining all the benefits familiar from the Azure cloud. The benefits Microsoft cites include simpler licensing, effective high availability, and a simpler update management process, all of which are discussed here.

Free Extended Security Updates

Extended Security Updates (ESUs) after the end of support (e.g., for Windows Server 2008/2008 R2 and Windows Server 2012/2012 R2) for up-to-date Azure VM operating systems have been free thus far. This point is interesting because even if companies use Windows Server 2016, they will slowly but surely have to start worrying about expiring support. Organizations migrating to Azure will therefore want to look into updating VMs right away in this context.

Even if you are not moving to the cloud, Microsoft offers customers with Software Assurance and various subscriptions the option of purchasing extended support that is valid for three years and continues to provide security updates. However, ESUs are not cheap. In the first year, 75 percent of the license fees of the current version are due, which rises to 100 percent in the second year. In the third year, costs rise to 125 percent. In comparison, if you migrate to Azure VMs, you will receive ESU security updates free of charge for the next three years, and extended support is included in the cost of ownership.

This policy applies to all operating systems that are no longer supported – licenses for SQL Server 2012 and Windows Server 2012, for example, can now be used in the cloud.

Strictly speaking, organizations will benefit from the free updates if they rely on Azure VMs, Azure Dedicated Host, Azure VMware Solution, Azure Nutanix Solution, or Azure Stack HCI. Servers with Azure Stack HCI can remain in the on-premises data center, although the use of a certified solution to run Azure Stack HCI on your premises is required. You can build an on-premises cluster that is connected to Azure but running in-house. The connection to Azure does not need to be persistent if the system exclusively relies on on-premises services.

Update Management from Azure

Updating servers is not just about extended support, of course, but also about patches for current servers that you run as VMs in Azure. Microsoft offers Azure Automation Update Management, which can automate patching of servers in on-premises data centers and virtual servers in Azure and other cloud services (Figure 1). This service is ideal for Azure VMs because all services run directly in Azure and no other services are needed. Azure Update Management is also capable of updating Linux servers running as Azure VMs. The service is available free of charge, but charges are incurred if you store logs. Azure stores the monitoring agent logs for update management in Log

Lead Image © Brian Welker, 123RF.com

Analytics, but you need to create your own workspace there, where the Azure Monitor data for telemetry and for logging-connected servers is written by default. The monitoring tool can perform automated queries, with the logs of the connected servers stored in Log Analytics, which will give you information about your servers, including missing updates. In turn, this information can be used by other services in Azure – Azure Update Management in this case.

Azure Monitor and Azure Update Management can be connected to create server logs and install updates at the same time. Simply put, Azure Update Management extends the capabilities of Azure Monitor to include update management. Besides Windows Server, the supported operating systems include CentOS, RHEL, and SUSE version 12 or newer, as well as Ubuntu. You cannot update Windows 7, 8.1, 10, and 11 with the tool. Microsoft recommends the use of Endpoint Manager for this.

Integrating Azure VMs

For Azure Update Management, it does not matter whether the connected computer is a physical or virtual server and whether it resides in the local data center or in the cloud. Integrating Azure VMs with Azure Update Manager is particularly easy because both resources reside in the cloud. The *Updates* item is available for this purpose on the Azure portal's Azure VM dashboard. You can create the link there by choosing to update with automation, and you can remove servers from update management in the same way. Azure VMs and on-premises VMs are visible in the web interface, allowing update rules to be applied by location.

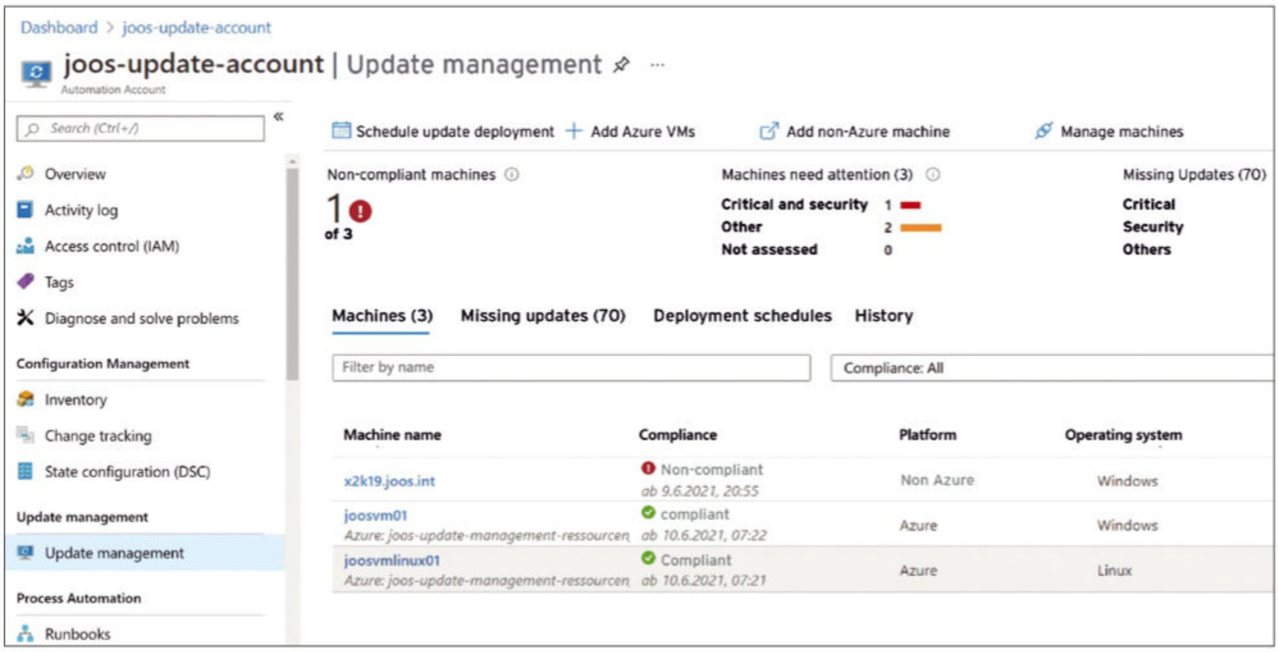
One of the strengths of Azure Update Management is that it can also integrate Linux servers and verify that they are correctly configured and have all updates. The configuration is similar to managing updates for Windows servers. To integrate Linux servers, open your Azure Update Management account and click *Update Management*. Use *Add Azure VMs* to add VMs to Azure, whether these be Windows or Linux machines.

If you want to add computers outside Azure, use *Add non-Azure machine*. These can be physical computers or VMs in Amazon AWS or Google Cloud Platform (GCP). When adding VMs, in the new window, first select which Azure subscription and locations you want to use; then, choose the resource groups in which the servers you are currently integrating reside. At the bottom of the window, the portal shows the individual VMs in Azure; you can see which VMs are already integrated with Azure Update Management. Azure does not differentiate between the various operating systems. Selecting *Enable* simply adds the computers.

Scheduling and Automating Updates

On the Azure Update Management web portal, you can see which servers are not up to date by clicking the Automation Account you created in the resource group where you integrated Azure Update Management, and then click *Update management*. You can view the noncompliant servers (i.e., the servers that are missing updates), the compliant servers, and other information here.

Integrating computers with Azure Update Management is the first step in providing updates to those computers. A deployment schedule then lets you automate update controls on the integrated computers. You can define schedules, release specific updates, and configure which updates you want the servers to install automatically – completely independent of the data center in which the computers are running. You can create schedules from the *Schedule update deployment* item, which can be found under *Update*



Non-compliant machines
1
of 3

Machines need attention (3)

Critical and security	1
Other	2
Not assessed	0

Missing Updates (70)
Critical
Security
Others

Machine name	Compliance	Platform	Operating system
x2k19.joos.int	Non-compliant ab 9.6.2021, 20:55	Non Azure	Windows
joosvm01 Azure: joos-update-management-ressourcen	compliant ab 10.6.2021, 07:22	Azure	Windows
joosvmlinux01 Azure: joos-update-management-ressourcen	Compliant ab 10.6.2021, 07:21	Azure	Linux

Figure 1: Azure VMs, VMs in other cloud services, and servers in the on-premises data center can be added to Azure Update Management.

