c't CLOUD

Einstieg und Praxis für Admins und Entwickler

CI/CD
7 Stdn. Videokurs
mit Leserrabatt (S. 53)

Nützliche Cloud-Dienste betreiben

Eigenen E-Mail-Server installieren & administrieren
Mitgliederversammlungen selbst hosten
Passwort-Server sicher und zuverlässig betreiben

Teamwork mit Nextcloud

Gruppenarbeit organisieren
Office im Browser nutzen
Online-Konferenzen per Chat und Video

Cloud-Programmiersprache Go

Effizient programmieren für viele Plattformen
Mit Go eigene Cloud-Dienste entwickeln



facebook.com/ctspecials

Profi-Know-how für Cloud-Admins & Entwickler

Cloud: Konzepte und Software, die Sie kennen sollten
Strategien für den Einstieg und den Umzug in die Cloud

€ 14,90
CH CHF 27.90
AT € 16,40
LUX € 17,10

4 196982 314900 05



heise Academy

Ob Newcomer oder Profi: Meistern Sie Ihren Cloud-Betrieb mit unserem Angebot aus Workshops – in gewohnter Heise-Qualität: unabhängig, praxisnah, kompetent.

Unser Angebot

Cloud Fast Track

In diesem Workshop erhalten Sie eine Übersicht über die Aspekte, die sie bei der Nutzung der Public Cloud (Azure, Google Cloud, AWS) beachten müssen.

27. Oktober

WORKSHOP

Sichere Cloudnutzung

In diesem Workshop erarbeiten wir mit Ihnen kompakt und praxisnah Schritt für Schritt Ihren Leitfaden für eine sichere Cloud-Nutzung.

16. – 17. November

WORKSHOP

Mehr Informationen:

heise-academy.de/cloud

Editorial

Mehr ausprobieren, weniger schwitzen

Früher, als Server noch im eigenen Haus liefen und dem Admin ihre Abwärme ins Gesicht bliesen, war ihr Betrieb harte und schweißtreibende Arbeit. Das ist jetzt vorbei, das Zeitalter der Cloud hat begonnen. Statt Server ins Rack zu wuppen, mietet man sie heute bequem vom Schreibtisch aus und statt des Admins schwitzt nur noch die Kreditkarte, die beim Vermieter hinterlegt ist.

Wer allerdings glaubt, in diesem Zeitalter mit weniger IT-Personal auszukommen, hat nur die Werbeprospekte der Cloudprovider gelesen und noch nie einen gemieteten Server eingerichtet. Daher keine Sorge: Arbeit gibt es noch immer genug, anstrengend ist sie weiterhin, nur anders. Statt im Kabelsalat im Netzwerkschrank verstrickt man sich jetzt in Konfigurationsdateien im YAML-Format.

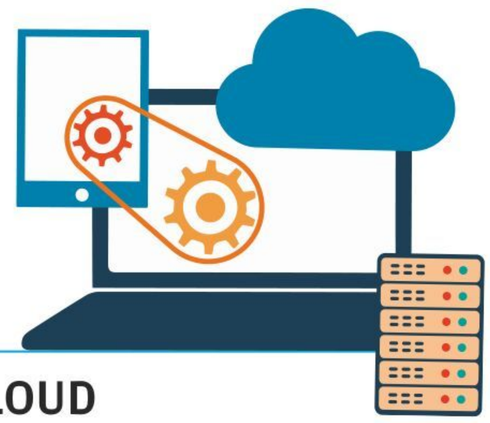
Dieses Heft soll Ihnen die Arbeit erleichtern und zum Ausprobieren anregen. Sie finden Hintergrundwissen zu Docker, Kubernetes und Ansible sowie Praxistipps und Rezepte, um populäre Software wie Nextcloud oder Rocket.Chat auf eigenen oder gemieteten Servern zu betreiben. Sie lesen, wie ein fiktives Start-up seine Probleme mit Servern aus der Cloud löst und wie Sie einen geeigneten Anbieter für Mietserver finden. Wenn Sie eigene Cloud-Dienste programmieren wollen, finden Sie einen Einstieg in die noch junge, aber in der Cloudszene populäre Programmiersprache Go.

In erster Linie macht das Hantieren mit Containern und flexiblen Servern Spaß und vieles geht heute viel, viel schneller als damals, als Server noch laut und schwer waren.



Jan Mahn

Inhalt



WEGE IN DIE CLOUD

Mit dem einfachen Anmieten von Servern allein ist es nicht getan. Erst mit dem passenden Anbieter und einer geeigneten Strategie gelingt ein echter Umzug in die Cloud.

6 Serverinfrastruktur mieten in der Cloud
14 Cloudprovider aus Europa
22 Cloudserver bestellen und einrichten
26 Automatisierung mit Ansible
36 Schnellstart mit Ansible
42 Cloud-Dienste mit Terraform
46 Kubernetes für Docker-Kenner

CLOUD AUF EIGENER HARDWARE

Nicht immer brauchen Sie gemietete Server, um Ihre Dienste zu betreiben. Sie können Ihre Cloud auch auf eigenen Maschinen einrichten.

54 Starthilfe für den eigenen Server
62 Dienste ins Internet bringen
68 Einstiegsfreundlicher Heimserver

DIENSTE FÜR DIE CLOUD

Viele nützliche Anwendungen sind dank Containern schnell auf Cloudservern oder eigener Hardware zum Laufen gebracht.

72 Wegweiser im Heimnetz
74 CI/CD mit GitHub Actions
80 Online-Dokumentationen mit MkDocs
86 Statusseiten für IT-Infrastruktur
90 Passwortmanager in Eigenregie hosten
98 Mitgliederversammlungen virtuell

EIGENE DATEN IN NEXTCLOUD

Die Open-Source-Komplettlösung Nextcloud unterstützt mit Dateispeicher, Kalenderserver und vielen Erweiterungen die Gruppenarbeit in der eigenen Cloud.

104 Nextcloud als Bausatz
108 Chat-Server hinter Nextcloud
112 Nextcloud: Tipps für die Gruppenarbeit
118 Videokonferenzen mit Nextcloud Talk



E-MAIL-SERVER SELBER HOSTEN

Ein eigener Mailserver ist Ihre Unabhängigkeitserklärung von Mailprovidern. Grundlagen und eine praktische Anleitung helfen bei der Einrichtung.

126 E-Mail: Begriffe und Protokolle
134 Mailserver einrichten mit Mailcow

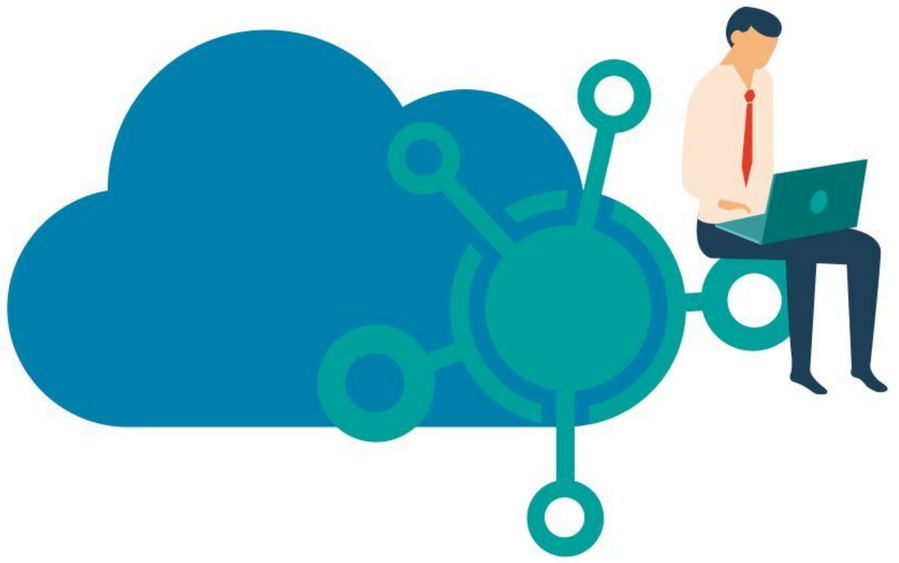
CLOUD-PROGRAMMIERSPRACHE GO

Für eigene Dienste, die bereit für den Einsatz in der Cloud sind, ist die Programmiersprache Go die erste Wahl. Ein praktischer Einstieg für Programmiererfahrene.

142 Entwickeln für die Cloud
148 Kommandozeile
154 Unit Tests
160 Server-Dienste
166 Zugangsdaten

ZUM HEFT

3 Editorial
53 **Aktion:** Videokurs Continuous Delivery von A bis Z
159 Impressum





Serverinfrastruktur mieten in der Cloud

Flexibilität, schnelle Skalierung bei Lastspitzen und niedrigere Kosten – das sind die Versprechen der Cloudanbieter. Mit dem Anmieten von Servern ist es aber nicht getan: Wer in die Cloud umzieht, sollte Anwendungen und Gewohnheiten anpassen. Nur dann profitiert man wirklich von der Cloudinfrastruktur.

Von **Jan Mahn**



Bild: Sven Hauth

Ihr Weg in die Cloud	6
Cloudprovider aus Europa	14
Cloudserver bestellen und einrichten	22
Automatisierung mit Ansible	26
Schnellstart mit Ansible	36
Cloud-Dienste mit Terraform	42
Kubernetes für Docker-Kenner	46

Dass Betreiber von Rechenzentren ihre Server vermieten, ist kein neues Geschäftsmodell, sondern seit Jahrzehnten üblich. Unter Bezeichnungen wie „Root-Server“, „vServer“ oder „Dedicated Server“ bekommt man ganze physische Server oder virtuelle Maschinen mit unterschiedlich viel zugesicherten Ressourcen bei zahlreichen Anbietern. Darauf läuft meist eine Linux-Server-Distribution, manchmal auch Windows Server. Einsetzen kann man die Maschinen für unterschiedlichste Serversoftware, vom Webserver über eine Videokonferenzsoftware bis zur KI-gestützten Datenauswertung. Gemein ist diesen Angeboten, dass man auf dem Server die volle Kontrolle über das Betriebssystem hat und nach Belieben Software installieren darf. Das unterscheidet das Serverhosting vom Webhosting – dort gibt es einen vorinstallierten Webserver (und oft einen SQL-Datenbankserver) und als Kunde darf man nur Dateien fürs eigene Webangebot in einem Ordner ablegen.

Üblich sind beim Serverhosting eher lange Vertragslaufzeiten: Mindestens einen Monat, gerne auch 12 oder 24 Monate muss man dem Vermieter treu bleiben und den Server nutzen. Der Wechsel auf ein anderes Paket, zum Beispiel mit mehr Leistung, ist meist mit Kosten oder gar manuellem Umzugsaufwand verbunden. Oft verlangen die Anbieter einen Obolus für die Ersteinrichtung.

Ab etwa 2010 nahm dann ein neues Geschäftsmodell Fahrt auf: **Cloud Computing**. Von den Marketingabteilungen wurde der Begriff seitdem inflationär benutzt und auch IT-Laien, die nie in Verlegenheit kommen würden, selbst Server zu mieten, haben schon von „der Cloud“ gehört. Dabei ist der Kern der Idee derselbe wie beim klassischen Hosting – vermietet werden vor allem Server, aber auch Speicherplatz oder gleich fertig installierte und vom Anbieter gepflegte Serverdienste wie zum Beispiel Datenbanken.

Minutenpreise

Der große Unterschied ist das Abrechnungsmodell: Abgerechnet wird bei Cloudanbietern in Sekunden, Minuten oder Stunden, nicht in Monaten oder Jahren, und die Mindestlaufzeit liegt oft nur bei einer Minute. Die Vermieter haben die Prozesse auf Basis von gängigen Virtualisierungstechniken so automatisiert, dass ein virtueller Server innerhalb von wenigen Sekunden einsatzbereit ist. Damit werden ganz neue Einsatzbereiche möglich: Ein Ingenieur beispielsweise, der für die Berechnung eines mathematischen Modells nur eine halbe Stunde am Tag eine Hochleistungsmaschine mit 12 Prozessorkernen und 64 GByte RAM braucht, kann eine solche in der Weboberfläche eines Cloudanbieters bestellen, darauf seine Berechnung ausführen und die Maschine nach 30 Minuten wieder löschen. Für ein paar Euro bekommt er so Rechenleistung, für die er sich sonst teure Hardware ins Büro stellen müsste.

Aber auch Dienste, die rund um die Uhr laufen, profitieren von der Flexibilität in der Cloud – bestes Beispiel ist ein typischer Webshop, der im Zeitraum von Ende November bis zum 24. Dezember im Weihnachtsgeschäft mehr Kunden bedienen muss als im restlichen Jahr zusammen. Mit einem klassischen virtuellen Server in einem langfristigen Tarif würde er in elf von zwölf Monaten für eine Infrastruktur bezahlen, die kaum ausgelastet ist, weil sie auf den Ansturm des Weihnachtsgeschäfts ausgelegt ist. Das lohnt sich nur für den Vermieter. Bei Cloudanbietern dagegen könnte der Händler die Ressourcen, die seiner gemieteten virtuellen Maschine zugewiesen werden, Mitte November in Erwartung des Advents-

Doppelte Bedeutung: Cloud für Anwender

Der Begriff „Cloud“ ist mindestens doppelt belegt. Admins und Entwickler verstehen darunter vor allem das Mietgeschäft mit virtuellen Servern – darum geht es auch in diesem und den folgenden Artikeln. Viele private Anwender denken vor allem an Online-Speicherplatz wie bei Dropbox, iCloud, & Co. Auch in Politik, Medien und Wirtschaft werden die Bedeutungen gern durcheinandergewürfelt.

ansturms anpassen und nur bezahlen, was auch gebraucht wird.

Milliardengeschäft

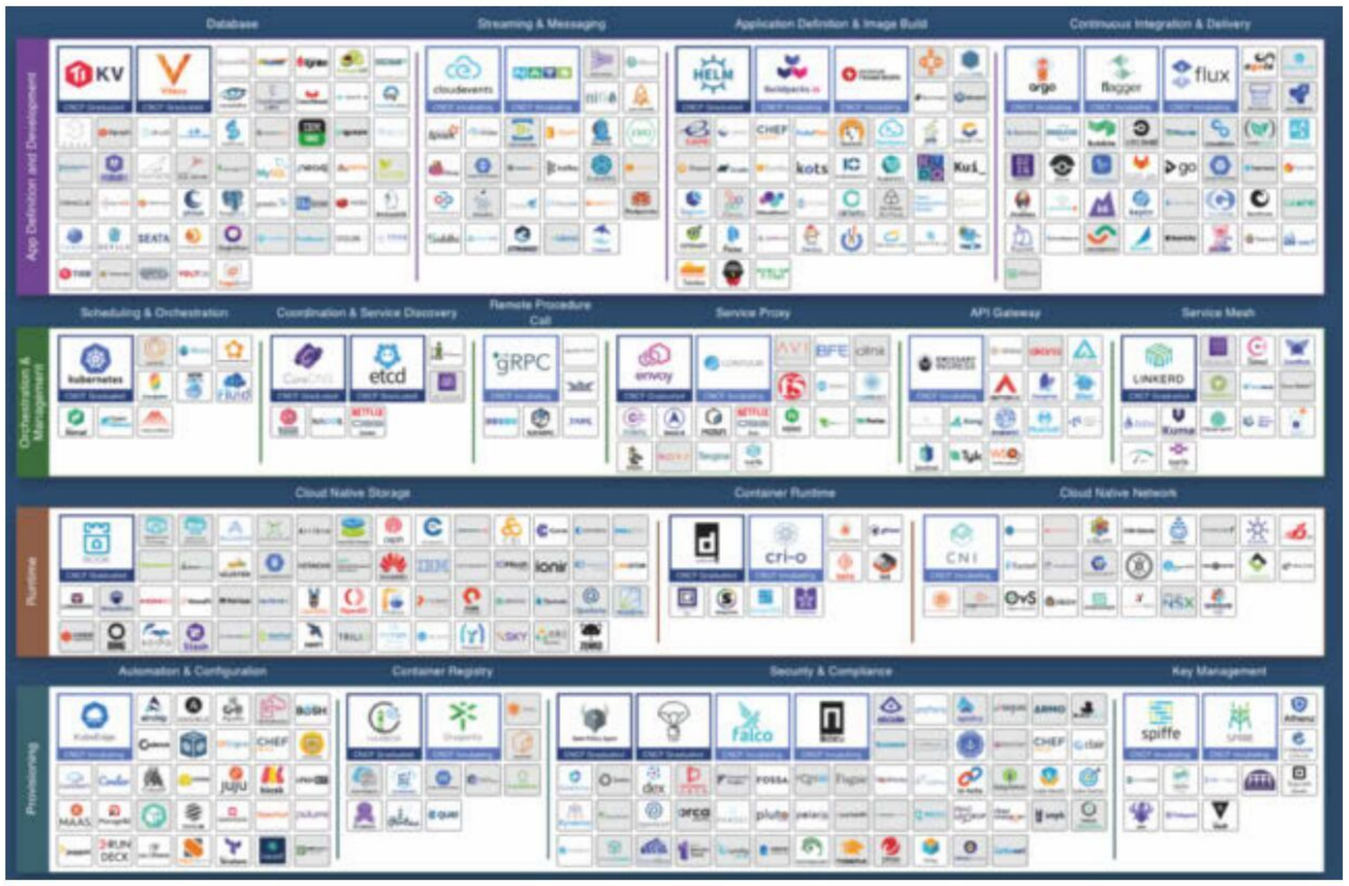
Dieses Modell ist für Kunden wie Anbieter gleichermaßen attraktiv. Allein im ersten Quartal 2021 hat Amazon mit seinen Clouddiensten „Amazon Web Services“ (AWS) 13,5 Milliarden US-Dollar umgesetzt. Seit 2014 konnte das Unternehmen den Umsatz in jedem einzelnen Quartal steigern. Entgegen der landläufigen Meinung bedeutet Cloud-Computing aber nicht zwangsläufig, sich in die Abhängigkeit von einem der drei großen US-Anbieter Google Cloud, Amazon AWS und Microsoft Azure zu begeben. Auch europäische und deutsche Anbieter vermieten Server, Speicherplatz und Datenbanken zum Minutenpreis – und für die DSGVO-konforme Datenverarbeitung in deutschen Rechenzentren müssen Sie nicht einmal mehr zahlen. In der Marktübersicht ab Seite 14 erfahren Sie, was Sie bei EU-Anbietern fürs Geld bekommen und worauf Sie neben dem Minutenpreis noch achten sollten.

Bei Unternehmenskunden ist die Flexibilität der Cloudanbieter beliebt: Für den im Juni 2021 veröffentlichten „Cloud-Monitor 2021“ befragte die Unternehmensberatung KPMG im Auftrag des Branchenverbands Bitkom über 550 deutsche Unternehmen nach ihrer Cloudnutzung. 82 Prozent gaben an, schon Clouddienste zu nutzen, 15 Prozent diskutieren und planen noch. 2016 waren es noch 65 Prozent Nutzer und 18 Prozent Planer. Cloudcomputer machen eigene Server meist nicht überflüssig: Nur 5 Prozent der befragten Unternehmen, die die Cloud verwenden, setzen auf eine Cloud-Only-Strategie, immerhin 31 Prozent haben eine Cloud-First-Strategie, versuchen also, neue Dienste primär auf Mietservern zu betreiben. Bemerkenswert an der Studie: 33 Prozent der Cloud-Kunden gaben an, keine Strategie zu verfolgen.

Vorteile ausnutzen

Dabei ist eine Strategie dringend empfohlen, bevor man aus dem eigenen Rechenzentrum in „die Cloud“ umzieht. Wenn man einfach die bisher im eigenen Haus oder beim Hoster laufenden Server zu einem Cloudanbieter umtopft, hat man nämlich nichts gewonnen.

Zum Umstieg gehört ein neuer Blick auf Server: Früher haben Administratoren ihre Server behandelt wie Haustiere. Die Maschinen hatten originelle Namen, wurden über Jahre gepflegt und bestenfalls von Betriebssystemversion zu Betriebssystemversion aktualisiert. Ging es ihnen schlecht, wurde der Fehler gesucht und ausgetrieben. Konfigurationsdateien wurden von Hand liebevoll aufgepäppelt. Ins Backup kamen nicht nur die anfallenden Daten, sondern gleich ein Abbild der ganzen Maschine. In vielen Umgebungen stehen noch immer solche in-



Mit dem Mieten von Servern ist es nicht getan. Bis Anwendungen stabil bei Cloudanbietern laufen, sind viele Probleme zu lösen. Eine Übersicht über Open-Source-Werkzeuge liefert das Angebot landscape.cncf.io.

Jetzt testen: ionos.cloud/trial

030 57700-840

IONOS CLOUD

Die europäische Cloud-Alternative

Performant: Bestes Preis-Leistungs-Verhältnis

Sicher: 100 % DSGVO-konform

Einfach: Intuitive & flexible Benutzeroberfläche

Fair: Pay-per-Minute Abrechnung & kein Vendor Lock-in

Kundenorientiert: Kostenloser 24/7 Enterprise-Level-Support

IONOS Cloud ist Teil von IONOS, Europas führendem Anbieter von Cloud-Infrastrukturen, Cloud-Services und Hosting-Dienstleistungen. Als einziger Cloud-Computing-Anbieter mit eigenem Code Stack "Made in Germany" wurden wir bereits mehrfach ausgezeichnet.

Public Cloud – Solutions and Services
Hyperscale Infrastructure and Platform Services
Rising Star Germany
ISG Provider Lens
2020

GOLD
INFRASTRUCTURE AS A SERVICE
CLOUDCOMPUTING INSIDER
AWARD 2020

techconsult
PUR 2020
★ CHAMPION ★
MANAGED CLOUD & DATACENTER SERVICES
MANAGED SERVICES

Internet made in Germany

IONOS

IONOS SE, Greifswalder Straße 207, 10405 Berlin • https://cloud.ionos.de • Tel.: (030) 57700-840

Konzepte, die man kennen sollte

Rund ums Cloudgeschäft gibt es Schlagworte, die immer wieder auftauchen. Einige werden inflationär eingesetzt, weil sie sich in Werbetexten gut machen. Dahinter stecken aber bewährte Konzepte und Denkweisen, von denen man in den meisten Umgebungen profitiert. Wer bisher seine Infrastruktur ganz klassisch administriert hat, sollte keinesfalls versuchen, alle Ideen auf einen Schlag umzusetzen.

CI/CD (Continuous Integration/Continuous Delivery): Früher, als Software noch auf CDs gebrannt und ausgeliefert wurde, waren die Release-Zyklen lang – gerne auch mal ein ganzes Jahr. Bis zu einem Stichtag sammelten die Entwickler alle Neuerungen, testeten sie und erstellten irgendwann einen „Golden Master“ für das Presswerk. Heute erwarten Kunden, dass neue Funktionen schneller fertig sind. **Continuous Integration** beschreibt eine Strategie, die beim Beschleunigen hilft. Jede Änderung am Code, die in der Versionsverwaltung (meist Git) eingecheckt wird, wird sofort automatisch von Skripten getestet und zu einem auslieferungsfertigen Paket verschnürt. Im Rahmen von **Continuous Delivery** werden die neuen Versionen schnell automatisch ausgeliefert. Meist erst in eine Testumgebung, dann ins Produktivsystem (Cloudserver helfen dabei). Das Motto „Release early, release often“ verhindert, dass sich viele kleine Änderungen ansammeln und das jährliche Update viel Ärger verursacht. CI/CD-Werkzeuge wie Jenkins, GitHub Actions oder GitLab übernehmen das automatische Bauen und Ausliefern.

Cloud-Native: Es muss nicht immer ein Clouddienst sein. Gerade aus Datenschutzperspektive ist es oft geboten, bestimmte Daten auf eigenen Maschinen zu verarbeiten. Das muss aber nicht dazu führen, dass man für Dienste im eigenen Haus auf die Vorteile der Cloud – Flexibilität, Skalierbarkeit und Reproduzierbarkeit – verzichtet. Software, die von den Entwicklern als **Cloud-Native** bezeichnet wird, ist meist für den Betrieb in Containern optimiert und kann sowohl auf gemieteten Servern als auch im eigenen Rechenzentrum eingesetzt werden.

Container: Mit Containern fing der Erfolg der Cloud so richtig an. Container sind Prozesse, die von der Container-Runtime so gekapselt sind, dass sie vom Rest des Betriebssystems nichts sehen. Anders als virtuelle Maschinen nutzen Container den Kernel des darunterliegenden Betriebssystems – sie blockieren also keinen Arbeitsspeicher und haben kaum Bootzeit. Container enthalten alle Abhängigkeiten, so kann man sie schnell starten, vervielfältigen, auf andere Maschinen oder gar in andere Rechenzentren übertragen. Erfolgreich wurde das Container-Konzept mit der Software Docker.

DevOps: Die Grundidee des Konzepts ist es, Entwicklung und Betrieb einer Software (**Dev**elopment und **Op**erations) nicht mehr strikt zu trennen. Entwickler sollen zum Beispiel schnell eine Kopie der Produktionsumgebung starten können, um ihre Änderungen zu testen. Dadurch kann man Änderungen zügig ausspielen, statt etwa halbjährlich große Versionssprünge zu machen. Flexible Mietserver und Konzepte aus dem Cloudumfeld (zum Beispiel CI/CD) helfen dabei, die DevOps-Konzepte umzusetzen.

Hybrid: Meist ist es nicht klug, auf einen Schlag die gesamte Serverinfrastruktur abzuschaffen und durch Mietserver zu ersetzen. In einer hybriden Umgebung laufen Dienste, die besser auf eigenen Servern laufen (weil sie zum Beispiel besonders sensible Daten verarbeiten), weiter im eigenen Haus. Dienste, die besonders flexibel sein müssen oder weltweit schnell erreichbar sein sollen, laufen in der Public Cloud. Hybride Strategien funktionieren besonders gut, wenn man auch lokal Software aus dem Cloud-Native-Umfeld einsetzt.

IaaS (Infrastructure as a Service): eine Wortkreation aus den Marketingabteilungen der Cloudanbieter. Gemeint sind mietdividuell hergerichteten Einzelstücke, die um jeden Preis am Leben erhalten werden müssen – weil sie eine wichtige Aufgabe erfüllen, aber niemand mehr genau weiß, welche Komponenten darauf in welcher Version installiert und konfiguriert werden mussten. Die Admins hängen daran wie an einem vergreisten Haustier, das astronomische Tierarztrechnungen verursacht. Das hat oft fatale Folgen: Diese Altlasten laufen gern mit hoffnungslos veralteten Betriebssystemen, weil sich niemand ans Upgrade traut. Änderungen müssen zwangsläufig im Produktivsystem ausprobiert werden, weil man keine Testumgebung mit vergleichbaren Bedingungen aufbauen kann.

bare Server und Speicherplatz – aber „Mietserver“ klingt nicht gerade attraktiv. Aus derselben Feder stammt die Bezeichnung **Platform as a Service (PaaS)**. Oft sind damit Angebote gemeint, bei denen man keine Server mit Root-Zugriff mietet, sondern zum Beispiel einen Webserver zum Minutenpreis.

Infrastructure as Code: Eine Idee der schönen neuen Cloudwelt ist es, dass man die Infrastruktur jederzeit wieder neu einrichten kann. Das klappt nicht, wenn man die Konfiguration per Hand per SSH vorgenommen hat. Stattdessen beschreibt man den gewünschten Zustand in Konfigurationsdateien (meist im YAML-Format) außerhalb der Infrastruktur und überlässt es Werkzeugen wie **Terraform** (siehe Seite 42) oder **Ansible** (siehe Seite 26), die Definition in die Realität umzusetzen. Die Konfiguration lagert man wie Programmcode in einer Versionsverwaltung (meist Git).

Public Cloud & Private Cloud: Als Public Cloud bezeichnet man alle Angebote für flexibel abgerechnete Server und weitere Dienstleistungen, die Kunden für Geld anmieten. Diese Flexibilität kann man auch innerhalb von Unternehmen erreichen: Mit Software wie OpenStack macht man eigene physische Server zur privaten Cloud: Ein virtueller Server ist dann schnell per Kommandozeile oder Weboberfläche bestellt. Unternehmen, die intern mit cloud-nativen Werkzeugen arbeiten, können ihre Dienste zügig aus der privaten in die öffentliche Cloud überführen, wenn es gute Gründe dafür gibt.

Semantic Versioning: Software-Versionsnummern sind meist nicht willkürlich, sondern folgen einem Schema. Besonders im schnelllebigen Cloudumfeld ist das wichtig. Die Versionsnummer 2.5.11 etwa besteht aus der Major-Version 2, der Minor-Version 5 und der Patch-Version 11. Patches sind kleinere Reparaturen, die Sicherheitsprobleme beseitigen oder Fehler abstellen. Minor-Änderungen bringen neue Funktionen mit, verändern aber keine bestehenden Funktionen. Entscheiden sich Entwickler für ein neues Major-Update, kann es elementare Änderungen an der Software geben, die manuelle Arbeit beim Umstieg erfordern. Wer Software im Cloudumfeld nutzt, sollte dieses Schema verinnerlichen und zum Beispiel nicht ohne Tests eine neue Major-Version einer Datenbank einsetzen.

SaaS (Software as a Service): ein Geschäftsmodell, bei dem eine Software meist zum Monatspreis vermietet wird. Das können webbasierte Anwendungen sein (wie etwa eine Buchhaltungssoftware), aber auch gemietete Datenbanken bei Cloudanbietern fallen in diese Kategorie. Bei SaaS muss man sich nicht um Administration, Updates oder Backups kümmern. Das übernehmen die Betreiber. SaaS-Anwendungen laufen oft auf Infrastruktur der großen Cloudanbieter.

Vendor Lock-in: Hat man sich nach gründlichem Nachdenken für eine Software entschieden, bleibt immer eine Angst: Was tun, wenn der Anbieter nächstes Jahr das Angebot einstellt oder unverhältnismäßig verteuert? Dahinter steckt die Befürchtung, dass man mit einer Entscheidung in einem Herstelleruniversum eingesperrt ist. Mit Open-Source-Software aus dem Cloud-Native-Umfeld kann man dem größtenteils aus dem Weg gehen. Wer sich etwa für Kubernetes als Plattform für seine Container entscheidet und dafür eine gemanagtes Kubernetes anmietet, kann die Umgebung zügig von einem zum anderen Anbieter verschieben. Vendor Lock-in gibt es aber auch bei Cloudanbietern, wenn man sich für Spezialösungen abseits von Speicherplatz und Rechenleistung entscheidet. Dazu gehören einige Datenbanken oder Dienste aus dem KI-Bereich, die man bei Cloudanbietern anmieten kann. Wer Vendor Lock-in vermeiden will, sollte nur Server und Speicherplatz mieten und damit Open-Source-Software selbst betreiben. Das macht mehr Arbeit beim Betrieb, verhindert später aber Probleme beim Umzug.

Dem gegenüber steht der neue Ansatz, der vor allem im Cloudumfeld verbreitet ist: Server werden nicht wie Haustiere, sondern wie Nutzvieh behandelt. Sie sind ersetzbar, müssen nur fleißig ihren Zweck erfüllen. Sie werden nicht gepflegt, sondern ausgetauscht, wenn sie schwächeln oder Ärger machen. Das funktioniert aber nur, wenn man seine Administrationsgewohnheiten ändert. Statt einen Server direkt per SSH oder RDP mit Klicken und Tippen zu verwalten, schreibt man reproduzierbare Rezepte, die jeden einzelnen Administrationsschritt enthalten. Diese Rezepte lagert man außerhalb der Server – das können im einfachsten Fall einfache Skripte sein, einfacher wird die Arbeit mit Anwen-

Software, die man kennen sollte

Neben Konzepten und Schlagworten begegnen einem früher oder später eine Menge Anwendungen mit teils originellen Namen. Erfolgreiche Software aus dem Cloudumfeld ist fast immer Open-Source-Software. Viele der erfolgreichsten Projekte werden unter dem Dach der Cloud Native Computing Foundation (**CNCF**) entwickelt, einer Stiftung unter dem Dach der Linux Foundation.

Ansible und Terraform: Mit Ansible erstellt man Rezepte, um Server zu bestellen und zu konfigurieren. Ansible ist ein Kommandozeilenwerkzeug auf dem Rechner des Admins. Zum Ausführen der Rezepte spricht es meist über SSH mit den Servern. Terraform ist spezialisiert auf das Bestellen von Servern bei zahlreichen Cloudanbietern, Ansible erledigt beide Aufgaben.

Docker: Mit Docker wurde die Idee von Softwarecontainern populär. Das Unternehmen hinter Docker betreibt den Docker Hub, noch immer die größte Ablage für Containerabbilder. Docker nutzt man meist in kleinen und mittleren Umgebungen und auf Entwicklermaschinen, um Container zu entwickeln.

Git: An Git führt kein Weg vorbei, wenn man programmiert. Git registriert Codeänderungen und erlaubt dabei verschiedene Entwicklungspfade zu verfolgen – so können zum Beispiel mehrere Entwickler parallel an Teilen der Software arbeiten und ihre Änderungen später zusammenführen.

GitHub und GitLab: die bekanntesten Hoster für Git-Repositories. Hier lagert man Code und kommuniziert mit anderen Entwicklern. Beide Anwendungen können mittlerweile aber noch viel mehr, haben zum Beispiel eingebaute CI/CD-Lösungen. Nach dem Hochladen von Code kann der automatisch kompiliert und verteilt werden. GitLab kann man selbst hosten.

Kubernetes: Werden Umgebungen auf Containerbasis größer, stößt man irgendwann an die Grenzen von Docker. Dann übernimmt Kubernetes (siehe Seite 46). Es verwaltet Cluster von Servern, die Container ausführen und kümmert sich um allerlei Probleme, die so nebenbei anfallen. Bevor man in Kubernetes einsteigt, sollte man Docker verstanden haben.

OpenStack: die Grundlage für viele private und einige öffentliche Clouds (zum Beispiel bei der Telekom) – eine Art Baukasten für Cloudserver. Die Open-Source-Software wird meist auf physischen Maschinen installiert und stellt Programmierschnittstellen bereit, über die man zum Beispiel virtuelle Maschinen und Laufwerke bestellen oder Netzwerke konfigurieren kann.

S3: S3 war ursprünglich nur das Blockspeicherplatz-Angebot von Amazon, hat es als proprietäre Software aber quasi zum Standard geschafft. In Blockspeichern werden Dateien in Buckets – quasi Ordner ohne Verschachtelung – abgelegt. Als Entwickler, der mit S3 arbeitet, muss man sich nicht mit typischen Fragen eines Dateisystems auseinandersetzen. Daten werden per HTTPS in S3 abgelegt und wieder abgerufen. Weil S3 einfach und gut skalierbar ist, nutzen es viele Anwendungen zum Ablegen von Dateien aller Art. Wer S3 nicht bei Amazon als Dienst anmieten will, bekommt mit dem Open-Source-Projekt **MinIO** ein S3-kompatibles System zum Selbsthosten.

dungen wie Ansible (siehe Seite 26). Weiter vereinfachen kann man sich das Leben, wenn man die Software auf den Servern noch in Container steckt und diese mit Docker oder Kubernetes ausführt. Im Kasten unten finden Sie kurze Erklärungen zu Software, die die Umsetzung einer Cloudstrategie einfacher macht und sich bereits bei Organisationen aller Größen bewährt hat.

Aber nicht nur die Servereinrichtung, auch die Serverbestellung muss reproduzierbar sein: Die

Cloudanbieter haben zwar alle Weboberflächen mit Bestellformularen für neue Server – diese nutzt man aber am besten so wenig wie möglich. Als professioneller Cloudkunde besucht man die Webseite nur am Anfang, um einen Account zu erstellen und die Kontaktdaten zu hinterlegen. Danach spricht man mit dem Anbieter nur über ein API – ebenfalls per Skript oder Ansible (siehe Seite B40). Diese Art der Arbeit ist am Anfang ungewohnt: Anstatt eine Umgebung an einem Stück einzurichten und nebenbei zu dokumentieren, tüftelt man einige Tage an den Rezepten, bestellt immer mal wieder Testserver, vernichtet sie wieder und bestellt neue.

In einer perfekten Welt hat man am Ende der Arbeit eine Rezeptsammlung, die alle wesentlichen Schritte enthält: Mit einem Befehl auf der Kommandozeile werden dann je nach Anforderung drei oder dreihundert Server bestellt, DNS-Einträge angelegt, mit Docker oder Kubernetes ausgestattet und die Container mit der Anwendung gestartet. Zugegeben, bis dieser Pfad reibungslos funktioniert, vergeht etwas mehr Zeit, als wenn man die Server per Hand zurechtbiegt. Aber mit einem guten Rezept startet man auch eine Testumgebung innerhalb von wenigen Minuten ohne Handarbeit oder eine Demo-Umgebung oder eine weitere Instanz für neue Kunden oder … Und auch ein Umzug von einem Cloudanbieter zum anderen sollte dann keine Panik verursachen: Rezept anpassen, ausführen und Anwendungsdaten kopieren.

Server in der Cloud werden nicht wie Haustiere, sondern wie Nutzvieh behandelt.

Gut geplant

Ein Umzug von eigenen Servern in die Cloud will gut überlegt sein. Mit einer Hauruck-Maßnahme, bei der man alte Probleme auf neue Infrastruktur umzieht, ist niemandem geholfen. Die Techniken und Strategien aus dem Cloud-Native-Umfeld lohnen sich aber auch, wenn man seine Dienste weiterhin nicht aus der Hand geben will. Auch im eigenen Rechenzentrum kann es nicht schaden, wenn man flexibel bleibt.

Wie ein Unternehmen ganz ohne eigene Server eine Anwendung für den Einsatz in der Cloud vorbereitet und welche Probleme es zu lösen gibt, lesen Sie ab Seite 22 – hat man sich erst an die neue Admin-Arbeit gewöhnt, macht das Umziehen durchaus Spaß. (jam) c't

Weitere Infos

ct.de/wgnx

Make:
KREATIV MIT TECHNIK

DAS KANNST DU AUCH!



2× Make testen und 6 € sparen!

Ihre Vorteile:

✓ **GRATIS dazu:** Arduino Nano
✓ **Zugriff auf Online-Artikel-Archiv***
✓ **NEU:** Jetzt auch im Browser lesen!
✓ Zusätzlich digital über iOS oder Android lesen

Für nur 15,60 Euro statt 21,80 Euro.

* Für die Laufzeit des Angebotes.

Jetzt bestellen:
make-magazin.de/miniabo

GRATIS!