Management in the Azure Update Management account area. To begin, assign a name to the schedule (e.g., *Monthly Patch Day*), and then select whether the schedule is for Windows or Linux computers. When done, decide on the computer groups you want to connect. On the Groups configuration page, you can configure whether you want to integrate VMs from Azure or from outside. Groups can be filtered by Azure subscription, location, storage locations, and tags. After defining the groups, add the machines you want to update with the schedule.
The section where you select individual update classifications is important. Conventional updates, roll-ups, security updates, critical updates, and feature packs are available, and you can exclude or include individual updates from installation by their knowledgebase IDs.
In the update management Overview, below the update management account, several menu items are listed for each computer; they play an essential role in managing the computers. The *Machines* tab lists the computers integrated with Azure Update Management along with some basic information. The information includes the number of updates missing on the machine and whether the management agent can currently connect to Azure, if the machine is not an Azure VM.
The *Missing updates* tab shows which updates are currently not installed on the computers and how many computers are missing updates. A distinction is made between updates for Windows and updates for Linux. If you have created a provisioning schedule, then it can be seen in the menu item *Deployment schedules*. Of course, multiple schedules are possible; you can click on a schedule to customize its settings.
The *History* tab tells you whether the deployment schedules are working on the computers. In deployment schedules you can add specific updates on the basis of knowledgebase IDs or exclude specific IDs. You can see the exact IDs again in *Missing updates*. Clicking on an update opens the Microsoft support page with detailed instructions on the update in question. If you double-click on a row with an update, the window changes to the Log Analytics area for update management.

VM-Specific Update Methods

As part of the VM creation process in Azure, you can make further adjustments that update the VMs – in Azure Update Management, in part, but also with functions that have nothing to do with Azure Update Management. For example, if you use Windows Server 2022 Datacenter: Azure Edition, you can set the hot patch function for Azure VMs in the cloud or on Azure Stack HCI. Hot patching lets you install updates without having to restart the entire server each time. If individual services or areas of a server require a restart after installing updates, then only those restart. This process takes a fraction of a second, and users do not notice any interruptions in most cases. In other words, the workloads remain permanently active.
From the Azure portal when creating a VM, four settings in *Patch orchestration options* (**Figure 2**) also have a permanent effect on how updates are installed for the various sources:

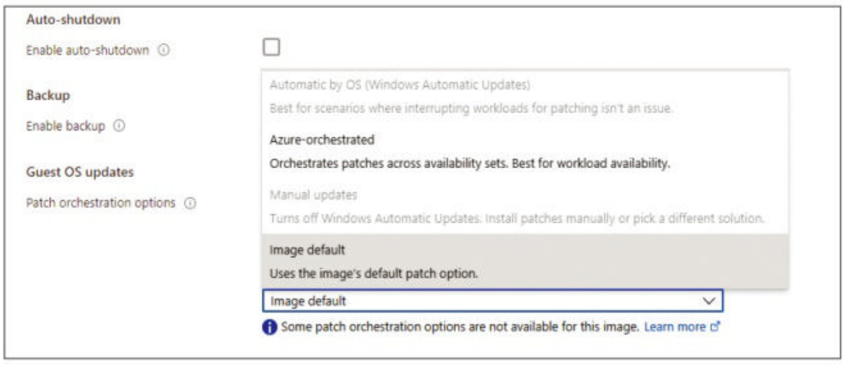

Auto-shutdown
Enable auto-shutdown ⓘ ☐
Backup
Enable backup ⓘ
Guest OS updates
Patch orchestration options ⓘ
Automatic by OS (Windows Automatic Updates)
Best for scenarios where interrupting workloads for patching isn't an issue.
Azure-orchestrated
Orchestrates patches across availability sets. Best for workload availability.
Manual updates
Turns off Windows Automatic Updates. Install patches manually or pick a different solution.
Image default
Uses the image's default patch option.
Image default
Some patch orchestration options are not available for this image. Learn more

Figure 2: The Guest OS updates section presents four patch orchestration options.

Automatic VM Guest Patches

Choosing the option to enable automatic VM guest patches for Azure VMs in the settings enables the automatic installation of patches for Azure VMs. This option installs all critical updates and security patches, but not the definition files for Microsoft Defender. To do this, Azure automatically finds a time when a VM has a low load level and installs the updates in the background. All actions for this update run automatically in Azure. The system regularly checks for updates for VMs and installs them. Restarts occur outside of peak periods. However, the feature does not support every image. Microsoft explains the options capabilities and tells what to watch out for on the automatic VM guest patching website [1].

- *Automatic by OS (Windows Automatic Updates)*
- *Azure-orchestrated*
- *Manual updates*
- *Image default*

Not all options are applicable to all images, however. If you select *Automatic by OS*, Windows servers can be updated automatically by the Windows Automatic Updates feature. One example of provisioning this is the update built into the VM operating system, or you can use Azure Update Management. (See also the "Automatic VM Guest Patches" box.)
The *Azure-orchestrated* option lets you specify that Windows and Linux servers are no longer updated by the operating systems' built-in update functions, but only by Azure itself. However, this only works for selected images in Azure. If the feature is not available for a particular image, the setup wizard grays out this option.
Manual updates means that Azure does not install any updates automatically, so manual work is required involving the use of policies to manage updates. You can do this in Azure with Azure Update Management, for example, but also with Windows Server Update Services (WSUS). In this case, you need a WSUS server in Azure or in the local data center if you are using Azure Stack HCI. The server then supplies the Azure VMs with updates. The option for *Image default* is used for Linux servers if *Azure-orchestrated* is not available.

Public Money
Public Code

Modernising Public Infrastructure with Free Software

fsfe Free Software Foundation Europe

Learn More: https://publiccode.eu/

After creating the VM, you can change the settings retroactively in many cases. To do this, look for the update settings button in the Updates menu. When you get there, select the update approach for your various Azure VMs. You need to use the *Try new Update Management Center* link for this menu item to appear. (I used a Preview edition of the Update Management Center, so slight changes in options and arrangements might occur as the product matures.) After Microsoft has activated this new view, the menu item immediately becomes available.

Azure Update Management Center

Once you have created an Azure VM, the *Updates* item is available on the dashboard; you can use it to connect the VM to Azure Update Management. More menu items appear here after enabling the new user interface, and if you don't need the new interface anymore, you can easily return by choosing the link to exit.

In parallel, you can open the new Azure Update Management Center at this point. This is where you control the installation of VM updates in the Azure portal. First, let *Check for updates* scan the VMs to check for missing updates. If some updates are missing, the portal displays them. You can then specify whether you want to handle the update process manually once only. In this case, select *One-time update*. If you want to install the updates at a later time, choose *Scheduled updates*.

After creating an Azure VM, it makes perfect sense to refresh it first to make sure that the options work. After selecting *One-time update*, select the VMs you want to update in this step. For each VM, Azure shows the status and how many updates are missing. Next, define which updates you want to apply. You can select the *Include update classification* item and check *Select all*.