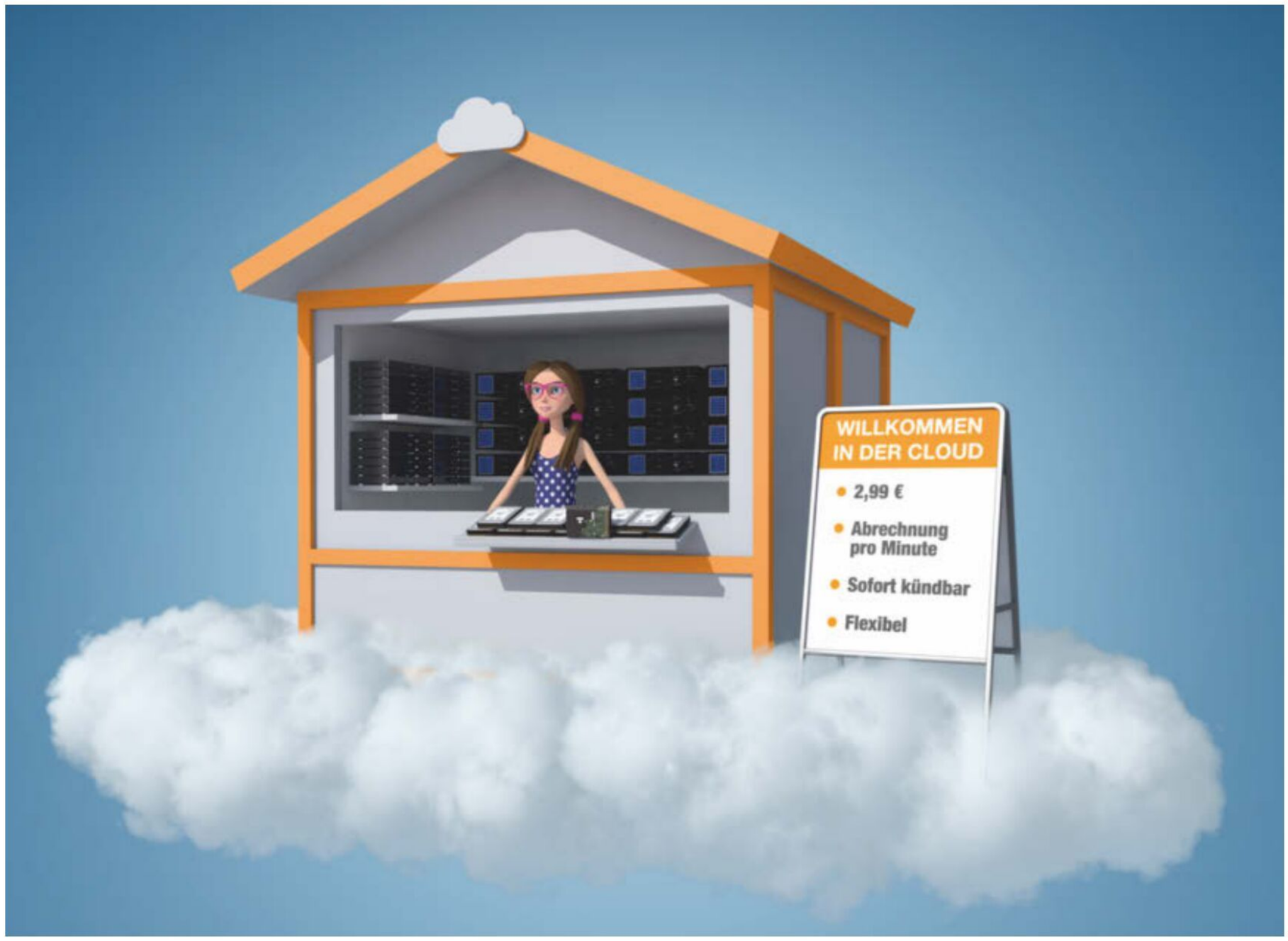


Bild: Sven Hauth

Cloudprovider aus Europa

Es muss nicht immer Amazon sein: Kleinere Cloudprovider bieten für einen schmalen Euro skalierbare Cloudressourcen ohne lange Vertragslaufzeiten. Wenige Klicks genügen meist, bis die Server am Netz sind. Alles geht, vom kleinen Gameserver bis zur Cluster-Infrastruktur fürs Online-Start-up.

Von **Holger Bleich und Jan Mahn**

Eigentlich sollte es selbstverständlich sein: Pay per Use – der Kunde bezahlt für das, was er in Anspruch nimmt. Doch lange Jahre haben sich vor allem bei deutschen Server- und Hosting-Providern Geschäftsmodelle etabliert, die auf langen Vertragslaufzeiten und Angebotspaketen mit vielen wenig genutzten Funktionen beruhen. Gut für den Provider, aber schlecht für seine Kunden.

Das hat sich gründlich geändert. Wer individuell konfigurierbare, skalierbare und vor allem nach tatsächlichem Verbrauch abgerechnete Server benötigt, steht vor einer stattlichen Angebotsvielfalt. Dank Servervirtualisierung und Optimierungen aufseiten der Hoster sind Cloudserver oft in Sekunden eingerichtet und noch schneller wieder entfernt. Und diese Server stehen nicht mehr nur dem Großkunden zur Verfügung, sondern auch dem freiberuflichen Entwickler, dem Start-up und sogar dem Gelegenheitsdaddler, der ab und an einen Game- oder TeamSpeak-Server starten und nur für die reine Nutzungszeit zahlen möchte.

Provider, die virtuelle Maschinen (VMs) anbieten, gibt es viele. Einige beschränken sich allerdings darauf, „Virtual Private Server“ (VPS) als unflexibles Produkt mit langer Vertragslaufzeit anzubieten. Diese Kategorie kommt hier nicht zur Sprache. Für diese Marktübersicht setzten wir voraus, dass VMs nach Belieben gestartet und gestoppt werden können und der Kunde nur die tatsächliche Betriebszeit bezahlt.

Schwer vergleichbar

Das Angebot lässt sich schwer direkt vergleichen. Während der eine Provider auf vorkonfigurierte

Amazon & Co. – für große und sehr große Wünsche

In diesem Artikel finden Sie eine Übersicht über deutsche und europäische Cloudanbieter für die typischen Anforderungen von kleinen und mittelgroßen Organisationen. Von der Nextcloud für einen Sportverein bis zum API für die Telemetrie von Industriemaschinen eines großen Mittelständlers kann man mit Mietservern viele Aufgaben abdecken.

Doch bei den hier vorgestellten Anbietern gibt es Grenzen, und zwar geografische: Bewusst haben wir deutsche und europäische Anbieter ausgewählt, die ihre Rechenzentren und den Firmensitz in der EU haben. So ist ein DSGVO-konformer Betrieb möglich. Wenn Sie beabsichtigen, Ihre Dienste auch für andere Märkte anzubieten und Wert auf extrem kurze Antwortzeiten legen, kann das ein Problem sein. An den Grenzen der Physik kann ein deutscher Hoster nicht viel ändern, und eine Anfrage aus Japan oder Australien ist etwas länger unterwegs. Für gelegentliche Anfragen aus dem Ausland ist das kein Problem. Wenn Ihr Kernmarkt aber in weiter Ferne liegt, ist ein lokaler Anbieter oder ein globaler Akteur mit Rechenzentren auf jedem Kontinent vielleicht besser geeignet.

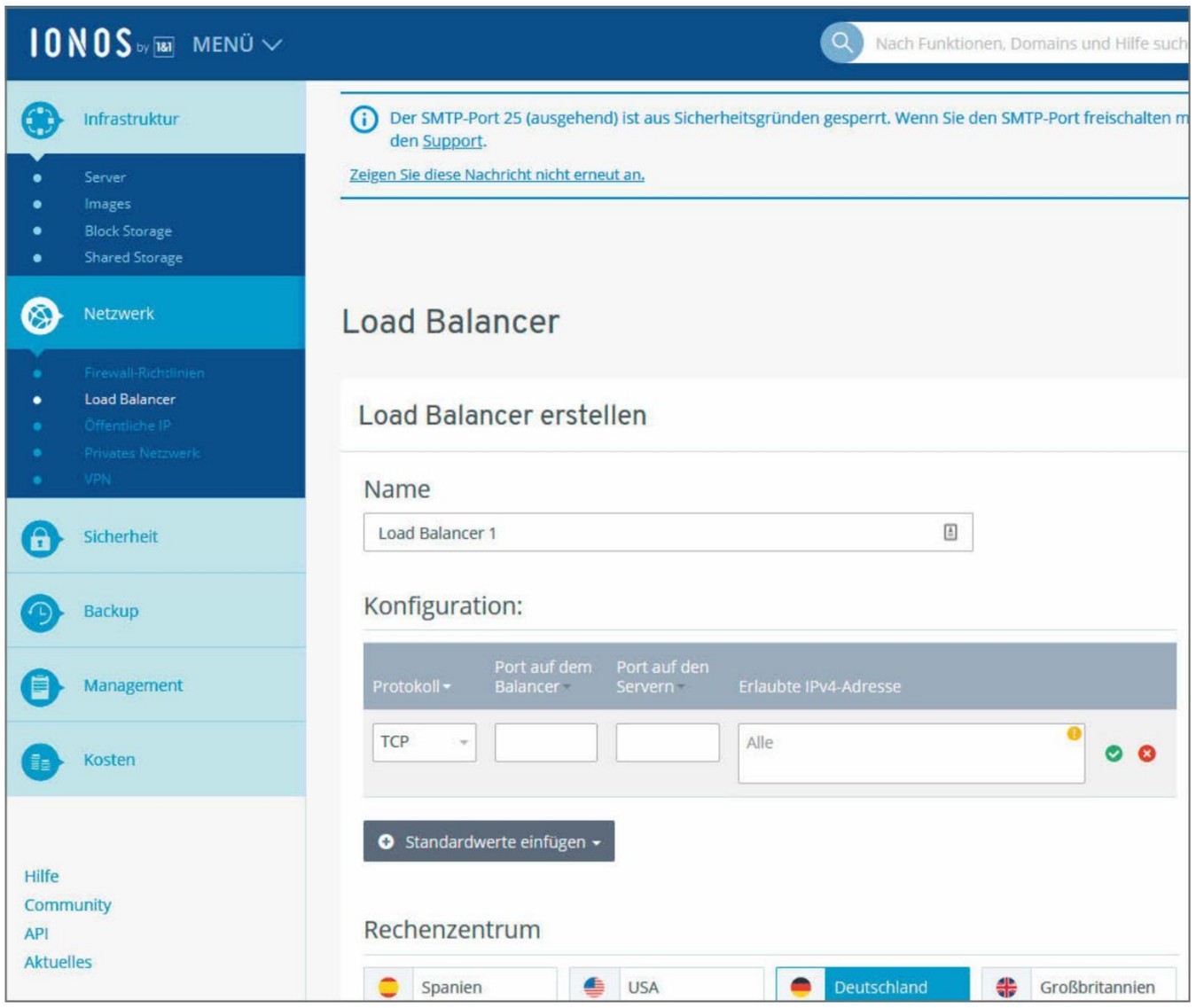
Die europäischen Anbieter stehen im Wettbewerb mit den drei großen Anbietern aus den USA: Amazon AWS, Microsoft Azure und Google Cloud. An dem Gespann führt auf der internationalen Bühne kein Weg vorbei, weil sie Rechenzentren auf der ganzen Welt betreiben und vor allem, weil sie außer Servern und Speicherplatz noch jede Menge andere Dienste vermieten: diverse Datenbanken, S3-Speicher, Archiv-Storage, aber auch so exotische Produkte wie gehostete Blockchains und APIs zur Text- und Bilderkennung.

Diese flexiblen und skalierbaren Pakete sind interessant, wenn Ihr Angebot global verteilt betrieben werden muss oder Sie eines der spezialisierten Produkte brauchen.

Im Gegenzug ist die Einrichtung von Servern, Netzwerken, IP-Adressen und Load Balancern wesentlich komplexer und schon die Benutzerverwaltung erfordert Einarbeitung. Das übertragene Datenvolumen („Traffic“) wird meist pro MByte abgerechnet – im Vorfeld ist es sehr schwer abzuschätzen, welche Kosten am Monatsende anfallen. Dass der Einstieg nicht gerade einfach ist, wissen auch Amazon, Google und Microsoft. Daher bieten sie Unternehmen zahlreiche Workshops (sicher irgendwann auch wieder in Präsenz) und Großkunden auch einen persönlichen Berater, der das Projekt betreut. Für Start-ups gibt es außerdem Förderprogramme mit großzügigem Budget für die ersten Monate.

Aber auch europäische Anbieter sind im Projektgeschäft für Großkunden aktiv. In Deutschland bekommen sie zum Beispiel bei der Open Telekom Cloud auf Wunsch einen persönlichen Ansprechpartner.

Unser Tipp: Starten Sie lieber klein bei einem europäischen Anbieter mit niedriger Einstiegshürde – solche finden Sie in diesem Artikel. Nur weil die Konfiguration von Servern und Netzwerken einfach ist, sind die angebotenen Dienste nicht schlechter oder ungeeignet für professionelle Ansprüche. Verwenden Sie von Beginn an nichts, was es exklusiv bei einem Anbieter gibt. Dann kann ein Umzug reibungslos in außereuropäische Rechenzentren gelingen, wenn Ihr Unternehmen irgendwann international expandiert.



In der Ionos-Cloud kann ein Load Balancer die Anfragelast über mehrere VMs verteilen.

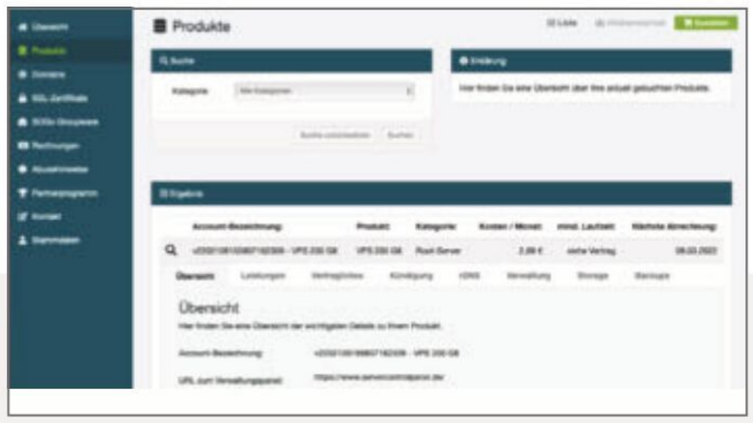
Bundles, also etwa eine fixe Kombination aus virtualisierten Prozessorkernen, RAM und Festplattenplatz setzt, gewährt der nächste mehr Flexibilität. Da sich diese Angebote keineswegs nur an Privatkunden richten, sondern genauso an Freiberufler und Unternehmen, spielen hinsichtlich des Datenschutzes der Serverstandort und der Hauptsitz des Providers große Rollen. In dieser Übersicht sind nur Provider berücksichtigt, die garantieren, dass ihre VMs innerhalb des DSGVO-Gebiets betrieben werden – die Daten also in Europa verbleiben.

In die Auswahl haben wir folglich europäische Provider aufgenommen, die mit Stunden- oder Minutenpreisen werben, unabhängig davon, ob sie ihr Produkt selbst als Cloud bezeichnen. Die Auswahlkriterien führen dazu, dass Sie in der Tabelle auf Seite 20 eine große Vielfalt vorfinden.

Stark am klassischen Hosting orientiert sich der deutsche Provider Netcup, der zwar stundengenaue Abrechnung verspricht, jeden gebuchten Server aber zunächst für sechs Monate in Rechnung stellt und nicht genutzte Zeit erstattet, wenn man vorher kündigt, was jederzeit möglich ist. Weil das API, über das man bei anderen Anbietern auch Server mieten kann, bei Netcup nur lesenden Zugriff gestattet, eignet sich dieses Angebot nicht, um automatisiert und spontan Maschinen zu starten (wie auf Seite 22 beschrieben).

Am anderen Ende des Feldes steht der französische Provider OVH mit seiner OVHcloud. Der hat sich ganz klar am Angebot der US-Schwergewichte im Cloudgeschäft orientiert und versucht, eine europäische Alternative für Profis anzubieten. Um hier Server oder eine der zahlreichen weiteren Ressourcen zu buchen, muss man mehr Hintergrundwissen mitbringen. Dennoch ist die Weboberfläche nicht mit den überkomplexen und verwirrenden Backends von Amazon oder Azure zu vergleichen (siehe Kasten auf Seite 15). Technisch setzt OVH auf dem Open-Source-Cloud-Unterbau OpenStack auf. Dieselbe Technik nutzt in Deutschland die Deutsche Telekom mit ihrer Open Telekom Cloud – die hat es nicht in diese Marktübersicht geschafft, weil sie keine Anmeldung für Privatpersonen gestattet.

Zwischen Netcup und OVH reihen sich Domainfactory, Ionos, Strato und Hetzner ein. Jiffybox, das Cloudserver-Produkt von Domainfactory, war einst Vorreiter in dieser Kategorie. Mittlerweile wirkt es nicht mehr ganz so frisch, weil die starre Vorkonfektionierung viel von eigentlich möglicher Flexi-

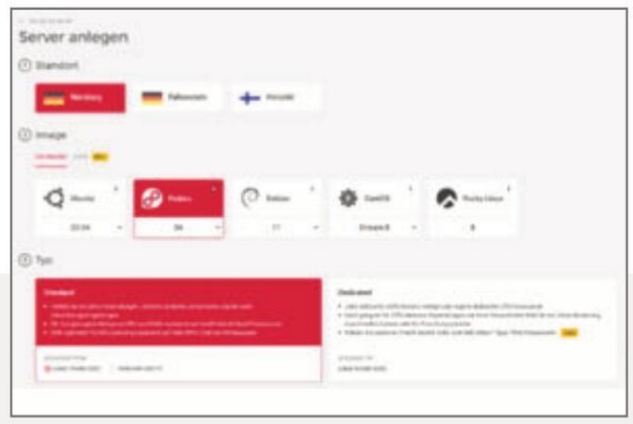


Netcup

Virtuelle Server vermietet Netcup unter dem Namen VPS sowohl zum Monats- als auch zum Stundenpreis. Ganz so flexibel wie die Cloudserver anderer Anbieter sind sie aber nicht: Jeden Server muss man im Kundenportal erstellen und bekommt dann – ebenfalls für jeden einzelnen Server – einen Account zum „Server Control Panel“, einer weiteren Oberfläche. Ein API zum Anlegen und Löschen von Servern gibt es nicht, das vorhandene API kann aber Informationen über einen Server auslesen. Server, die zum Stundenpreis gemietet werden, rechnet Netcup direkt nach der Bestellung für sechs Monate im Voraus ab. Wenn man sie vorher kündigt, wird die nicht genutzte Zeit erstattet.

All diese Einschränkungen qualifizieren Netcup nur für lang- und mittelfristige Projekte, nicht für Strategien mit automatisierter Erstellung und Einrichtung. Das cloudtypische Wegwerfen und Ersetzen von Ressourcen funktioniert trotz stundenbasierten Preisen mit Netcup VPS nicht sehr gut.

- **keine Bestellung von Servern per API**
- **trotz stundengenauer Abrechnung werden bei Bestellung 6 Monate im Voraus berechnet**



Hetzner

Der Einstieg ins Cloudgeschäft mit der Hetzner-Cloud geht recht einfach von der Hand. Die Weboberfläche ist aufgeräumt, und zwischen dem Anlegen eines Kontos und dem ersten laufenden Server vergehen nur wenige Minuten. Auch die Preisstruktur ist sehr transparent. Zahlen kann man per Lastschrift, Rechnung, Kreditkarte und PayPal.

Die Server kann man in Nürnberg, Falkenstein oder in Finnland hochfahren und zusätzlich noch skalierbare Volumes zum Stundenpreis mieten und einem Server hinzufügen. Wenn mehrere Server redundant arbeiten sollen, gibt es Load Balancer zur Miete. Die Netzwerkkonfiguration ist in der Standardeinstellung sehr einfach. Interne Netzwerke für Server untereinander lassen sich optional einrichten, eine kostenpflichtige „Floating IP“ kann man flexibel einem seiner Server zuweisen. Hetzner selbst pflegt eine Ansible-Erweiterung zum automatisierten Bestellen von Ressourcen. Nützlich gerade in der Ausprobierphase ist ein Preisalarm per Mail, den man für jedes Projekt einrichten kann.

- **übersichtliche Oberfläche und transparente Preisstruktur**
- **für Einsteiger und große Projekte gleichermaßen geeignet**

bilität nimmt. Ionos, das aus der Hosting-Sparte von 1&1 hervorgegangen ist, gehört zum United-Internet-Konzern. Das Cloudserver-Angebot strotzt vor Funktionen, die sich der Provider aber meist zusätzlich bezahlen lässt. Dies sollten Sie für Ihren eigenen Preisvergleich immer im Hinterkopf behalten.

Strato wurde vor einigen Jahren von United Internet geschluckt, und hier wirken augenscheinlich Synergieeffekte: Das Servercloud-Produkt sieht nur



OVH

Der französische Anbieter OVH präsentiert sich selbst gern als europäischer Gegenpol zu den US-Schwergewichten und ist auch bei der deutsch-französischen Europa-Cloud-Initiative Gaia-X seit der ersten Stunde dabei. Das Angebot von OVHcloud muss sich nicht hinter AWS oder Google verstecken. Außer Servern an Standorten in Frankreich, Kanada, Deutschland und Polen kann man auch Managed Kubernetes oder Rechner mit dicken GPUs für KI-Anwendungen buchen.

Obwohl man in der Registrierung (die Website spricht auch Deutsch) auch „Privatperson“ auswählen kann, richtet sich das Angebot klar an Profis – nicht nur, weil alle Preise ohne Mehrwertsteuer angegeben sind. Für absolute Neulinge lohnt es, gründlich die vielen anschaulichen Erklärtexte in der Oberfläche zu lesen. Fortgeschrittene finden sich auf der gut strukturierten Plattform schnell zurecht. Weil die OVHcloud auf OpenStack basiert, ist eine Fernsteuerung per Ansible und Terraform kein Problem.

- **großer Funktionsumfang, für große Projekte**
- **OpenStack als Unterbau, daher gängiges API**
- **Preise ohne Mehrwertsteuer**



Domainfactory

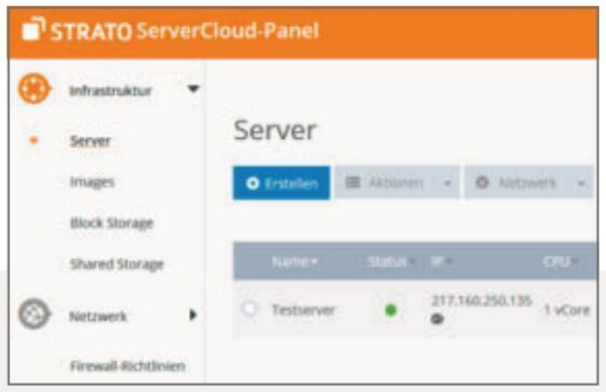
Flexible Cloudserver heißen bei Domainfactory Jiffybox. Hat man sich für eine der konfektionierten Konfigurationen entschieden, gelingt der Einstieg mit wenigen Klicks. Über die klar strukturierte Oberfläche erstellt, startet oder stoppt man maximal 20 VMs.

Einmal aktiviert, erstellt der Provider ohne Mehrkosten täglich Back-ups von jeder Maschine. Über das API lassen sich viele Vorgänge automatisieren. Leider kann man keine Ressourcen erweitern, es hilft nur ein generelles Upgrade in einen höheren (und teureren) Level. Dies wiederum erzwingt eine Downtime von etwa einer Stunde, was im Produktivbetrieb lästig sein kann.

Die Abrechnung erfolgt sekundengenau. Der Punkt „Verbräuche“ informiert über aktuell aufgelaufene Kosten. Praktisch: Mit der Preisbremse kann man einen Schwellenwert für im laufenden Monat auflaufende Kosten aktivieren. Wird er überschritten, erhält man je nach Einstellung eine Warnmail oder die VMs werden „eingefroren“. Wer die Maschinen als Test- oder Entwicklungsumgebung einsetzt und nur gelegentlich benötigt, wird diese Sicherheitsschicht zu schätzen wissen.

- **angenehm transparent und einfach zu bedienen**
- **eingebaute Preisbremse**
- **Upgrades mit langer Downtime**

von außen betrachtet anders aus. Die Kunden erwartet ein mit dem des Schwesterunternehmens Ionos nahezu identisches Produkt, dem eine Preisstruktur verpasst wurde, die eher ans althergebrachte Hosting erinnern, also beispielsweise mit Grundgebühr und Vertragslaufzeit. Auch das deutsche Webhosting-Urgestein Hetzner mischt seit 2018 im Cloudgeschäft mit und vermietet zum Stundenpreis. Außer Servern gibt es wie bei Ionos einen kostenpflichtigen Load Balancer, der eingehenden Verkehr



Strato

Wer spontan einen Cloudserver benötigt, wird bei Strato nicht glücklich: Der Hoster besteht auf einen länglichen Bestellprozess, bei dem er zudem lediglich SEPA als Zahlungsmittel gestattet. Selbst nach den versprochenen 24 Stunden war unser Account nicht freigeschaltet. Ein Anruf bei der Hotline ergab, dass die Order wohl „irgendwo im System hängengeblieben“ sei und nun reaktiviert würde. Aha.

Die Weboberfläche ist nahezu identisch mit der von Ionos, was kein Wunder ist, wurde Strato doch von 1&1 übernommen, wozu auch Ionos gehört. Augenscheinlich handelt es sich um dasselbe Produkt. Auf den zweiten Blick fällt aber auf, dass Strato einige Funktionen von Ionos nicht im Programm hat, vor allem die überaus sinnvolle Back-up-Option.

Strato setzt auf eine Preisstruktur, die eher dem Hosting entlehnt ist und damit spontaner Nutzung im Weg steht. So wird je nach „Level“ eine monatliche Grundgebühr ab 1 Euro fällig. Daraus resultieren eine Vertragslaufzeit und Kündigungsfrist von einem Monat und weniger Optionen innerhalb eines Levels. Beispielsweise darf man als „Entry“-Kunde nur bis zu zehn VMs anlegen.

- **sehr übersichtlich**
- **großer Funktionsumfang**
- **träger Bestellprozess und Grundgebühren**



Ionos

Ein vormodellierter Cloudserver ist bei Ionos fix bestellt und in wenigen Minuten ohne Laufzeitvertrag und Vorabzahlung startklar. Allerdings buchte uns der Provider sofort 30 und 10 Euro vom hinterlegten PayPal-Konto ab, um sie dann direkt zurückzuschicken. Offenbar checkt man so die Kontodeckung, ohne jede Erläuterung wirkt dieses Vorgehen freilich befremdlich. Auf unsere Nachfrage hin, gelobte der Provider Besserung.

Im Cloud-Panel von Ionos dürften sich sowohl Neulinge als auch Erfahrene sofort wohlfühlen: Die vielen Funktionen erschließen sich mit guten Hilfetexten sofort. Vieles kostet allerdings extra, was man erst erfährt, wenn man es aktivieren möchte, etwa ein Back-up oder zuschaltbarer Shared Storage, auf den mehrere Server zugreifen können.

Gut für Gruppen, etwa eine IT-Abteilung: Die Rollenvergabe ermöglicht es, mehreren Nutzern abgestufte Rechte zuzuweisen. Wem die Cloudserver mit ihren festen Ressourcen pro Instanz noch zu starr sind, der wechselt bei Ionos besser in die „Compute Engine“, wo er ähnlich wie bei OVH sehr flexible Ressourcen findet.

- **sehr übersichtlich**
- **großer Funktionsumfang**
- **viele Funktionen kosten extra**

auf mehrere Server verteilt, und skalierbaren Speicherplatz, den man auf gemieteten Servern einhängen kann. Die Weboberfläche beweist, dass auch ein Produkt für Profis aufgeräumt und leicht verständlich sein darf.

Prioritäten

Welche Funktionen, die über das Mieten eines einzelnen Servers hinausgehen, unbedingt notwendig sind, hängt vom geplanten Projekt und den Aufgaben

Marktübersicht Europäische Cloudanbieter

Anbieter	Domainfactory	Hetzner	Ionos	Netcup	OVH	Strato
Produkt	**JiffyBox Cloudlevel**	**Cloud**	**Cloud Server**	**VPS**	**OVHcloud Public Cloud**	**ServerCloud (Entry)**
Internet	https://www.df.eu/de/cloud-hosting/	https://www.hetzner.com/cloud	https://www.ionos.de/cloud/cloud-server	https://www.netcup.de/vserver/vps.php	https://www.ovhcloud.com/de/public-cloud/compute/	https://www.strato.de/server/server-cloud/
Standorte der Rechenzentren	Köln, Straßburg	Nürnberg, Falkenstein, Finnland	Baden-Baden	Nürnberg	Beauharnois (Kanada), Straßburg, Frankfurt, Gravelines, London, Warschau	Baden-Baden
Ausstattung						
max. vCores pro VM	12	32	24	16	32	16
max. RAM pro VM	32 GByte	128 GByte	48 GByte	64 GByte	240 GByte	32 GByte
max. Plattenpeicher pro VM	500 GByte SSD	600 GByte SSD	480 GByte SSD	2000 GByte SSD	400 GByte SSD	500 GByte SSD
max. Anzahl VMs	20	10 (mehr auf Anfrage im Control Panel)	unlimitiert	unlimitiert	unlimitiert	10
IP-Traffic (extern)	unlimitiert (100 MBit/s)	20 TByte	unlimitiert (400 MBit/s)	80 GByte/Monat unlimitiert, danach 100 Bit/s-Drossel	k. A.	unlimitiert (400 MBit/s)
Betriebssysteme	CentOS 7, Debian Buster, Debian Stretch, Dedora 24, open Suse 43.3, Ubuntu 18.04 LTS	CentOS 8, Fedora 34, Debian 11, Rocky Linux 8, Ubuntu 20.04	CentOS 8, Debian 10, OpenSuse Leap 15, Ubuntu 20.4, Windows Server 2019 (kostenpflichtig)	CentOS, Debian, Ubuntu	CentOS 8, Debian 10, Fedora 34, Ubuntu 21.04	Debian 10, CentOS 8, Ubuntu 20.04, Windows Server 2019
IPv4-Adressen	1 (3 weitere gegen Set-up-Gebühr möglich)	1	1	1	1	1 (bis zu 3 pro VM kostenpflichtig)
IPv6-Adressen	✓ (beta)	✓	✓	✓	✓	✓
API	✓	✓	✓	✓ (nur lesend)	✓	✓
Verwaltung per Ansible	–	✓	✓	–	✓	✓ (API)
Ansible-Collection	–	hetzner.hcloud	github.com/ionos-cloud/module-ansible	–	openstack.cloud	–
Verwaltung per Terraform	–	✓	✓	–	✓	✓ (API)
Load Balancer	–	✓ (kostenpflichtig)	✓	–	✓ (kostenpflichtig)	– (erst ab „pro“)
Back-up-Platz	✓	✓	✓ (kostenpflichtig)	✓ (kostenpflichtig)	✓	–
Verfügbarkeit (SLA)	99,9 % p.A.	99,9 % p.A.	99,99 % p.A.	99,6 % p.A.	99,999 % p.A.	99 % p.A.
Ressourcen-Änderung ohne Reboot	–	–	✓	–	–	–
VM Einfrieren	✓	–	✓	–	✓	✓
VM Klonen	✓	✓	✓	–	✓	✓
Kostenbremse	✓	–	–	–	–	–
Vertrag						
Zahlungsart	SEPA / PayPal / Rechnung	SEPA / Überweisung / PayPal / Kreditkarte	SEPA / PayPal / Kreditkarte	SEPA / Überweisung / PayPal	SEPA / Überweisung / PayPal / Kreditkarte	SEPA
Startguthaben erster Monat	–	–	100 €	–	– (teilweise Gutschein-Aktionen)	–
Zahlungsintervall	monatlich	monatlich	monatlich	6 Monate	monatlich	monatlich
Abrechnungstakt für Ressourcen	angefangene Sekunde	angefangene Stunde	angefangene Minute	angefangene Stunde	angefangene Stunde	angefangene Minute
Kündigungsfrist	–	–	–	–	–	1 Monat
Mindestvertragslaufzeit	–	–	–	– (Zahlung für 6 Monate im Voraus, Erstattung bei Kündigung)		1 Monat
Kosten						
Setup-Kosten (einmalig)	–	–	–	–	–	– (aber 1 € mtl. Grundkosten)
Preisbeispiele (naheliegendste, nächst höhere Konfiguration), pro Monat						
1 vCore + 1 GByte RAM + 20 GByte HDD	15 €	4,15 €	8 €	2,69 €	26,18 € (7 GByte, 2 vCores)	max. 7,92 €
4 vCore + 4 GByte RAM + 100 GByte HDD	40 €	14,76 €	50 €	14,30 €	49,98 € (15 GByte, 4 vCores)	max. 33 €

✓ ja – nein k. A. = keine Angaben

der Server ab. Bestellen und Löschen von Ressourcen über eine Programmierschnittstelle (API) ist erforderlich, wenn Sie planen, per Skript oder mit Werkzeugen wie Ansible und Terraform zu arbeiten und Ihre Infrastruktur mit einem Befehl auf der Kommandozeile hoch- und herunterzufahren.

Sobald Sie planen, mehrere Server anzumieten und diese zu einem Cluster zu verbinden, wünschen Sie sich von Ihrem Provider eine Möglichkeit, die Server untereinander in privaten Netzwerken zu verbinden und einen Load Balancer als Dienst anzumieten. Wenn Sie ein solches Szenario planen oder sich die Option zumindest für die Zukunft offenhalten wollen, kommen Hetzner, Ionos, Strato und OVH infrage.

Im Unterschied zu den großen IaaS-Angeboten (Erklärungen zu Abkürzungen finden Sie im Glossar ab Seite 6) von Amazon & Co. ist der ein- und ausgehende IP-Traffic bei den Angeboten in der Marktübersicht enthalten. Allenfalls eine Drossel kann eingebaut sein (siehe Tabelle unten). Das ist gerade dann ein großer Pluspunkt, wenn ein Projekt auch dann finanziell kalkulierbar bleiben soll, wenn der Traffic plötzlich durch die Decke schießt. Interner Datenverkehr von Servern im Rechenzentrum untereinander wird ohnehin nie berechnet und läuft mindestens im GBit/s-Tempo.

Apropos Kosten: Die Angebote verleiten zum munteren Ausprobieren. Schnell kann man sich eine Infrastruktur zusammenklicken, die mit mehreren Hundert Euro monatlich zu Buche schlägt. Deshalb zu guter Letzt ein Tipp aus leidvoller Redakteurserfahrung: Alle Provider zeigen irgendwo im Menü die aufgelaufenen Kosten und manchmal auch Kostenprognosen. Werfen Sie gerade zu Beginn öfter mal einen Blick drauf, um am Ende des Monats böse Überraschungen zu vermeiden. Bei Hetzner kann man sich eine Mailbenachrichtigung bei Überschreiten eines Limits einrichten, bei Domainfactory gibt es eine harte Preisbremse. Nutzen Sie derlei Optionen, um sich aufs Wesentliche konzentrieren zu können. (hob) c't

Alte Schwachstellen in neuer Technologie: IT Security hört nicht in der Cloud auf

Sicherheit in der Cloud ist so vielfältig wie der Begriff Cloud selbst. Darauf muss IT Security reagieren.

Cloud: ein wolkiger Begriff
Auch wenn der Name es suggeriert: Nextcloud muss nicht Cloud sein. Containertechnologien wie Kubernetes können Cloud sein, offene Cloudsysteme wie z. B. OpenStack sind meistens Cloud, Cloud Service Provider wie AWS oder MS Azure sind Cloud.

Alte Probleme in neuer Technologie
Einige Schwachstellen aus der alten Welt sind in die Cloud mit umgezogen. Manche Probleme im Bereich Netzwerk- und Betriebssystemsicherheit bleiben bestehen. In Penetrationstests kombiniert die SySS ihre langjährige Erfahrung mit neu aufgebautem Cloudwissen, um alle Schichten von Cloudinfrastrukturen abzusichern.

Pentests neu denken
Von der Microsoft 365-Analyse bis hin zum AWS-Organisationsaudit: Cloudtests gibt es in allen Größen und Formaten. Dabei hat sich ein offenes Vorgehen als zielführend erwiesen und steht komplementär zum klassischen Pentest: Audit der Cloudinfrastruktur, Pentest der Applikation.

Geteilte Verantwortung – ein Sicherheitskonzept
Cloudanbieter sprechen vom „Shared Responsibility"-Konzept. D. h. ein Teil der Verantwortung liegt immer noch bei den Nutzern. Die SySS unterstützt ihre Kunden, genau diese Bereiche zu identifizieren und zu prüfen. Die Themen reichen dabei von kleinteiliger Konfigurationsanalyse bis hin zu allgemeinen Prozessthemen.

Finden Sie Cloud Security genauso spannend wie wir? Haben Sie Spaß an der sich ständig ändernden Technologielandschaft Cloud? Dann helfen Sie uns, alle flAWS zu finden und bewerben Sie sich:
https://www.syss.de/jobs-karriere/syss-als-arbeitgeber





Bild: Sven Hauth

Cloudserver bestellen und einrichten

Wer heute eine gute Idee für ein Online-Angebot hat, braucht keine eigenen Server und kein Rechenzentrum, wohl aber einen Schlachtplan für den Weg in die Cloud. Viele kleine und größere Probleme müssen Start-ups und große Unternehmen gleichermaßen lösen, bevor der eigene Code in gemieteter Umgebung läuft.

Von **Jan Mahn**

Am Anfang steht die Idee: Eine App und eine Website sollen entstehen, über die man Fotos hochladen und mit Freunden teilen kann. Das Entwicklungsteam ist klein: Einer kümmert sich um die mobilen Apps, einer um das Web-Frontend und ein Dritter baut das Backend, das Bilder und Benutzerinformationen speichert und wieder anzeigt. So oder so ähnlich sind viele IT-Start-ups personell aufgestellt, für eine ganze Abteilung, die sich um den Betrieb eigener Server kümmert, ist weder Geld noch Platz im Co-Working-Bürogebäude da. Außerdem soll sich die App bestenfalls gegen Instagram, Twitter und Facebook durchsetzen und zum weltweiten Erfolg werden – dann wäre ein einziges Rechenzentrum in Deutschland zu wenig und zu unflexibel.

Also entscheiden sich die Start-up-Gründer dafür, die Anwendung von Anfang an auf gemieteter Infrastruktur zu betreiben. Das Start-up, das sich diese Anwendung ausgedacht hat, heißt Piqup Ventures und ist rein fiktiv. Dieser Artikel vermittelt einen Eindruck, welche Probleme die Entwickler auf dem Weg zu einer skalierbaren Cloud-Anwendung lösen müssen: Von DNS über Containerisierung, automatische Updates, skalierbaren Speicherplatz bis hin zu Monitoring gibt es viel einzurichten. Auf den nächsten Seiten geht es zunächst um ein Rezept für einen einzelnen Server – wie man es schafft, dass beliebig viele gemietete Server im Verbund arbeiten, erfahren Sie in einer der nächsten Ausgaben.