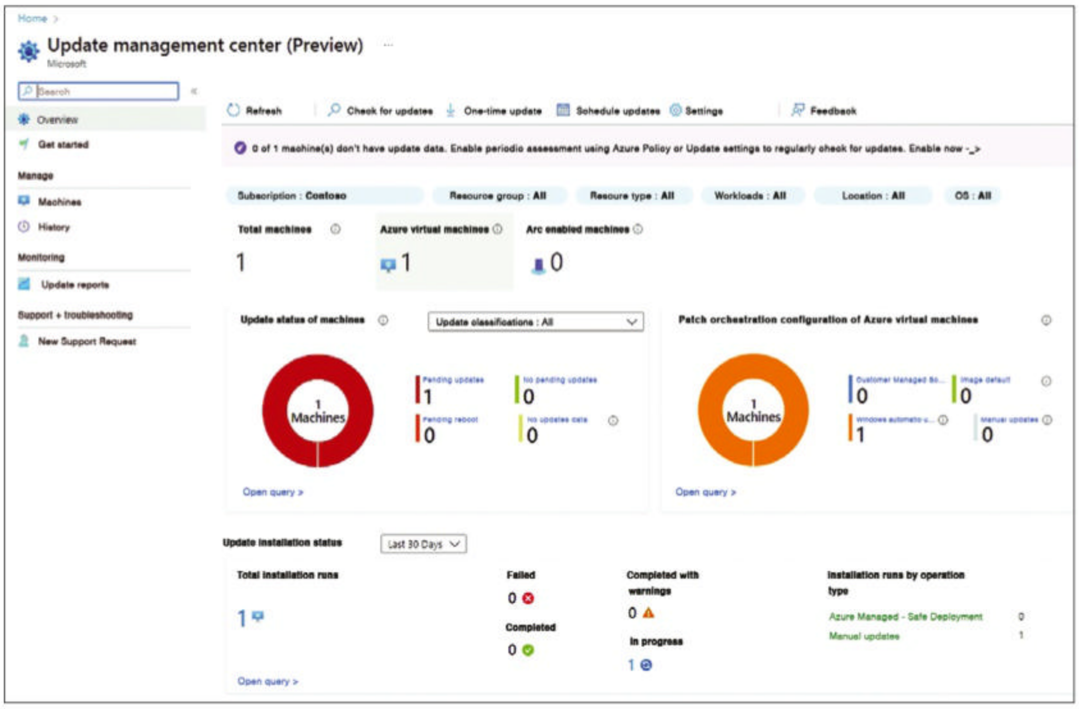
When done, you still need to decide whether the VMs will always restart or whether you leave the restart decision to the VM. Finally, you will see a summary and Azure will proceed to install the updates on the VM. You can view the status on the Azure portal. You do not need to switch to the operating system interface to do this.

If you use the Azure Update Management Center, all connected Azure VMs can be managed with a single action in the portal. You can view various charts on the Update Management Center dashboard showing which VMs are missing updates and how many VMs are connected to the environment (Figure 3). For selected machines, select *Machines* | *Update settings* and enable periodic assessment.

Azure Policy lets you run an automated scan across all connected VMs. To do this, in the Azure Update Management Center, enable periodic assessment in *Machines*. After doing so, Azure automatically scans all your Azure VMs for missing updates and installs them according to the settings you stored in the Update Management Center and in the VM settings.

Conclusions

You have numerous ways at your disposal to update Azure VMs, some of which can be defined when a VM is created. In general, all Azure VMs and all supported operating systems let you deploy updates with on-board operating system resources, but running Azure Update Management will make more sense in many cases. ■



0 of 1 machine(s) don't have update data. Enable periodic assessment using Azure Policy or Update settings to regularly check for updates. Enable now ->

Subscription : Contoso | Resource group : All | Resource type : All | Workloads : All | Location : All | OS : All

Figure 3: Microsoft visualizes update management in the Azure Update Management Center, which is still in preview.

Info

[1] Automatic VM guest patching for Azure VMs: [https://learn.microsoft.com/en-us/azure/virtual-machines/automatic-vm-guest-patching]

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].

FEBRUARY 26 & 27

2024

KICKSTART

ANNUAL EUROPEAN STRATEGY & NETWORKING **EUROPE**

| TRENDS | INVESTMENTS | CLOUD |
| CONNECTIVITY | DATA CENTERS |

AMSTERDAM RAI

Accelerated and targeted search and find with Ripgrep

Rusty Finds

Ripgrep combines the best features of tools like Grep, Ack, and Silver Searcher when it comes to using search patterns in a terminal window. By Ferdinand Thommes

If you want to search for specific strings in code or files, you can turn to a number of powerful Unix tools for the command line, such as Ack [1] and Grep [2]. Both use regular expressions for the search patterns. Grep is often used for this type of search, although Ack has a slight edge in terms of functionality. Silver Searcher [3], with a similar orientation, is an Ack fork that aims to boost the search speed. But these old-timers are not the subject of this article. In fact, I'm only mentioning them because they form the basis of Ripgrep.

Ripgrep [4] is a speedy implementation of Grep in the Rust language. The tool searches directories recursively with a pattern of regular expressions (regexes) and outputs all the matches it finds sorted by file. It is part of a collection of modernized Unix tools [5]. Ripgrep additionally adopts some of the features of Ack and Silver Searcher, such as searching a complete directory tree. However, it does not try to be a complete replacement for Grep, because it does not cover 100 percent of Grep's use cases.

One advantage of the Rust programming language is its speed, which is why many Linux tools have been rewritten in Rust in recent years. They include Ripgrep, which is specifically designed to make searching for strings in large files or directories as efficient as possible. The application's syntax is intuitive and easy to learn. The tool offers clear and consistent output that highlights the lines found and presents the results clearly, without requiring actions you would need to achieve comparable results in Grep, including searching for regular expressions, ignoring certain file types, or recursively searching directories. Ripgrep ignores symbolic links and hidden and binary files by default and helps Git users search for code by supporting `.gitignore` [6].

Another advantage is that Ripgrep is available on macOS and Windows, not just on Linux, so if you work on multiple platforms, you just need to learn the syntax for one search utility. Moreover, the current version of Ripgrep [7] is available in the repositories of many distributions. On macOS you can use Homebrew to install the package, whereas Chocolatey, Winget, or Scoop do the job on Windows.

As already mentioned, Ripgrep uses fairly easy to learn syntax. I will be looking at a few simple examples to get you started. The name of the Ripgrep executable is `rg`. In the simplest case, you would use the command

```
rg -i <search key>
```

to find a string in the current directory and its subdirectories. As with Grep, the `-i` option stands for ignoring case. If you explicitly want to see only matches with the upper- or lower-case spelling of a term, use the `-s` parameter followed by the search term with the desired spelling.

Ripgrep first displays the path and file name of the match before listing all occurrences of the search term with their line numbers. The application highlights the term in red (Figure 1). Depending on the search term entered, the output can be very long, even without path specification. In this case, it makes sense to limit the search command to the extent possible by entering a path (Figure 2):

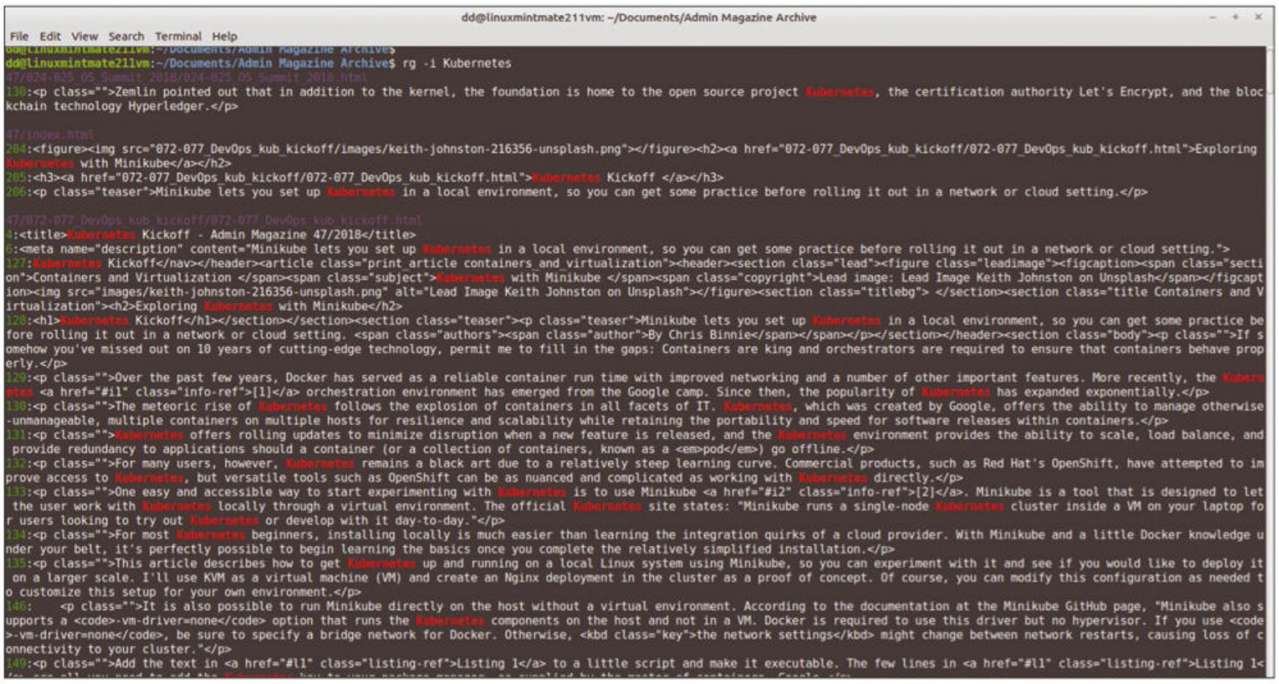
```
$ rg -i bicycle Nextcloud3/Linux-User/
```