Dieser Überblick sollte nicht als Schritt-für-Schritt-Anleitung zum Nachbauen verstanden werden – ei-

nige der vorgestellten Werkzeuge erfordern durchaus eine längere Einarbeitungszeit. Wie immer in der IT führen bei jedem Teilproblem zahlreiche Wege zum Ziel und es gibt immer gute Gründe, andere als die hier beschrittenen Pfade zu wählen. Wenn Sie Ideen aus diesem Artikel für die eigene Cloud-Strategie übernehmen wollen, finden Sie das beschriebene Rezept über ct.de/w8ma.

Reproduzierbar und flexibel

Wichtig ist den fiktiven Unternehmern von Piqup, dass die gesamte Einrichtung ihrer Infrastruktur ohne Handarbeit funktioniert. Die Entwickler träumen davon, mit nur einem Befehl neben der produktiven Umgebung app.piqup.de schnell eine baugleiche Testumgebung unter der Adresse testing.piqup.de hochzufahren und der Geschäftsführer würde gern auf Knopfdruck eine Demo-Umgebung für Produktpräsentationen starten können. Damit das klappt, sollen alle Schritte reproduzierbar sein und in einem Rezepte-Ordner liegen. Im einfachsten Fall könnte das eine Sammlung von Bash-Skripten sein.

Bequemer wird die Arbeit mit dem populären Werkzeug Ansible. Eine ausführliche Einführung in das Open-Source-Programm lesen Sie auf Seite 26. Die Grundidee: In sogenannten Playbooks im YAML-Format definiert man, welche Konfigurationsschritte Ansible erledigen soll. Über Plug-ins kann Ansible zum Beispiel mit den APIs von Cloudanbietern sprechen und Ressourcen einkaufen, es kann sich aber auch per SSH mit Servern verbinden und dort Einrichtungsschritte vornehmen und so auch komplexeste Anwendungen einrichten. Der große Vorteil: Die Einrichtungsschritte sind idempotent – das bedeutet, dass man das Rezept ohne Schaden immer wieder ausführen kann. Ansible achtet darauf, dass nur Änderungen umgesetzt werden und die mehrfache Ausführung eines Rezepts keine negativen Folgen hat. Vereinfacht gesagt: Wenn man schon drei Server bestellt hat und gerne noch einen vierten hätte, muss man nur im Rezept eine 3 in eine 4 ändern und das Rezept erneut ausführen.

DNS-Automation

Der erste Kontakt des Nutzers mit der vom Beispiel-Start-up entwickelten Anwendung führt immer über das DNS (Domain Name System). Ein Hostname wie app.piqup.de muss zu einer IP-Adresse aufgelöst werden, unter der einer der Webserver zu erreichen ist. Dafür braucht das Unternehmen Piqup einen sogenannten Provider, der die Domain piqup.de bei der deutschen Registrierungsstelle Denic registriert und dort die Adresse zu einem Nameserver hinterlegt. Viele Hoster bieten ihren Kunden diese Dienste im Paket mit anderen Dienstleistungen an. Gerade in klassischen Webhoster-Paketen oder bei virtuellen Servern im langfristigen Tarif ist eine Domain oft im Preis enthalten und der Hoster stellt auch den DNS-Server bereit.

Es gibt aber gute Gründe, die Aufgaben von Provider und DNS-Anbieter von Anfang an zu trennen und die eigene Domain nicht an ein vermeintlich günstiges Paket vom Hoster zu binden. Solche in Webhosting integrierten DNS-Verwaltungen kann man meist nur über eine Weboberfläche steuern – in der schnelllebigen Cloud-Welt möchte man aber nicht jede IP-Adresse jedes Mietservers per Hand dort eintippen. Wenn man sich für einen unabhängigen Dritten als reinen Domainverwalter entscheidet, erspart man sich außerdem Probleme, die entstehen können, wenn man sich irgendwann einem anderen Anbieter für die Server zuwendet.

Die Start-up-Gründer sind daher gut beraten, sich einen Provider zu suchen, der Domains als lose Ware zum Stückpreis anbietet. Der Betreiber des DNS-Servers kann davon völlig unabhängig sein. Viele setzen für die DNS-Verwaltung ihrer Domains mittlerweile auf das US-Schwergewicht Cloudflare: Dessen Schwerpunkt ist die Abwehr von DDoS-Attacken, ganz nebenbei betreiben sie aber auch DNS-Server – wenn man keine Extras braucht, ist das Angebot kostenlos.

Bei einem deutschen Domainhändler bestellen die Piqup-Admins daher nur die Domain und hinterlegen dort die Adressen zu zwei Nameservern von Cloudflare. Der große Vorteil: Alle Funktionen der Weboberfläche von Cloudflare kann man auch per API fernsteuern und damit gut skripten. Bestenfalls nutzt man die Weboberfläche nur einmalig, um das API-Token zu generieren, und arbeitet dann mit reproduzierbaren Rezepten per API. Besonders komfortabel wird das mit Ansible – weil Cloudflare weit verbreitet ist, gibt es ein Plug-in für Ansible, das die API-Kommunikation abstrahiert. Mit wenigen Zeilen entsteht ein Rezept für den A-Record (app.piqup.de) der Piqup-App:

```
- name: Create A record
  community.general.cloudflare_dns:
    zone: piqup.de
    record: app
    type: A
    value: "<IP-Adresse>"
```

```
    account_email: john.doe@piqup.de
    account_api_key: "<API-Token>"
  register: record
```

Fertige Server, bitte

Für einen DNS-Eintrag fehlt bisher noch ein entscheidendes Detail: die IP-Adresse. Die bekommt man vom Cloud-Hoster erst, wenn man einen Server bestellt, und beim Umgang mit Netzwerkkonfiguration und Firewall unterscheiden sich die Hoster stark. Teilweise muss man eine externe IP zusätzlich konfigurieren. Der Einfachheit halber hat sich Piqup für einen Anbieter entschieden, der jedem gemieteten Server für den Mietzeitraum eine öffentliche IPv4-Adresse zuteilt.

Als Entwickler, der jetzt ein Ansible-Rezept schreiben soll, beginnt man mit einem Blick in die Weboberfläche des Serververmieters und verschafft sich einen Überblick, welche Produkte und Optionen es überhaupt gibt. In der Oberfläche hinterlegt man dann seine Zahlungsdaten und erzeugt ein API-Token, das Ansible für die Kommunikation erwartet. Anschließend muss man sich mit der Syntax des Ansible-Moduls vertraut machen, über das man mit seinem Hoster kommuniziert. In der Tabelle ab Seite 20 erfahren Sie, wie Sie die vorgestellten Anbieter per Ansible fernsteuern.

Als Beispiel hält ein Server vom Typ CX11 des deutschen Anbieters Hetzner her, für den es ein Ansible-Plug-in gibt:

```
- name: create a server
  hetzner.hcloud.hcloud_server:
    name: "server1"
    server_type: cx11
    location: fsn1
    image: ubuntu-20.04
    ssh_keys:
    - piqup-ansible
    state: present
    api_token: "<API-Token>"
  register: server
```

Passende Werte (etwa für den Standort des Rechenzentrums oder die installierbaren Betriebssysteme) muss man sich aus Dokumentation und schlimmstenfalls aus Forenbeiträgen zusammensuchen. Beim Zusammenbau der Ansible-Rezepte sollte man nicht geizig sein, sie durchaus mal mutig ausführen, die Server wieder löschen und solange neu bauen, bis alles rund läuft. Bei Stundenpreisen im Centbereich kann man problemlos mal ein paar Testmaschinen starten und kurz darauf vernichten. Wichtig ist nur, am Ende des Arbeitstages mit einem Blick in die Weboberfläche sicherzustellen, dass man keine Kostenverursacher vergessen hat.

Im obigen Beispiel wird direkt ein SSH-Schlüssel hinterlegt, mit dem sich Ansible in weiteren Schritten am neuen Server anmelden kann. Den Public Key muss man vorher beim Anbieter hinterlegen, das private Gegenstück muss auf der Entwicklermaschine liegen, auf der Ansible ausgeführt wird. Abgerundet wird der obige Ansible-Schnipsel mit dem Befehl `register: server`. Ansible speichert damit die Rückgabewerte des APIs in der Variable `server`, die man zum Beispiel hervorragend nutzen kann, um im zweiten Schritt einen DNS-Eintrag bei Cloudflare mit der gerade vergebenen IP-Adresse anzulegen:

```
value: "{{ server.hcloud_server.↵
                  ↳ipv4_address }}"
```

Bei aller Flexibilität und Abstraktion muss man sich bewusst sein, dass ein Umzug zu einem anderen Hoster mit etwas Arbeit verbunden ist: Die Zeilen im Ansible-Rezept, die in diesem Beispiel für Hetzner ausgelegt sind, müsste man ersetzen, wenn man zum Beispiel zu OVH wechseln will. Vergleichsweise einfach ist ein Umzug unter allen Anbietern, die im Hintergrund OpenStack einsetzen und das OpenStack-API zur Steuerung öffentlich zugänglich machen.

Tücken lauern auch bei der eingangs gelobten Idempotenz: Nicht jeden Wert kann man nach dem Erstellen einer Ressource noch ändern. Wenn Sie gehofft hatten, nur den Namen des Rechenzentrums in der Ansible-Datei austauschen zu müssen, um einen Server umziehen zu lassen, werden Sie meist enttäuscht. In diesem Fall muss man den Server löschen und einen neuen anlegen – sofern er automatisch eingerichtet wird, ist das aber kein Problem.

Containerisierung

Server und DNS-Einträge sind jetzt fertig und ein Rezept kann die Linux-Maschinen mit Software ausstatten. Mit nur drei Zeilen in einem Playbook kümmert sich Ansible zum Beispiel darum, dass der Webserver Nginx per Apt aus den Paketquellen installiert wird:

```
- name: Install nginx
  apt:
    name: nginx
```

Grundsätzlich spricht nichts dagegen, Cloudserver auf diese Weise einzurichten. Es gibt jedoch überzeugende Argumente dafür, einem der größten IT-Trends der letzten Jahre zu folgen und die Anwendungen in Container zu stecken. Im einfachsten Fall startet man solche mit Docker; wenn man größere Umgebungen plant, setzt man auf Kubernetes (siehe Seite 46).

Vorteile von Containerisierung gibt es viele: Weil ein Container alle Abhängigkeiten in der genau passenden Version mitbringt, kann man auch mehrere Versionen derselben Software auf einem Server betreiben. Außerdem verschwinden Container nach dem Löschen spurlos. Der größte Vorteil: Eine Zusammenstellung aus denselben Container-Abbildern verhält sich immer gleich (Ausnahmen bestätigen auch hier die Regel). Die Entwickler können ganz ohne Aufwand mit virtuellen Maschinen eine exakt identische Testumgebung auf ihren Laptops starten.

Die Piqup-Gründer entscheiden sich, zunächst einen Prototyp mit Docker auf einem einzigen Mietserver zu starten und heben sich den Umzug auf eine redundante und skalierbare Umgebung für später auf. Das ist eine empfehlenswerte Herangehensweise: Versuchen Sie nicht, alles auf einen Schlag zu erzwingen und das perfekte System zu entwerfen. Sammeln Sie zunächst Erfahrungen mit einem einzelnen Docker-Host. Beim redundanten Betrieb im Cluster wird alles schlagartig komplexer. Docker für Linux bekommt man mit Ansible zügig auf die Maschine, das komplette Ansible-Rezept zur Docker-Installation (etwa 40 Zeilen YAML) finden Sie über ct.de/w8ma.

Container-Zusammenstellungen für Docker beschreibt man ebenfalls im YAML-Format für das Kommandozeilenprogramm Docker-Compose. Darin definieren die Piqup-Entwickler die Container, die für ihre Anwendung zusammenarbeiten müssen: eine Datenbank, ein Container mit dem Backend und einen mit der Weboberfläche. Außerdem einen HTTP-Router, der den eingehenden Verkehr abwickelt und sich um ein TLS-Zertifikat von Let's Encrypt kümmert. Ein Projekt mit dem Open-Source-Router Traefik finden Sie auf Seite 108.

Damit die von den Piqup-Entwicklern hergestellten Container-Abbilder installiert werden können, müssen sie irgendwie in eine Registry gelangen. Lange Zeit war die Docker Registry die erste Wahl, um eigene Abbilder abzulegen. Weil die PiqupEntwickler ihren Code ohnehin bei GitHub verwalten, entscheiden sie sich, die GitHub-Registry zu nutzen und die Container automatisch von GitHub Actions bauen zulassen (siehe Seite 74). Ein Umzug in eine eigene Registry (zum Beispiel ein selbst gehostetes GitLab) ist später kein großer Aufriss.

Die Docker-Compose-Datei legen die Entwickler zusammen mit ihren Ansible-Dateien in einem Rezepteordner ab. Um die Zusammenstellung zu starten, muss niemand per Hand auf der Kommandozeile `docker-compose up` eintippen. Zum Starten von Compose-Zusammenstellungen gibt es ein Ansible-Modul:

```
community.docker.docker_compose:
  project_src: /srv/piqup
  remove_orphans: true
```

Seinem Ziel, reproduzierbare Umgebungen bei Cloud-Providern zu erzeugen, ist das Start-up bereits sehr nahe gekommen. Kombiniert man all diese Bausteine zu einem Playbook, reicht am Ende ein Befehl wie

```
ansible-playbook -i inventory site.yml
```

Danach kann man etwa fünf Minuten lang Ansible bei der Arbeit zusehen. Am Ende läuft, ganz ohne Benutzerinteraktion, ein Cloudserver mit installiertem Docker, es gibt einen DNS-Eintrag, auf dem Server läuft die Anwendung, veröffentlicht per HTTPS.

Datenhalde

In schicken Whitepapern und werblichen Konferenzvorträgen endet die Erzählung hier. In Sekunden, so das Versprechen, das man dort liest und hört, läuft eine „stateless application". Was dabei unterschlagen wird: Nur die allerwenigsten Anwendungen sind „stateless", produzieren also keinerlei Nutzdaten, die dauerhaft gespeichert werden müssen. Das trifft eigentlich nur auf statische Websites zu – schon ein einfaches Blog mit einem CMS-Backend ist nicht stateless, sondern braucht zwei Formen von Speicherplatz: ein Dateisystem für hochgeladene Bilder und Dokumente sowie eine (SQL-)Datenbank.

Und genau dort wird es schnell kompliziert, wenn die Software auf mehreren Servern redundant laufen soll und die Daten überall aktuell sein müssen. Statt Docker ist dann Kubernetes (siehe dazu Seite 46) als Container-Orchestrator das Mittel der Wahl. (jam) c't

Ansible-Beispiel

ct.de/w8ma



Bild: Albert Hulm

Automatisierung mit Ansible

Ansible hilft, die Zauberformel „agile IT“ in die Tat umzusetzen. Es erledigt Konfigurationsaufgaben sicher, reproduzierbar und zuverlässig. Mit seiner Hilfe entsprechen Betriebssysteme, Applikationen und andere Komponenten den gesetzten Standards – die IT versinkt nicht im Wildwuchs.

Von **Götz Rieger**

Automatisierungslösungen in der IT sind schon lange Bestandteil des Werkzeugkastens von Admins. Sie versorgen frisch eingerichtete Server mit einer Grundkonfiguration, etwa SSH-Keys und Adressen von Nameservern, oder lassen ganze Cloud-Infrastrukturen aus dem Nichts entstehen. Gemessen an Software wie CFEngine und Puppet ist Ansible ein Frischling – das erste Release erschien 2012. Die Software wurde von Beginn an konsequent von der Firma Ansible zusammen mit der Community als Open Source entwickelt und machte es zu einem der aktivsten Projekte auf GitHub. Red Hat hat die Firma Ansible 2015 übernommen und treibt die Entwicklung mit voran.

„Simple – Powerful – Agentless“ lautet der Ansible-Slogan. Der Lernfortschritt soll im täglichen Einsatz kontinuierlich sein. Die Beschreibung der durchzuführenden Schritte soll verständlich ausfallen und

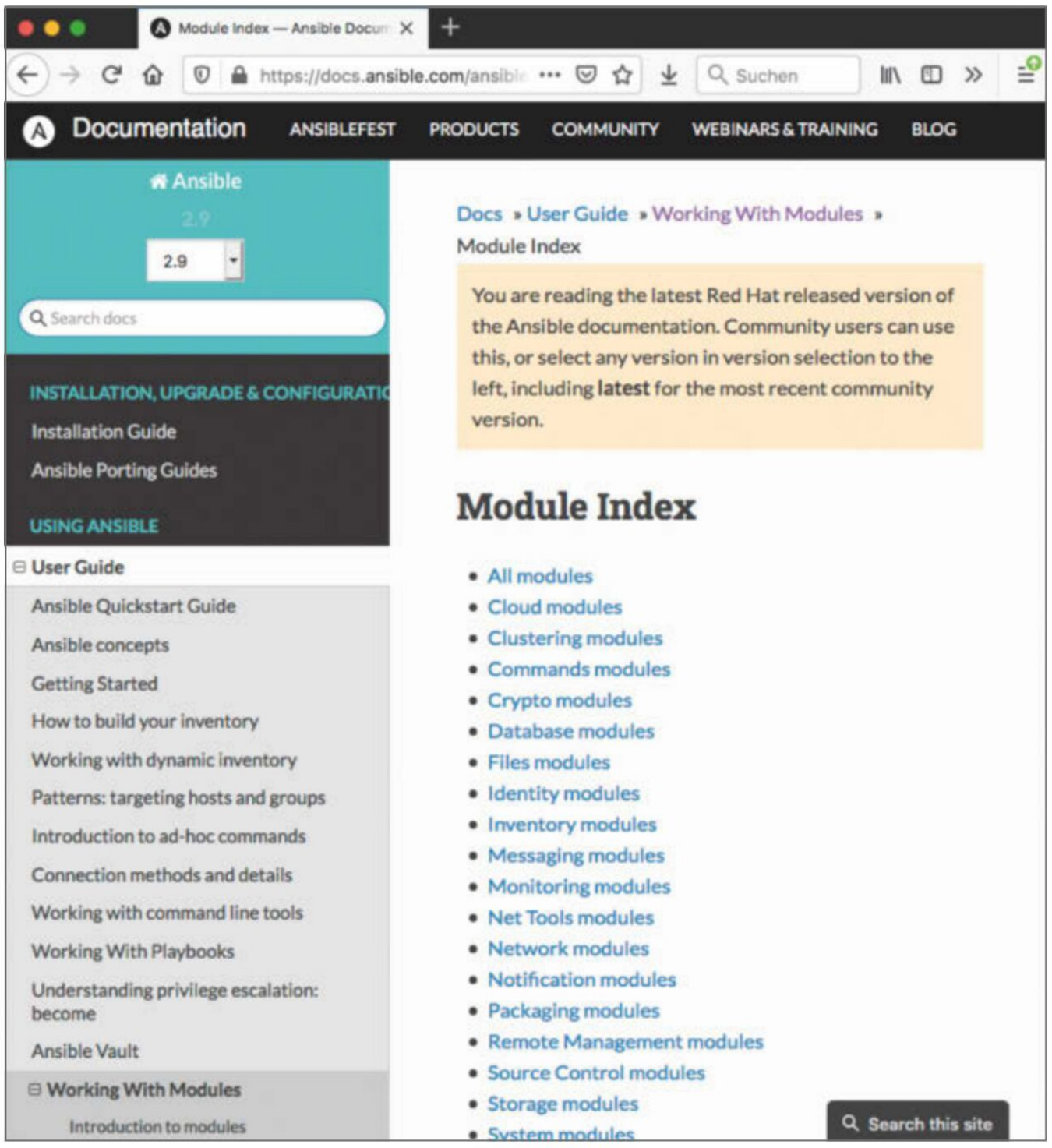
nicht komplexem Programmcode ähneln. Dazu bedarf es einer durchdachten Architektur. Ansible verfolgt einen konsequent modularen Ansatz, indem es die eigentlichen Aktivitäten Modulen überlässt. Die stammen von großen Hardware- und Softwareherstellern ebenso wie von individuellen Entwicklern. Die wirkliche Stärke entwickelt sich dann im Zusammenspiel der verschiedenen Module.

Im Unterschied zu den meisten anderen Lösungen am Markt nutzt Ansible für die Kommunikation bereits vorhandene Wege. Für die Kontaktaufnahme mit unixoiden Betriebssystemen spannt es SSH ein, unter Windows greift es auf das Windows Remote Management (WinRM) zurück. Der Vorteil liegt auf der Hand: Die Installation und Konfiguration eigener Agents entfällt. Ansible verwendet Dienste, die Standard in der jeweiligen Welt sind, täglich millionenfach genutzt werden und mit dem Betriebssystem aktuell gehalten werden.

Inventory

Für erste Gehversuche genügt jedes gängige Linux. Indem Sie darauf per Paketmanager Ansible samt Modulen installieren, wird es zu Ihrem Control-Host. Ansible braucht also keine dedizierte Maschine. Legen Sie auf dem Control-Host ein beliebiges Verzeichnis für die im Folgenden erstellten Dateien an. Für den produktiven Einsatz sollten Sie die Dateien einer Versionsverwaltung wie Git anvertrauen.

Die Grundlage für alle weiteren Schritte bildet dann das „Inventory“, ein Verzeichnis der Ziel-Hosts, auf denen Sie automatisierte Aufgaben („Tasks“



Steter Begleiter beim Automatisieren mit Ansible: Die Dokumentation zu den Modulen enthält viele Beispiele.

genannt) durchführen möchten. In der einfachsten Form ist ein Inventory eine einfache Textdatei im INI-Stil:

```
mail.example.com

[web]
web1.example.com
web2.example.com

[dbserver]
db1.example.com
db2.example.com
```

Das Beispiel beschreibt fünf Ziel-Hosts. Es ordnet jeweils zwei mit den eckigen Klammern einer Gruppe zu. Ein Inventory kann deutlich mehr, etwa Untergruppen bilden, Variablen definieren und unterschiedliche Zugangsmethoden und Parameter vorgeben.

Ein Inventory muss nicht statisch sein, sondern kann auch mit Skripten generiert werden; das bietet sich in dynamischen Umgebungen wie Virtualisierungsplattformen oder Cloud-Umgebungen an. Aber für den Einstieg reicht obiges Beispiel. Die Ziel-Hosts müssen in diesem Fall auf IP-Adressen auflösen. Ein Inventory kann aber auch „sprechende“ Namen (im Folgenden prod-db) verwenden, die dann auf IP-Adressen oder DNS-Namen verweisen:

```
prod-db ansible_host=1.2.3.4
```

Linux-Distributionen richten das Gerüst für eine Ansible-Vorkonfiguration in /etc/ansible ein, etwa /etc/ansible/ansible.cfg. Es ist aber gängige und empfohlene Praxis, benutzerspezifische Konfigurationseinstellungen zu nutzen, zum Beispiel mit einer Datei namens .ansible.cfg im Home-Verzeichnis des aufrufenden Nutzers oder ansible.cfg im aktuellen Verzeichnis.

Wer, wann, was

Sie müssen Ansible mitteilen, auf welchen Ziel-Hosts es bestimmte Tasks ausführen soll. Sehr selten werden das alle eines Inventory sein. Um die Hosts einzuschränken, können Sie neben dem Namen der Inventory-Datei ein „pattern“ übergeben.

Um herauszufinden, welche Hosts eine Aktion überhaupt betreffen würde, kann man Ansible befragen. Der folgende Befehl gibt alle Hosts des Inventory aus, das Pattern ist „all“, die Option `--list-hosts` sorgt für eine Ausgabe der passenden Hosts ohne weitere Aktionen:

```
ansible all -i inventory --list-hosts
```

Nach dem gleichen Muster kann man das Inventory auf die Hosts einer Gruppe beschränken:

```
ansible web -i inventory --list-hosts
```

Hiermit erhält man, wenig überraschend, nur die Hosts der angegebenen Gruppe. Die Filter kann man weiter verfeinern und zum Beispiel Wildcards verwenden:

```
ansible 'd*' -i inventory --list-hosts
```

Die Anführungszeichen sind nötig, um zu verhindern, dass die Shell die Wildcard interpretiert.

Ansible benutzt für die Kommunikation mit den Ziel-Hosts einen vorhandenen Mechanismus (connection method). Dazu müssen Sie vor dem ersten Kontakt einige Vorbereitungen treffen. Für SSH sollten Sie einen Login mit SSH-Key-Authentication ohne Passwortabfrage für einen unprivilegierten Benutzer auf den Ziel-Hosts einrichten, den nur Ansible nutzt.

In .ansible.cfg können Sie im Abschnitt `[defaults]` den Namen (`remote_user`) und privaten Schüssel (`private_key_file`) dieses Ansible-Nutzers eintragen. Für typische Admin-Aufgaben muss dieser Benutzer auf den Ziel-Hosts bei Bedarf Root-Rechte erlangen können, auf Unix-Systemen nutzt man dazu den sudo-Mechanismus. Es sollte klar sein, dass Sie die SSH-Schlüssel für diesen Benutzer gut schützen müssen!

Kommandos

Funktioniert das Inventory wie gewünscht und sind die Hosts vorbereitet, kann man die nächsten Schritte zur Automatisierung unternehmen. Ansible kennt zwei Betriebsmodi: Ad-hoc-Kommandos, die direkt ein Modul samt Parametern aufrufen, und Playbooks, die mehr-schrittige Automatisierungsanweisungen enthalten.

Ein Ad-hoc-Kommando könnte zum Beispiel die Kernel-Versionen aller Datenbankserver ermitteln. Dafür bietet sich das Modul `command` an, dem man den eigentlichen auf dem Host auszuführenden Befehl als Parameter mitgeben muss: `-m` übergibt das Modul, `-a` die Modul-Parameter und das optio-

nale `-o` sorgt für die Ausgabe auf einer Zeile pro Host:

```
ansible dbserver -i inventory \
-m command -a "uname -r" -o
```

Ad-hoc-Kommandos sind nützlich, um schnell etwas über Hosts in Erfahrung zu bringen, oder ein Modul auszuprobieren.

Einer der Gründe für den Erfolg von Ansible ist das konsequent modulare Konzept. Es gibt Module für so ziemlich jede IT-Komponente, allen gemeinsam ist aber die sogenannte Idempotenz: Wird ein Modul ein- oder mehrfach ausgeführt, wird es immer ein definiertes Ergebnis zurückliefern. So meldet ein Modul zur Installation eines RPM-Paketes als Ergebnis nur „changed“ oder „ok“ zurück, aber keinen Fehler, egal, ob es das Paket installieren musste oder ob es schon vorhanden war. Es zählt das definierte Ergebnis („Paket ist installiert“). Der Ziel-Host wird immer in den beschriebenen Zustand versetzt.

Durch die Anzahl und Vielfalt der Module ist es anfangs manchmal schwer, den Überblick zu behalten. Die erste Anlaufstelle sollte immer der Modul-Index der Ansible Dokumentation sein. Mit Einführung der Version 2.10 ist der Index in sogenannte Collections aufgeteilt worden (Hinweise auf die Dokumentation im Web via ct.de/wp22). Das mit Ansible installierte Kommando `ansible-doc` stellt die Dokumentation auch lokal zur Verfügung. Sie unterscheidet Module und Plug-ins, letztere wirken auf Funktionen des Ansible-Kerns.

Mal eben

Ad-hoc-Kommandos sind für den Einstieg gut geeignet, aber seine volle Leistungsfähigkeit entfaltet Ansible erst beim Einsatz von Playbooks. Das sind in YAML geschriebene, klar strukturierte und verständliche Tätigkeitsanweisungen. Das folgende Playbook installiert Apache in der aktuellsten Version auf allen Hosts der „web“-Gruppe:

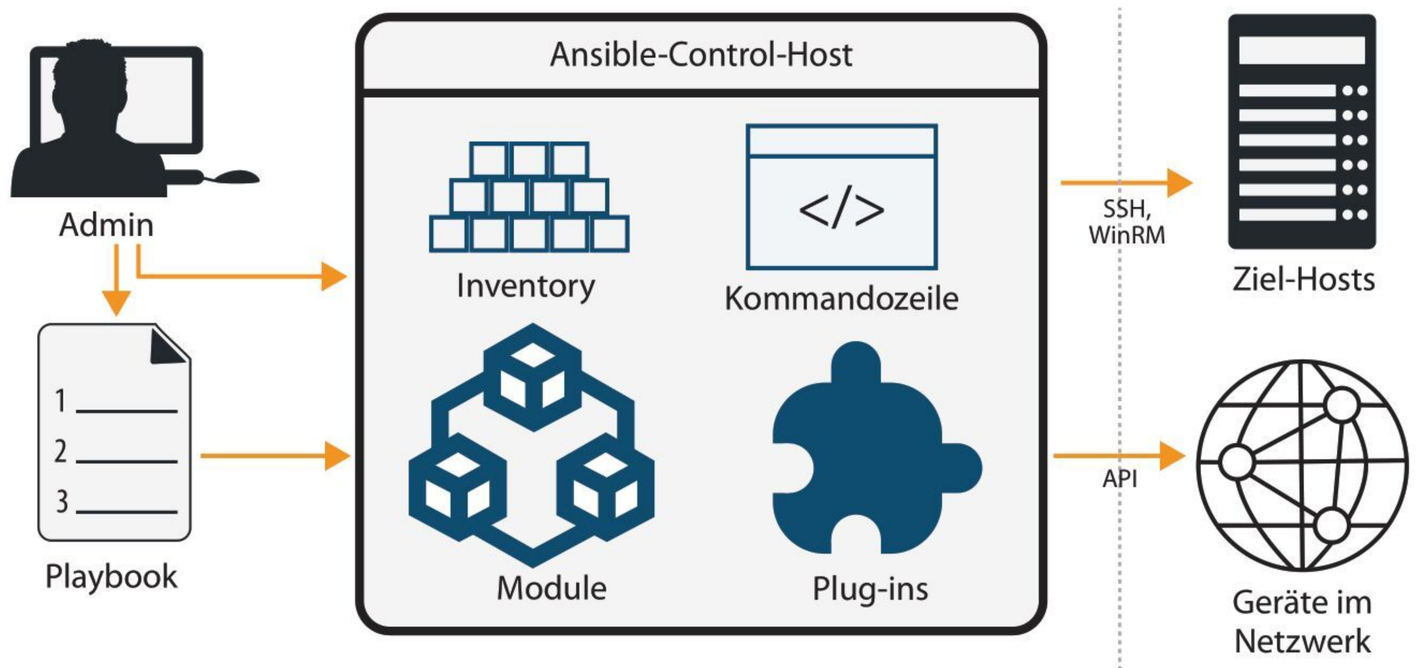
```
---
- name: Install Apache Webserver
  hosts: web
  become: yes

  tasks:
  - name: Install Latest Version
    yum:
      name: httpd
      state: latest
```

Die drei `---` kennzeichnen den Beginn einer YAML-Datei. Danach beginnt, was im Ansible-Slang ein „Play“

Was bei Ansible zusammenwirkt

Ansible hilft, IT-Ausrüstung reproduzierbar zu konfigurieren. Ein Admin notiert die nötigen Arbeitsanweisungen in einem Playbook. Die dort abgelegten Aufgaben (Tasks) rufen Module und Plug-ins auf.



ist: das Zuweisen von Tasks zu einer Auswahl von Hosts. Ein Playbook kann mehrere Plays enthalten. Im vorangehenden Beispiel wählt `hosts:` die Hosts aus dem Inventory aus und weist Ansible mit `become:` an, die folgenden Tasks mit Root-Rechten auszuführen.

Danach folgt eine Liste mit einem oder mehreren Tasks. Ein Task besteht immer aus einem Modulaufruf und den entsprechenden Parametern, Tasks werden sequenziell abgearbeitet. Die `name:` Zeilen sind übrigens optional, aber für den Überblick dringend zu empfehlen – Ansible gibt sie beim Ausführen aus.

Mit grundlegendem Linux-Admin-Wissen ergibt sich dafür der folgende Ablauf: Das Modul `yum` bekommt einen Paketnamen (`httpd`) und die Paketversion übergeben (hier `latest`) und stellt sicher, dass Apache in der aktuellsten Version, die im Repository liegt, installiert wird (oder ist).

Möchte man jetzt den Apache-Service auch aktiviert und gestartet haben, genügt es, dem Playbook einen weiteren Task hinzuzufügen:

```
- name: Enable and Start Apache
  service:
    name: httpd
    enabled: true
    state: started
```

Er ruft das Modul `service` auf. Die Parameter stellen sicher, dass Apache läuft, und aktivieren den automatischen Start. Beide Module verstehen weitere Parameter, das hier gezeigte ist ein Minimal-Task. Was ein Modul leistet, lässt sich in der Web-Dokumentation oder lokal mit `ansible-doc -t module <module name>` in Erfahrung bringen (`-t module` kann meist entfallen).

Strategie

Zum Ausführen von Playbooks bringt Ansible das Kommando `ansible-playbook` mit. Im Gegensatz zum bereits vorgestellten `ansible` benötigt es kein Host-Pattern oder Modul, weil die Definitionen komplett im Playbook hinterlegt sind. Für das obige Beispiel genügt

```
ansible-playbook -i inventory install_apache.yml
```

um das Playbook auszuführen. Zur Erinnerung: Ansible verbindet sich standardmäßig unter der Identi-

```
PLAY [Install and Enable Apache Webserver]************

TASK [Gathering Facts] *******************
ok: [host1.example.com]
ok: [host2.example.com]

TASK [Install Apache Latest Version] *****************
changed: [host2.example.com]
changed: [host1.example.com]

TASK [Enable and Start Apache] ******************
changed: [host2.example.com]
changed: [host1.example.com]

PLAY RECAP ***********************
host1.example.com          : ok=3    changed=2    unreachable=0    failed=0
host2.example.com          : ok=3    changed=2    unreachable=0    failed=0
```

Beim Ausführen eines Playbook gibt Ansible Schritt für Schritt aus, was es tut. Die „name:“-Parameter im Playbook wie „Install Apache Latest Version“ liefern Orientierung.

Ansible-Architektur

Unter der Haube führt Ansible im Standardmodus nicht etwa einzelne Kommandos über SSH aus, sondern kopiert ausführbare, kleine Programme (im Prinzip die Module) auf die Ziel-Hosts, führt sie dort aus und entfernt sie danach wieder. Es ist deshalb auch nicht unbedingt ersichtlich, dass Ansible einen Host verwaltet. Sinnvolle Praxis ist es aber, von Ansible bearbeitete Dateien automatisch mit Kommentaren zu kennzeichnen (natürlich automatisiert). Diese Arbeitsweise gelingt in erster Linie auf Unix-artigen Systemen. Soll Ansible Netzwerkgeräte, Virtualisierungsplattformen oder Cloud-Umgebungen konfigurieren, eignet sich diese Methode nicht. Der aktuelle Weg ist die Nutzung von Httpapi-Plug-ins, um die Geräte über APIs anzusprechen.

tät des ausführenden Nutzers mit den Ziel-Hosts. Ein anderer Nutzer lässt sich zum Beispiel im Playbook mit dem Parameter `remote_user` konfigurieren.

Den Ausgaben, die der Aufruf erzeugt (siehe Listing auf S. 30) kann man entnehmen, dass Ansible auf beiden Hosts drei Tasks ausgeführt hat. Zwei ergeben sich aus dem Playbook und haben etwas verändert („changed“): Einer hat Apache installiert und der andere den Dienst gestartet. Den in der Ausgabe erstgenannten Task („Gathering Facts“) ergänzt Ansible von sich aus. Er führt das Modul „setup“ aus, das Informationen über die Hosts als sogenannte „Facts“ sammelt und in den Playbooks verfügbar macht. Dazu gleich mehr. Am Ende gibt Ansible als „Play Recap“ eine Zusammenfassung aus.

Um Ansible effektiv und flexibel nutzen zu können, reichen statische Playbooks nicht aus. Ziel-Hosts unterscheiden sich: Sie sollen etwa in Abhängigkeit vom Standort unterschiedliche NTP-Server bekommen oder die Konfiguration eines Hosts beeinflusst die Parameter eines anderen. Hierfür bietet Ansible ein durchdachtes System für die Nutzung von Variablen an. Variablen in Playbooks kennzeichnet Ansible mit doppelten geschweiften Klammern, so steht zum Beispiel `{{ variablen_name }}` für den Inhalt der Variable.

Variableninhalte lernt Ansible an verschiedenen Stellen, so zum Beispiel auf der Kommandozeile, in einem Playbook, in der Inventory-Datei oder in speziellen Variablendateien. Zwar sind die Rangfolge der Definition und der Scope genau dokumentiert, trotzdem ist es ein typisches Problem, dass man sich nach einigem Einarbeiten und etwas komplexeren Playbooks fragt: Wo hat Ansible denn jetzt diesen Wert her? Klare Vorgehensweisen, wo Variablen definiert werden, sind dringend zu empfehlen – gerade zu Beginn. Als Best Practice gilt die Nutzung von Host- und Group-Variablen.

Variablen

Dazu legt man auf der gleichen Ebene mit dem Inventory-File die Verzeichnisse host_vars/ und group_vars/ an. Die Dateien in YAML-Syntax definieren in group_vars/ jetzt Variablen für Inventory-Gruppen und in host_vars/ für Hosts. Variablen, die für alle Hosts gelten sollen, nimmt die Datei group_vars/all auf. Gruppenspezifische Variablen landen in Dateien, die eben nach der Gruppe benannt sind. Spezifische Definitionen überschreiben hierbei die generische Definition. Ein Beispiel: Ausgehend von dem anfangs gezeigten Inventory könnte man folgende Struktur bauen:

```
inventory
group_vars/all
          /web
          /dbserver
```

Als Beispiel sollen alle Hosts im Inventory, die keiner Gruppe zugeordnet sind, als „Message of the day“ beim Login „Generischer Server“ ausgeben, die Hosts einer Gruppe sollen sich aber als „Web Server“ oder „Database Server“ zu erkennen geben. Dazu muss Ansible den Inhalt der Datei /etc/motd auf den Hosts in Abhängigkeit von der Gruppenzugehörigkeit anpassen. Dafür definiert man eine Variable, zum Beispiel „type“. Die drei Dateien sehen dann folgendermaßen aus, in group_vars/all steht:

```
---
type: "Generischer Server"
```

In group_vars/web:

```
---
type: "Web Server"
```

Und in group_vars/dbserver:

```
---
type: "Database Server"
```

Ein Playbook, das diese Variable nutzt, um /etc/motd zu erstellen, könnte folgendermaßen aussehen:

```
---
- name: Create motd file
  hosts: all
  become: yes

  tasks:
  - name: Write file
    copy:
      content: Dies ist ein {{ type }}
      dest: /etc/motd
```

Beim Ausführen dieses Playbook liest Ansible die Dateien aus group_vars/ und host_vars/ automatisch ein. Zuerst setzt Ansible dabei die Variable `type` aus der Datei group_hosts/all. Steckt der Ziel-Host, den Ansible gerade verarbeitet, aber in einer Inventory-Gruppe, nutzt es die Definition der Variablen aus der gruppenspezifischen Datei. Das Modul `copy` aus dem Beispiel-Playbook stellt dann sicher, dass eine Datei /etc/motd mit dem Inhalt „Dies ist ein {{ type }}“ existiert.