If you don't know the exact path or file name but know what type of file it is, the `-g` option is useful:

```
$ rg <Search key> -g '*.<Type>'
```

Conversely, you can exclude file types with the `--type` parameter:

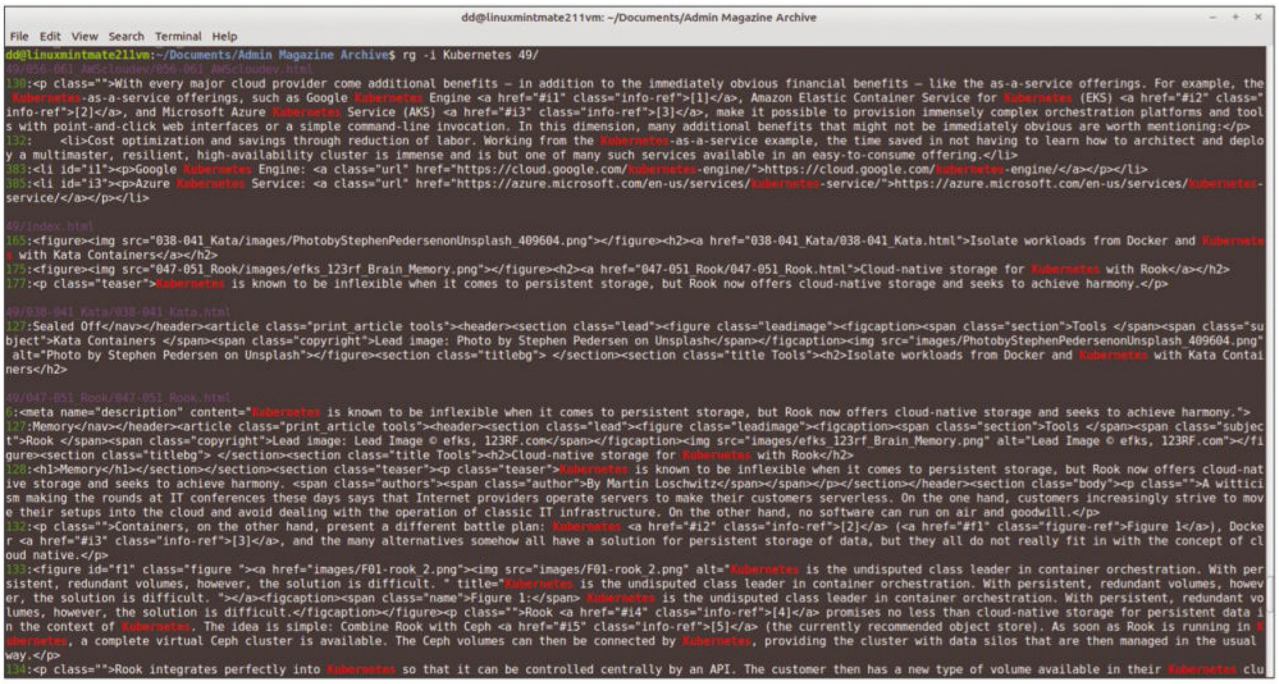
```
$ rg -i bicycle --type not txt
```



```
dd@linuxmintmate211vm:~/Documents/Admin Magazine Archive$
dd@linuxmintmate211vm:~/Documents/Admin Magazine Archive$ rg -i Kubernetes
47/024-025_OS_Summit_2018/024-025_OS_Summit_2018.html
130:<p class="">Zemlin pointed out that in addition to the kernel, the foundation is home to the open source project Kubernetes, the certification authority Let's Encrypt, and the bloc
kchain technology Hyperledger.</p>

47/index.html
204:<figure><img src="072-077_DevOps_kub_kickoff/images/keith-johnston-216356-unsplash.png"></figure><h2><a href="072-077_DevOps_kub_kickoff/072-077_DevOps_kub_kickoff.html">Exploring
Kubernetes with Minikube</a></h2>
205:<h3><a href="072-077_DevOps_kub_kickoff/072-077_DevOps_kub_kickoff.html">Kubernetes Kickoff </a></h3>
206:<p class="teaser">Minikube lets you set up Kubernetes in a local environment, so you can get some practice before rolling it out in a network or cloud setting.</p>

47/072-077_DevOps_kub_kickoff/072-077_DevOps_kub_kickoff.html
4:<title>Kubernetes Kickoff - Admin Magazine 47/2018</title>
6:<meta name="description" content="Minikube lets you set up Kubernetes in a local environment, so you can get some practice before rolling it out in a network or cloud setting.">
127:Kubernetes Kickoff</nav></header><article class="print_article containers_and_virtualization"><header><section class="lead"><figure class="leadimage"><figcaption><span class="secti
on">Containers and Virtualization </span><span class="subject">Kubernetes with Minikube </span><span class="copyright">Lead image: Lead Image Keith Johnston on Unsplash</span></figcapt
ion><img src="images/keith-johnston-216356-unsplash.png" alt="Lead Image Keith Johnston on Unsplash"></figure><section class="titlebg"> </section><section class="title Containers and V
irtualization"><h2>Exploring Kubernetes with Minikube</h2>
128:<h1>Kubernetes Kickoff</h1></section></section><section class="teaser"><p class="teaser">Minikube lets you set up Kubernetes in a local environment, so you can get some practice be
fore rolling it out in a network or cloud setting. <span class="authors"><span class="author">By Chris Binnie</span></span></p></section></header><section class="body"><p class="">If s
omehow you've missed out on 10 years of cutting-edge technology, permit me to fill in the gaps: Containers are king and orchestrators are required to ensure that containers behave prop
erly.</p>
129:<p class="">Over the past few years, Docker has served as a reliable container run time with improved networking and a number of other important features. More recently, the Kubern
etes <a href="#i1" class="info-ref">[1]</a> orchestration environment has emerged from the Google camp. Since then, the popularity of Kubernetes has expanded exponentially.</p>
130:<p class="">The meteoric rise of Kubernetes follows the explosion of containers in all facets of IT. Kubernetes, which was created by Google, offers the ability to manage otherwise
-unmanageable, multiple containers on multiple hosts for resilience and scalability while retaining the portability and speed for software releases within containers.</p>
131:<p class="">Kubernetes offers rolling updates to minimize disruption when a new feature is released, and the Kubernetes environment provides the ability to scale, load balance, and
 provide redundancy to applications should a container (or a collection of containers, known as a <em>pod</em>) go offline.</p>
132:<p class="">For many users, however, Kubernetes remains a black art due to a relatively steep learning curve. Commercial products, such as Red Hat's OpenShift, have attempted to im
prove access to Kubernetes, but versatile tools such as OpenShift can be as nuanced and complicated as working with Kubernetes directly.</p>
133:<p class="">One easy and accessible way to start experimenting with Kubernetes is to use Minikube <a href="#i2" class="info-ref">[2]</a>. Minikube is a tool that is designed to let
 the user work with Kubernetes locally through a virtual environment. The official Kubernetes site states: "Minikube runs a single-node Kubernetes cluster inside a VM on your laptop fo
r users looking to try out Kubernetes or develop with it day-to-day."</p>
134:<p class="">For most Kubernetes beginners, installing locally is much easier than learning the integration quirks of a cloud provider. With Minikube and a little Docker knowledge u
nder your belt, it's perfectly possible to begin learning the basics once you complete the relatively simplified installation.</p>
135:<p class="">This article describes how to get Kubernetes up and running on a local Linux system using Minikube, so you can experiment with it and see if you would like to deploy it
 on a larger scale. I'll use KVM as a virtual machine (VM) and create an Nginx deployment in the cluster as a proof of concept. Of course, you can modify this configuration as needed t
o customize this setup for your own environment.</p>
146:    <p class="">It is also possible to run Minikube directly on the host without a virtual environment. According to the documentation at the Minikube GitHub page, "Minikube also s
upports a <code>-vm-driver=none</code> option that runs the Kubernetes components on the host and not in a VM. Docker is required to use this driver but no hypervisor. If you use <code
>-vm-driver=none</code>, be sure to specify a bridge network for Docker. Otherwise, <kbd class="key">the network settings</kbd> might change between network restarts, causing loss of c
onnectivity to your cluster."</p>
149:<p class="">Add the text in <a href="#l1" class="listing-ref">Listing 1</a> to a little script and make it executable. The few lines in <a href="#l1" class="listing-ref">Listing 1<
```

dd@linuxmintmate211vm: ~/Documents/Admin Magazine Archive
File Edit View Search Terminal Help

Figure 1: You should only use the search without a path specification if you know that the output will be manageable.



```
dd@linuxmintmate211vm:~/Documents/Admin Magazine Archive$ rg -i Kubernetes 49/
49/056-061_AWScloudev/056-061_AWScloudev.html
130:<p class="">With every major cloud provider come additional benefits – in addition to the immediately obvious financial benefits – like the as-a-service offerings. For example, the
 Kubernetes-as-a-service offerings, such as Google Kubernetes Engine <a href="#i1" class="info-ref">[1]</a>, Amazon Elastic Container Service for Kubernetes (EKS) <a href="#i2" class="
info-ref">[2]</a>, and Microsoft Azure Kubernetes Service (AKS) <a href="#i3" class="info-ref">[3]</a>, make it possible to provision immensely complex orchestration platforms and tool
s with point-and-click web interfaces or a simple command-line invocation. In this dimension, many additional benefits that might not be immediately obvious are worth mentioning:</p>
132:    <li>Cost optimization and savings through reduction of labor. Working from the Kubernetes-as-a-service example, the time saved in not having to learn how to architect and deplo
y a multimaster, resilient, high-availability cluster is immense and is but one of many such services available in an easy-to-consume offering.</li>
383:<li id="i1"><p>Google Kubernetes Engine: <a class="url" href="https://cloud.google.com/kubernetes-engine/">https://cloud.google.com/kubernetes-engine/</a></p></li>
385:<li id="i3"><p>Azure Kubernetes Service: <a class="url" href="https://azure.microsoft.com/en-us/services/kubernetes-service/">https://azure.microsoft.com/en-us/services/kubernetes-
service/</a></p></li>

49/index.html
165:<figure><img src="038-041_Kata/images/PhotobyStephenPedersenonUnsplash_409604.png"></figure><h2><a href="038-041_Kata/038-041_Kata.html">Isolate workloads from Docker and Kubernete
s with Kata Containers</a></h2>
175:<figure><img src="047-051_Rook/images/efks_123rf_Brain_Memory.png"></figure><h2><a href="047-051_Rook/047-051_Rook.html">Cloud-native storage for Kubernetes with Rook</a></h2>
177:<p class="teaser">Kubernetes is known to be inflexible when it comes to persistent storage, but Rook now offers cloud-native storage and seeks to achieve harmony.</p>

49/038-041_Kata/038-041_Kata.html
127:Sealed Off</nav></header><article class="print_article tools"><header><section class="lead"><figure class="leadimage"><figcaption><span class="section">Tools </span><span class="su
bject">Kata Containers </span><span class="copyright">Lead image: Photo by Stephen Pedersen on Unsplash</span></figcaption><img src="images/PhotobyStephenPedersenonUnsplash_409604.png"
 alt="Photo by Stephen Pedersen on Unsplash"></figure><section class="titlebg"> </section><section class="title Tools"><h2>Isolate workloads from Docker and Kubernetes with Kata Contai
ners</h2>

49/047-051_Rook/047-051_Rook.html
6:<meta name="description" content="Kubernetes is known to be inflexible when it comes to persistent storage, but Rook now offers cloud-native storage and seeks to achieve harmony.">
127:Memory</nav></header><article class="print_article tools"><header><section class="lead"><figure class="leadimage"><figcaption><span class="section">Tools </span><span class="subjec
t">Rook </span><span class="copyright">Lead image: Lead Image © efks, 123RF.com</span></figcaption><img src="images/efks_123rf_Brain_Memory.png" alt="Lead Image © efks, 123RF.com"></fi
gure><section class="titlebg"> </section><section class="title Tools"><h2>Cloud-native storage for Kubernetes with Rook</h2>
128:<h1>Memory</h1></section></section><section class="teaser"><p class="teaser">Kubernetes is known to be inflexible when it comes to persistent storage, but Rook now offers cloud-nat
ive storage and seeks to achieve harmony. <span class="authors"><span class="author">By Martin Loschwitz</span></span></p></section></header><section class="body"><p class="">A wittici
sm making the rounds at IT conferences these days says that Internet providers operate servers to make their customers serverless. On the one hand, customers increasingly strive to mov
e their setups into the cloud and avoid dealing with the operation of classic IT infrastructure. On the other hand, no software can run on air and goodwill.</p>
132:<p class="">Containers, on the other hand, present a different battle plan: Kubernetes <a href="#i2" class="info-ref">[2]</a> (<a href="#f1" class="figure-ref">Figure 1</a>), Docke
r <a href="#i3" class="info-ref">[3]</a>, and the many alternatives somehow all have a solution for persistent storage of data, but they all do not really fit in with the concept of cl
oud native.</p>
133:<figure id="f1" class="figure "><a href="images/F01-rook_2.png"><img src="images/F01-rook_2.png" alt="Kubernetes is the undisputed class leader in container orchestration. With per
sistent, redundant volumes, however, the solution is difficult. " title="Kubernetes is the undisputed class leader in container orchestration. With persistent, redundant volumes, howev
er, the solution is difficult. "></a><figcaption><span class="name">Figure 1:</span> Kubernetes is the undisputed class leader in container orchestration. With persistent, redundant vo
lumes, however, the solution is difficult.</figcaption></figure><p class="">Rook <a href="#i4" class="info-ref">[4]</a> promises no less than cloud-native storage for persistent data i
n the context of Kubernetes. The idea is simple: Combine Rook with Ceph <a href="#i5" class="info-ref">[5]</a> (the currently recommended object store). As soon as Rook is running in K
ubernetes, a complete virtual Ceph cluster is available. The Ceph volumes can then be connected by Kubernetes, providing the cluster with data silos that are then managed in the usual
way.</p>
134:<p class="">Rook integrates perfectly into Kubernetes so that it can be controlled centrally by an API. The customer then has a new type of volume available in their Kubernetes clu
```

dd@linuxmintmate211vm: ~/Documents/Admin Magazine Archive
File Edit View Search Terminal Help

Figure 2: Narrowing down by stating the path gives you more targeted output. In addition to the path to the file, the tool displays the number of the line where the paragraph with the search term starts.

If you don't need the individual matches, but only the files in which they occur, the `-l` option is useful:

```
$ rg -l <Search key>
```

You also can achieve alphabetical sorting in the context of `-l` by appending `--sort path` to the command (Figure 3).