Soll die Variable `type` jetzt noch für einzelne Ziel-Hosts anders belegt werden, legt man in host_vars/ eine Variablen-Definition mit dem Dateinamen des Hosts an. Mit diesem Variablensystem lassen sich einfache wie auch komplexe Fälle abdecken. Es ist gleichzeitig ein Ansible-typisches Beispiel, wie ein recht einfacher und verständlicher Ansatz viele Anwendungsfälle abbilden kann.

Bedingungen

Ansible kennt weitere Konstrukte, die das Automatisieren erleichtern. Ein typisches Beispiel ist die Verwendung der bereits erwähnten „Facts“ in Tasks. Das sind Systemeigenschaften, die das „setup“-Modul zu Beginn eines Playbook-Laufs automatisch sammelt und dann als Datenstruktur in der Variable `ansible_facts` zur Verfügung stellt. Alle Facts eines Hosts kann man sich folgendermaßen ansehen:

```
ansible web1 -m setup
```

Die Ausgabe ist eine verschachtelte Datenstruktur (siehe Listing rechts unten). Der Zugriff auf einzelne Werte mag anfangs gewöhnungsbedürftig sein. Ist man zum Beispiel an der IP-Adresse interessiert, könnte man diese in einem Task folgendermaßen referenzieren:

```
- debug:
        msg: "{{ ansible_default_ipv4.address }}"
```

Das Modul `debug` hilft während der Entwicklung von Playbooks, etwa um über den Parameter `msg` Variablen-Werte auszugeben. Solche Variablen können Tasks an beliebiger Stelle verwenden.

Ansible-Facts sind nicht nur auf die Daten beschränkt, die das „setup“-Modul von sich aus sammelt. Über den Local-Facts-Mechanismus ist es möglich, eigene Facts zu definieren, die Ansible dann in die „setup“-Facts aufnimmt. Dazu legt man lokal auf einem Host im Verzeichnis /etc/ansible/facts.d/ eine Datei mit den Facts an, zum Beispiel während einer automatisierten Installation. Die liest das „setup“-Modul automatisch aus. Ansible kennt auch spezialisierte Fact-Module, zum Beispiel für eine Liste der installierten Pakete.

Um Tasks nur unter bestimmten Bedingungen auszuführen, kommen Abfragen ins Spiel. Ein `when`-Statement definiert dann, wann ein Task ausgeführt wird. Ein Task, den MariaDB nur auf einem Host in-

```
ok: [webserver1] => {
    "ansible_facts": {
        [...]
        "ansible_default_ipv4": {
           "address": "192.168.1.2",
        [...]
```

Ansible sammelt automatisch Daten über einen Ziel-Host, die man als Variablen in Playbooks einsetzen kann. Das „setup“-Modul liest sie in einem Schwung aus – die Ausgabe hier ist stark gekürzt.

stalliert, der mehr als 3000 MByte RAM hat, kann etwa wie folgt aussehen:

```
- name: MariaDB on host > 3000MB RAM
    yum:
      name: mariadb-server
      state: latest
    when: ansible_memtotal_mb > 3000
```

Das `when`-Statement verknüpft den Fact `ansible_memtotal_mb` als Variable zusammen mit dem Vergleichsoperator `>` und einem Wert. Ansible verwendet die Python-Templating-Bibliothek „Jinja2“, aus der auch die Ausdrücke für „when“ stammen. Aber `when` erlaubt mehr als nur simple Vergleiche: Zum Beispiel kann man einen Task abhängig davon machen, ob ein anderer Task erfolgreich war oder nicht. Es lohnt sich ein Blick in die Ansible-Dokumentation.

Schleifen

Schleifen vervollständigen die Liste der grundlegenden Ansible-Konstrukte, mit denen man sich früh vertraut machen sollte. Sie helfen, Tasks mehrfach mit unterschiedlichen Parametern zu wiederholen. Das kann eine Liste mit anzulegenden Nutzerkonten sein oder Software-Pakete, die man installieren möchte. Das folgende Beispiel überprüft, ob die angegebenen Dienste laufen:

```
tasks:
  - name: Check if service started
    service:
      name: "{{ item }}"
      state: started
    loop:
      - httpd
      - sshd
```

Das Modul `service` bekommt hier anstelle eines Service-Namens die Variable `{{ item }}`. Die Spezialvariable belegt Ansible in einem `loop`-Statement der Reihe nach mit den Elementen der folgenden YAML-Liste. So läuft das `service`-Modul dann für jedes Listenelement ein Mal. Wichtig ist die korrekte Einrückung des `loop`-Blocks auf die Ebene des Moduls. In vielen Ansible-Skripten findet sich auch noch der Vorläufer namens `with`, den `loop` aber ersetzen soll.

Ausgerüstet mit den bisher vorgestellten Möglichkeiten kann man daran gehen, die ersten IT-Aufgaben mit Ansible zu automatisieren. Mit wachsender Erfahrung steigen meist auch die Anforderungen, dann hilft die ausführliche Dokumentation und die rege Community von Ansible, für fast alle Herausforderungen findet sich ein Lösungsansatz.

Ab in die Cloud

Als Beispiel für ein komplexeres Szenario bietet sich das Bändigen von Cloud-Ressourcen an. Auch hier steht am Anfang der Blick in den Ansible-Modul-Index, zum Beispiel für die Amazon Web Services. Ansible kennt eine Vielzahl von Cloud-Providern – die Dokumentation widmet den Details unter „Public Cloud“ ein ganzes Kapitel. Amazon dient hier nur als Beispiel.

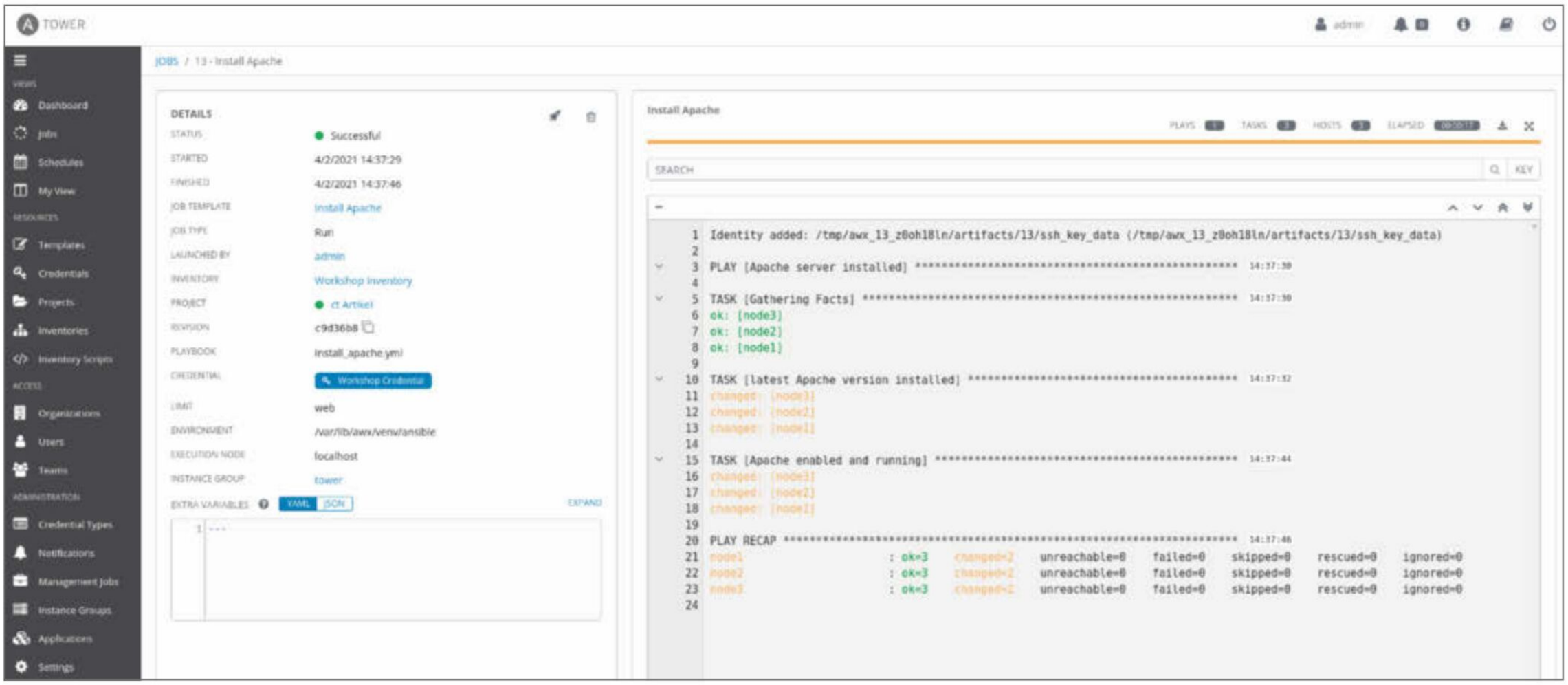
Der klassische Einstieg ist das Verwalten von einer oder mehreren EC2-Instances, Ansible-seitig übernimmt das Modul `ec2` diese Aufgabe. Die Ansible AWS-Module setzen eine Installation der Python-Bibliothek Boto voraus; solche Abhängigkeiten schlüsselt die Ansible-Dokumentation zum Modul auf. Außerdem benötigen sie einen AWS-Account sowie ein Zugriffsschlüsselpaar (access und secret key) für die Authentifizierung.

In der einfachsten Form startet der Task drei Instanzen vom Typ t2.micro von einem Amazon Machine Image (AMI) und versieht sie mit einer öffentlichen IP-Adresse:

```
- ec2:
    key_name: sshkey
    instance_type: t2.micro
    image: ami-123456
    wait: yes
    count: 3
    vpc_subnet_id: subnet-29e63245
    assign_public_ip: yes
```

Zur Übergabe der Zugriffsschlüssel verwenden alle AWS-Module entweder Umgebungsvariablen oder eine Variablendatei. Mit der Ausführung eines entsprechenden Playbooks sind die drei Instanzen schnell gestartet. Allerdings möchte man gerade in der dynamischen Cloud-Welt die Umgebung schnell verändern können, etwa um die Anzahl der Instanzen anzupassen. Führt man das Playbook aber ein zweites Mal aus, wird Ansible brav drei weitere Instanzen starten.

Diese Herausforderung ist ein gutes Beispiel dafür, dass Ansible Module meist deutlich mehr als die Basisfunktion abdecken. Ein Blick in die Dokumentation zeigt, dass ein paar Zeilen der Task genau dieses Verhalten beibringen können:



Ausgehend vom Open-Source-Projekt AWX liefert Red Hat seinen „Ansible Tower“ als Weboberfläche, um die Ansible-Engine unter anderem um Aufgabenplanung, Zugangsschutz und zentrales Logging zu ergänzen.

```
- ec2:
    key_name: sshkey
    instance_type: t2.micro
    image: ami-123456
    wait: yes
    count: 3
    vpc_subnet_id: subnet-29e63245
    assign_public_ip: yes
    instance_tags:
        Name: webserver
        type: nginx
  exact_count: 5
  count_tag:
        Name: webserver
        type: nginx
```

Den Instanzen verpasst `instance_tags` jeweils ein AWS-Tag `Name` mit dem Wert `webserver` und ein Tag `type` mit dem Wert `nginx`. Die beiden Tags lassen sich dann als `count_tag` einspannen. Mit `exact_count` legt die Task die absolute Anzahl der Instanzen fest. Führt Ansible dieses Playbook aus, bringt es die Anzahl der laufenden Instanzen mit dem Namen `webserver` und dem `type nginx` automatisch auf den Wert von `exact_count`.

Aller Anfang ist einfach

Es ist beeindruckend, wie schnell man sich mit Ansible nerviger IT-Alltagsjobs entledigt. Mit zunehmenden Kenntnissen und ambitionierten Zielen sollte man auf eine sinnvolle Struktur der Playbooks achten. Versionieren ist dafür obligatorisch. Ansible macht darüber hinaus mit Roles und Collections eigene Angebote für eine wartbare Umgebung. Inspiration für die eigenen Gehversuche liefern Blogs, Tutorials und Fachbücher zuhauf – Sie kennen jetzt die wesentlichen Begriffe und Techniken, um sich in dieser Welt zu orientieren.

Ansible wandelt sich kontinuierlich. Wenn Sie sich von Tasks und Playbooks anderer Nutzer aus dem Netz inspirieren lassen, sollten Sie sich in der Ansible-Dokumentation vergewissern, dass die nach den aktuellen Regeln der Kunst geschmiedet sind. Die Beispiele aus dem Artikel sollten mit allen aktuellen Versionen ab 2.9 funktionieren. Lassen Sie sich nicht von groben Versionssprüngen irritieren: Von Version 2.9 auf 2.10 haben die Entwickler das Versionierungsschema umgestellt, dabei entsprach im Kern das neu eingeführte Ansible Community 3.0 der Version 2.10. (ps) c't

Links zur Ansible-Dokumentation

ct.de/wp22



Jetzt gleich bestellen:
ct.de/angebot
+49 541/80 009 120
leserservice@heise.de

ICH KAUF MIR DIE c't NICHT. ICH ABONNIER SIE.

Ich möchte c't 3 Monate lang mit 35 % Neukunden-Rabatt testen.
Ich lese 6 Ausgaben als Heft oder digital in der App, als PDF oder direkt im Browser.

Als Willkommensgeschenk erhalte ich eine Prämie nach Wahl, z. B. einen RC-Quadrocopter.



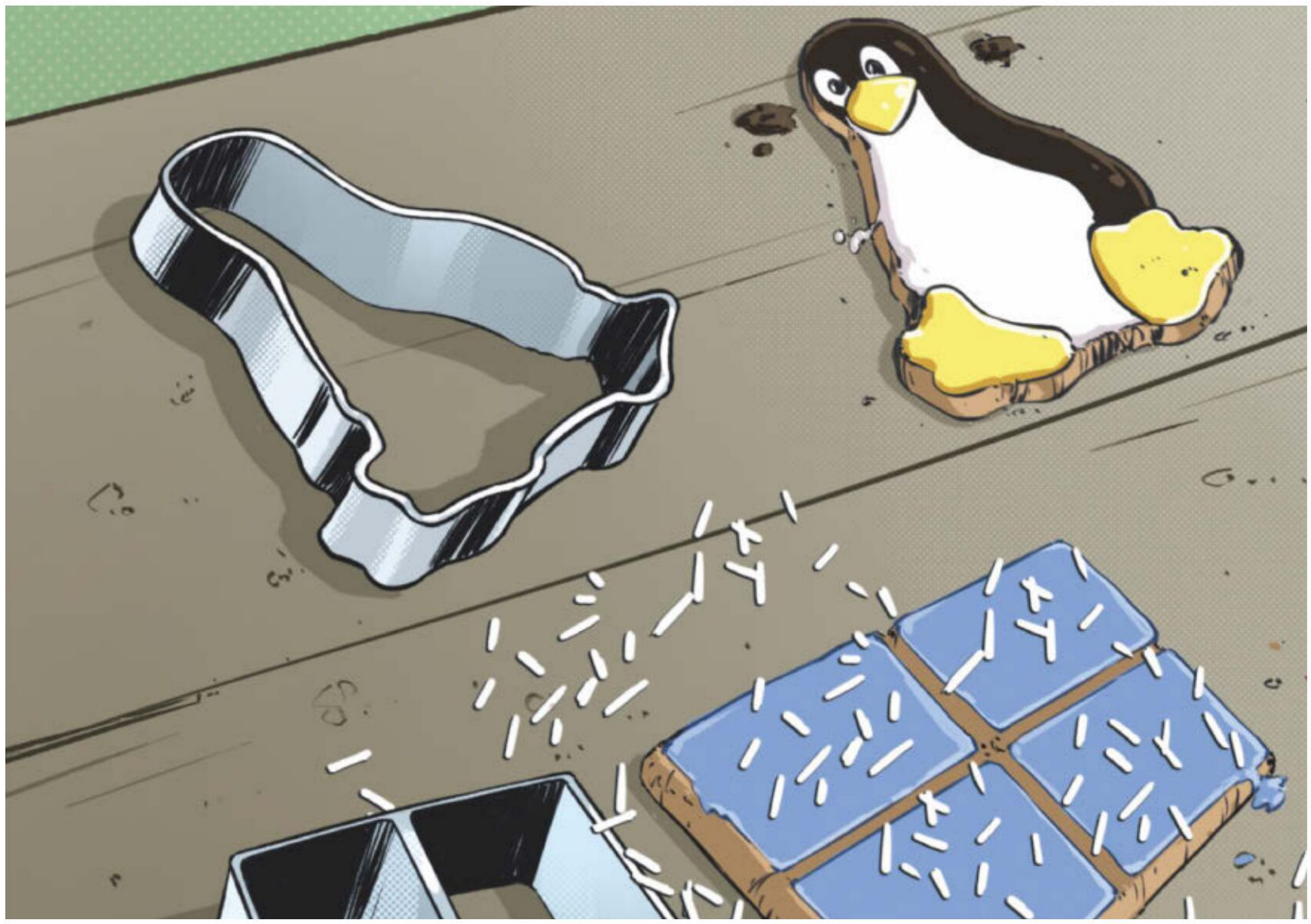


Bild: Albert Hulm

Schnellstart mit Ansible

Die Pakete für Ansible installieren, SSH-Schlüssel generieren und verteilen, sudo einrichten – alles für sich genommen ganz einfach, doch unterm Strich Aufwand. Unser Bootstrap-Skript und einige weitere Beispiele kürzen Ihren Weg zum ersten eigenen Playbook ab.

Von **Peter Siering**

Wenn nach der Lektüre des vorangehenden Artikels der Tatendrang groß ist, mit Ansible loszulegen, sind Sie hier richtig: Das Folgende liefert den Grundstock dafür, um nach dem Einrichten von Ansible auf dem Control-Host sofort einen Ziel-Host damit zu konfigurieren. Control-Host kann Ihr Arbeitsplatz-PC, ein Notebook oder eine Linux-VM sein. Auf dem Gerät müssen Sie Ansible installieren und es empfiehlt sich, eine einfache Verzeichnisstruktur anzulegen, um darin Dateien für

Ansible abzulegen; idealerweise verwalten Sie die Dateien mit Git.

SSH

Erzeugen Sie ein Verzeichnis „ansible“ und erstellen Sie darin das Verzeichnis .ssh. Wechseln Sie in das Verzeichnis und erzeugen Sie in diesem Verzeichnis ein SSH-Schlüsselpaar:

```
mkdir -p ansible/.ssh
cd ansible/.ssh
ssh-keygen -f ansible -t ecdsa -b 521
```

Dieses Schlüsselpaar in den Dateien ansible und ansible.pub soll Ansible zukünftig nutzen, um sich mit Ziel-Hosts per SSH zu verbinden. Ansible sieht keine Möglichkeit vor, eine Passphrase für den Schlüssel zu hinterlegen. Wenn Sie sich vorerst Tricksereien mit `ssh-agent` ersparen wollen, geben Sie keine Passphrase ein, wenn `ssh-keygen` danach fragt.

Für den administrativen Zugriff auf die aus dem Verzeichnis „ansible“ heraus verwalteten Ziel-Hosts empfiehlt es sich, zusätzlich ein SSH-Schlüsselpaar für den Root-Zugriff zu erzeugen:

```
ssh-keygen -f root -t ecdsa -b 521
```

Für diesen Key sollten Sie eine Passphrase vergeben. Die im späteren Verlauf vorgestellten Playbooks werden den öffentlichen Schlüssel auf den verwalteten Ziel-Hosts verteilen. Wenn Sie weitere Schlüssel für den Zugriff auf die Ziel-Hosts brauchen, etwa fürs Monitoring, können Sie die jetzt ebenfalls mit `ssh-keygen` erzeugen und in dem Verzeichnis ansible/.ssh ablegen.

Ziel-Host

Außer dem Control-Host brauchen Sie für die ersten Schritte einen Ziel-Host, um Ihre Automatisierungsschritte auf die Probe zu stellen. Das kann eine VM sein, in der Sie ein minimales Linux-System einrichten, etwa ein Debian-Netinst-Image. Wenn Sie als Ziel-Host ein produktiv genutztes System nehmen, achten Sie darauf, dass die folgenden Schritte nichts ruinieren.

Um den Ziel-Host für Ansible zugänglich zu machen, sind einige Handgriffe nötig, die für Debian ein einfaches Shell-Skript erledigen kann (siehe Listing oben rechts). Unser Skript „debansstrap.sh“ können Sie via ct.de/wmvu herunterladen. Es erstellt

```
#!/bin/bash
# enable host for ansible
set -e
PUBKEY="AAAAE2VjZh...f1B2A== ps@"
id -u ansible /dev/null 2>&1|| \
adduser ansible --disabled-password \
--gecos "" --quiet
mkdir -p /home/ansible/.ssh
echo "$PUBKEY" \
> /home/ansible/.ssh/authorized_keys
chown -R ansible /home/ansible/.ssh
apt-get update
apt-get install sudo
grep -q ansible /etc/sudoers || \
echo "ansible ALL = (ALL) \
NOPASSWD: ALL" >> /etc/sudoers
```

Ein Shellskript (debansstrap.sh) hilft Ihnen, ein frisches Debian-System mal eben für die weitere Konfiguration durch Ansible vorzubereiten.

den Nutzer `ansible` und deaktiviert seine Anmeldung per Passwort. Es legt in der nutzerspezifischen SSH-Konfiguration (in ~/.ssh/authorized_keys) auf dem Ziel-Host den öffentlichen Teil des zuvor erstellten Schlüsselpaars ab. Dadurch kann sich der Nutzer `ansible` mit dem privaten Schlüssel mit dem Host verbinden. Den Schlüssel müssen Sie in Ihr Skript übertragen (Variable `PUBKEY`, im Skript gekürzt).

Als Eigentümer für die SSH-Konfigurationsdatei setzt das Skript den Nutzer `ansible` ein. Anschließend holt es aktuelle Paketlisten, richtet `sudo` ein und erlaubt dem Nutzer `ansible` ohne Eingabe eines Passworts Root-Rechte zu erlangen – damit sind auf einem Ziel-Host alle Grundlagen geschaffen, damit sich Ansible dorthin verbinden und dort schalten und walten darf. Das Einrichten von Ansible auf dem Ziel-Host ist nicht nötig. Es genügt, einmal das Skript als Nutzer root auf dem Ziel-Host auszuführen.

Control-Host

Erstellen Sie nun auf dem Control-Host ein Inventory – als Datei inventory.ini im Verzeichnis ansible.

Für den Anfang steht da nur Ihr Ziel-Host drin. Die IP-Adresse müssen Sie anpassen, den hier vorgegebenen Namen „debtst“ können Sie frei wählen:

```
debtst ansible_host=192.168.178.23
```

Zu guter Letzt müssen Sie Ansible auf dem Control-Host vermitteln, mit welchem Nutzernamen es Kontakt zu den Ziel-Hosts aufnehmen und welchen SSH-Schlüssel es verwenden soll. Dazu legen Sie in Ihrem Home-Verzeichnis die Datei .ansible.cfg mit folgendem Inhalt an:

```
[defaults]
remote_user = ansible
private_key_file = ~/ansible/.ssh/ansible
```

Wenn Sie `remote_user` nicht setzen, würde Ihr Anmeldename auf dem Control-Host verwendet. Nunmehr sind alle Anforderungen erfüllt. Ob alles funktionstüchtig ist, finden Sie mit einem einfachen Aufruf heraus (alle Beispielaufrufe erfolgen hier aus dem Verzeichnis „ansible“ heraus):

```
ansible '*' -i inventory.ini -m ping
```

Das sollte eine Antwort aller im Inventory definierten Hosts ergeben, ansonsten stimmt mit der Einrichtung auf dem Control- oder Ziel-Host etwas noch nicht – die Ausgabe sollte wie folgt aussehen:

```
debtst | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Das ping-Modul liefert ein „pong“ zurück, der Aufruf ändert am Ziel-Host nichts und führt auch keine Konfigurationsänderung aus. Das ping-Modul stellt außerdem sicher, dass Ansible sich verbinden und Python-Code ausführen kann.

SSH-Key verteilen

Ein erstes nützliches Playbook (myroot.yml) nimmt sich die SSH-Konfiguration auf dem Ziel-Host vor. Es deaktiviert im ersten Task das Anmelden per Passwort. Damit dieser Eingriff sofort gültig wird, benachrichtigt das Playbook über einen speziellen Mechanismus einen sogenannten Handler, der den SSH-Daemon neu startet. Außerdem überspielt der zweite Task den öffentlichen Schlüssel des auf dem Ziel-Host angelegten Schlüsselpaares für root in die Datei .authorized_keys für den Nutzer root.

```
---
 - name: add ssh, deploy root key
   hosts: all
   become: yes

   tasks:
   - name: disable password authentication
     lineinfile:
       path: /etc/ssh/sshd_config
       regexp: '^PasswordAuthentication'
       line: 'PasswordAuthentication no'
       state: present
       backup: yes
       notify:
         - restart ssh

   - name: deploy key
     authorized_key:
       user: root
       key: "{{ lookup('file',
'/ansible/
.ssh/root.pub') }}"

   handlers:
     - name: restart ssh
       service:
         name: sshd
         state: restarted
```

Wenige Anweisungen (myroot.yml) ergänzen auf dem Ziel-Host für den User root einen SSH-Key und deaktivieren die Anmeldung per Passwort.

Letzteres ließe sich auch über Dateioperationen auf dem Ziel-Host bewerkstelligen, würde aber viel mehr Aufwand erfordern. Das authorized_key-Modul, das Ansible schon mitbringt, erledigt das gezielt. Generell sollte man, bevor man sich daran macht, einzelne Konfigurationsdateien (schlimmstenfalls mit regulären Ausdrücken) zu befummeln, nach einer besseren, schon fertigen Alternative im Ansible-Modul-Index suchen.

Um das Playbook auf dem Control-Host auszuführen und den Ziel-Host zu konfigurieren, genügt ein einfacher Aufruf:

```
---
 - name: set name, adjust /etc/hosts
   hosts: all
   become: yes

# order matters!
   tasks:
   - name: adjust /etc/hosts
     replace:
       path: /etc/hosts
       regexp: '(\s*){{ ansible_hostname }}(\s*)'
       replace: '\1{{ hostname }}\2'

   - name: set hostname
     hostname:
       name: '{{ hostname }}'
```

Den Hostnamen ändert unser Playbook (hostname.yml) in zwei Schritten: Es ersetzt ihn zunächst in /etc/hosts und bemüht dann den hostname-Befehl.

```
ansible-playbook -i \
inventory.ini myroot.yml
```

Die Ansible-typische Ausgabe gibt der Artikel hier nicht wieder. Wichtig ist, dass Ansible am Ende zwei Änderungen ausweist. Bei einem zweiten Aufruf sollte das Playbook keine Änderung mehr zählen, sondern stattdessen drei Tasks als „OK“ melden.

Wenn Sie Ihre Ziel-Server beispielsweise über VM-Templates anlegen, wollen Sie danach meist deren Hostnamen neu setzen. Um das für ein frisch installiertes Debian-System vollständig zu erledigen, eignet sich das Playbook „hostname.yml“. Es greift im ersten Task mangels besserer Alternativen auf das Modul `replace` zurück, das Ersetzungen mit regulären Ausdrücken in Textdateien vornimmt, um in der /etc/hosts-Datei den alten durch den neuen Namen zu ersetzen.

Den alten Hostnamen ermittelt es über Ansible-Facts, im Playbook durch `{{ ansible_hostname }}` vertreten. Den neuen Hostnamen entnimmt es der Variable `{{ hostname }}`, die Sie als host-spezifische Variable zufüttern können, indem Sie die Datei an-



```
---
 - name: set up docker & compose
   hosts: docker
   become: yes

   tasks:
   - name: update package index
     apt:
       name: "*"
       state: latest
       update_cache: yes
       force_apt_get: yes

   - name: add packages
     apt:
       name: "{{ packagesdep }}"
       update_cache: yes
       force_apt_get: yes
       state: latest
     vars:
       packagesdep:
       - apt-transport-https
       - ca-certificates
       - curl
       - gnupg-agent
       - software-properties-common

   - name: add docker signing key
     apt_key:
       url: https://download.docker.com/linux/debian/gpg
       state: present

   - name: add docker repository
     apt_repository:
      repo: deb [arch=amd64] https://download ... stable
      state: present

   - name: update package index
     apt:
       name: "*"
       state: latest
       update_cache: yes
       force_apt_get: yes

   - name: install docker
     apt:
       name: "docker-ce"
       state: latest
```

Für das Einrichten von Docker (dockersetup.yml) sind eine Reihe von Handgriffen nötig – sie zeigen die Vielfalt der Ansible-Module. Die URLs sind gekürzt.

sible/host_vars/debtst mit folgendem Inhalt anlegen:

```
---
hostname: debtst
```

Im zweiten Task nutzt das Playbook das Modul `hostname`, um den Namen des Ziel-Hosts via `hostname`-Befehl zu setzen und diesen Namen in der Datei /etc/hostname zu verewigen.

Die Reihenfolge der beiden Tasks ist bewusst gewählt und der Idempotenz geschuldet (siehe vorangehender Artikel): Wenn beim Ersetzen in /etc/hosts nämlich etwas schiefläuft, würde Ansible die Ausführung des Playbooks für den Host stoppen. Das ist wünschenswert, weil der Task den aktuellen Hostnamen via Fact referenziert. Würde das Playbook zuerst das Modul `hostname` aufrufen und das Ersetzen in /etc/hosts anschließend fehlschlagen, würde ein erneuter Aufruf schon den neuen Hostname als Ansible-Fact liefern und das Ersetzen kann nicht mehr klappen.

Ein weiteres Beispiel für diesen Artikel finden Sie auf ct.de/wmvu. Es richtet Docker auf einem Debian-Ziel-Host ein (dockersetup.yml) und zeigt die vielseitige Verwendung des `apt`-Moduls, um den Paketindex zu aktualisieren und die Pakete zu installieren – dazu verwendet es unter anderem eine Listenvariable (packagesdeb). Spezialisierte Module (`apt_key` und `apt_repository`) übernehmen das Hinzufügen von GPG-Schlüsseln für das Docker-APT-Repository und das Ergänzen des Repository.

Und nun: Viel Erfolg beim Schreiben eigener Playbooks. (ps) c't

Beispiele via GitHub

ct.de/wmvu

» Continuous Lifecycle »
[Container Conf]

Die Konferenzen für Continuous Delivery, DevOps, Containerisierung und Cloud Native

Online 17. – 18. November 2021

Frühbucher-Rabatt bis zum 26.10.2021

Highlights aus dem Vortragsprogramm

- » Sichere Container-Automatisierung: Docker versus Podman
- » Infrastructure-as-Code: Terraform & Co.
- » Kubernetes-Operator in der Praxis: Monitoring mit Prometheus
- » Die Rolle von GitOps – so gelingt es in der Praxis
- » Qualitätssicherung: Testcontainers und Integrationstests
- » Site Reliability Engineering: Wie geht das und wie wirkt es?

www.continuouslifecycle.de

+++ Tiefergehende Workshops zu Kubernetes, Docker, GitOps, Terraform und Microservices ergänzen das Konferenzprogramm. +++

Veranstalter

iX heise Developer dpunkt.verlag

Platinsponsor

snyk

Goldsponsor

Capgemini

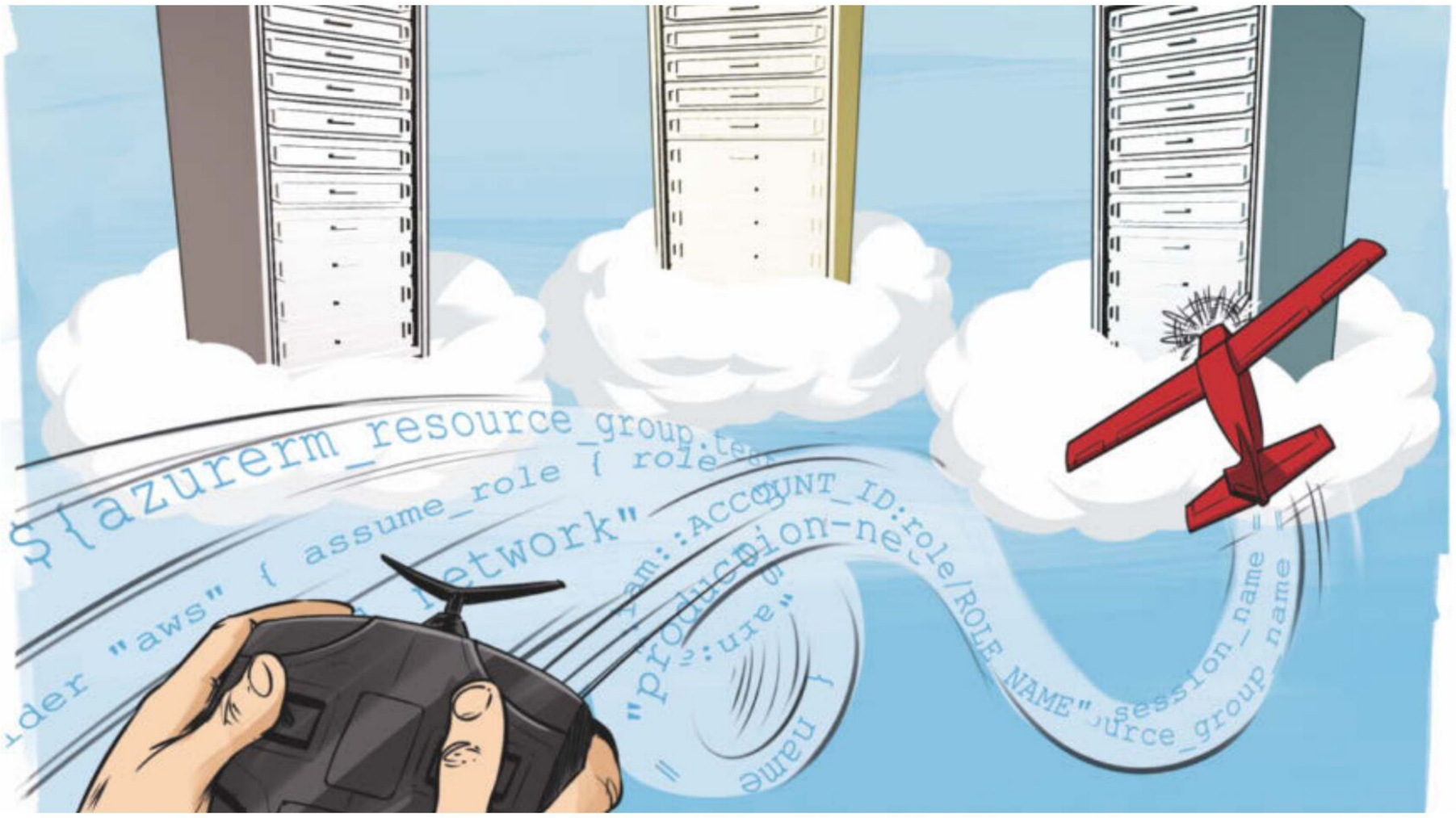


Bild: Albert Hulm

Cloud-Dienste mit Terraform

Mit Terraform definieren Admins in Textdokumenten, aus welchen Komponenten eine Cloud-Infrastruktur bestehen soll. Damit ist es leicht, Server, Speicherplatz und weitere Ressourcen aus unterschiedlichen Clouds zu kombinieren.

Von **Jan Mahn**

Vieles ist einfacher geworden für Admins und Entwickler: Code liegt in Git-Repositories, Software wird in Container verpackt, automatisch gebaut und nach Updates verteilt. Im Hintergrund laufen aber weiterhin Server, die irgendjemand einrichten muss. Auch das ist dank Cloud-Anbietern einfacher geworden – ohne Fax-Bestellformular sind die Ressourcen nach wenigen Sekunden einsatzbereit. Selbst Schlüssel zum Anmelden über SSH kann man häufig direkt auf der Webseite hinterlegen und sofort mit der Arbeit auf dem Server beginnen. Werden Server nicht mehr benötigt, kann man sie genauso schnell wieder entfernen, abgerechnet wird meist nach Stunden.

So einfach das Bestellen und Löschen auch ist, so hinderlich ist die dokumentarische Arbeit drumherum – besonders, wenn man sich nicht auf einen Anbieter festlegen will. Schließlich möchte man auch nach einigen Wochen noch wissen, bei welchem Cloud-Dienst man welche Ressourcen bestellt hat und welchen Zweck sie erfüllen.

Wer beruflich oder privat Server für mehrere Projekte bei Cloud-Anbietern verwaltet, wünscht sich eine reproduzierbare Alternative zu den Webober-

flächen der Cloud-Anbieter. Genau diese Aufgabe löst das vom Unternehmen HashiCorp entwickelte Open-Source-Werkzeug Terraform. Terraform löst ein Teil des Problems, das auch Ansible löst (siehe Seite 26), fokussiert sich aber auf das Anlegen von Servern und ähnlichen Ressourcen. Wenn es um das Einrichten von Servern geht, ist Ansible klar überlegen.

Infrastruktur als Code

Der Administrator definiert die Infrastruktur in einer Textdatei mit einer leicht zu verstehenden Syntax, Terraform spricht mit den APIs der Cloud-Anbieter und bestellt die Ressourcen. Ändert man die Definition, berechnet Terraform die Unterschiede und bestellt Ressourcen nach oder löscht nicht benötigte. Die Definition kann in einer Versionsverwaltung wie Git liegen – ein Rollback zu einer älteren Version ist dann schnell erledigt.

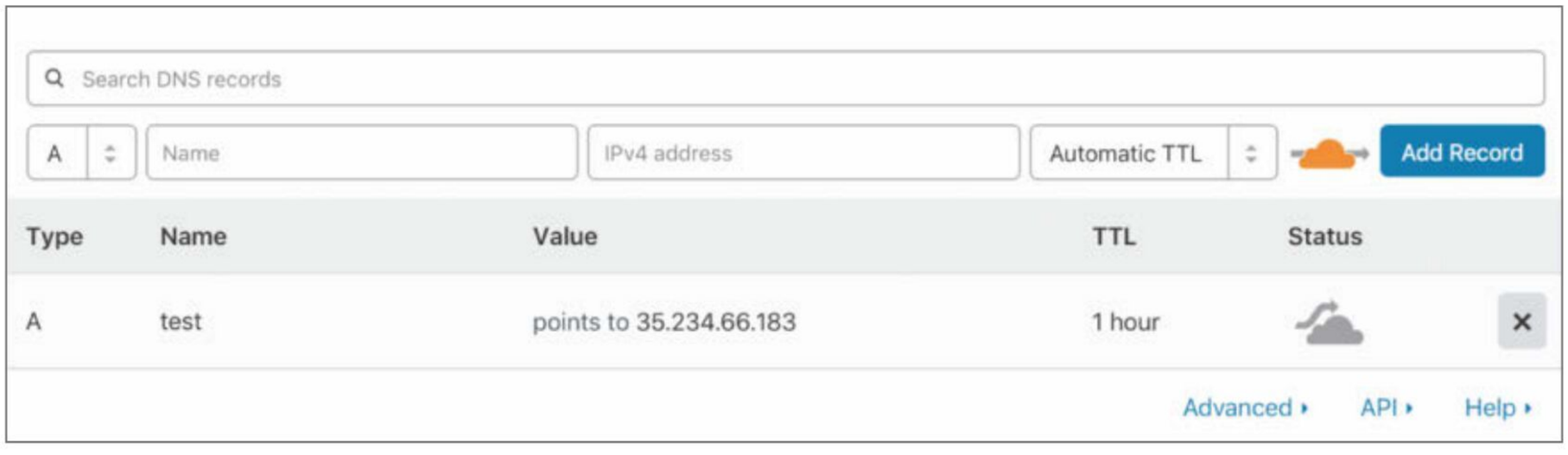
Terraform arbeitet mit sogenannten Providern. Dabei handelt es sich um Plug-ins, die zwischen API und der Terraform-Auszeichnungssprache übersetzen. Für zahlreiche Anbieter stellt Terraform solche Provider bereit. Als Beispiel soll ein einfacher Server eingerichtet werden, der per SSH über einen Hostnamen erreichbar ist. Beim Cloud-Anbieter wird dafür nur eine Ressource benötigt: eine virtuelle Maschine mit einem vorinstallierten Linux. Gleichzeitig soll ein DNS-Eintrag angelegt werden, der auf diesen Server zeigt. In diesem Artikel kommt das Angebot von Google Cloud zum Einsatz, da der kostenlose Account einen Server in kleinster Konfiguration abdeckt. Über ct.de/w7b3 finden Sie passende Code-Beispiele. Für den zugehörigen DNS-Eintrag zum Server kommt der DNS- und DDoS-Abwehrdienst Cloudflare zum Einsatz. Der kostenlose Account reicht für einfache DNS-Verwaltung aus.

Vorbereitung

Kern von Terraform ist ein Kommandozeilenwerkzeug, das der Administrator auf seiner Maschine installieren muss. Über ct.de/w7b3 finden Sie ausführbare Dateien für Windows, Linux, macOS, Solaris und BSD zum Download. Zur Installation reicht es, die Datei herunterzuladen und an den Ort zu verschieben, an dem das Betriebssystem nach ausführbaren Dateien sucht – oder den Ordner zur Systemvariable `PATH` hinzuzufügen. Unter Unix-Betriebssystemen müssen Sie die Datei noch ausführbar machen.

Öffnen Sie eine Kommandozeile und führen Sie `terraform version` aus. Wenn die Antwort eine Versionsnummer ist, kann die Arbeit beginnen. Legen Sie für das Projekt einen Ordner an und navigieren Sie dorthin. Um die erste Ressource, einen DNS-Eintrag bei Cloudflare, zu erstellen, legen Sie eine Datei dns.tf an. Bei der Namensvergabe und der Aufteilung der Dateien können Sie sich frei entfalten. Terraform berücksichtigt alle Dateien mit der Endung .tf in einem Ordner gemeinsam.

Bevor es losgehen kann, brauchen Sie einen Account bei Cloudflare, den Sie unter cloudflare.com anlegen können. Nach der Erstanmeldung müssen Sie eine Domain hinzufügen. Am Ende des Assistenten wählen Sie das kostenlose Paket aus. Wenn Sie nur testen möchten, ignorieren Sie die Anweisung, Cloudflares Nameserver bei Ihrem Domain-Registrar zu hinterlegen. Die Funktionsweise von Terraform können Sie auch so kennenlernen. In der Weboberfläche müssen Sie nun den API-Key auslesen. Dieser verbirgt sich in den Kontoeinstellungen, zu finden über das Menü oben rechts auf der Seite hinter dem Punkt „My Profile“. Scrollen Sie in den Einstellungen bis zum Ende, zeigen Sie den „Global API Key“ an und kopieren Sie ihn.



Terraform erzeugt einen DNS-Eintrag und verwendet die IP-Adresse der virtuellen Maschine.

So sei es

Mit dem Schlüssel ausgerüstet, können Sie Ihre Wunschvorstellung eines DNS-Eintrags definieren. Öffnen Sie die Datei dns.tf und beginnen Sie mit dem Anlegen des Cloudflare-Providers und einer Ressource:

```
provider "Cloudflare" {
  email = "mailadresse"
  token = "token"
}
resource "Cloudflare_record" "test" {
  domain = "example.org"
  name = "test"
  value = "10.0.0.1"
  type = "A"
  ttl = 3600
}
```

Der erste Block versorgt den Provider mit Zugangsdaten. Mailadresse und Token müssen Sie durch Ihre Daten ersetzen. Mit dem zweiten Block `resource` bestellen Sie eine Ressource vom Typ „Cloudflare_record“. Alle möglichen Parameter dieser Ressource finden Sie über ct.de/w7b3. In diesem Beispiel wird der A-Record test.example.org angelegt und mit der IP-Adresse 10.0.0.1 befüllt. 3600 Sekunden sollen Clients ihn zwischenspeichern.

Sichern Sie die Datei und wechseln Sie auf die Kommandozeile. Im ersten Schritt muss der Ordner für den Einsatz von Terraform vorbereitet werden. Das erledigt `terraform init`. Der Befehl sucht nach allen in den tf-Dateien verwendeten Providern und installiert diese. Jetzt soll Terraform berechnen, welche Ressourcen es ändern oder anlegen muss: `terraform plan`. Wenn Sie später mit zahlungspflichtigen Ressourcen operieren, sollten Sie die Ausgabe dieses Befehls gewissenhaft lesen. Terraform listet auf, was es beabsichtigt zu erstellen, zu ändern und zu löschen. In diesem Fall muss nur eine Ressource angelegt werden. Ernst wird es mit `terraform apply`. Dieser Befehl erwartet, dass Sie mit einem ausgeschriebenen `yes` zustimmen. Um das zu umgehen, wenn Sie den Vorgang später zum Beispiel in einem Skript automatisieren wollen, hängen Sie den Parameter `-auto-approve` an.

Terraform sollte mit einer Erfolgsmeldung abschließen. Werfen Sie einen Blick in die Weboberfläche von Cloudflare, sehen Sie den neuen Eintrag. Im Ordner auf Ihrem Rechner finden Sie eine neue Datei mit dem Namen terraform.tfstate. Sie enthält als JSON-Objekt den aktuellen Zustand der Infrastruktur und Terraform nutzt sie, um Unterschiede zu berechnen. Bearbeiten Sie diese niemals selbst und überlassen Sie die Verwaltung dem Programm. Ändern Sie die IP-Adresse in der Datei dns.tf und führen Sie `terraform plan` erneut aus. Das Werkzeug erkennt die Veränderung und der Aufruf von `terraform apply` setzt sie um.

Einen Server bitte

Zum Bestellen von virtuellen Maschinen müssen Sie ein paar Vorbereitungen in der Google-Cloud-Console erledigen. Öffnen Sie diese über console.cloud.google.com und melden Sie sich mit Ihren Zugangsdaten an. Alle Ressourcen werden Projekten zugeordnet, das erste Projekt mit dem Namen „test“ müssen Sie zunächst anlegen. Google erstellt im Feld darunter eine Projekt-ID mit einer angehängten Zahlenkette. Kopieren Sie diese heraus, sie wird in der Terraform-Datei benötigt.

Jetzt benötigen Sie ein Dienstkonto, das zum Anlegen von virtuellen Maschinen berechtigt ist. Klicken Sie links im Menü auf „IAM & Verwaltung“, dann auf „Dienstkonten“ und schließlich auf „Dienstkonto erstellen“. Wählen Sie einen eindeutigen Namen wie „terraformtest“. Im zweiten Schritt müssen Sie eine Rolle zuweisen. Damit der Test nicht an Zugriffsrechten scheitert, wählen Sie in der obersten Kategorie „Projekt“ die Rolle „Inhaber“, die Vollzugriff hat. Im dritten Schritt können Sie den Schlüssel herunterladen. Wählen Sie als Format „JSON“ und legen Sie die heruntergeladene Datei in den Ordner mit den Terraform-Dateien und benennen Sie sie um in „google.json“.

Bevor es an das Erstellen der Terraform-Datei geht, müssen Sie noch die „Compute Engine“ aktivieren. Klicken Sie dazu oben im Menü auf „Google Cloud Platform“ und im Menü auf „Compute Engine“. Die Aktivierung dauert etwa fünf Minuten. Öffnen Sie jetzt mit „Erstellen“ den Dialog zum Anlegen neuer VMs. Diesen Dialog können Sie als Spickzettel nutzen, um zu sehen, welche Optionen möglich sind.

Im Kasten auf der nächsten Seite finden Sie das Terraform-Rezept für eine virtuelle Maschine (Download über ct.de/w7b3). Der erste Abschnitt legt den Provider an – ersetzen Sie die Projekt-ID durch die aus der Oberfläche. Der Provider bekommt die Zugangsdaten aus der Datei google.json. Dann wird eine statische IP-Adresse bestellt. Der letzte Block bestellt den Server. Er bekommt den Namen „test-server“ und den Typ „n1-standard-1“ – eine Nano-Instanz,

```
provider "google" {
  credentials = "${file("google.json")}"
  project = "cttest"
  region = "europe-west3"
}
resource "google_compute_address" "server-static" {
  name = "test-ip-address"
}
resource "google_compute_instance" "test2" {
  name = "test-server"
  machine_type = "n1-standard-1"
  zone = "europe-west3-c"
  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-9"
    }
  }
  metadata_startup_script = "curl -fsSL https://get.docker.com -o ↵
↳get-docker.sh && sudo sh get-docker.sh"
  metadata {
    ssh-keys = "jam:${file("~/.ssh/id_rsa_CT.pub")}"
  }
  network_interface {
    network = "default"
    access_config {
      nat_ip = "${google_compute_address.server-static.address}"
    }
  }
}
```

Mittels dieses Skripts setzt Terraform eine VM-Instanz bei Google mit Debian 9 und einer statischen IP-Adresse auf.

die vom kostenlosen Kontingent ein Jahr lang abgedeckt ist.

Als Boot-Image kommt Debian 9 zum Einsatz. Es folgt ein Skript, das beim ersten Start ausgeführt werden soll. Hier kann man zum Beispiel Docker installieren. Im Block `metadata` wird ein öffentlicher SSH-Schlüssel übertragen und dem Benutzer „jam“ (Namen nach Belieben anpassen) zugewiesen. Mit diesem Schlüssel können Sie sich später beim Server per SSH anmelden.

Im letzten Schritt bekommt die VM eine NAT-Regel. Hier sehen Sie erstmals Terraforms mächtige Fähigkeit, Variablen auszutauschen. Die Auszeichnung `${name}` weist eine Variable aus, in diesem Fall `google_compute_address.server-static.address`, also den Wert `address` der Ressource `server-static`, die oben definiert wurde. Der Wert `address` ist beim Anlegen des Skripts noch nicht bekannt – Terraform bestellt eine statische IP bei Google und bekommt eine freie Adresse zurück. Diese wird dann in der Variable eingesetzt.

Zum fertigen Setup fehlt noch eine Änderung in der Datei dns.tf. Ersetzen Sie hier die Fantasie-IP aus dem ersten Test durch `"${google_compute_address.server-static.address}"`.

Alles zusammen

Immer, wenn ein neuer Provider ins Spiel gekommen ist, braucht es den Befehl `terraform init`. Mit `terraform apply` beginnt die Arbeit. Schon nach etwa zwei Minuten ist das Skript fertig und Sie sollten Ihren Server per SSH erreichen können. Bei Cloudflare sollte sich die IP-Adresse auf die öffentliche IP geändert haben.

Im Beispiel sind viele Einstellungen hart in die Terraform-Dateien eingetragen. Um die Stärken von Terraform auszuspielen, sollten Sie alle Optionen, die flexibel sein sollen, in Variablen auslagern. Anstatt die Domain einzugeben, schreiben Sie zum Beispiel:

```
resource "Cloudflare_record" "test" {
  domain = var.domain
       [...]
```

Legen Sie eine neue Datei mit dem Namen „terraform.tfvars“ an und definieren Sie hier die Werte für die Platzhalter:

```
domain = "example.org"
```

Dateien mit der Endung .tfvars liest Terraform automatisch ein und ersetzt die darin definierten Variablen. Auch die API-Keys sollten besser in einer separaten Datei gespeichert werden. Wenn Sie mit mehreren Admins zusammenarbeiten und das Terraform-Projekt per Git verwalten, schließen Sie Datei mit den Schlüsseln aus der Versionsverwaltung aus.

Um nach getaner Experimentierarbeit nicht per Hand aufräumen zu müssen, kennt Terraform einen Befehl, der im Produktivsystem verheerend sein kann: `terraform destroy`. Er löscht alle von Terraform erzeugten Dienste restlos. So können Sie sicher sein, dass nach dem Experimentieren keine Kosten bei Cloud-Anbietern mehr entstehen. (jam) c't

Download und Beispiele

ct.de/w7b3



Kubernetes für Docker-Kenner

Betreiber von großen Container-Umgebungen stoßen irgendwann an die Grenzen dessen, was mit Docker möglich ist. Der Container-Verwalter Kubernetes verspricht Abhilfe, der Einstieg ist aber auch für erfahrene Docker-Nutzer nicht ganz leicht. Mit einem Beispiel-Setup aus Raspis kann man erste Erfahrungen sammeln.

Von **Jan Mahn und Merlin Schumacher**

Docker hat die Technik der Linux-Container handlich und damit populär gemacht und ist die erste Wahl für kleine und mittelgroße Umgebungen.

Bei wirklich großen Umgebungen – etwa einer Webseite, die weltweit erreichbar sein soll, hohe Lastspitzen abfangen muss und auf vielen Servern in mehreren Clouds und auch im eigenen Rechenzentrum zugleich laufen soll, braucht es einen ausgewachsenen Container-Orchestrator wie Kubernetes. Die Software ist schnell zum Industriestandard für die Container-Verwaltung geworden und die Nachfrage nach Kubernetes-Experten in Unternehmen ist groß. Wer aus Neugier oder beruflichem Interesse den Schritt von Docker zu Kubernetes gehen will, steht vor der Herausforderung, nicht nur neue Begriffe lernen zu müssen, sondern nebenbei eine realistische Testumgebung, am besten aus mehreren Maschinen, betreiben zu müssen – oder Geld bei Cloud-Anbietern auszugeben. Mit ein paar Raspis kann man ein ausfallsicheres Kubernetes-Cluster zum Ausprobieren schnell selbst einrichten

und in Kubernetes einsteigen. Wer keine Raspis zur Hand hat, kann diese Anleitung ohne Anpassungen auch auf drei gemieteten Servern von einem Cloudprovider (siehe Seite 14) nachbauen.

Kubernetes ist Open-Source-Software, die 2015 von Google an die Cloud Native Computing Foundation übergeben wurde. Das Projekt entstand aus den Erfahrungen, die Googles Entwickler mit ihrem eigenen Container-Orchestrator Borg gemacht haben. Borg ist noch heute als Grundlage für praktisch alle von Google betriebenen Dienste im Einsatz. Der etwas sperrige Name Kubernetes kommt aus dem Griechischen und bedeutet Steuermann – um Platz zu sparen, wird er gern abgekürzt zu K8S (ein K, acht Zeichen und ein S). Kubernetes gehört mit mehr als 88000 Commits von 2400 Entwicklern und zu den größten Open-Source-Projekten.

Einige der Funktionen, die Kubernetes mitbringt, hat Docker inzwischen auch gelernt. Dennoch ist Kubernetes in diesem Bereich der Cloud-Dienste der Industriestandard. Ohne sich um das zugrundeliegende Betriebssystem Gedanken machen zu müssen, kann man bei Google, Amazon, Microsoft & Co. in den Clouds fertige Kubernetes-Nodes starten und ganz nebenbei der Kreditkarte beim Schwitzen zusehen.

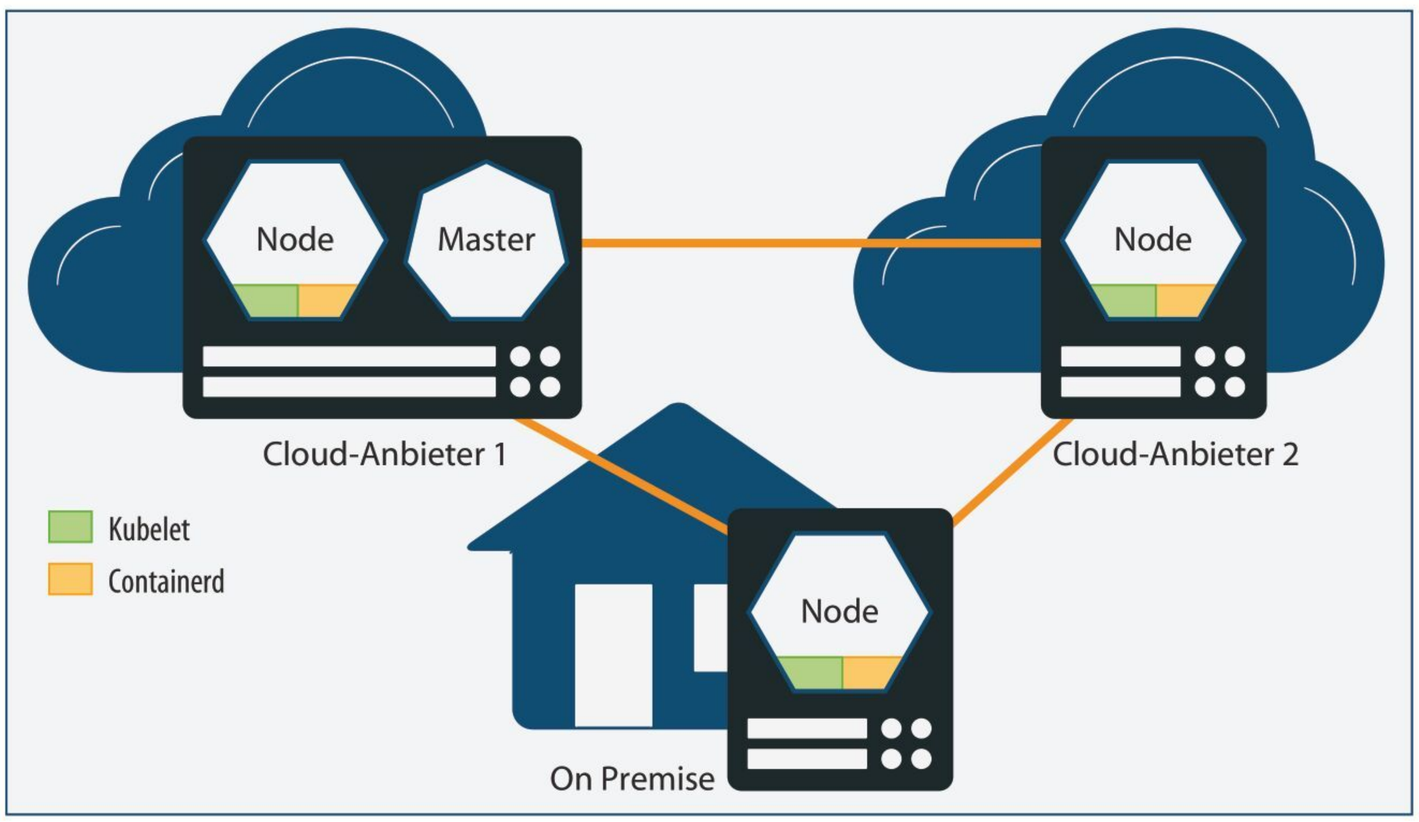
Um die Beispiele in diesem Artikel nachzuvollziehen, brauchen Sie nicht so tief in die Tasche zu greifen. Für den Betrieb eines ersten Kubernetes-Clusters reichen schon zwei Raspis. Version 2 sollte es mindestens sein. Die Anleitung funktioniert aber auch mit beliebigen x86-PCs oder virtuellen Maschinen. Solide Docker-Kenntnisse sind für das Verständnis erforderlich.

Begriffsklärung

Die Architektur und die Begriffswelt von Kubernetes ist auf den ersten Blick kaum zu durchschauen, was vor allem daran liegt, dass Dinge aus der realen Welt (Computer, Festplatten, Prozessoren) über mehrere Ebenen bis zur Unkenntlichkeit wegabstrahiert werden. Es gibt also einige Vokabeln zu lernen:

Aufbau eines Kubernetes-Clusters

Ein Kubernetes-Cluster umfasst alle Dienste und Ressourcen, die für die Bereitstellung einer oder mehrerer Anwendungen nötig sind. Innerhalb des Clusters gibt es verschiedene Nodes. Das können Server, VMs oder auch Container sein, in denen eine Container-Runtime wie Containerd und ein Verwaltungsdienst namens Kubelet läuft, über den der Node verwaltet wird. Die Verwaltung übernimmt der Master.



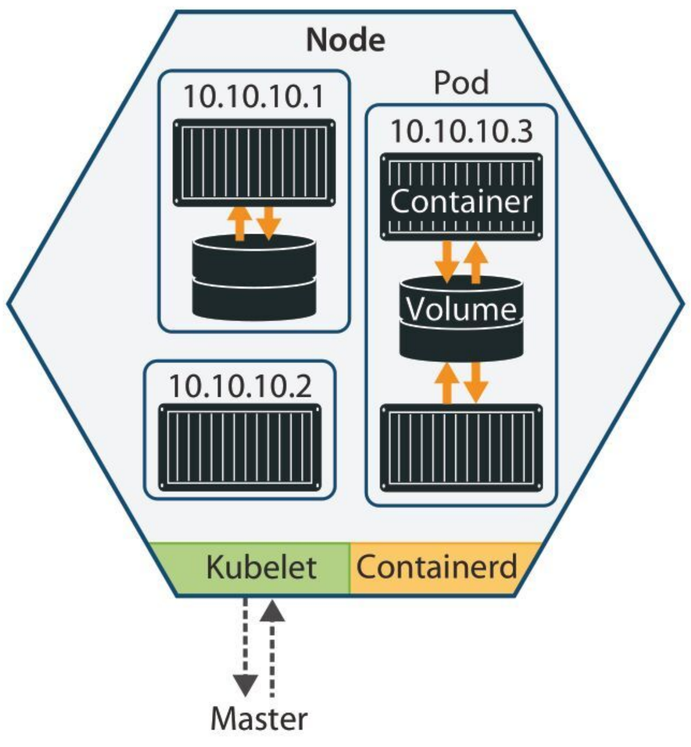
Die gesamte Einheit von Diensten, Containern und Servern wird als **Cluster** bezeichnet. Die Struktur eines Clusters sehen Sie in der Infografik auf Seite 47.

Die kleinste Einheit ist der **Pod**. Er fasst mehrere **Container** und zugehörige Ressourcen (zum Beispiel persistenten Speicher) zusammen und stellt deren Dienste über eine gemeinsame IP-Adresse bereit. Ein Pod kann beliebig oft kopiert und verschoben, allerdings nie aufgeteilt werden.

Wie die Pods ausgestaltet sind, also welche Container darin laufen und wie diese konfiguriert sind, definiert ein **Deployment**. Das Deployment beschreibt auch, wie viele Replicas, also Kopien, eines Pods existieren sollen. Kubernetes verfolgt permanent das Ziel, diese Zahl zu erreichen: Fällt ein Pod aus, erzeugt Kubernetes automatisch einen neuen, um die Anforderungen des Deployments zu erfüllen.

Aufbau eines Kubernetes-Nodes

Innerhalb eines Kubernetes-Nodes laufen immer zwei Dinge: Der Verwaltungsdienst Kubelet, der ein API bereitstellt, über das der Master Dienste und Ressourcen verwalten kann. Dazu kommt eine Container-Runtime. Bei Kubernetes ist das üblicherweise Containerd. Innerhalb der Nodes laufen sogenannte Pods. Diese können verschiedene Ressourcen beinhalten, wie etwa Container oder Volumes.



Die verschiedenen Kubernetes-Instanzen, auf denen Pods laufen, werden als **Nodes** bezeichnet. Wo diese Nodes liegen, ist unerheblich. So kann ein Node ein physischer Server (etwa ein Raspi), eine virtuelle Maschine oder sogar selbst ein Container sein, in dem wiederum Kubernetes läuft.

Pods sind flüchtig, das heißt, dass sie jederzeit gelöscht und wieder neu erzeugt werden können. Auf welchem Node ein Pod läuft oder welche IP-Adresse er bekommt, weiß man nicht und braucht man auch nicht zu wissen – man sollte auf keinen Fall mit festen IP-Adressen hantieren.

Eine Ansammlung von identischen Pods wird zu einem **Service** zusammengefasst. Angesprochen wird also nie ein Pod direkt, sondern immer der Service. Kommt eine Anfrage von außerhalb beim Service an, leitet er diese an einen der Pods weiter. An welchen Pod Kubernetes die Anfrage leitet, weiß man nicht – das ist aber auch unerheblich, denn jeder Pod arbeitet gleich. Es gibt unterschiedliche Typen von Services, die entweder ausgeklügelte Load-Balancing-Funktionen bereitstellen oder nur der Reihe nach die Pods durchwechseln.

Die Steuerung der Nodes, Verteilung der Pods und die Verwaltung des Clusters übernimmt der **Master**. Der Master nimmt Anweisungen über ein API entgegen und wendet diese an. Wenn man zum Beispiel die Menge der Kopien eines Pods erhöht, erzeugt der Master neue, indem er den Nodes mitteilt, wie viele Pods welchen Typs sie erzeugen müssen. Der Master kontrolliert auch, ob die Anwendungen in den Containern wie geplant ihrer Arbeit nachgehen und stellt sicher, dass die Anforderungen der Deployments eingehalten werden.

Eine Besonderheit von Kubernetes ist der **Ingress-Router** für eingehenden Verkehr. Er kann HTTP(S)-Anfragen von Diensten an Services extrem flexibel verteilen. Der Administrator kann Regeln aufstellen, die dafür sorgen, dass Anfragen auf www.beispiel.de von einem anderen Service beantwortet werden, als solche an www.example.com. Auch TLS-Terminierung gehört zu den Aufgaben des Ingress-Routers. Der Webserver im Service muss sich also nicht mit Zertifikaten belasten und kann hinter dem Ingress HTTP ausliefern. Die für das Routing eingesetzte Software wird als Ingress-Controller bezeichnet. Es gibt einige Programme, die als Ingress-Controller für Kubernetes arbeiten können. In diesem Artikel kommt die leichtgewichtige Kubernetes-Distribution K3S zum Einsatz, die den Router Traefik 1.7 nutzt. Traefik ist im Container-Umfeld aufgrund seiner Flexibilität sehr populär.

```
01 ---
02 apiVersion: apps/v1
03 kind: Deployment
04 metadata:
05   name: whoami-deployment
06 spec:
07   selector:
08     matchLabels:
09       app: whoami
10   replicas: 6
11   template:
12     metadata:
13       labels:
14         app: whoami
15     spec:
16       containers:
17       - image: ctmagazin/whoami
18         name: whoami-container
19         ports:
20         - containerPort: 80
21 ---
22 apiVersion: v1
23 kind: Service
24 metadata:
25   name: whoami-port
26 spec:
27   selector:
28     app: whoami
29   type: NodePort
30   ports:
31   - nodePort: 30123
32     port: 80
33     targetPort: 80
```

whoami.yml: Eine YAML-Datei reicht, um einen einfachen, aber redundanten Dienst über Kubernetes bereitzustellen.

Auch wenn Kubernetes nach all den Begriffen komplex und unbezwingbar wirkt, muss man nicht viel dafür tun, Nodes, Services und Pods zu starten – wie auch bei Docker-Compose passiert das meiste in schön strukturierten YAML-Dateien.

K3S einrichten

Der Kubernetes-Download ist mit über 400 MByte ein ziemlicher Brocken. Entpackt werden daraus etwa 1,6 GByte. Da man auf dem Raspi mit dem Arbeitsspeicher meist haushalten muss, sind Binaries in dieser Größe zu viel.

Das K3S-Paket ist entpackt nur etwa 80 MByte groß und läuft somit auch auf dem Raspi oder kleinen Cloudservern mit wenig Arbeitsspeicher. Das Schöne an echter Hardware, also einer Handvoll Raspis ist aber, dass man wirklich mal testen kann, was mit einem Cluster passiert, wenn ein kompletter Node ausfällt, weil jemand gerade den Stecker gezogen hat.

Die Installation von K3S auf dem Raspi ist schnell erledigt. Schreiben Sie ein aktuelles Raspbian-Buster-Image (zu finden über ct.de/wzx3) auf eine SD-Karte und bringen Sie es auf den neusten Stand. Falls Sie bereits Raspis am Laufen haben, können Sie auch die bestehenden Raspbian-Installationen weiternutzen. Ob Buster oder Stretch ist unerheblich.

Laden Sie jetzt das Installations-Skript herunter und führen Sie es im selben Schritt aus:

```
curl -sfL https://get.k3s.io | sh -
```

Wenn Sie auf Nummer sicher gehen wollen, laden Sie die Datei erst herunter und kontrollieren Sie, ob in der Datei k3s.sh das korrekte Shell-Skript und kein Müll gelandet ist.

Zentrales Werkzeug bei der Arbeit mit Kubernetes ist das Programm Kubectl. Es spricht über ein API mit dem Master. Nach der Installation können Sie mit Kubectl testen, ob Kubernetes auf Ihrer Maschine läuft:

```
sudo kubectl get nodes
```

Eventuell dauert es einen Moment, bis der Master antwortet. Anschließend sollten Sie eine Ausgabe bekommen, die der folgenden ähnelt:

```
NAME STATUS ROLES AGE VERSION
k3smaster Ready master 47h v1.20
```

Der Name k3smaster ist der Hostname des Systems. Haben Sie den Ihres Raspi nicht geändert, dürfte dort raspberrypi stehen.

Öffnen Sie nun die Datei `/var/lib/rancher/k3s/server/node-token`. Sie enthält den Node-Token. Er wird benötigt, um weitere Nodes beim Master anzumelden.

Anschließend müssen Sie weitere Nodes – in diesem Fall Raspberry Pis – in das Cluster aufnehmen.

Installieren Sie Raspbian und verbinden Sie sich per SSH. Zwei Umgebungsvariablen müssen gesetzt sein, bevor das Installationsskript startet: Die IP-Adresse des Masters und das Token:

```
export K3S_URL=https://[MEINSERVER]:6443
export K3S_TOKEN=[MEINTOKEN]
curl -sfL https://get.k3s.io | sh -
```

Diesen Prozess wiederholen Sie nun für alle weiteren Nodes. Wenn Sie nun sudo kubectl get node auf dem Master aufrufen, sollten Sie alle hinzugefügten Nodes sehen können:

```
NAME STATUS ROLES AGE VERSION
```

```
k3smaster Ready master 5m v1.14.4
k3snode1 Ready worker 5m v1.14.4
```

Das Kubernetes-Cluster ist nun bereit für einen Test. Nach oben gibt es für die Anzahl der Nodes keine Grenze – zumindest keine, die Sie mit Raspis erreichen können.

Ein Beispiel bitte

Kubernetes wird wie Docker-Compose auch über YAML-Dateien gesteuert. Die Syntax ist aber wesentlich komplexer. Die von der YAML-Datei beschriebenen Dienste werden als **Objekte** bezeichnet. Im Listingkasten auf Seite 49 sehen Sie ein Beispiel für einen grundlegenden HTTP-Dienst mit allen dafür nötigen Objekten. Basis ist ein Container, der per HTTP Informationen über sich selbst ausgibt. Darunter auch seinen Hostnamen. Darüber kann man erkennen, in welchem Pod ein Aufruf gelandet ist.

Alle Beispieldateien, die in diesem Artikel auftauchen, finden Sie über ct.de/wzx3. Das spart bei den arg länglichen YAML-Blöcken viel Tipparbeit.

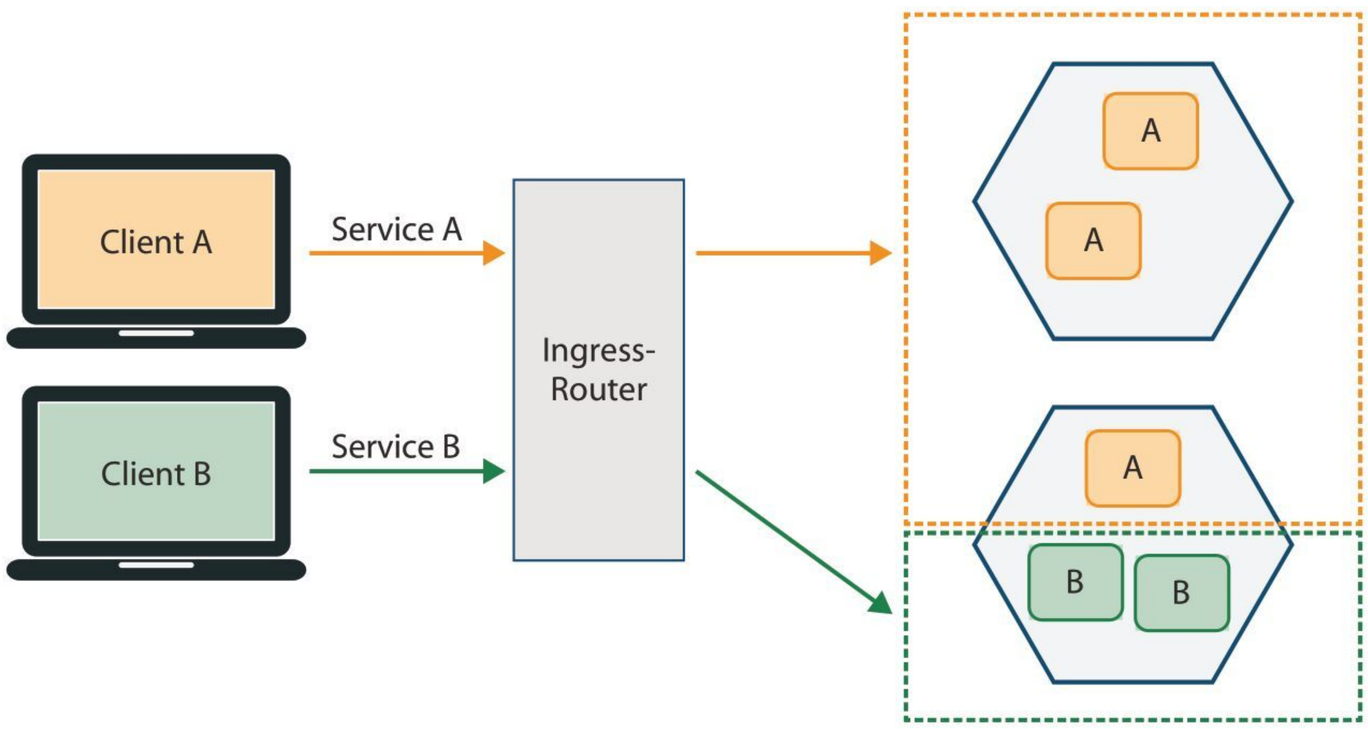
Die YAML-Datei gliedert sich in zwei Definitionen, die durch drei Minuszeichen getrennt sind. Der obere Block legt die Eigenschaften des Deployments fest. Der untere definiert den Service. Die erste Zeile beider Blöcke legt jeweils die API- Version fest, die Kubectl nutzen soll, um die Anweisungen umzusetzen.

Darauf folgt die Art (`kind`) von Element, die dieser Block definiert. Je nach Art unterscheidet sich teilweise auch die API-Version. Eine Übersicht der aktuellen API-Versionen finden Sie über ct.de/wzx3. Im anschließenden Block `metadata` legt man den Namen des bereitgestellten Dienstes fest. Das hilft bei der eindeutigen Identifikation. Mit dem `spec`-Abschnitt beginnen die ersten großen Unterschiede, denn dieser beschreibt die Eigenschaften des Objekts. Die möglichen Werte und Strukturen unterscheiden sich aber je nach Objekt. Eine vollständige Dokumentation aller möglichen `spec`-Definitionen wäre zu ausführlich für einen Artikel, daher beschreiben wir nur die Grundlagen – mehr ist für die ersten Gehversuche aber auch nicht nötig.

Beim Deployment wird zunächst der `selector` definiert. Seine Aufgabe ist es festzulegen, welche Pods (neu erzeugt oder bereits bestehend) von diesem Deployment und den damit einhergehenden Änderungen betroffen sind. Das Attribut `matchLabels` legt fest, welches Label die Pods tragen müssen,

Routing und Services

Damit Clients immer den gewünschten Dienst innerhalb eines Kubernetes-Clusters erreichen, müssen deren Anfragen geroutet werden. Dafür ist der Ingress-Router zuständig. Pods sind bei Kubernetes flüchtig, das heißt, sie können jederzeit rückstandslos zerstört werden. Pods werden in Services zusammengefasst. Kubernetes weiß, welche Pods den gewünschten Dienst bereitstellen und kann externe Anfragen durchleiten. Der Ingress-Router greift wiederum auf die Services zurück und leitet die externen Anfragen weiter.



```
01 ---
02 apiVersion: v1
03 kind: Service
04 metadata:
05   name: whoami-service
06 spec:
07   ports:
08   - targetPort: 80
09     port: 80
10   selector:
11     app: whoami
12 ---
13 apiVersion: networking.k8s.io/v1beta1
14 kind: Ingress
15 metadata:
16   name: whoami-ingress
17 spec:
18   rules:
19   - http:
20       paths:
21       -   path: /
22           backend:
23             serviceName: whoami-service
24             servicePort: 80
```

whoami-ingress.yaml: Die Konfiguration eines Ingress-Routers macht kaum Aufwand, bietet aber sehr viel Flexi bilität und automatisches Load-Balancing. Den Block für das Deployment können Sie aus dem ersten Listing übernehmen.

damit das Deployment sie mit einbezieht. Die Labels können auch dazu dienen, Objekte aufzufinden und sind in der Form „Schlüssel=Wert“ definiert. So kann man zum Beispiel mit folgendem Befehl alle Pods anzeigen lassen, die das Label `app=whoami` tragen.