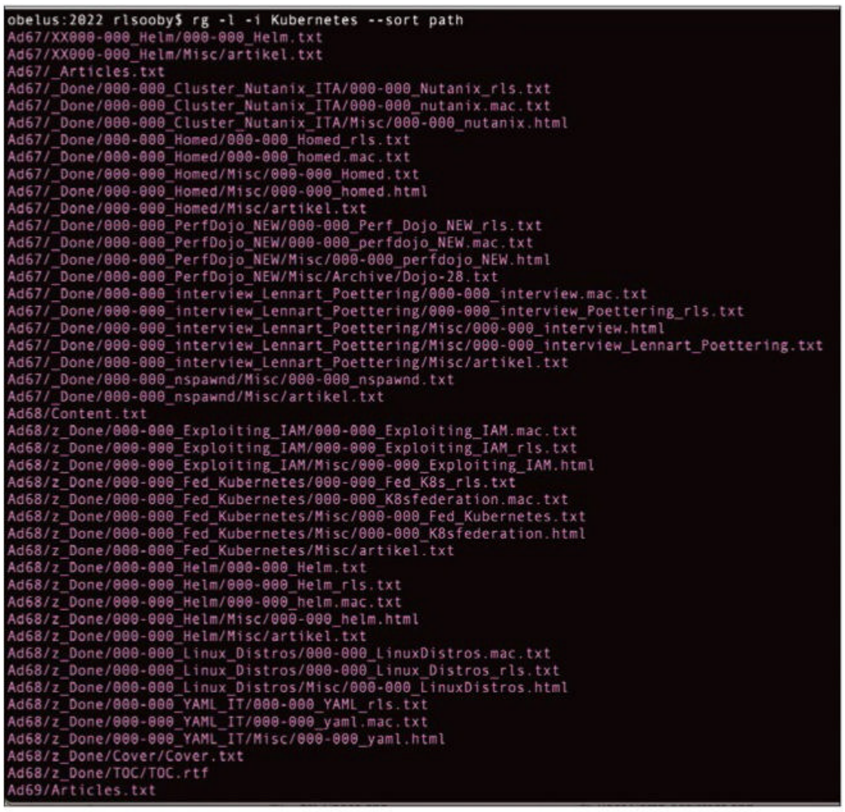
Assume you want to change the SSH port for security reasons. To do this, you want to find the line in the configuration file and jump to it directly in an editor. In this case, you would not just specify the path, but also the file containing the term. First, use

```
$ rg -i port /etc/ssh/sshd_config
```

to determine the line number. The port specification in the file shown in Figure 4 occurs in line 40. In editors like Nano or Vim, you can stipulate `+40` in the call to jump directly to this line and change the port.

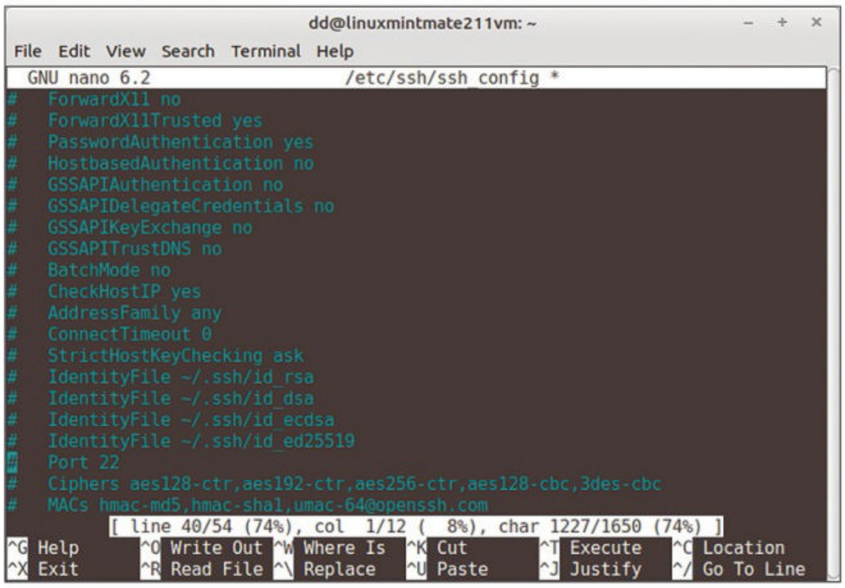
Additionally, statistical values often play a key role. For example: How often does a term occur in how many lines? How long did the search take? How many files were browsed? This information appears at the end of the output if you append the `--stats` option to the search command (Figure 5).

If you want to have a little more context than Ripgrep gives you by default, set the `-C <n>` option, where `<n>` denotes the number of lines before and after the location you want to view. To check a certain number of lines exclusively before the find location, use `-B <n>`; `-A <n>` does the same thing for lines after the find. You can search compressed text archives with `-z -a` and a combination of options (Figure 6).



```
obelus:2022 rlsooby$ rg -l -i Kubernetes --sort path
Ad67/XX000-000_Helm/000-000_Helm.txt
Ad67/XX000-000_Helm/Misc/artikel.txt
Ad67/_Articles.txt
Ad67/_Done/000-000_Cluster_Nutanix_ITA/000-000_Nutanix_rls.txt
Ad67/_Done/000-000_Cluster_Nutanix_ITA/000-000_nutanix.mac.txt
Ad67/_Done/000-000_Cluster_Nutanix_ITA/Misc/000-000_nutanix.html
Ad67/_Done/000-000_Homed/000-000_Homed_rls.txt
Ad67/_Done/000-000_Homed/000-000_homed.mac.txt
Ad67/_Done/000-000_Homed/Misc/000-000_Homed.txt
Ad67/_Done/000-000_Homed/Misc/000-000_homed.html
Ad67/_Done/000-000_Homed/Misc/artikel.txt
Ad67/_Done/000-000_PerfDojo_NEW/000-000_Perf_Dojo_NEW_rls.txt
Ad67/_Done/000-000_PerfDojo_NEW/000-000_perfdojo_NEW.mac.txt
Ad67/_Done/000-000_PerfDojo_NEW/Misc/000-000_perfdojo_NEW.html
Ad67/_Done/000-000_PerfDojo_NEW/Misc/Archive/Dojo-28.txt
Ad67/_Done/000-000_interview_Lennart_Poettering/000-000_interview.mac.txt
Ad67/_Done/000-000_interview_Lennart_Poettering/000-000_interview_Poettering_rls.txt
Ad67/_Done/000-000_interview_Lennart_Poettering/Misc/000-000_interview.html
Ad67/_Done/000-000_interview_Lennart_Poettering/Misc/000-000_interview_Lennart_Poettering.txt
Ad67/_Done/000-000_interview_Lennart_Poettering/Misc/artikel.txt
Ad67/_Done/000-000_nspawnd/Misc/000-000_nspawnd.txt
Ad67/_Done/000-000_nspawnd/Misc/artikel.txt
Ad68/Content.txt
Ad68/z_Done/000-000_Exploiting_IAM/000-000_Exploiting_IAM.mac.txt
Ad68/z_Done/000-000_Exploiting_IAM/000-000_Exploiting_IAM_rls.txt
Ad68/z_Done/000-000_Exploiting_IAM/Misc/000-000_Exploiting_IAM.html
Ad68/z_Done/000-000_Fed_Kubernetes/000-000_Fed_K8s_rls.txt
Ad68/z_Done/000-000_Fed_Kubernetes/000-000_K8sfederation.mac.txt
Ad68/z_Done/000-000_Fed_Kubernetes/Misc/000-000_Fed_Kubernetes.txt
Ad68/z_Done/000-000_Fed_Kubernetes/Misc/000-000_K8sfederation.html
Ad68/z_Done/000-000_Fed_Kubernetes/Misc/artikel.txt
Ad68/z_Done/000-000_Helm/000-000_Helm.txt
Ad68/z_Done/000-000_Helm/000-000_Helm_rls.txt
Ad68/z_Done/000-000_Helm/000-000_helm.mac.txt
Ad68/z_Done/000-000_Helm/Misc/000-000_helm.html
Ad68/z_Done/000-000_Helm/Misc/artikel.txt
Ad68/z_Done/000-000_Linux_Distros/000-000_LinuxDistros.mac.txt
Ad68/z_Done/000-000_Linux_Distros/000-000_Linux_Distros_rls.txt
Ad68/z_Done/000-000_Linux_Distros/Misc/000-000_LinuxDistros.html
Ad68/z_Done/000-000_YAML_IT/000-000_YAML_rls.txt
Ad68/z_Done/000-000_YAML_IT/000-000_yaml.mac.txt
Ad68/z_Done/000-000_YAML_IT/Misc/000-000_yaml.html
Ad68/z_Done/Cover/Cover.txt
Ad68/z_Done/TOC/TOC.rtf
Ad69/Articles.txt
```

Figure 3: If you want to know which files contain a search term, the `-l --sort path` option sorts the results alphabetically.



```
  GNU nano 6.2                    /etc/ssh/ssh_config *
#   ForwardX11 no
#   ForwardX11Trusted yes
#   PasswordAuthentication yes
#   HostbasedAuthentication no
#   GSSAPIAuthentication no
#   GSSAPIDelegateCredentials no
#   GSSAPIKeyExchange no
#   GSSAPITrustDNS no
#   BatchMode no
#   CheckHostIP yes
#   AddressFamily any
#   ConnectTimeout 0
#   StrictHostKeyChecking ask
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
#   IdentityFile ~/.ssh/id_ecdsa
#   IdentityFile ~/.ssh/id_ed25519
#   Port 22
#   Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
#   MACs hmac-md5,hmac-sha1,umac-64@openssh.com
          [ line 40/54 (74%), col  1/12 (  8%), char 1227/1650 (74%) ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

Figure 4: If you know the line number of the search term in a configuration file, you can jump to it directly with Nano or Vim.

Conclusions

Ripgrep does not claim to replace Grep. On the one hand, it behaves differently in some cases; on the other hand, it does not cover all the functions of its role model.

The tool is aimed more at pattern searching, whereas Grep developer Ken Thompson designed his tool mainly for stream processing [8] on AT&T Unix v6. For example, you need to run GNU Grep in combination with `find` for recursive searching in directories; Ripgrep handles this task without external support.

Ripgrep offers many more functions than I can hope to describe in this article. The detailed documentation [9] will help you explore the full feature set. If you are interested in the differences between various search tools, it is also worth visiting Beyondgrep.com [10]: The site compares the features of Ack, Silver Searcher, Git-Grep, GNU Grep, and Ripgrep in detail. ■

Info

[1] Ack: [https://beyondgrep.com/documentation/]

[2] Grep: [https://www.man7.org/linux/man-pages/man1/grep.1.html]

[3] Silver Searcher: [https://github.com/ggreer/the_silver_searcher]

[4] Ripgrep: [https://github.com/BurntSushi/Ripgrep#installation]

[5] Modern Unix: [https://github.com/ibraheemdev/modern-unix]

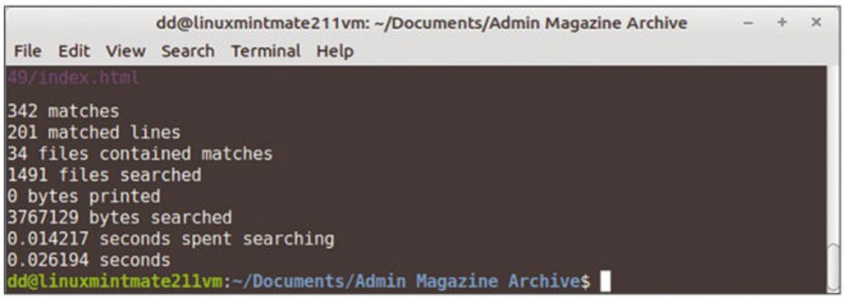
[6] gitignore: [https://www.atlassian.com/git/tutorials/saving-changes/gitignore]

[7] Versions: [https://repology.org/project/ripgrep/versions]

[8] Data stream: [https://en.wikipedia.org/wiki/Data_stream]

[9] Documentation: [https://github.com/BurntSushi/Ripgrep/blob/master/GUIDE.md]

[10] Comparison: [https://beyondGrep.com/feature-comparison/]

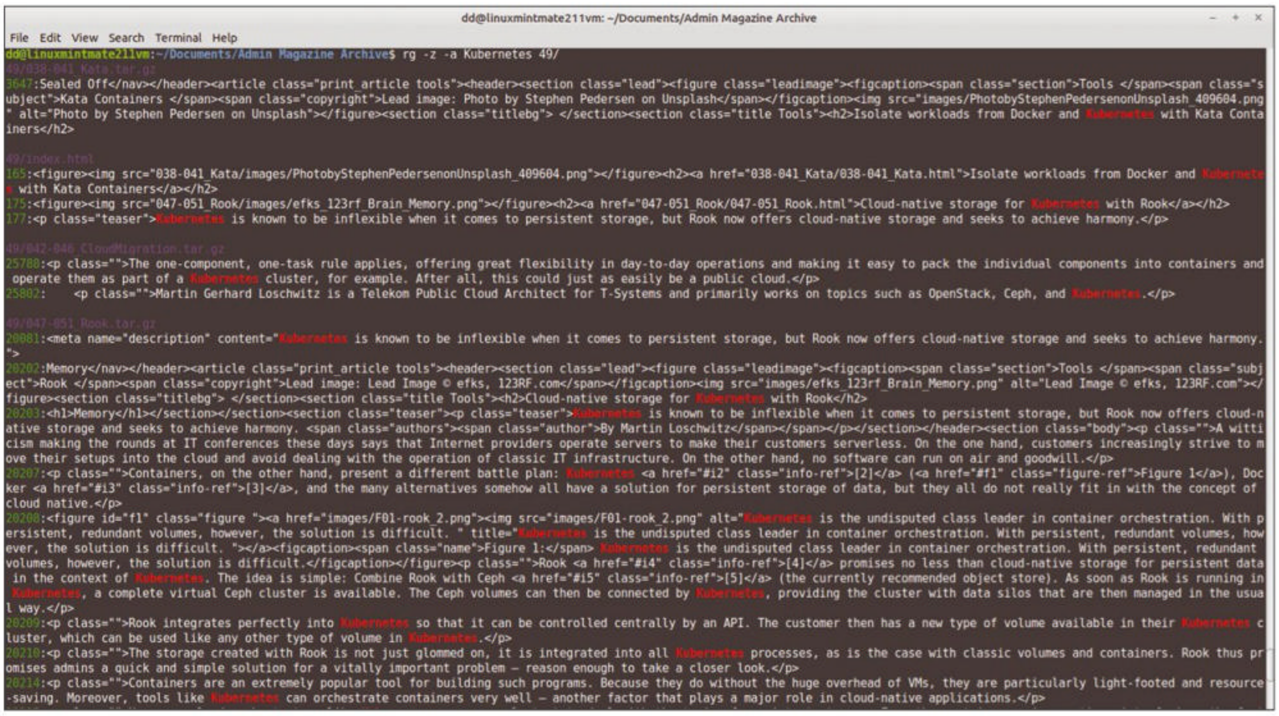

dd@linuxmintmate211vm: ~/Documents/Admin Magazine Archive
File Edit View Search Terminal Help

```
49/index.html

342 matches
201 matched lines
34 files contained matches
1491 files searched
0 bytes printed
3767129 bytes searched
0.014217 seconds spent searching
0.026194 seconds
dd@linuxmintmate211vm:~/Documents/Admin Magazine Archive$
```

Figure 5: Statistical values for the number of finds, the number of files searched, the time required, and more are provided by the `--stats` option.

The Author

Ferdinand Thommes lives and works as a Linux developer, freelance writer, and tour guide in Berlin.


dd@linuxmintmate211vm: ~/Documents/Admin Magazine Archive
File Edit View Search Terminal Help

```
dd@linuxmintmate211vm:~/Documents/Admin Magazine Archive$ rg -z -a Kubernetes 49/
49/038-041_Kata.tar.gz
3647:Sealed Off</nav></header><article class="print_article tools"><header><section class="lead"><figure class="leadimage"><figcaption><span class="section">Tools </span><span class="s
ubject">Kata Containers </span><span class="copyright">Lead image: Photo by Stephen Pedersen on Unsplash</span></figcaption><img src="images/PhotobyStephenPedersenonUnsplash_409604.png
" alt="Photo by Stephen Pedersen on Unsplash"></figure><section class="titlebg"> </section><section class="title Tools"><h2>Isolate workloads from Docker and Kubernetes with Kata Conta
iners</h2>

49/index.html
165:<figure><img src="038-041_Kata/images/PhotobyStephenPedersenonUnsplash_409604.png"></figure><h2><a href="038-041_Kata/038-041_Kata.html">Isolate workloads from Docker and Kubernete
s with Kata Containers</a></h2>
175:<figure><img src="047-051_Rook/images/efks_123rf_Brain_Memory.png"></figure><h2><a href="047-051_Rook/047-051_Rook.html">Cloud-native storage for Kubernetes with Rook</a></h2>
177:<p class="teaser">Kubernetes is known to be inflexible when it comes to persistent storage, but Rook now offers cloud-native storage and seeks to achieve harmony.</p>