```
kubectl get pods -l app=whoami
```

Mit `replicas`, gefolgt von einer Zahl definieren Sie, wie viele Pods erzeugt werden sollen. Kubernetes verteilt diese dann gleichmäßig auf die vorhandenen Nodes. Wenn Sie also wie im Beispiel sechs Pods und zwei Nodes haben, landen jeweils drei Pods auf den Nodes. Die Zahl lässt sich theoretisch beliebig skalieren – in der Praxis ist Schluss, wenn die Ressourcen der Nodes erschöpft sind.

Die nun folgende `template`-Definition legt die Ausgestaltung der Pods fest. Mit `metadata.label` bekommen die Pods ein oder mehrere Label zugewiesen. Das sind die Label, nach denen in `matchLabels` gesucht wird. Stimmen diese nicht überein, wird auch das spätere Deployment fehlschlagen, denn aus Sicht von Kubernetes gibt es keine Pods, die von diesem Deployment betroffen wären. Damit wäre das Deployment nichtig.

`spec` legt die innerhalb des Pods benötigten Ressourcen fest. Das ist in diesem Beispiel ein einzelner Container. Es könnten aber auch mehrere Container und auch Volumes enthalten sein. Der Container wird aus einem `image` erzeugt und bekommt einen Namen (`name`) zugewiesen. Der Block `ports` legt fest, welche Ports des Containers erreichbar sind.

Anschließend kommt die Service-Definition. Der hier erzeugte Service ist vom Typ `NodePort`. Das bedeutet, dass der Dienst auf jedem Node über einen festgelegten Port von außen erreichbar ist. Jeder Node stellt aber nur Ausgaben der auf ihm laufenden Pods bereit. Es findet also kein Routing oder Load-Balancing statt.

Ein NodePort-Service muss nur wissen, für welche Pods er zuständig ist. Dafür kommt der `selector` zum Einsatz. `type` bestimmt den Typ des Services. Die `ports` gliedern sich in drei Typen: `nodePort` ist der Port, über den der Service extern, also auf der IP-Adresse des Nodes erreichbar ist. `port` bestimmt, über welchen Port der Service innerhalb des Clusters erreichbar ist. Auf welchen Port die jeweiligen Pods lauschen, legt `targetPort` fest.

Um das Beispiel zu starten, geben Sie folgenden Befehl auf dem Master ein:

```
kubectl apply -f whoami.yaml
```

Kubectl schickt die Anweisungen der YAML-Datei per API-Aufruf an den Kubernetes Master, der dann die Anforderungen des Deployments umsetzt. Er weist die Nodes an, die Pods und Services zu erzeugen. Um zu sehen, ob alles umgesetzt ist, können Sie

kubectl get services und kubectl get deployments aufrufen. Laufen Deployment und Service, starten Sie Ihren Browser und öffnen Sie die URL http://<RASPI-IP>:30123. Daraufhin sollten Sie nun Informationen über den Container bekommen. Wenn Sie die Seite mit abgeschaltetem Cache (etwa über Strg+F5) neu laden, werden Sie sehen, dass der Hostname sich ändert. Der Service wechselt die antwortenden Pods einfach durch.

Neben `NodePort` gibt es noch weitere Typen von Services: Der Standard heißt `ClusterIP` und bedeutet lediglich, dass der Service nur innerhalb des Clusters erreichbar ist. Mit `LoadBalancer` erzeugt man eine Load-Balancer-Instanz, die versucht, die Last auf alle Nodes gleichmäßig zu verteilen. Das Load-Balancing kann man sowohl lokal als auch über einen externen Cloud-Provider bereitstellen. Auch hier richtet Kubernetes die nötigen Routen und notwendige Portfreigaben automatisch ein. Der letzte Typ heißt `ExternalName` und stellt den Service unter einem festlegten DNS-Namen bereit.

Aktuell antworten immer nur die Pods auf dem Node, den man per Browser angesprochen hat. Eleganter ist eine Lösung, die Load-Balancing kann – das ist das Problem, das Kubernetes lösen soll. Die eleganteste Lösung dafür ist der Ingress-Router. Er ist weit mächtiger als der Service-Typ `LoadBalancer`. Für den Einsatz des Ingress-Routers muss die Konfiguration des Services nur minimal angepasst werden. Hinzu kommt noch die Ingress-Konfiguration. Am Deployment ändert sich nichts. Eine Beispielkonfiguration für Ingress-Router und Service sehen Sie im Listingkasten auf Seite 51. Beim Service verschwindet der `type`, sodass er im ClusterIP-Modus läuft und damit nur innerhalb des Clusters erreichbar ist. Das reicht für den Ingress-Router. Die Option `nodePort` fehlt hier ebenfalls.

Die Konfiguration des Ingress-Routers ist genauso aufgebaut wie die der anderen Objekte. Lediglich die API-Version und `spec` unterscheiden sich. Bei der `spec` legt `rules` Regeln fest, nach denen der Ingress-Router entscheidet, welche Anfragen wo landen sollen. In diesem Beispiel soll der Router lediglich `http`-Anfragen für den Root-Pfad / beantworten. Für die Antwort ist dann der Service zuständig.

Wenden Sie die geänderte Konfiguration mit dem folgenden Befehl an:

```
kubectl apply -f whoami-ingress.yaml
```

Daraufhin wird Kubectl Ihnen mitteilen, dass es das Deployment nicht geändert hat, aber den Service aktualisiert und einen Ingress-Router erzeugt hat.

```
Hostname: whoami-deployment-7f444c5bc-fh7kk
IP: 127.0.0.1
IP: ::1
IP: 10.42.0.9
IP: fe80::9842:59ff:fe26:21d4
GET / HTTP/1.1
Host: 192.168.73.210
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.8,en-US;q=0.5,en;q=0.3
Cache-Control: no-cache
Cookie-Installing-Permission: required
Dnt: 1
Pragma: no-cache
Upgrade-Insecure-Requests: 1
X-Forwarded-For: 10.42.0.1
X-Forwarded-Host: 192.168.73.210
X-Forwarded-Port: 80
X-Forwarded-Proto: http
X-Forwarded-Server: traefik-56688c4464-p7j4q
X-Real-Ip: 10.42.0.1
```

Dank des whoami-Containers sieht man, welcher Pod antwortet.

Wenn Sie nun die IP-Adresse des Masters im Browser aufrufen – diesmal auf dem normalen HTTP-Port 80, sehen Sie wieder die Ausgabe eines Pod. Wenn Sie die Seite neu laden, antwortet jedes Mal ein anderer Pod von einem beliebigen Node. Dadurch wird die Last auf alle Pods auf allen Nodes gleichmäßig verteilt und alle Pods sind über eine IP-Adresse erreichbar.

Weiter gedacht

Damit haben Sie einen ersten einfachen Dienst auf Basis von Kubernetes erzeugt – und damit etwa 1 Prozent des Funktionsumfangs von Kubernetes ausprobiert. Der Dienst kann zum Beispiel noch keine Daten speichern, weil noch keine Volumes enthalten sind. Auch das automatische Skalieren von Diensten oder Multi-Cloud-Umgebungen und zahlreiche andere Kubernetes-Spezialitäten sind nicht zur Sprache gekommen.

Alle Funktionen zu beschreiben, würde schätzungsweise mehrere c't-Ausgaben füllen. Für den Einstieg und das weitere Erforschen von Kubernetes haben Sie jetzt aber das Rüstzeug.

Weiterführende Dokumentationen und Beispiele haben wir für Sie unter dem Link ct.de/wzx3 hinterlegt. (jam) c't

Links und Beispiele

ct.de/wzx3

AKTION
7 STUNDEN VIDEOKURS
MIT LESERRABATT

heise Academy

Continuous Delivery von A bis Z

In diesem Videokurs für Software-Entwickler erklärt Alexander Birk, wie man CI/CD-Pipelines mit GitLab und Jenkins entwickelt, konfiguriert, testet und deployt.

Continuous Delivery ist nicht mehr wegzudenken aus der Welt der Software-Entwicklung (siehe Seite 6). Hat man die in diesem Grundlagenkurs der heise Academy vermittelten Prinzipien einmal verstanden, kann man sie auf beliebige Continuous-Delivery-Szenarien anwenden – von On-Premises bis Cloud, vom Multi-Module-Monolith bis zu einer verteilten Microservice-Architecture.

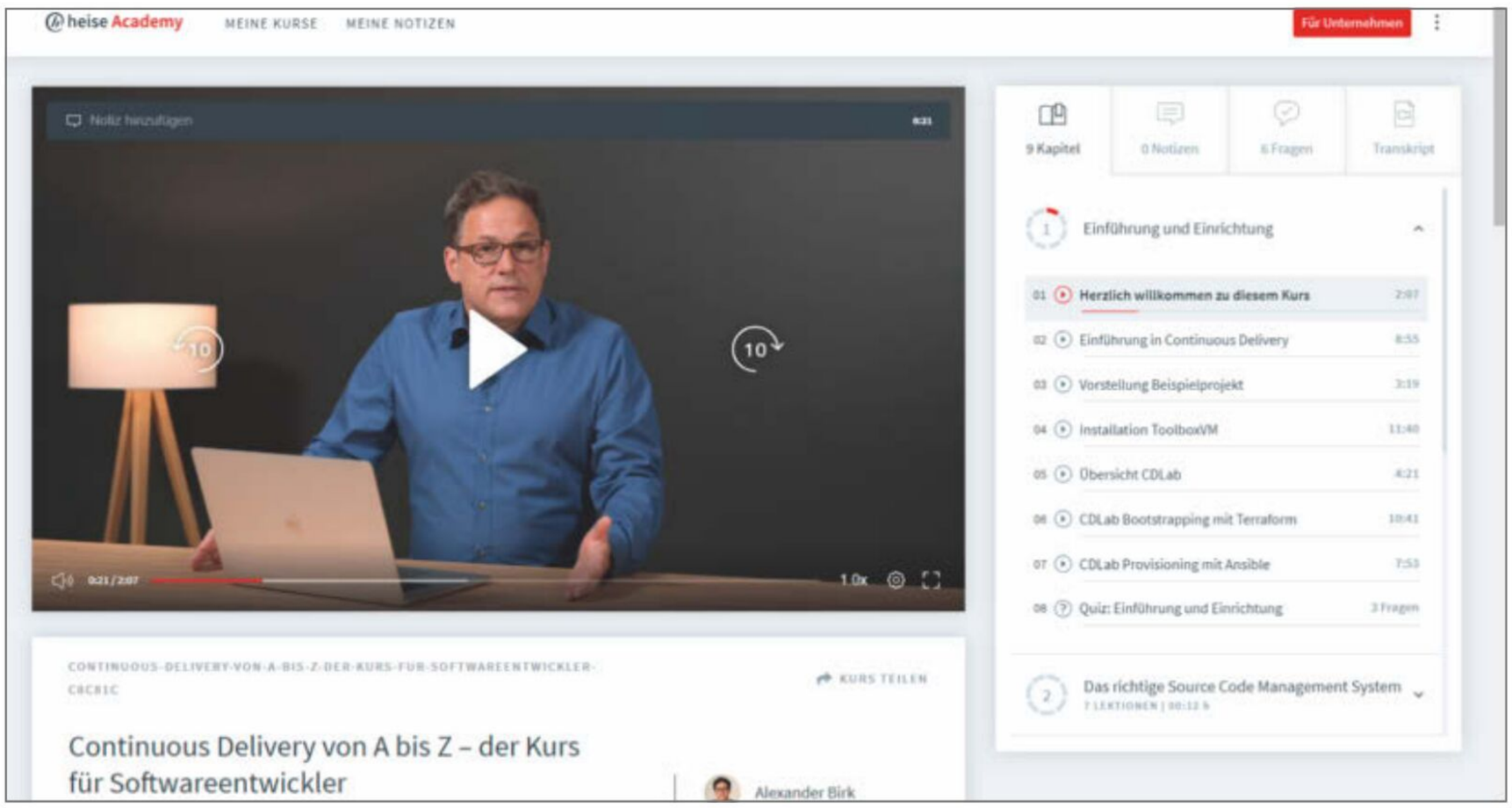
Der Trainer Alexander Birk führt Sie in diesem Videokurs Schritt für Schritt durch die typischen Aufgaben der einzelnen Stages einer CD-Pipeline. Sie machen sich mit Infrastructure as Code vertraut und seiner wichtigen Rolle für die Erstellung von Ziel- und Testumgebungen. Und Sie beschäftigen sich eingehend mit dem Konzept „Pipeline as Code“ und lernen, wie Sie alle für Ihre Pipeline nötigen Build- und Testtools als Docker Images packen und nutzen können.

Anhand einer Beispielanwendung, bestehend aus Frontend und Backend inklusive PostgresSQL, wenden Sie das Kernkonzept „Continuous Delivery Pipeline“ direkt an und entwickeln dazu Ihr eigenes arbeitsfähiges Continuous Delivery Lab inklusive eigenem GitLab und eigenem Jenkins. Das im Kurs gelernte Docker-Compose-basierte Deployment auf einer Single-VM-Zielumgebung lässt sich auch sehr gut auf helm/kustomize-basiertes Kubernetes übertragen.

Die heise Academy ist eine neue digitale Anlaufstelle für IT-Fachkräfte, die sich regelmäßig zu den wichtigsten Themen rund um Softwareentwicklung, Administration und Security fortbilden wollen. Die Auswahl an Videokursen, Online-Konferenzen und Praxis-Webinaren wird kontinuierlich durch aktuelle Neuzugänge erweitert.

Über ct.de/wxs5 erhalten Sie diesen Videokurs einmalig für 19,90 Euro* statt 119 Euro.

*Das Angebot ist gültig bis zum 13.12.2021



In 9 Kapiteln mit insgesamt 74 Video-Lektionen lernen Sie, wie Sie Ihre aktiven Anwendungen mit Continuous Delivery schneller und besser pflegen, entwickeln und erweitern.

Starthilfe für den eigenen Server

Für einen eigenen Server sprechen starke Argumente: Er liefert maximales Tempo im Netz. Er verhindert, dass Sie Ihre Daten Fremden aushändigen. Er erlaubt individuelle Lösungen. Er vermittelt aktuelle Techniken im Selbstversuch. Aber: Er erfordert ein hohes Maß an Verantwortung und birgt – besonders bei Vernachlässigung – Risiken. Unsere Starthilfe stellt sich diesen Anforderungen.

Von **Peter Siering**

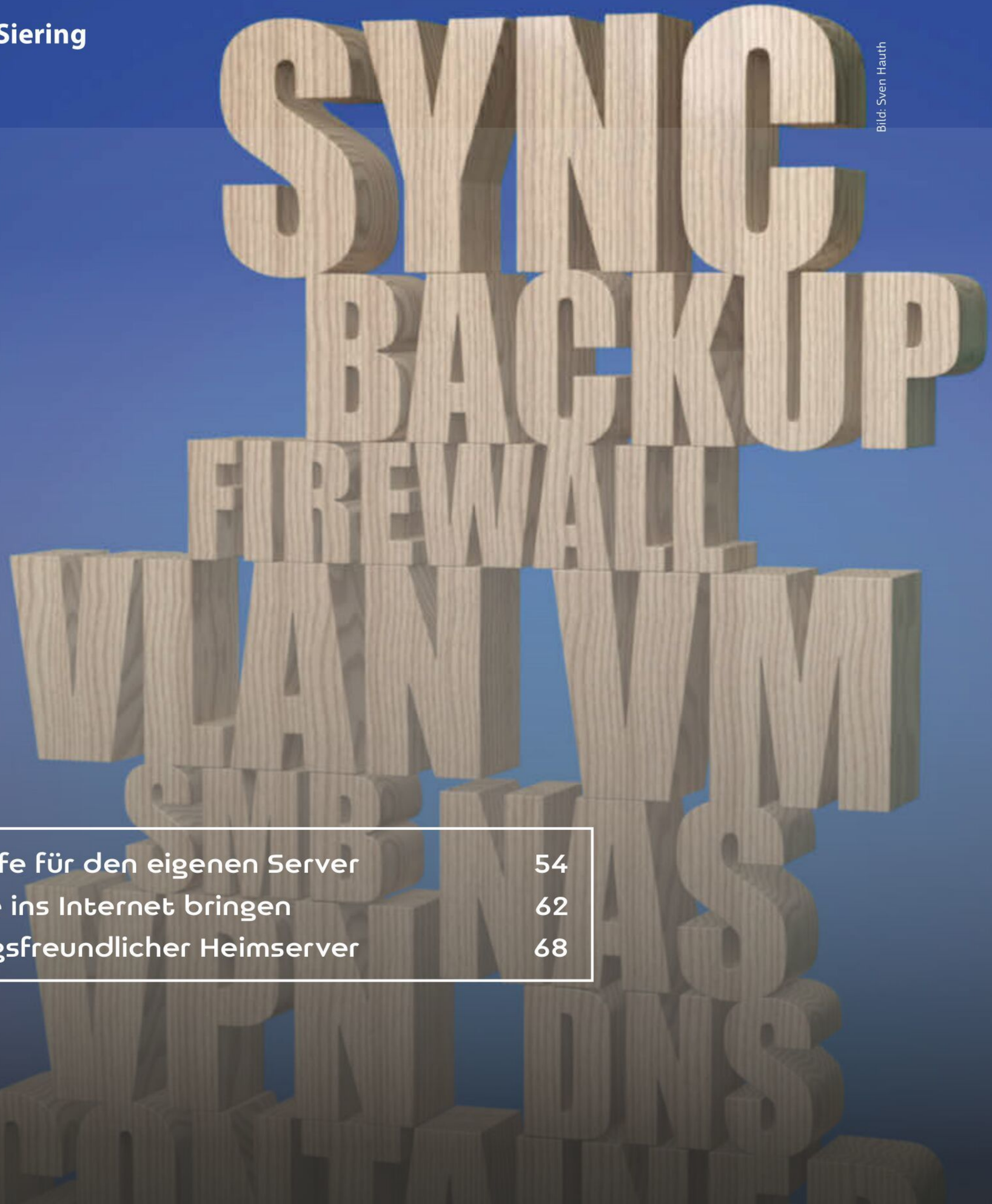


Bild: Sven Hauth

Starthilfe für den eigenen Server	54
Dienste ins Internet bringen	62
Einstiegsfreundlicher Heimserver	68

Sogar Internet-Anschlüsse für Privatleute sind oft schon schnell genug, um einen eigenen Server hinter den Router zu stellen und damit nicht nur das lokale Netzwerk zu versorgen, sondern auch für das Smartphone und das Notebook unterwegs sinnvolle Dienste anzubieten. So brauchen Sie Ihre Daten nicht außer Haus zu geben und müssen trotzdem nicht auf cloudartige Dienste wie Kalender, Dateisynchronisierung et cetera verzichten. Tipps, um Dienste hinter einem Router im Internet erreichbar zu machen, gibt der Beitrag auf Seite 62. Dieser Artikel widmet sich der Software-Ausstattung eines eigenen Servers.

Ausgelagert

Ein eigener Server muss nicht zwangsläufig heißen, dass Sie um Angebote einen Bogen machen, die man mithin auch Cloud nennen kann: Statt selbst Hardware anzuschaffen, können Sie ebenso gut

Kurzcharakteristik: Weitere Spezialdistributionen für Server

Betriebssysteme beziehungsweise Distributionen, die sich für die Installation auf einem Server anbieten, gibt es reichlich. Die meisten großen Distributionen wie Debian bieten eine Installationsvariante an, die ohne grafische Bedienoberfläche mit reiner Textoberfläche daherkommen. Der Server-Admin muss vieles selbst machen.

Red Hat und SUSE veröffentlichen eigene Serverversionen mit kommerziellem Support und Zertifizierung für Hardware. Die vom Anbieter ergänzte Verwaltungsoberfläche unterstützt den Admin bei vielen gängigen Aufgaben. Der Einsatz erfordert einen Support-Vertrag mit jährlichen Lizenzkosten.

Kommerzielle Anbieter von Linux-Serverdistributionen erlauben oft den kostenlosen Einsatz ihrer Produkte mit Einschränkungen: Mal ist die Anzahl der Nutzer begrenzt, mal dient der Admin als Testhelfer, weil er seinen Server mit aktuellen, aber wenig intensiv getesteten Paketen der Distribution betreibt.

OpenMediaVault (OMV) nennt sich selbst „freies Betriebssystem für Netzwerk-Speicher (NAS)“. Es stammt von Volker Theile, der einst FreeNAS entwickelt hat. Längst kann die Distribution auf Debian-Basis viel mehr als nur Dateien bereitstellen. Eine Weboberfläche erleichtert Einsteigern den Umgang. OMV läuft auch auf Raspis.

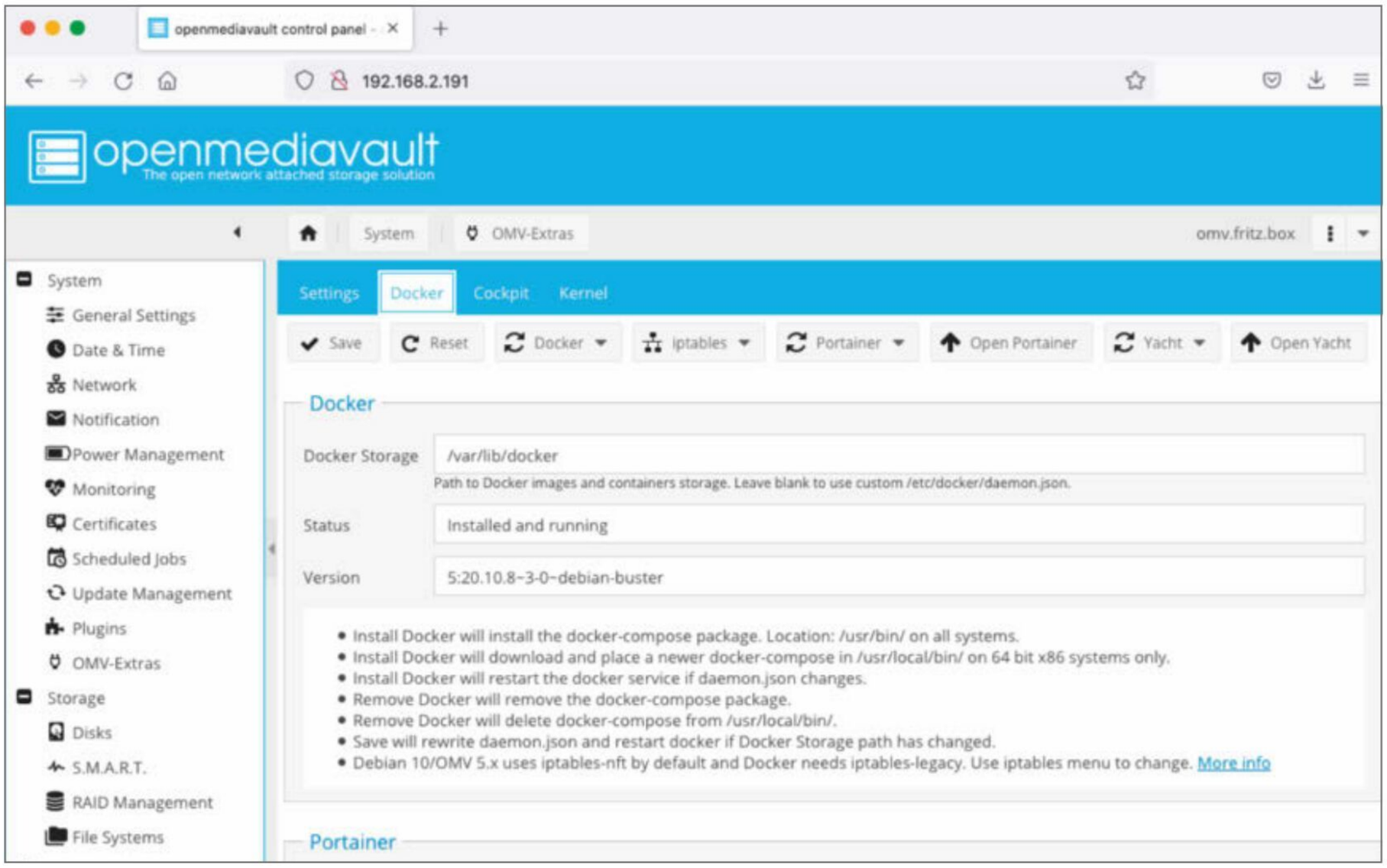
OPNSense glänzt als Routerdistribution mit durchgängiger IPv6-Fähigkeit – etwas, was die bekannten Linux-Pendants in dem Feld vermissen lassen. Es baut auf FreeBSD auf und wird ausschließlich in einer Community entwickelt.

Proxmox Virtual Environment stellt auf Debian-Basis eine featurereiche Virtualisierungsumgebung bereit, die im Artikel an einigen Stellen als leuchtendes Beispiel auftaucht.

TrueNAS als Nachfolger von FreeNAS lässt sich kostenlos nutzen. Es ist die Wahl schlechthin, wenn man den Server nur als Datei- und Datenablage braucht, bietet aber auf Virtualisierungsfunktionen und viele Plug-ins. Als Systembasis dient derzeit noch FreeBSD, aber langfristig soll es Debian werden. Für das Dateisystem ZFS bringt TrueNAS interessante Features wie Snapshots, Datensicherheit und verschiedene Redundanz-Level. Ein Web-GUI hilft beim Verwalten.

Univention Corporate Server (UCS) zielt auf größere Multi-Server-Installationen. Die Software schafft auf Debian-Basis um einen Verzeichnisdienst herum, der kompatibel zu Microsofts Active Directory ist, eine eigene Systemlandschaft. Über das Univention „App Center“ lassen sich auch die Serverdienste von Fremdanbietern nahtlos in eine UCS-Installation integrieren und damit per Browser verwalten, etwa Groupware, Backup, Archivierung …

Unraid OS, das der Artikel auf Seite 68 vorstellt, geht einen kreativen Weg, um einen Storage-Server einzurichten. Die Distribution läuft vom USB-Stick und kann ein Sammelsurium verschiedener Festplatten zu einem redundanten Datenspeicher zusammenfassen. Plug-ins, die oft Docker als Technik nutzen, erschließen viele weitere Funktionswelten. Als Basis dient eine weitgehend aus dem Sichtfeld verschwundene Distribution: Slackware. Der Einsatz über die Testphase hinaus ist nur durch eine einmalige Lizenzzahlung möglich.



Linux gibt den Ton an, egal ob auf den NAS-Geräten der großen Hersteller oder beim NAS-Selbstbau mit Distributionen wie OpenMediaVault. Letzteres läuft ebenso auf dem Raspi und erschließt mit dem Community-Repository auch Container.

Hardware mieten. Der eigene Server darf auch im Rechenzentrum stehen. Dann sind die Stromkosten mit der Miete bezahlt und Sie brauchen sich nicht um Platz und Lärmdämmung zu kümmern.

Hostingfirmen überlassen Ihnen virtuelle Server für weniger als fünf Euro monatlich – dort können Sie zum Beispiel einen eigenen VPN-Server betreiben, der Ihr Smartphone in offenen WLANs sichert und nicht mehr kostet, als bei einem kommerziellen VPN-Provider zu berappen wäre. Die Monatsmieten für reale Hardware starten bei rund 30 Euro. Viele der deutschen Hoster haben einen guten Ruf, was Service und Zuverlässigkeit angeht. Sie sollten aber in der Lage sein, sich bei Software-Problemen selbst zu helfen. Das können die Hoster bei den Kampfpreisen nicht leisten.

Bei nahezu sämtlichen Mietangeboten und auch Betriebssystemen, die man auf den meisten Fertigservern wie NAS-Geräten findet, handelt es sich um Linux. Auf Mietservern herrscht es eher in roher Form gängiger Distributionen wie Debian, Ubuntu oder Fedora vor – zumeist ohne grafische Bedienoberfläche. Auf NAS-Geräten finden sich Spezialdistributionen der Gerätehersteller, die schon mit einer Weboberfläche zur Verwaltung ausgestattet sind. Bei den rohen Distributionen ist man selbst dafür verantwortlich, etwas Passendes einzurichten oder sich selbst auf der Befehlszeile auszutoben – je nach Geschmack.

Wer den Server im eigenen Netz betreibt, kann auch dort auf eine spezialisierte Distribution zurückgreifen, die dann wie ein NAS-Gerät mit einer eigenen Verwaltungsoberfläche glänzt. Diese Serverdistributionen nutzen nicht alle Linux, sondern auch andere freie Betriebssysteme als Basis. Für den Betreiber macht das keinen großen Unterschied, solange er nicht tief in die Innereien abtauchen muss, um individuelle Erweiterungen oder Korrekturen vorzunehmen – hierfür hätte er es mit Linux oft leichter, weil da mehr und ausführlichere Hilfestellung im Netz zu haben ist, die indes in der Masse oft schwerer zu finden ist.

Vergeigt

Windows spielt in dem Segment autonom betriebener Server für Privathaushalte und kleinere Büros kaum mehr eine Geige: Microsoft hat systematisch alle Angebote wie Home und Small Business Server für diese Zielgruppe vor fast zehn Jahren eingestellt. Das günstigste Angebot von Microsoft für bis zu 25

Nutzer ist eine Lizenz des Windows Server Essentials für rund 300 Euro. Im Vergleich zu Linux fällt die Auswahl der Software eher schmal aus, die man darauf aufbauend einsetzen könnte und die nicht wieder erhebliche Kosten nach sich zieht; Microsofts Exchange etwa fordert zusätzlich eine Standard- oder Datacenter-Ausgabe des Servers.

Kurzum, Windows-Server wird man dann einsetzen, wenn es eine Serveranwendung gibt, die ohne nicht zu haben ist – oft handelt es sich um spezialisierte Software, deren Hersteller womöglich auch gleich zertifizierte Hardware fordert, wenn man Support erwartet.

Andererseits: Vielleicht muss es für serverartige Anwendungen auf Windows-Seite gar kein Serverbetriebssystem sein? Die Pro-Version eigne sich durchaus als Dienstleister im Netz. Sie lässt sich per Remote-Desktop-Protokoll (RDP) fernsteuern, kann Freigaben per SMB bereitstellen. Software zur Dateisynchronisierung und Verschlüsselung lässt sich ebenfalls installieren, um einen Backup-Server für Windows-Clients nach eigenem Gusto zu kreieren. Wer sich mit Windows auskennt, mag diese Lösung bevorzugen.

Root-Server: Anders starten

Wer einen virtuellen oder realen Server bei einem Hoster mietet, muss sich darüber klar sein, dass von der ersten Sekunde an Besucher an Türen, Fenstern und Mauern rütteln werden. Es darf also niemals ein Dienst oder ein Port offenstehen, der nicht abgesichert ist. Nicht abgesichert heißt: unverschlüsselt, mit schwachem oder gar ohne Kennwort zugänglich. Schützen Sie deshalb auch das Fundament: Firewallregeln, die am Anfang alles außer einem Wartungszugang ablehnen, sind eine gute Idee.

Als Wartungszugang empfiehlt sich ausschließlich SSH, das Sie nur mit einem per Passphrase geschützten Schlüsselpaar nutzen; das Anmelden mit Nutzername und Passwort per SSH verbieten Sie. Wenn Sie zur Konfiguration etwa die Weboberfläche der eingerichteten Virtualisierung erreichen müssen, geben Sie nicht deren Port frei, sondern nutzen Sie eine Port-Weiterleitung per SSH (über die Option `-L`). Nach dem Durchkonfigurieren können Sie die Firewall für die angebotenen Dienste mit Ausnahmen öffnen. So ist sichergestellt, dass versehentlich veröffentlichte Dienste nicht gleich für die Welt erreichbar sind.

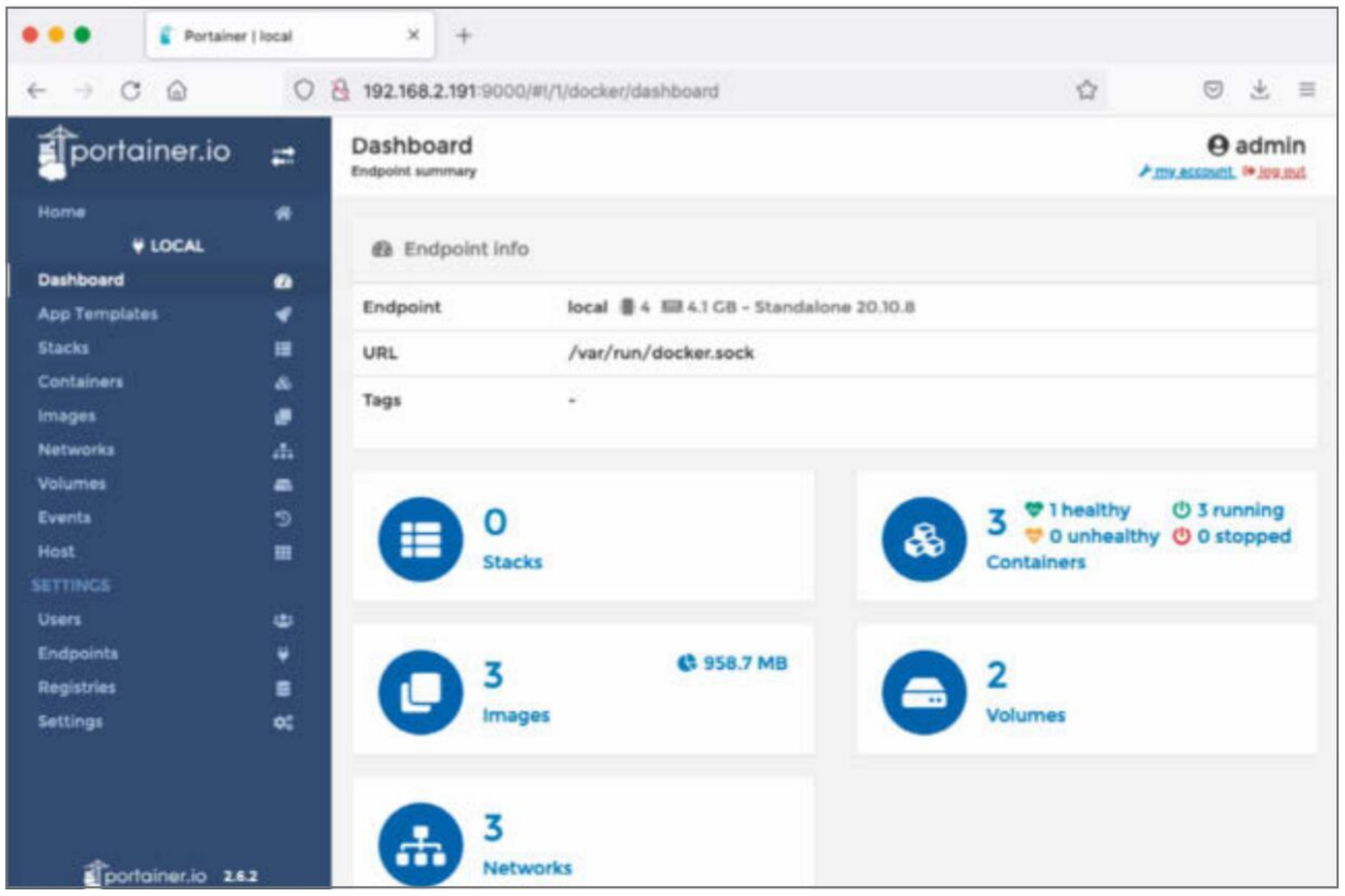
Auf Windows als Betriebssystem auf dem Server sollte man sich nur dann einlassen, wenn man entweder die volle Systemleistung für Windows wirklich braucht oder sich mit Windows alle Ansprüche an den Server erfüllen lassen. Wer mehr vorhat, sollte darüber nachdenken, ein Betriebssystem auf dem Server zu nutzen, das Virtualisierung beherrscht. Der Server lässt sich so in mehrere logische Server (virtuelle Maschinen) aufteilen, in denen getrennt voneinander unterschiedliche Betriebssysteme laufen können. So können Windows und Linux friedlich nebeneinander existieren. Jedes kann mit seinen Stärken glänzen.

Theoretisch führt Windows ab der Pro-Ausgabe dank der von Microsoft integrierten Technik Hyper-V selbst virtuelle Maschinen aus. Das läuft besonders gut, wenn Windows in den VMs arbeiten soll. Die Integration von Linux-Gästen ist aber weit fortgeschritten. Die Verwaltungsoberfläche von Hyper-V ist jedoch eher spröde. Linux-basierte Lösungen und das, was die NAS-Hersteller integrieren, bringen meist eine einfach bedienbare Weboberfläche mit. Ein Browser genügt. Man ist nicht auf unter Windows laufende Spezialsoftware angewiesen, wie sie auch für VirtualBox oder VMware Player und Workstation nötig wäre, was die für eigene Server eher disqualifiziert.

Und noch ein letztes Argument für Linux und gegen Windows als Betriebssystembasis: Heute vereinfachen die in Clouds gebräuchlichen Container-Techniken den Einsatz und die Wartung von Serverdiensten erheblich – wer sich das Leben leicht machen möchte, setzt diese auch auf dem eigenen Server ein. Hierfür hat Windows derzeit wenig zu bieten und die dort verfügbare Docker-Integration ist an vielen Ecken und Kanten kaputt. Linux-Container sperrt Windows in VMs weg, was durchaus sinnvoll ist. Windows-Container haben nie Fahrt aufgenommen. Die in Windows integrierten Techniken wie das Subsystem für Linux sind nicht für den dauerhaften Betrieb von Diensten ausgelegt.

Containert

Obschon Container meistens mit Clouds assoziiert werden, sind sie auch für autonome Serverbetreiber eine begnadete Erleichterung: Im Docker-Hub gibt es einen riesengroßen Fundus von direkt einsetzbaren Images, um Installationen von Nextcloud, ganzen Mailserver-Stacks, Smart-Home-Zentralen, Videokonferenzsystemen und vielem mehr mit wenigen Handgriffen einsatzfertig einzurichten. Viele Projekte stellen diese Images inzwischen automa-



Container helfen, komplexe Anwendungen mit weniger Handgriffen einzurichten und aktuell zu halten. Sie unterstützen auch die Wartung ihrer selbst: Portainer als Weboberfläche und Watchtower für auto-matische Updates stecken dabei selbst in einem Container.

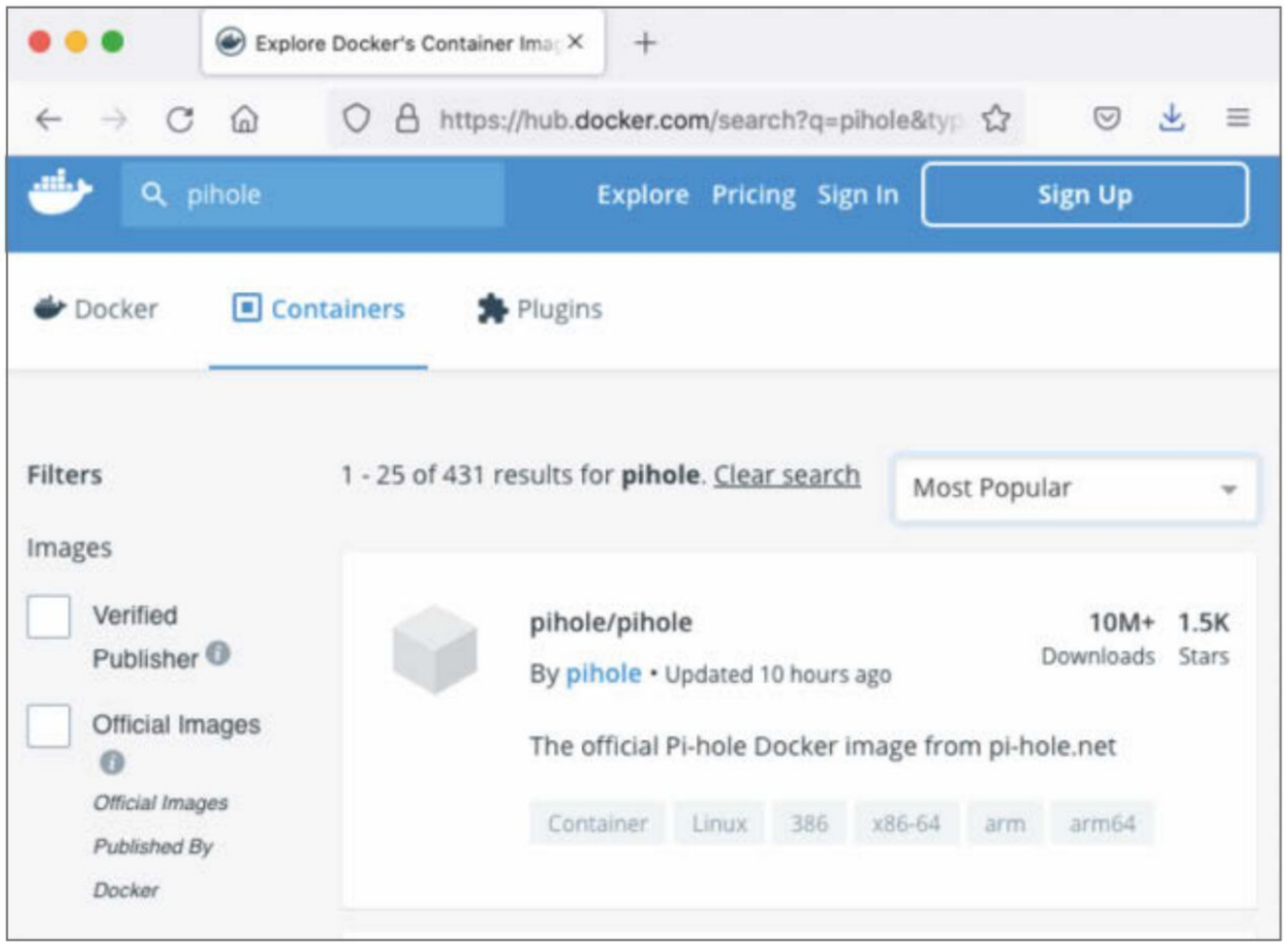
tisiert selbst her, sodass man sie aus solchen Quellen bedenkenlos einsetzen kann, wenn man die wesentlichen Vorbehalte kennt – ein älterer c't-Artikel hilft dabei (siehe ct.de/wmgz).

Weil sie stets aktuelle Software versprechen, sind Docker-Images beliebt. NAS-Hersteller verwenden sie, um ihre Geräte mit weiteren Funktionen zu versorgen. Viele Spezialdistributionen für Server können im Auslieferungszustand Docker-Images beziehen und ausführen, bei anderen kann man solche Fähigkeiten nachrüsten. Eine browserbedienbare Verwaltungsoberfläche ist Standard. Allerdings bildet die oftmals nicht alle Möglichkeiten ab, die man mit Docker auf der Kommandozeile selbst hätte. Manche Distribution, beispielsweise Unraid, ergänzen Docker-Images mit etwas Zuckerguss, um sie hübsch in das eigene Universum zu integrieren. Docker geht natürlich auch zu Fuß auf der Kommandozeile und ist auch auf dem Raspi nützlich.

Das Grundprinzip ist bei allen gleich: Sie laden das Docker-Image herunter und starten die im Container verdrahteten Prozesse in einer geringfügig

CLOUD: Can't Locate Our User's Data

Einen Server selbst zu betreiben, heißt digital souverän zu sein. Man sollte sich aber keinen Illusionen hingeben, diesen Job besser zu können als die Admins in den Cloud-Rechenzentren. Die tun im Schichtbetrieb nichts anderes, als die Hardware am Laufen zu halten und die Sicherheitslage zu studieren, um die Schäfchen in ihrer Serverherde vor den Wölfen zu schützen. Der 1-Server-Admin merkt im Zweifel erst, dass sein Schäfchen zur Beute wurde, wenn der Server nicht mehr zuckt. Selbst wenn der 1-Server-Admin seiner Sorgfaltspflicht nachkommt, zeitnah Sicherheitsupdates einspielt, regelmäßig die Aktivitäten auf dem System beobachtet, die normale Lastsituation kennt und sich bei Abweichungen alarmieren lässt, muss er dennoch damit rechnen, dass unerwünschte Besucher über die Schwelle treten. Das heißt, auch Notfallpläne gehören zur Verantwortung: Backups abseits des Servers und Dokumentation zum Neuaufsetzen von Diensten. Souverän ist nur, wer am Ende nicht sagen muss „Can't locate my user's data".



Vom simplen Web-Server bis zum vollständigen Mail-Server-Stack ist so ziemlich alles als Docker-Image zu haben. Einen Überblick verschafft der Docker Hub.

vom restlichen System abgetrennten Umgebung. Die Abtrennung erfüllt keine großartigen Sicherheitsansprüche, sondern ist eher eine logische, um Abhängigkeiten abzuschirmen, etwa unterschiedliche Bibliotheken. Die Detailkonfiguration im Container, etwa angelegte Benutzerkonten oder abgelegte Dateien landen idealerweise in separaten Verzeichnissen (benannte Volumes); oft verlangen Images, dass man ihnen für die Daten eine Datenbank zur Seite stellt, die dann ihrerseits als Container daherkommt und selbst ein benanntes Volume verwendet.

Dadurch, dass Container auf diese Weise den Zustand (Daten) und die Systemumgebung (die Software) trennen, lässt sich letztere bei einem Update wegwerfen und durch eine neue Version ersetzen. Das Update stoppt den Container, holt das neue Image und startet es erneut mit den gleichen Optionen wie beim initialen Start. So gelangen die in den benannten Volumes abgelegten Daten wieder in die Anwendung. Von Hand ist das mühsam. Deswegen gibt es Container-Images, die diese Aufgabe automatisch erledigen. Bewährt in Docker-Umgebungen ist Watchtower. Es sollte in keiner produktiven Umgebung fehlen.

Die Dienste eines Containers gelangen trickreich ins lokale Netz: Stellt ein Container zum Beispiel einen Webserver bereit, so nimmt der Docker-Host, also das System auf dem Docker läuft, diese Anfragen an und reicht sie in den Container weiter. Auf einem Linux-System gelingt das durch virtuelle Netzwerke, die Docker einrichtet. Jeder Container hat eine IP-Adresse in diesem Netzwerk und der Docker-Host dient als Vermittler dorthin; ähnlich wie ein Router benutzt Docker dabei Network Address Translation (NAT).

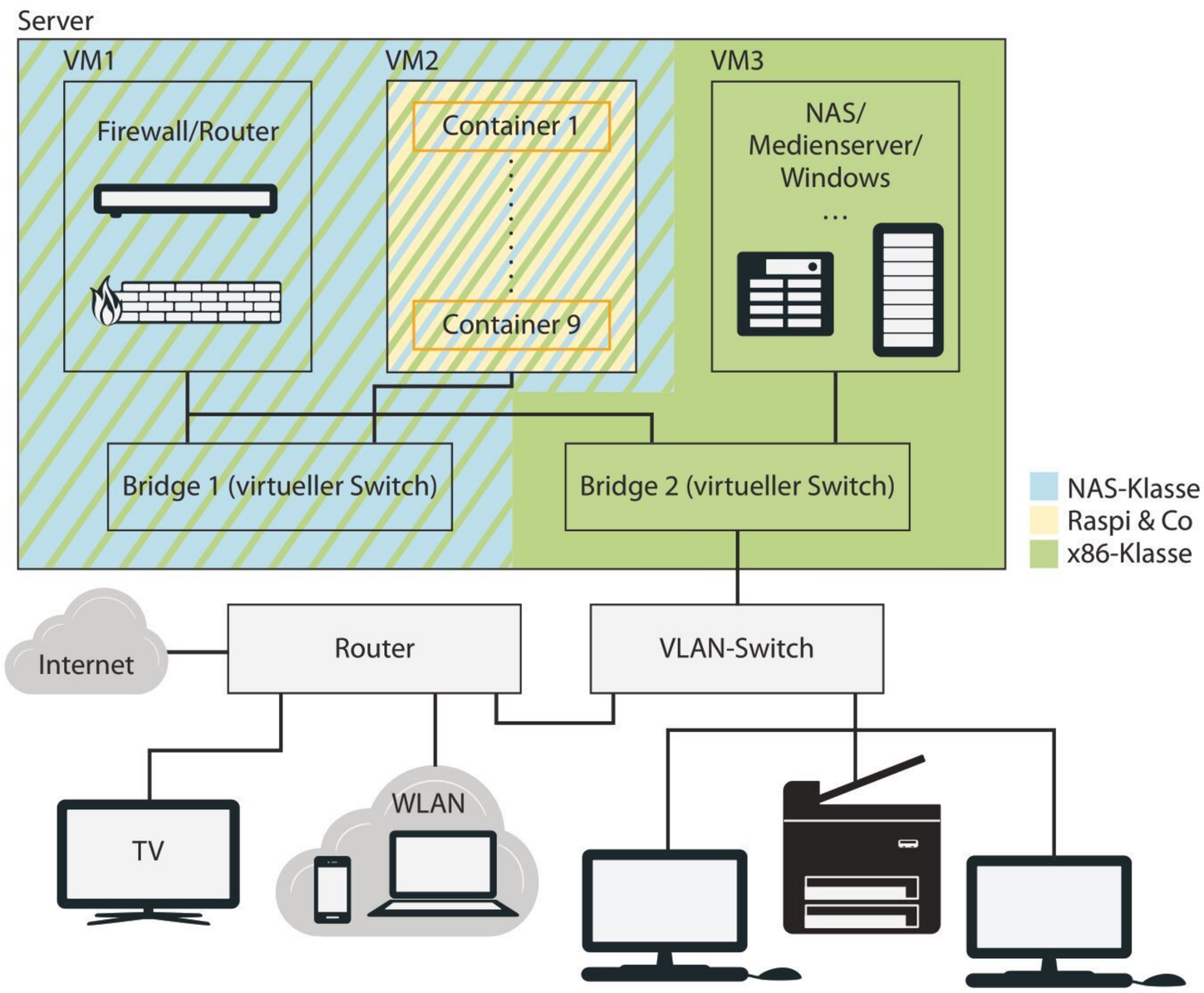
Ein letzter wichtiger Punkt beim Container-Einsatz ist noch, wie man sie mit der Außenwelt verbindet. Wer Docker in Eigenregie betreibt, muss sich darum selbst kümmern. Wie schon bei den Updates bietet sich hier wieder ein Docker-Image als Lösung an: Traefik (siehe ct.de/wmgz). Es arbeitet als sogenannter Edge-Router vergleichbar mit einem Reverse Proxy. Es wird dazu mit der Außenwelt verbunden, etwa über eine Freigabe im Router, die dafür sorgt, dass dieser Zugriffe an den Traefik-Container weiterleitet. Traefik sortiert die dann und leitet sie an den passenden Container weiter. Für Traefik spricht auch seine Fähigkeit, automatisch kostenlose TLS-Zertifikate bei Let's Encrypt zu beschaffen.

Empfohlen

Abseits der bereits erwähnten Fertigdistributionen, die Serverdienste und Virtualisierung beziehungsweise Container-Technik vereinen, gibt es derzeit vor

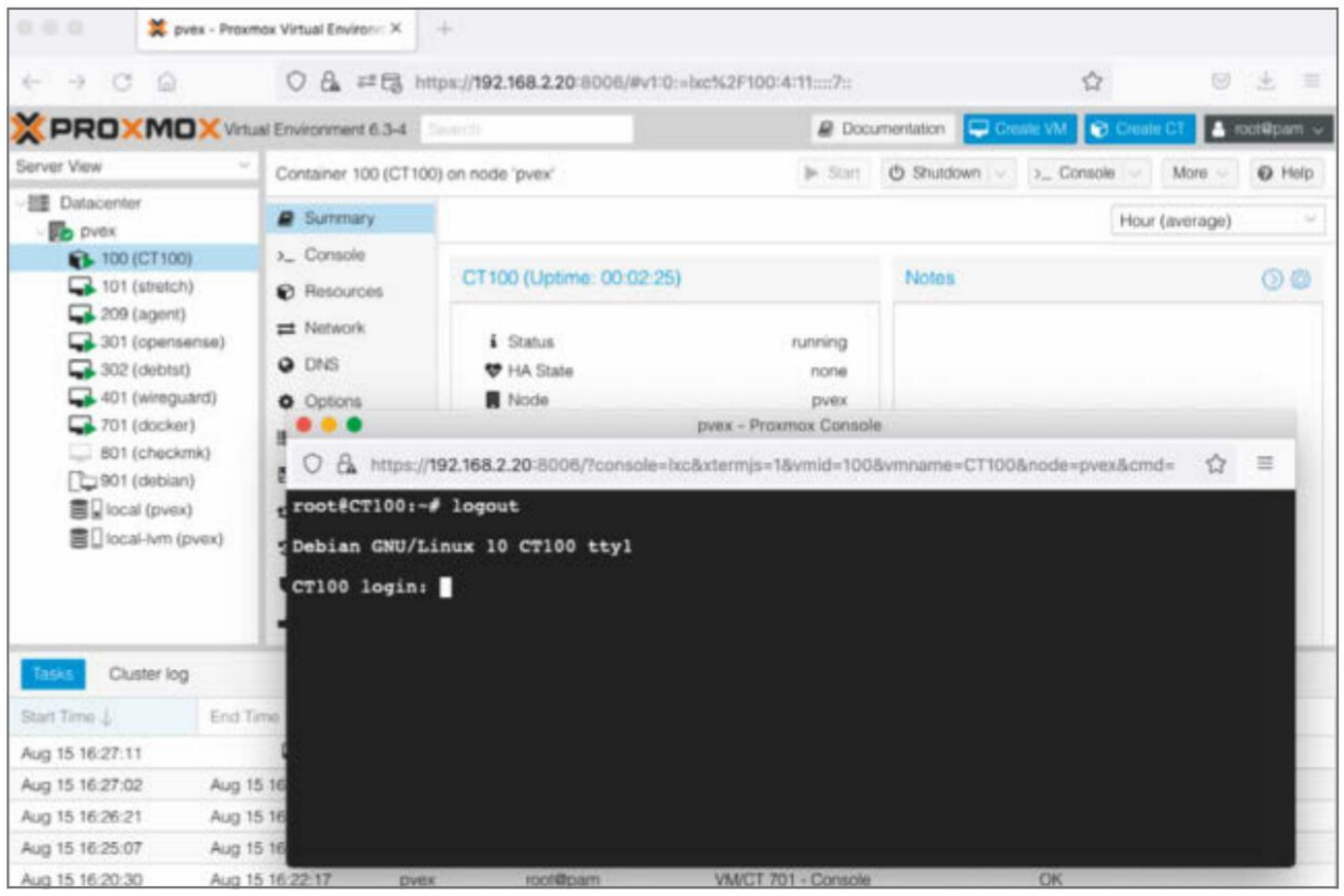
Server als Minirechenzentrum

Ein einzelner Server kann heute mehrere Systeme und Netze nachbilden, wie sie früher in einem Rechenzentrum standen. Je nach Anspruch genügen für den eigenen Bedarf aber schon einzelne Bausteine, um Arbeitslasten zu trennen. Für Container reicht ein Raspi. NAS-Geräte stellen virtuelle Maschinen bereit. Ein x86-Server kann mit bis ins Netzwerk fortgeschriebener Virtualisierung glänzen; ein VLAN-tauglicher Switch hilft der Router-VM dann, Netze mit Smart-Home- und Media-Gedöns von gefährdeten PCs zu trennen.



allem eine empfehlenswerte Software, um virtuelle Maschinen auf dem eigenen (Miet-)Server zu betreiben: „Proxmox Virtual Environment“, kurz Proxmox. Die Distribution der gleichnamigen Wiener Firma versieht die Open-Source-Techniken KVM zur Virtualisierung und ZFS als Dateisystem mit einer einfach zu bedienenden Weboberfläche. Als Basis nutzen die Entwickler ein geringfügig aufgemotztes Debian – Systemupgrades über viele Generationen von Proxmox hinweg gelingen wohl auch wegen der Debian-Basis seit Jahren sehr geschmeidig.

Proxmox lässt sich vollständig kostenlos nutzen, wenn man sich darauf einlässt, Aktualisierungen aus einem Repository zu beziehen, dessen Pakete nicht durch und durch getestet sind. Der Hersteller empfiehlt es nur für Testbetrieb, aber nicht für den produktiven Einsatz. Der Preis für ein Jahresabo der Software beträgt für einen Ein-Sockel-Server gerade mal 90 Euro, verschafft Zugriff auf das Repository mit den intensiv getesteten Enterprise-Paketen und erhöht die eigene Sichtbarkeit im Community-Forum.



Mit Proxmox macht Virtualisieren Spaß: Die Weboberfläche lässt kaum Wünsche offen. Läuft der Server nicht vor Ort, lässt sich seine Konsole und die der VMs per Browser erreichen. Wenn die Ressourcen knapp sind, können Linux-Umgebungen auch als Container neben den VMs laufen.

Die Voraussetzungen für den Einsatz von Proxmox fallen moderat aus. Für Testzwecke genügt laut Hersteller eine 64-Bit-Intel- oder AMD-CPU mit Virtualisierungsunterstützung. Das erfüllt heute bei passenden BIOS-Einstellungen (Virtualisierung eingeschaltet) so ziemlich jeder PC. Die Menge des nötigen Hauptspeichers hängt von den Aufgaben ab, die Ihr Server mal erfüllen soll: Für zwei kleinere VMs genügen 4 GByte RAM insgesamt. Schon ein Zotac-x86-Mini-PC mit Core i3 kann als Minimallösung in einer VM seinen Dienst als SMB-Server oder Freifunk-Offloader verrichten und als Brückenkopf zur Verwaltung eine zweite VM ausführen.

Das Coole an Proxmox ist aber, dass es nicht nur kleine Arbeitslasten hebt, sondern auch in professionellem Einsatz glänzt: Aus mehreren Proxmox-Maschinen lässt sich ein Cluster bauen, der dank integriertem Ceph-Dateisystem die virtuellen Platten der VMs auf einem gemeinsamen, automatisch replizierten Speicher ablegt, sodass sich laufende VMs von Knoten zu Knoten verschieben lassen. Ein solcher Hyper-Converged-Cluster, der sowohl Rechen- als auch Speicherkapazität bereithält, stellt natürlich höhere Anforderungen und braucht mindestens drei aktive Knoten.

c't-Artikelliste zum Containern mit Docker und Einrichten von Server-Diensten

ct.de/wmgz

Proxmox kennt noch weitere Spielarten, um Speicher und VMs über mehrere Knoten in einem Cluster zu verteilen (siehe ct.de/wmgz). Doch es beherrscht auch ganz pragmatische Ansätze, ausgeführte VMs abzusichern: Promox-Server können sich, wenn sie wie empfohlen ZFS als Dateisystem verwenden, über SSH-Verbindungen regelmäßig inkrementelle Snapshots der virtuellen Platten von VMs zuspielen. Da die dazu eingesetzte Software PVE-zsync nur die Änderungen überträgt, funktioniert das nach einer initialen, eher langwierigen Übertragung auch über asymetrische DSL-Anschlüsse wunderbar flüssig.

Wenn Sie vorhaben, intensiv mit virtuellen Maschinen zu hantieren, brauchen Sie nicht nur reichlich Hauptspeicher, sondern vor allem eine SSD, auf der die virtuellen Platten der VMs landen. Liegen die auf einer Magnetplatte, werden viele Operationen, zum Beispiel das Klonen, zur Geduldsprobe. Für Experimente mit Systemumgebungen, etwa einer Debian-Installation für Versuchszwecke, bietet Proxmox zusätzlich Linux Containers (LXC), für die es reichlich fertige System-Images bereitstellt; die Linux-Container begnügen sich im Vergleich zu VMs mit einem schmaleren Satz Systemressourcen.

Strukturiert

An den Möglichkeiten wird deutlich, dass ein eigener Server heute mehr als eine Dateiablage mit einigen Extradiensten sein kann. Die von ihm angebotenen Dienste lassen sich mit virtuellen Maschinen und Containern zu einem Gesamtgebilde kombinieren, das mehr ein Rechenzentrum früherer Tage nachbildet.

Statt einzelne Dienste zu konfigurieren, können Sie fertige Distributionen in VMs starten. Statt viele Pakete für einen Mailserver einzeln zu installieren und aufeinander abzustimmen, greifen Sie auf ein Container-Image zurück. Unter ct.de/wmgz haben wir Artikel, die dabei helfen, aus den letzten zwei c't-Jahrgängen zusammengestellt.

Lassen Sie sich von der Komplexität nicht schrecken und versuchen Sie am Anfang nicht alles auf einmal. Sie können auch mit einer Fertiglösung wie NAS oder einer Spezialdistribution starten. Wenn die nicht mehr alle Wünsche erfüllt, ist vielleicht ein Raspi mit ein paar Containern eine geeignete Ergänzung. Ist der Gerätezoo dann zu stark angewachsen, haben Sie genug Erfahrung, um sich an eine komplexere Lösung auf Basis von Proxmox zu machen, oder Sie vertrauen sich doch lieber Dritten an. (ps) c't



Bild: Sven Hauth

Dienste ins Internet bringen

Oft sollen Dienste des eigenen Servers hinter dem heimischen Router aus dem Internet erreichbar sein, etwa der Kalender von Nextcloud oder die Weboberfläche der Smart-Home-Zentrale. Mit einer Fritzbox geht das ganz leicht – ähnlich einfach gelingt das aber auch mit Routern anderer Hersteller.

Von **Peter Siering**

Ein Server, der hinter einem DSL- oder Kabel-Router steht, wird durch dessen Firewall geschützt. Wer unterwegs etwa mit dem Smartphone auf die Dienste zugreifen will, muss hinter die Firewall gelangen. Dafür bieten die meisten Router Portfreigaben oder -weiterleitungen an (gebräuchlich sind beide Begriffe, sie meinen dasselbe). Die Geräte registrieren dann Netzwerkanfragen von außen und leiten sie an vom Nutzer festgelegte Ports interner Geräte weiter – ähnlich wie sie Anfragen aus dem lokalen Netzwerk ins Internet durchreichen, eben nur in umgekehrter Richtung.

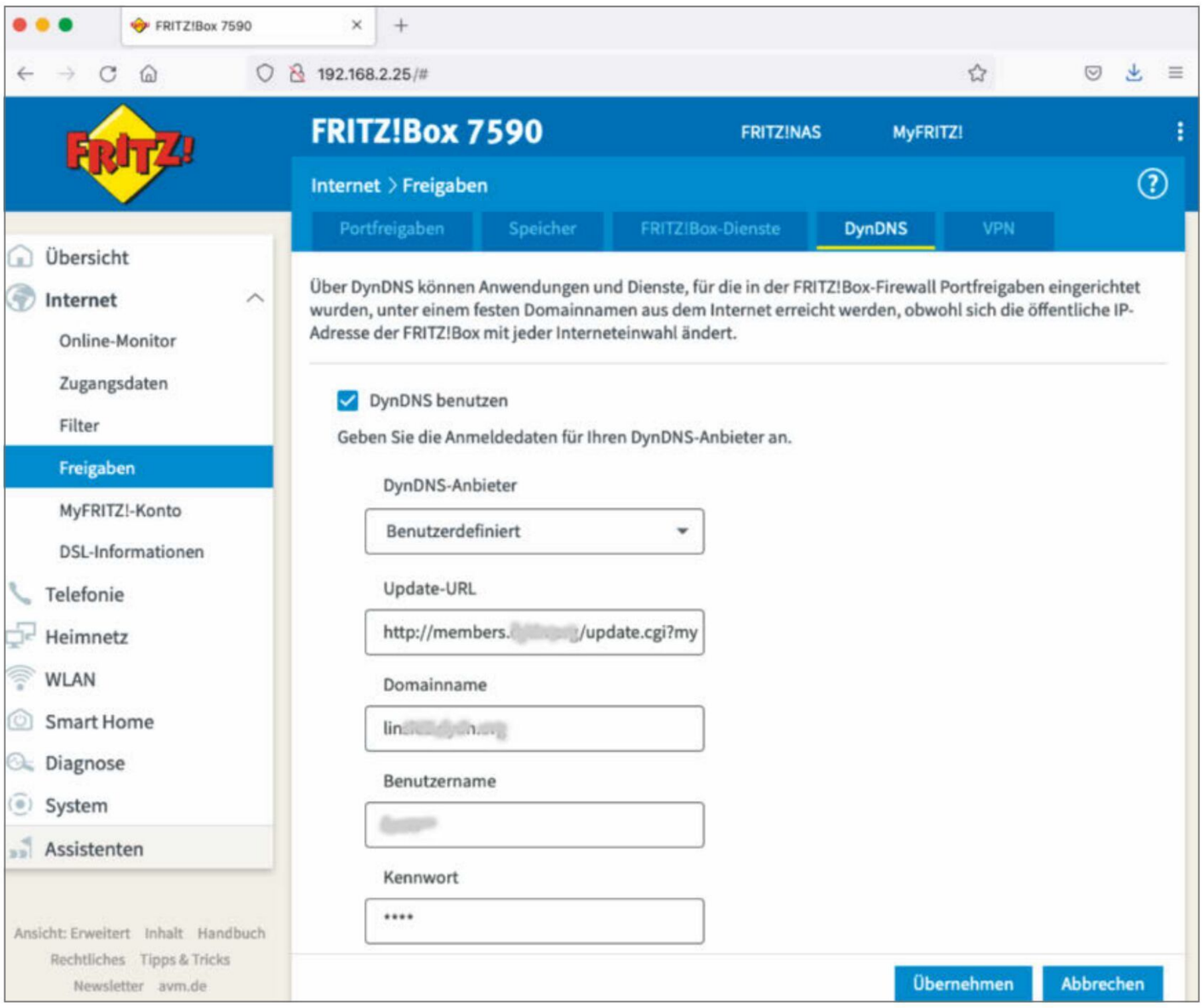
Die ominösen Ports, von denen die Rede ist, sind letztlich Teil einer Netzwerkadresse und tauchen

manchmal auch am Ende von URLs auf. So weist „https://ct.de:443“ in der Adresszeile einen Browser an, eine Verbindung zum Webserver auf ct.de aufzubauen und dazu den Port 443 anzusprechen (der für HTTPS verwendet wird). Für eine Fülle der Ports ist definiert, welcher Dienst dahintersteht. Somit muss man den Port üblicherweise nicht angeben.

Die Idee, einen privaten Webserver auf einem ungebräuchlichen Port laufen zu lassen, um ihn notdürftig zu verstecken, ist übrigens doof: Rechner im Internet werden regelmäßig von Portscans heimgesucht, sodass so ein Versteck schnell auffliegt. Die Häscher erkennen nicht nur, dass ein Port offensteht, sondern finden ebenso schnell heraus, mit welchem Dienst sie es dort zu tun haben [1]. Auf einem SSH-Server, den man auf einem ungewöhnlichen Port sicher wähnte, häufen sich schnell die Anmeldeversuche Unbefugter.

Namenssachen