49/042-046_CloudMigration.tar.gz
25780:<p class="">The one-component, one-task rule applies, offering great flexibility in day-to-day operations and making it easy to pack the individual components into containers and
 operate them as part of a Kubernetes cluster, for example. After all, this could just as easily be a public cloud.</p>
25802:    <p class="">Martin Gerhard Loschwitz is a Telekom Public Cloud Architect for T-Systems and primarily works on topics such as OpenStack, Ceph, and Kubernetes.</p>

49/047-051_Rook.tar.gz
20081:<meta name="description" content="Kubernetes is known to be inflexible when it comes to persistent storage, but Rook now offers cloud-native storage and seeks to achieve harmony.
">
20202:Memory</nav></header><article class="print_article tools"><header><section class="lead"><figure class="leadimage"><figcaption><span class="section">Tools </span><span class="subj
ect">Rook </span><span class="copyright">Lead image: Lead Image © efks, 123RF.com</span></figcaption><img src="images/efks_123rf_Brain_Memory.png" alt="Lead Image © efks, 123RF.com"></
figure><section class="titlebg"> </section><section class="title Tools"><h2>Cloud-native storage for Kubernetes with Rook</h2>
20203:<h1>Memory</h1></section></section><section class="teaser"><p class="teaser">Kubernetes is known to be inflexible when it comes to persistent storage, but Rook now offers cloud-n
ative storage and seeks to achieve harmony. <span class="authors"><span class="author">By Martin Loschwitz</span></span></p></section></header><section class="body"><p class="">A witti
cism making the rounds at IT conferences these days says that Internet providers operate servers to make their customers serverless. On the one hand, customers increasingly strive to m
ove their setups into the cloud and avoid dealing with the operation of classic IT infrastructure. On the other hand, no software can run on air and goodwill.</p>
20207:<p class="">Containers, on the other hand, present a different battle plan: Kubernetes <a href="#i2" class="info-ref">[2]</a> (<a href="#f1" class="figure-ref">Figure 1</a>), Doc
ker <a href="#i3" class="info-ref">[3]</a>, and the many alternatives somehow all have a solution for persistent storage of data, but they all do not really fit in with the concept of
cloud native.</p>
20208:<figure id="f1" class="figure "><a href="images/F01-rook_2.png"><img src="images/F01-rook_2.png" alt="Kubernetes is the undisputed class leader in container orchestration. With p
ersistent, redundant volumes, however, the solution is difficult. " title="Kubernetes is the undisputed class leader in container orchestration. With persistent, redundant volumes, how
ever, the solution is difficult. "></a><figcaption><span class="name">Figure 1:</span> Kubernetes is the undisputed class leader in container orchestration. With persistent, redundant
volumes, however, the solution is difficult.</figcaption></figure><p class="">Rook <a href="#i4" class="info-ref">[4]</a> promises no less than cloud-native storage for persistent data
 in the context of Kubernetes. The idea is simple: Combine Rook with Ceph <a href="#i5" class="info-ref">[5]</a> (the currently recommended object store). As soon as Rook is running in
 Kubernetes, a complete virtual Ceph cluster is available. The Ceph volumes can then be connected by Kubernetes, providing the cluster with data silos that are then managed in the usua
l way.</p>
20209:<p class="">Rook integrates perfectly into Kubernetes so that it can be controlled centrally by an API. The customer then has a new type of volume available in their Kubernetes c
luster, which can be used like any other type of volume in Kubernetes.</p>
20210:<p class="">The storage created with Rook is not just glommed on, it is integrated into all Kubernetes processes, as is the case with classic volumes and containers. Rook thus pr
omises admins a quick and simple solution for a vitally important problem – reason enough to take a closer look.</p>
20214:<p class="">Containers are an extremely popular tool for building such programs. Because they do without the huge overhead of VMs, they are particularly light-footed and resource
-saving. Moreover, tools like Kubernetes can orchestrate containers very well – another factor that plays a major role in cloud-native applications.</p>
```

Figure 6: If so desired, Ripgrep will also dig into archives to find your quarry.

FOSSLIFE

Open for All

News • Careers • Life in Tech
Skills • Resources

FOSSlife.org

ADMIN Network & Security

NEWSSTAND

Order online:
https://bit.ly/ADMIN-library

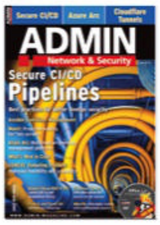
ADMIN is your source for technical solutions to real-world problems. Every issue is packed with practical articles on the topics you need, such as: security, cloud computing, DevOps, HPC, storage, and more! Explore our full catalog of back issues for specific topics or to complete your collection.

#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

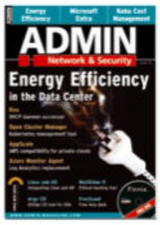


#76 – July/August 2023

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

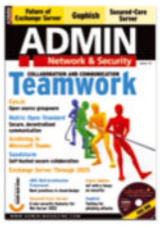


#75 – May/June 2023

Teamwork

Groupware, collaboration frameworks, chat servers, and a web app package manager allow your teams to exchange knowledge and collaborate on projects in a secure environment.

On the DVD: Ubuntu 23.04 "Lunar Lobster" Server Edition

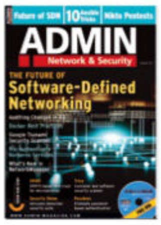


#74 – March/April 2023

The Future of Software-Defined Networking

New projects out of the Open Networking Foundation provide a glimpse into the 5G network future, most likely software based and independent of proprietary hardware.

On the DVD: Kali Linux 2022.4

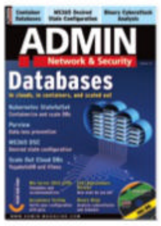


#73 – January/February 2023

Databases

Cloud databases can be useful in virtually any conceivable deployment scenario, come in SQL and NoSQL flavors, and harmonize well with virtualized and containerized environments.

On the DVD: Manjaro 22.0 Gnome

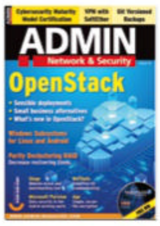


#72 – November/December 2022

OpenStack

Find out whether the much evolved OpenStack is right for your private cloud.

On the DVD: Fedora 36 Server Edition



WRITE FOR US

Admin: Network and Security is looking for good, practical articles on system administration topics. We love to hear from IT professionals who have discovered innovative tools or techniques for solving real-world problems.

Tell us about your favorite:

- interoperability solutions
- practical tools for cloud environments
- security problems and how you solved them
- ingenious custom scripts
- unheralded open source utilities
- Windows networking techniques that aren't explained (or aren't explained well) in the standard documentation.

We need concrete, fully developed solutions: installation steps, configuration files, examples – we are looking for a complete discussion, not just a "hot tip" that leaves the details to the reader.

If you have an idea for an article, send a 1-2 paragraph proposal describing your topic to: *edit@admin-magazine.com*.

Authors

Amber Ankerholz	6
Tam Hanna	22
Ken Hess	3
Stefan Hofer	10
Thomas Joos	86
Joydip Kanjilal	38
Ankur Kumar	78
Jeff Layton	32
Dr. Carola Lilienthal	18
Martin Gerhard Loschwitz	26, 42
Benjamin Pfister	56
Henning Schwentner	18
Artur Skura	48
Andreas Stolzenberger	72
Ferdinand Thommes	92
Kevin Wittmer	62
Eberhard Wolff	14
Matthias Wübbeling	70

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 armmypicca, 123RF Free Images

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7779420

Publisher
Brian Osborn

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2023 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by Kolibri Druck.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN (Print ISSN: 2045-0702, Online ISSN: 2831-9583, USPS No: 347-931) is published bimonthly by Linux New Media USA, LLC, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. November/December 2023. 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.

Want to work from anywhere?

Find remote jobs now!

OpenSource

JOB HUB

opensourcejobhub.com/jobs

Highly efficient athlete

TUXEDO Pulse 14 - Gen3 with **AMD**

CPU performance	GFX performance	Mobility	Battery life

Linux compatible · Up to 5 Years Guarantee · Immediately ready for use · Made in Germany · German Data Privacy · German Tech Support

TUXEDO

tuxedocomputers.com