Damit das Weiterleiten von Ports allein genügt nicht: Die meisten Internetzugänge nutzten dynamische IP-Adressen. Das heißt, dass sich die Adresse gelegentlich oder sogar regelmäßig ändert, unter der der Router und somit auch dahinter angebotene Dienste erreichbar sind. Indem man den Router dazu anweist, nach erfolgreicher Verbindung seine Adresse bei einem Dienst für dynamisches DNS zu hinterlegen, erhält der einen weltweit eindeutigen Namen wie „heise.de“. Dann kann man unter dem registrierten Namen stets die eigene externe IP-Adresse



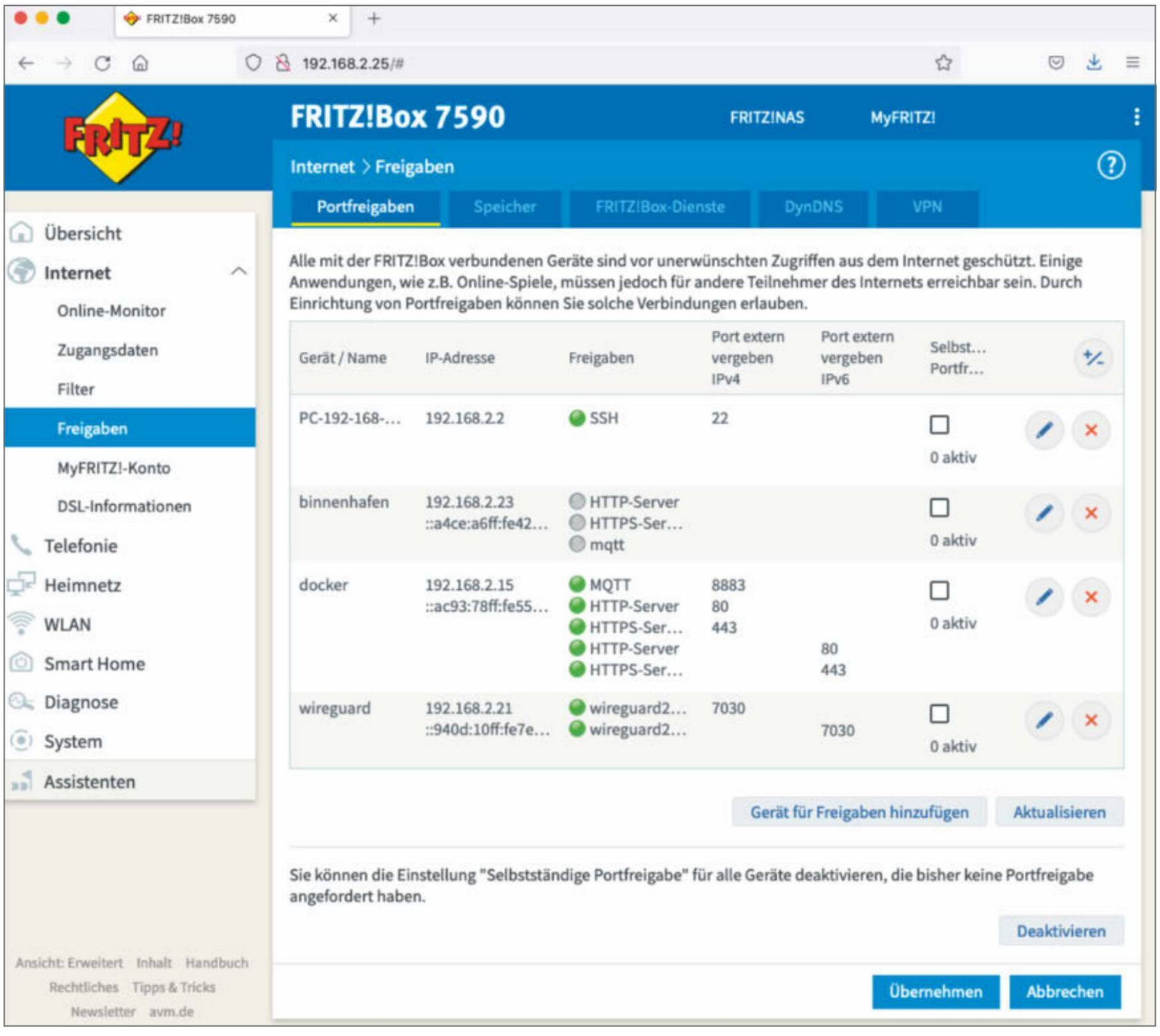
Bei wechselnden IP-Adressen unverzichtbar: ein DynDNS-Dienst, der den Serverdiensten einen Namen gibt. Fritzboxen kennen dafür nicht nur einige vorgefertigte Dienste, sondern lassen sich im Detail konfigurieren.

beziehungsweise dahinter angebotene Dienste erreichen.

Die Namen, die Dienste für dynamisches DNS (DynDNS) anbieten, kann man sich indes nicht einfach ausdenken. Sie sind in der Regel begrenzt auf eine Auswahl vorgegebener Internet-Domains wie „dyndns.org“. Lediglich den Host-Anteil der Adresse wie „icke23“, der „dyndns.org“ vorangestellt wird, kann man sich dabei aussuchen, solange das nicht bereits ein anderer Kunde getan hat. Große Provider lassen obendrein in die Firmware ihrer Router nur wenige Dienste für dynamisches DNS zu – oft solche, die ihr kostenloses Modell auf bezahlte Dienste umgestellt haben.

AVM ist da gnädiger: Fritzboxen kennen viele Anbieter und über die Konfiguration von benutzerdefinierten URLs für das Namensupdate bekommt man so ziemlich jeden DynDNS-Dienst eingerichtet. Auch bietet der Routerhersteller selbst im Rahmen seiner MyFritz-Dienste einen DynDNS-Dienst an. Die Fritzbox eines dort registrierten Nutzers erhält einen eindeutigen Namen wie „rtvw4nftt8qxpudp.myfritz.net“. Die Konfiguration läuft unter „Internet“ in „MyFRITZ!-Konto“. Die Eingabe einer E-Mail-Adresse genügt für die Registrierung. Die Konfigurationsoberfläche Ihrer Fritzbox öffnen Sie dadurch nicht ins Internet.

Wenn Ihnen der Sinn nach schöneren Namen als „icke23.dyndns.org“ oder „rtvw4nftt8qxpudp.myfritz.



Für jeden Dienst, der hinter einer Fritzbox erreichbar sein soll, braucht man passende Portfreigaben – bei vielen Diensten ist das umständlich, aber allemal besser als den Server den zahlreich einprasselnden Einbruchsversuchen auszusetzen.

net“ steht, dann bietet sich das Registrieren einer eigenen Domain an. Eine „.de“-Domain gibt es bei den einschlägigen Domain-Hostern für rund 10 Euro pro Jahr. In der Domain-Verwaltung können Sie üblicherweise solche DynDNS-Namen als CNAME eintragen, also eine Weiterleitung von „www.ihrefirma.de“ auf den DynDNS-Namen „icke23.dyndns.org“ einrichten. So brauchen Sie keinen Domain-Hoster zu suchen, der seinen Kunden DynDNS für die eigene Domain erlaubt.

Nach dem Eintragen eines CNAME für die eigene Domain, also Änderungen an der Konfiguration des Nameservers des Domain-Hosters, brauchen Sie etwas Geduld. Oft dauert es mehrere Stunden, bis sich solche Änderungen im dezentralen DNS-System herumgesprochen haben, also auch der Nameserver des Providers den neuen Namen zur Kenntnis genommen hat.

Weiterleitungen

Die Portfreigaben finden sich bei Fritzboxen unter „Internet“ und dort in „Freigaben“. Das Gerät beziehungsweise die VM, der Container, was auch immer Sie aus dem Internet erreichbar machen wollen, sollte zu diesem Zeitpunkt der Fritzbox bekannt sein; das ist dann der Fall, wenn das Gerät aktiv war und von der Fritzbox eine Adresse erhalten hat. Dann können Sie es bequem auswählen. Andernfalls müssen Sie die IP-Adresse von Hand eingeben, was gerade bei IPv6-Geräten mühsam sein kann.

Bei IPv4 können Sie ein Gerät vollständig dem Internet aussetzen (Exposed Host). Die Fritzbox reicht dann alle eingehenden IPv4-Netzwerkanfragen auf allen Ports an dieses Gerät durch. In der Regel will man das nicht: Wenn Sie gezielt nur die Ports für aktive Dienste öffnen, können Sie entspannt die Gesamtkonfiguration des Geräts vornehmen, etwa einen SSH-Dienst laufen lassen, auch wenn der übers Internet nicht erreichbar sein soll – als Exposed Host wäre jeder Dienst im Internet sichtbar.

Für IPv6 erlaubt es die Fritzbox, mehrere Exposed Hosts einzurichten. Das liegt daran, dass sie bei vielen Internet-Providern auch alle Geräte im lokalen Netz mit einer globalen, also weltweit eindeutigen und erreichbaren IPv6-Adresse versorgt. Das klappt, wenn der Provider der Fritzbox zusätzlich zu ihrer eigenen IPv6-Adresse auch ein IPv6-Präfix aushändigt; ein solches Präfix enthält mehrere IPv6-Netze, aus dem die Fritzbox den Clients Adressen zuteilt. Kein Grund zur Panik: Die Firewall der Fritzbox lässt Pakete an die IPv6-Adressen nicht passieren. Es sei denn, Sie konfigurieren den Client IPv6-seitig als Exposed Host.

Sowohl im Fall von IPv4 als auch von IPv6 ist es zumeist besser, ausgewählte Ports an die Geräte weiterzuleiten, weil man so kaum versehentlich Dienste ins Internet bringen kann; bei einem Exposed Host ist nur eine Frage der Zeit, dass Sie versehentlich nur für den lokalen Einsatz gedachte Dienste öffentlich zugänglich machen. Für eine gezielte Portfreigabe müssen Sie nach der Auswahl des Geräts eben auch auswählen, welchen Netzwerkport Sie erreichbar machen wollen. Beim Aufrufen des Assistenten bietet die Oberfläche die Wahl zwischen MyFritz-Freigabe oder Portfreigabe.

Portfreigabe ist die bessere Option: Damit können Sie auswählen, welcher externe auf welchen internen Port weitergeleitet werden soll. Das kann praktisch sein, wenn Sie mehrere Webserver nach außen freigeben wollen und beide, etwa in einer VM oder einem Container, intern ihre Dienste auf Port 80 anbieten. Den einen können Sie dann auf Port 80 und den anderen auf Port 81 von außen zugänglich machen. Außerdem erlaubt die Portfreigabe auch die Auswahl von Protokollen wie UDP und TCP.

Zwei zusätzliche Gedanken noch zur Freigabe von Hosts mit IPv6-Adresse: Der erste betrifft die Adresse selbst. Achten Sie darauf, dass die Fritzbox deren dauerhafte, meist von der MAC-Adresse des Interfaces abgeleitete IPv6-Adresse in der Konfigurationsoberfläche als „IPv6 Interface-ID“ zeigt – eine über die Privacy-Extenions von IPv6 generierte Adresse eignet sich zur Freigabe nicht, weil sie schließlich nach ein paar Stunden automatisch durch eine andere ersetzt wird.

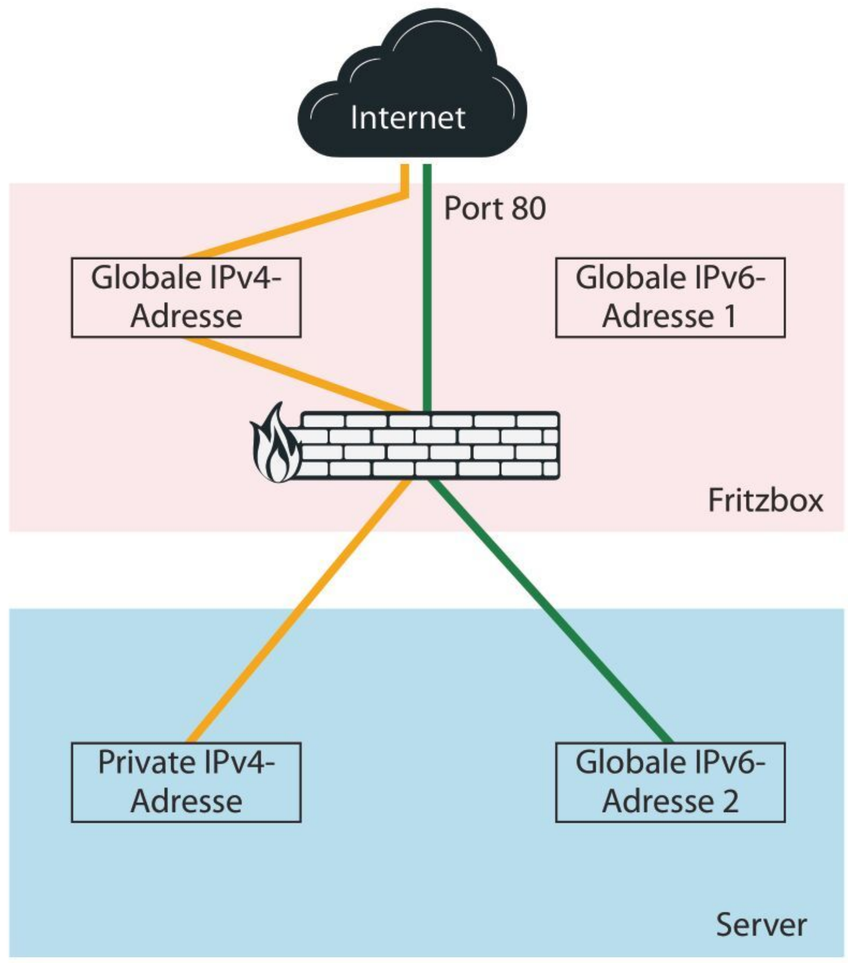
Der zweite Gedanke betrifft die Art und Weise, wie Sie den per IPv6 erreichbaren Dienst im Internet wiederfinden: Die eingangs beschriebenen Funktionen in einer Fritzbox, um per DynDNS die externe IP-Adresse des Routers bekannt zu machen, greifen nur für deren eigene IPv4- und IPv6-Adresse. Bei IPv6 erhalten die Geräte hinter der Fritzbox aber eigene IPv6-Adressen. Das bedeutet zusätzlichen Aufwand: Die Geräte müssen sich selbst mit ihrer eigenen IPv6-Adresse bei einem DynDNS-Dienst registrieren.

Zertifikate

Soweit ist die Fritzbox-Oberfläche hilfreich. Bei einem heute erstrebenswerten Detail kann sie leider nicht helfen, nämlich beim Beschaffen von TLS-Zertifikaten. Alle Dienste, die Sie im Internet freigeben,

Portfreigaben: IPv4 versus IPv6

Mit einer Portfreigabe erlauben Sie einem Router wie der Fritzbox, eingehende Netzwerkverbindungen weiterzuleiten. Die sind an einen Port gerichtet, zum Beispiel an den für unverschlüsselte Webzugriffe verwendeten Port 80. Den öffnen Sie in der Firewall wahlweise für IPv4 und für IPv6. Die Wege, die die Pakete dann nehmen, unterscheiden sich: Bei IPv4 kann die Fritzbox die Anfrage an genau ein Gerät mit privater IP-Adresse im Intranet weiterleiten, denn die Fritzbox hat selbst nur eine globale IPv4-Adresse. Bei vollumfänglicher IPv6-Unterstützung erhält jedes Gerät hinter der Fritzbox eine eigene globale IPv6-Adresse. Dann reicht die Fritzbox die Pakete direkt an die globale IPv6-Adresse des Geräts im Intranet durch. Die IPv6-Adresse der Fritzbox ist nicht beteiligt.



sollten Sie nur transportverschlüsselt zugänglich machen, damit niemand mitlesen kann. Dazu brauchen Sie Zertifikate, die eine Stammzertifizierungsstelle ausgestellt hat, deren Zertifikat Browser und andere Geräte akzeptieren. Bei selbst signierten Zertifikaten würden Sie Nutzer womöglich dazu animieren, Zertifikatswarnungen zu ignorieren – was die besser nicht lernen sollten.

Geeignete Zertifikate liefert die Zertifizierungsstelle Let's Encrypt kostenlos. Um ein solches Zertifikat zu erhalten, kann man auf oftmals in einzelnen Serverdiensten vorhandene Funktionen dafür zurückgreifen. Das hat allerdings den Nachteil, dass der Prozess, den eine Automatik ja für die nur 90 Tage gültigen Zertifikate wiederholen muss, nur für genau diesen einen Dienst funktioniert. Der für die Aktualisierung gebräuchliche Mechanismus von Let's Encrypt (ACME) setzt voraus, dass auf Port 80 ein speziell präparierter Webserver für die Zertifikatsausstellung und -aktualisierung antwortet. Sie können aber immer nur einen Dienst an einem Port lauschen lassen.

Wenn Sie mehrere Dienste mit Zertifikaten ausstatten wollen, lassen Sie am besten hinter Ihrer Fritzbox einen Edge-Router laufen, der alle Anfragen annimmt und zentral die TLS-Abwicklung für alle Dienste übernimmt. Für den richten Sie eine Portfreigabe zumindest für Port 80 und 443 ein (für weitere Dienste auch zusätzliche Ports), sodass er alle an diese Ports gerichteten Anfragen erhält. Anhand des dabei mit übermittelten Servernamens leitet der Edge-Router die Anfrage dann an die eigentlich zuständigen Dienste im lokalen Netz weiter. Den gesamten Verkehr, der für die Zertifikatsbeschaffung und -erneuerung notwendig ist, wickelt er selbst ab.

Als Edge-Router empfiehlt sich die Open-Source-Anwendung Traefik [2]. Wenn Sie bereits mit Containern hantieren, wird Ihnen Traefik möglicherweise vertraut sein. Alle anderen sollten sich etwas Zeit nehmen: Bis man Traefik soweit durchschaut hat, dass

Literatur

[1] Jan Mahn, Merlin Schumacher, **c't deckt auf**, Dateien, IoT und Industrieanlagen ungeschützt im Netz, c't 23/2020, S. 14

[2] Jan Mahn, **HTTP-Einweiser**, Eingehenden HTTP-Verkehr mit Traefik routen, c't 17/2019, S. 158

Traefik-Beispiel für Stand-alone-Betrieb

ct.de/wsr3

man es im Blindflug einsetzen kann, vergeht schon einige Zeit. Besonders gilt das, wenn Sie es manuell betreiben und nicht den Komfort einer Container-Umgebung nutzen, in der Traefik idealerweise neue Container namentlich kennt und sogar im DNS registrieren kann.

Aber es geht auch zu Fuß: Das in Go geschriebene Traefik gibt es auf der Release-Page in GitHub für diverse Plattformen als ausführbare Datei (siehe ct.de/wsr3). Diese Programmdatei sowie eine Datei mit einer Grundkonfiguration genügen, um Traefik auf einem beliebigen Host hinter Ihrem Router in Betrieb zu nehmen. Für jeden Dienst, für den es Vermittlung und Zertifikatsbeschaffung erledigen soll, müssen Sie dann eine weitere Beschreibung in einer Konfigurationsdatei ablegen. Ein Beispiel für einen einfachen Webserver stellen wir zum Download bereit.

Sicheres Ende

Dienste hinter einem Router zu betreiben, ist kein Hexenwerk. Wichtig ist, dass Sie stets sicherstellen, nicht die falschen Dienste ins Netz zu bringen und diese akribisch zu konfigurieren. Diese sollten nur nach einer Anmeldung zugänglich sein. Sie sollten nur starke Kennwörter nutzen und eventuelle Mitbenutzer anhalten, es ebenso zu halten. Dass Sie regelmäßig – am besten automatisiert – Sicherheitsupdates einspielen sollten, sei nur der Vollständigkeit halber erwähnt. Falls Sie sehr hohe Ansprüche an die Sicherheit haben, wäre auch ein VPN-Zugang in Ihr Netz eine Option, um darüber interne Dienste erreichbar zu machen – das allerdings erfordert zumindest eine Portfreigabe für den VPN-Verkehr, wenn der Router nicht selbst den VPN-Server gibt. (ps) c't

Es gibt 10 Arten von Menschen. iX-Leser und die anderen.

Jetzt Mini-Abo testen:
3 Hefte + Bluetooth-Tastatur nur 16,50 €
www.iX.de/testen

www.iX.de/testen | 49 (0)541 800 09 120 | leserservice@heise.de

iX MAGAZIN FÜR PROFESSIONELLE INFORMATIONSTECHNIK



Einstiegsfreundlicher Heimserver

Unraid setzt auf Flexibilität und maximalen Speicherplatz im NAS und schützt mit Parität vor Datenverlust. Wer das System ausreizen will, braucht einen SSD-Cache. Die Weboberfläche nimmt Nutzern viel Arbeit ab.

Von **Niklas Dierking**

Unraid erfüllt nicht nur die typischen Aufgaben eines NAS wie Speicherplatz für Backups oder das Fotoarchiv zentral im Netzwerk bereitzustellen. Das System dient auch als VM-Server und Docker-Host und öffnet Nutzern so viele Türen. Für ein (Heim-)Serverbetriebssystem setzt es dabei wenig Vorkenntnisse voraus. Die meisten Handgriffe erledigen Nutzer in der intuitiven Weboberfläche. Kniffe auf der Kommandozeile werden in der Regel erst bei Spezialanforderungen nötig. Unraid baut zwar auf Linux und diversen Open-Source-Komponenten auf, ist selbst aber nicht vollständig quelloffen.

Wie viele Speichergeräte (Festplatten, exklusive Boot-Stick und Cache) Nutzer in ihr Storage-Array einbinden dürfen, knüpft Hersteller Limetech an eine einmalig zu erwerbende Lizenz, siehe Tabelle. Eine kostenlose Probelizenz läuft 30 Tage. Andere NAS-Betriebssysteme wie das BSD-lizensierte TrueNAS oder Open Media Vault, das auf Debian aufbaut, haben keine künstlichen Beschränkungen.

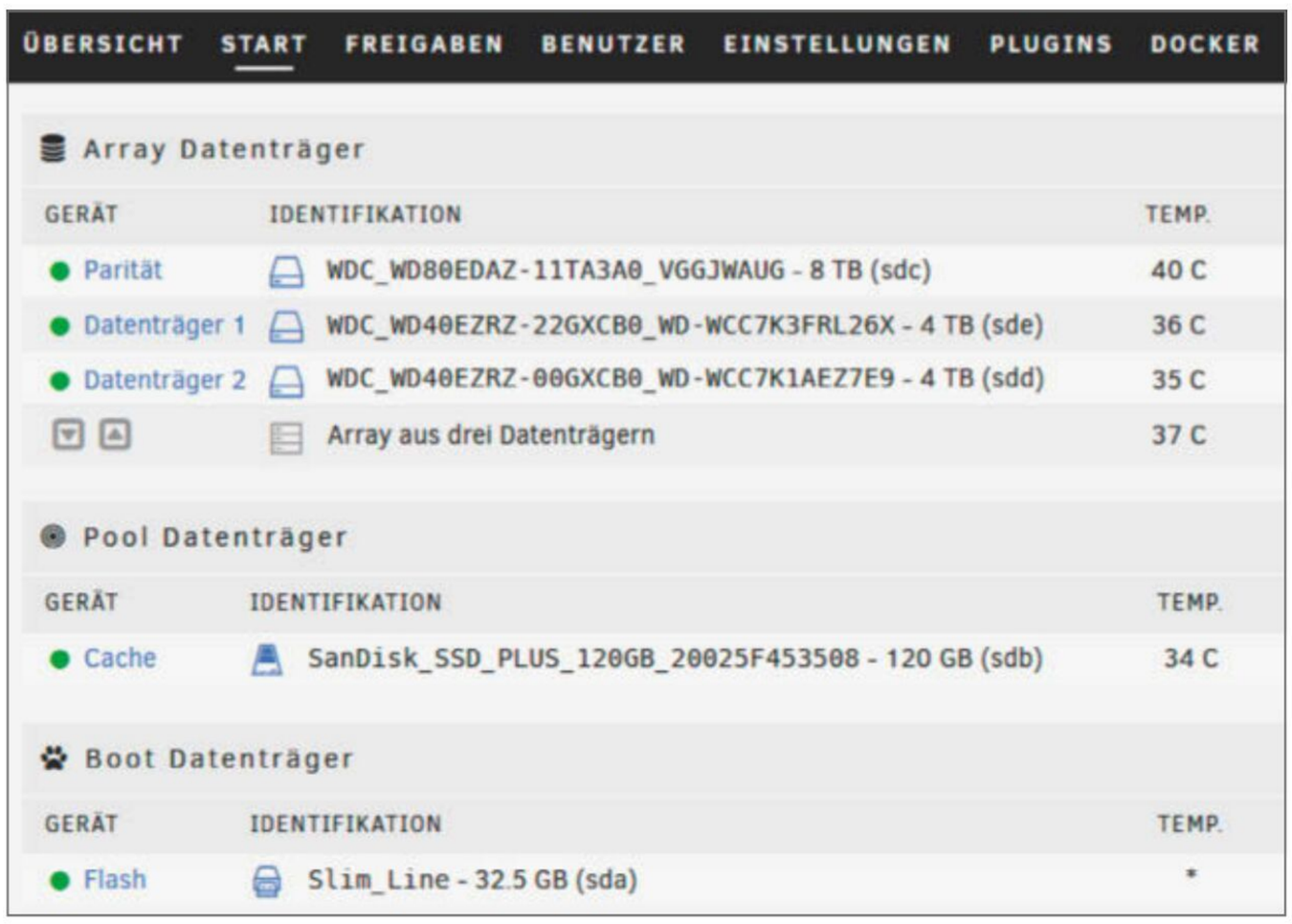
Nutzer installieren Unraid nicht auf einer der verbauten Festplatten, sondern auf einem USB-Stick. Das Betriebssystem wird mitsamt Konfigurationsdateien beim Systemstart einmalig in den Arbeitsspeicher

geladen und verrichtet von dort seine Arbeit. Weil Limetech die erworbene Lizenz an die GUID des USB-Sticks bindet, dient er gleichermaßen als physischer Lizenzschlüssel. Die Lizenz lässt sich via Weboberfläche auf einen anderen Stick transferieren. Danach können Nutzer den Stick nur alle 12 Monate wechseln. Wer die Lizenz öfter transferieren muss, ist auf den Support angewiesen.

Linux-Basis: Slackware

Unraid baut auf einer angepassten Version der Linux-Distribution Slackware auf. Limetech stutzt Slackware für Unraid stark zurecht und liefert nur die nötigsten Pakete aus. Nutzer finden an Bord zwar zentrale Utilities wie htop, nano und curl, die Liste an vorinstallierten Paketen in /var/logs/packages bleibt jedoch überschaubar. Das abgespeckte System sitzt wiederum auf einem Custom-Kernel, aktuell 5.10.28-Unraid. Den reichern die Entwickler mit Treibern an, die nicht standardmäßig im Linux-Kernel (out-of-tree) enthalten sind, um beispielsweise bestimmte Netzwerkadapter lauffähig zu machen.

Slackware kommt ohne Annehmlichkeiten wie Paketmanager, die Nutzer in modernen Linux-Distributionen erwarten. Die Installation von Slackware-Paketen ist in Unraid zwar möglich, im Design des Systems allerdings nicht unbedingt vorgesehen, weil alle Dateien auf dem USB-Stick außerhalb von /boot flüchtig sind und einen Neustart nicht überleben. Weil Abhängigkeiten nicht automatisch aufgelöst werden, besteht die Gefahr, dass Nutzer ein instabiles System erzeugen. Die Entwickler empfehlen, Anwendungen grundsätzlich in Docker-Container oder virtuelle Maschinen auszulagern. Wer trotzdem Software auf dem Basissystem einrichten möchte, bedient sich des Plug-in-Managers in der Weboberfläche oder legt Pakete als *txz-Archive im Verzeichnis /boot/config/extra ab. Sie werden dann bei jedem Systemstart automatisch installiert.



Die Weboberfläche informiert über Systemauslastung, laufende Container und VMs sowie (hier) den Status des Array.

keinRAID

Einen Festplattenverbund bezeichnet Unraid je nach Einsatzgebiet als Array oder Cache-Pool. Bei der Einrichtung des Arrays als Dateiablage legt der Nutzer fest, welche Festplatten als Datenspeicher dienen. Um das Array gegen einen Laufwerksausfall abzusichern, bestimmt er zusätzlich ein oder zwei Paritätslaufwerke. Mit einem Paritätslaufwerk kann Unraid den Ausfall eines beliebigen Datenlaufwerks ausgleichen. Bei zwei Paritätslaufwerken verkraftet das Array zwei gleichzeitige Ausfälle.

Das Dateisystem (XFS oder BTRFS) umspannt nicht mehrere Festplatten. Dateien werden ungestückelt auf eine Festplatte im Array geschrieben. Das System spiegelt die Dateien auch nicht zwischen den Festplatten wie bei einem RAID 1. Um trotzdem den Ausfall eines Laufwerks zu kompensieren, berechnet Unraid den Paritätsstatus des Arrays auf Blockebene. Dabei werden bestimmte Bits der Datenlaufwerke addiert und bei ungeradem Ergebnis ein Paritätsbit auf das Paritätslaufwerk geschrieben. Ersetzen Nutzer das defekte Laufwerk, „rekonstruiert“ das System die Daten, indem es erneut eine gerade Parität erzeugt (siehe Kasten zu Parität). Das spart gegenüber einem RAID 1 Speicherplatz, weil ein Paritätslaufwerk mehrere Platten absichert und nicht für jede Platte eine weitere installiert werden muss. Das Array kann außerdem leicht erweitert und Festplatten unterschiedlicher Kapazitäten kombiniert werden. Das größte Datenlaufwerk darf allerdings nicht größer als die Paritätsplatte(n) sein.

Der Preis, den Nutzer für diese Flexibilität zahlen, ist eingeschränkte Performance. Bei jedem Dateitransfer auf das Array wird gleichzeitig auf einem Datenlaufwerk und dem Paritätslaufwerk geschrie-

So schützt Parität vor Datenverlust

Um Datenverlust zu vermeiden, wahrt das System eine gerade Parität. In diesem Beispiel enthält das Array zwei Datenlaufwerke und ein Paritätslaufwerk. Kopiert ein Nutzer über eine Freigabe Daten auf das Laufwerk B, addiert die Software beispielsweise das elfte Bit von Laufwerk A und Laufwerk B. Kommt eine ungerade Zahl heraus, schreibt die Software ein Paritätsbit auf das Paritätslaufwerk, das den elften Bits von Laufwerk A und B zugeordnet ist.

Fällt Laufwerk B aus, gleicht die Software die Binärdaten des verbliebenen Laufwerks A mit den Paritätsdaten ab und berechnet, welche Daten fehlen. Nutzer können das defekte Laufwerk B ersetzen, ohne Daten zu verlieren. Auch bei mehr als zwei Datenlaufwerken bleibt das Prinzip identisch.

Laufwerk A	Laufwerk B	Parität
0	0	0
0	1	1
1	0	1
1	1	0

Laufwerk A	Laufwerk B	Parität	Rechenoperationen zur Wiederherstellung
0	-	0	0 (A) + 0 (P) = 0 (B)
0	-	1	0 (A) + 1 (P) = 1 (B)
1	-	1	1 (A) + 1 (P) = 0 (B)
1	-	0	1 (A) + 0 (P) = 1 (B)

ben und gelesen. Die langsamste beteiligte Festplatte begrenzt den Datendurchsatz. Diesen Nachteil kompensiert Unraid mittels Cache, der am sinnvollsten mit SSD-Laufwerken bestückt wird. Fügen Nutzer mehr als eine SSD zum Cache hinzu, fasst Unraid diese als Cache-Pool mittels BTRFS-RAID 1 zusammen, um Datenverlust vorzubeugen. Ein Schreib-Cache empfiehlt sich damit erst ab zwei SSDs. Richten Nutzer eine Netzwerkfreigabe ein, konfigurieren sie, ob neue Dateien zunächst im Cache landen und zu einer festgelegten Uhrzeit auf das Array geschoben werden, im Cache verbleiben oder ohne Umwege auf das langsamere Array geschrieben werden. Die Schreibgeschwindigkeit eines paritätsgeschützten Festplatten-Arrays pendelt sich in unserem Test nach kurzer Zeit bei langsamen 45 bis 60 MByte/s ein (Western Digital WD40EZRZ mit 4 TByte, Intel Pentium J5040 mit 2 GHz). Das überrascht, weil die beteiligte Festplatte technisch schneller schreiben könnte. Kommt flotter NVMe-Speicher als Schreib-Cache zum Einsatz, stellt ein Gigabit-Netzwerk den limitierenden Faktor dar.

Virtualisierung und Container

Abseits vom Netzspeicher fungiert Unraid als Hypervisor und Docker-Host. Als Virtualisierungstechnik kommt KVM zum Einsatz, libvirt verwaltet die virtuellen Maschinen. Unraid bringt Annehmlichkeiten wie

Unraid 6.9

NAS/-Heimserverbetriebssystem	
Hersteller	Limetech, www.unraid.net
Systemvoraussetzungen	USB-Stick mit GUID (min. 1 GByte, max 32 GByte), 64-Bit-Prozessor[1] mit mind. 1,0 GHz, 2 GByte Hauptspeicher, mind. 2 Festplatten für optionale Parität
Preis (einmalig)	59 US-$ (Standard, max. 6 Speichergeräte[2]), 89 US-$ (Plus, max. 12 Speichergeräte[2]), 129 US-$ (Pro, max. 30 Speichergeräte[2])

[1] mit Intel VT-x / AMD-V für Virtualisierung, IOMMU für PCIe-Passthrough, [2] inkl. Paritätslaufwerke

VNC, USB- und PCIe-Passthrough von Haus aus mit. Bei Bedarf mounten Nutzer direkt bestehende Netzwerkfreigaben und weisen VMs bestimmte CPU-Kerne zu. Ist ein Cache(-Pool) eingerichtet, legt Unraid die virtuellen Disk-Images für verbesserte I/O-Performance standardmäßig auf dem SSD-Speicher ab.

Durch die Installation des Plug-ins Community Applications bekommen Nutzer Zugang zum App-Store, der eine Fülle von Plug-ins und Anwendungen aus der Unraid-Community bereithält. Bei einem Großteil der Apps handelt es sich um Docker-Container. Viele Container-Templates beinhalten bereits simple Anpassungen für Unraid. Der Reiter „Docker“ in der Weboberfläche listet laufende und gestoppte Container sowie deren Volumes und TCP-Ports übersichtlich auf. Veraltete Images ersetzen Nutzer mit einem Mausklick auf „Apply Update“ bequem durch neuere.

Durch Plug-ins, Skripte, VMs und Container erweitern Nutzer den Funktionsumfang von Unraid und passen das System mit wenigen Handgriffen den eigenen Anforderungen an. Wem die Kombination aus paritätsgeschütztem Array und Cache(-Pool) nicht taugt, kann beispielsweise mit etwas Bastelarbeit das ZFS-Dateisystem installieren. Auch deutsche Sprachpakete gibt es im App-Store. Nutzer können sich über diverse Kanäle über den Zustand des Systems informieren lassen. Remote-Zugriff gibt es entweder via VPN oder mit dem offiziellen Plug-in „My Servers“. Um das (Heim-)Serverbetriebssystem besteht eine aktive Community, die Impulse für die weitere Entwicklung setzt. Nachholbedarf gibt es bei der Dokumentation, die für die aktuelle Version Lücken aufweist.

Fazit

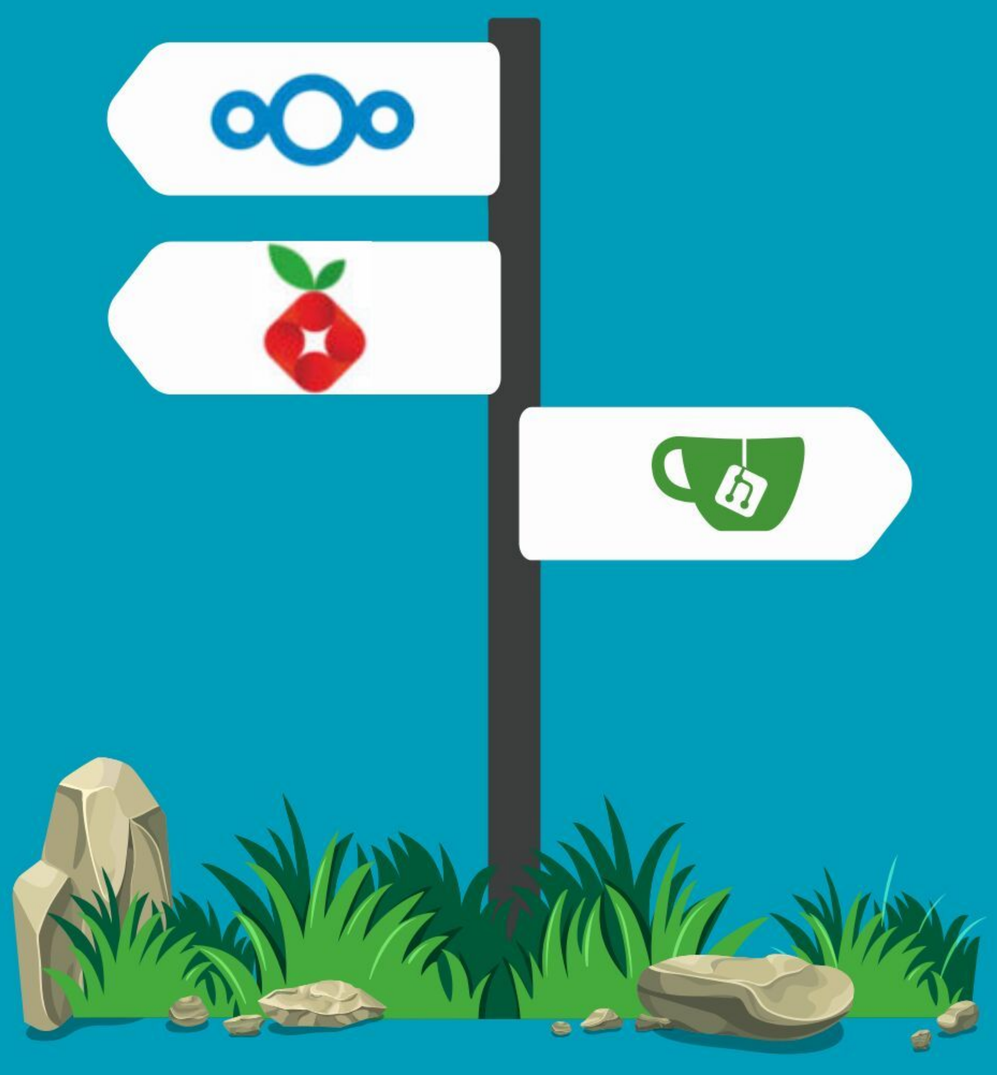
Mit Unraid klicken Nutzer sich ihren Heimserver zusammen, der nicht nur Dateien verwahrt und schützt, sondern bei Bedarf auch Webanwendungen hostet, Medien ausspielt und als VPN-Endpunkt ins Heimnetz dient. Diese Aufgaben erfüllen moderne Linux-Distributionen ebenfalls, setzen aber mehr Konfigurationsaufwand voraus. KVM/QEMU gehört zu den Linux-Bordmitteln, und Management-Software für VMs und Container gibt es reichlich. Profis, die Enterprise-Features wie ZFS und Ceph benötigen oder mehrere Speicher-/Virtualisierungscluster vernetzen wollen, greifen eher zu TrueNAS oder Proxmox. Für Heimanwender ist Unraid interessant, weil man es leicht erweitern kann und sich schnell eingearbeitet hat. (ndi) c't



Wegweiser im Heimnetz

Mit Containern laufen viele Anwendungen schnell auf dem eigenen Server. Da kann man schon mal den Überblick verlieren. Das Dashboard „Homer“ macht die Lesezeichensammlung überflüssig.

Von **Niklas Dierking**



Dashboard Homer	72
CI/CD mit GitHub Actions	74
Online-Dokumentationen mit MkDocs	80
Statusseiten für IT-Infrastruktur	86
Passwortmanager in Eigenregie hosten	90
Mitgliederversammlungen virtuell	98

Wie war noch mal die IP-Adresse des Raspi, auf dem die Nextcloud läuft? Auf welchem Port lauscht doch gleich der Webserver der Wiki-Anwendung? Wer Gefallen daran gefunden hat, diverse Webdienste in Eigenregie zu hosten, sieht sich früher oder später mit diesen Fragen konfrontiert. Homer verlinkt alle Anwendungen in einem Dashboard und hilft so dabei, den Durchblick zu behalten.

Beim Homer-Dashboard handelt es sich um eine simple, statische Website. Am schnellsten nimmt man das Dashboard mittels Docker-Container in Betrieb. Dafür gibt es im Docker Hub ein gut gepflegtes Container-Image (b4bz/homer). Im GitHub-Repository des Projektes stehen jedoch auch alle benötigten Dateien als Archiv zur Verfügung, die sich mit einem beliebigen Webserver ausliefern lassen.

Nutzer befüllen und konfigurieren das Dashboard über eine zentrale YAML-Datei. Das Container-Image und das Download-Archiv bei GitHub enthalten beide eine YAML-Datei mit einem Grundgerüst für eine Demo-Seite, sodass weitere Anpassungen mit dem Texteditor leicht von der Hand gehen. Um die Änderungen zu sehen, reicht es aus, die Seite im Browser neu zu laden.

Beispieldateien Font-Awesome

ct.de/w7zz

Homer	
Webbasiertes Dashboard	
Hersteller	Bastien Wirtz, github.com/bastienwirtz/homer
Systemanforderungen	Docker (Windows, macOS, Linux), Webserver (Apache, Nginx, Python http.server)
Preis	**kostenlos, Open Source (Apache 2.0 Lizenz)**

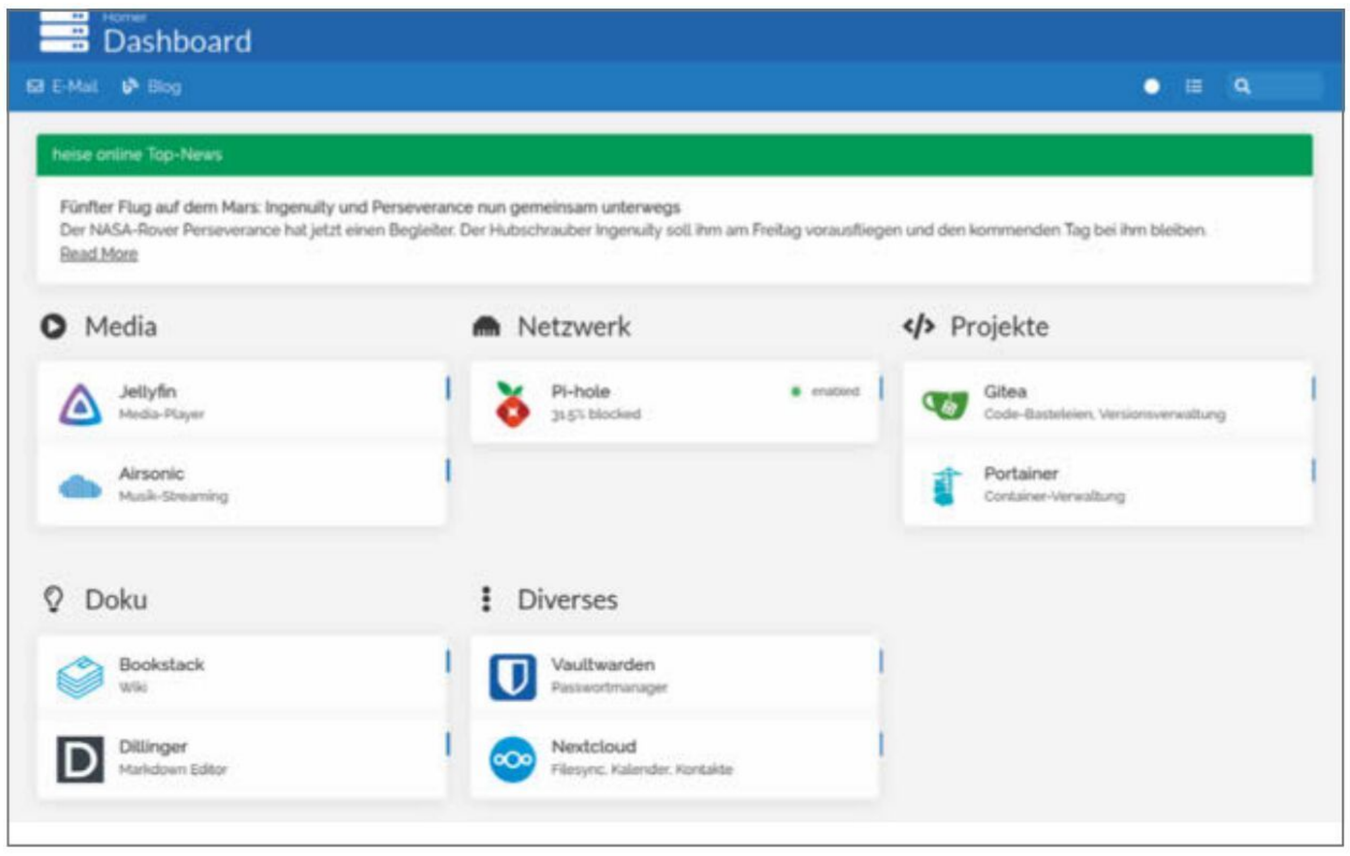
Im Einsatz

Wenn Sie Homer einsetzen wollen, müssen Sie sich also ans Bearbeiten der YAML-Datei machen. Mit eigenen Einträgen lassen sich Apps hinzufügen und in Kategorien einsortieren. Der dafür relevante Abschnitt in der Homer-Konfiguration heißt `services`. Über das erste Array fassen Sie die Dienste zu Gruppen zusammen, beispielsweise Medienanwendungen zu einer Gruppe mit dem Namen „Media“. Über ct.de/w7zz finden Sie eine YAML-Datei als Beispiel.

Das Homer-Dashboard skaliert flexibel je nach Bildschirmgröße und macht auch auf Smartphones und Tablets eine gute Figur. Die Kopfzeile beinhaltet neben zusätzlichen Shortcuts weitere Bedienelemente. Über eine Schaltfläche lässt sich hier in eine Listenansicht wechseln. Sollte das Dashboard irgendwann überquellen, hilft die Suchfunktion, die auch per Tastenkürzel startet. Ein Button zum Umschalten zwischen einem hellen und einem dunklen Theme sowie eine Option, die sich nach dem Farbschema der Systemeinstellungen richtet, stehen ebenfalls zur Verfügung.

Wer möchte, ändert die Farben des Dashboards oder verwendet ein eigenes Wallpaper. Mit etwas Bastelarbeit sind auch Spielereien wie ein Videohintergrund möglich. Tiefergehende Anpassungen nehmen Nutzer mit eigenen CSS-Stilen vor. Via Node-Red-Flow kann Homer einen RSS-Feed abrufen und so eine dynamische Nachricht anzeigen. Bei ausgewählten Diensten, wie beispielsweise Pi-hole, lädt Homer Statistiken des Dienstes über das Pi-hole-API und bindet diese ein. Andere Dashboards wie Heimdall sind bereits weiter und zeigen bei deutlich mehr Anwendungen zusätzliche Informationen an. Der Homer-Entwickler will zukünftig weitere Dienste integrieren, dabei jedoch den simplen und leichtgewichtigen Charakter des Dashboards beibehalten.

Wenn Ihnen noch Projekte fehlen, die Sie im Homer-Dashboard anzeigen können, finden Sie auf den folgenden Seiten einige Projekte für Cloudserver oder die eigene Hardware. (ndi) c't



Mit Homer bauen Sie ein Dashboard, damit Sie Ihre selbstbetriebenen Dienste schnell wiederfinden.



Bild: Thorsten Hübner

CI/CD mit GitHub Actions

Mit Actions nimmt GitHub Entwicklern nervige Fleißarbeit ab. Das Versionieren, Testen, Paketieren und Veröffentlichen von Software passiert damit auf einer einzigen Plattform.

Von **Merlin Schumacher**

Ein automatisierter Entwicklungs-Workflow, der einem die Pflichtaufgaben abnimmt, ist nicht nur ein Segen, sondern gehört heutzutage zum guten Ton in der Softwareentwicklung. Dienste wie Travis, CircleCI oder GitLab bieten solche CI/CD-Lösungen (Continuous Integration/Continuous Delivery) in der Cloud schon lange an. Mit GitHub Actions kommt nun eine mächtige in die Plattform integrierte CI/CD-Lösung hinzu.

Workflows werden in Actions durch YAML-Dateien definiert. Diese Dateien liegen zusammen mit dem Code im Repository. Die erste Version von Actions

verwendete die Hashicorp Configuration Language (HCL), mit Version 2 hat GitHub zu YAML gewechselt. Deshalb finden sich allerhand Tipps im Netz, die HCL verwenden. All diese Hinweise sind überholt. Auch die Actions für Version 1 funktionieren nicht mehr. Actions, die mit Version 2 funktionieren, finden Sie zuhauf im GitHub-Marketplace.

GitHub stellt für die Ausführung der Action-Workflows, also der vom Entwickler festgelegten Abläufe, virtuelle Maschinen, sogenannte Runner, bereit. Windows- und Linux-Runner laufen in der Azure-Cloud des Mutterkonzerns Microsoft. macOS-Runner stellt der Anbieter MacStadium bereit. Unabhängig vom Betriebssystem ist jeder Runner mit zwei CPUs, 7 GByte Arbeitsspeicher und 14 GByte SSD-Speicherplatz ausgestattet. Das dürfte auch für aufwendige Projekte reichen. Sollte es doch knapp werden, kann man die Actions-Aufgaben auch auf eigenen Runnern starten.

Wer keinen kostenpflichtigen GitHub-Tarif gebucht hat, darf bis zu 2000 Minuten rechnen lassen – eine Minute unter macOS zählt dabei für 10 Minuten. Letzteres ist wohl den höheren Hosting-Kosten von macOS geschuldet, da Apple keinerlei Serverhardware mehr liefert. Laufen die Jobs länger, bricht GitHub sie ab. Wer zahlt, erhält größere Kontingente. Eine Übersicht finden Sie in der Tabelle oben.

Viele Programme und Dienste sind auf den Runnern bereits installiert. Sie können diese direkt innerhalb eines Jobs starten. Welche Programme in welchen Versionen auf den verschiedenen Plattformen verfügbar sind, listet GitHub auf einer sehr ausführlichen Dokumentationsseite auf. Diese Seite und alle weiteren Links finden Sie über ct.de/wbbc. Wenn in einem Runner Software fehlt, können Sie fehlende Abhängigkeiten nachinstallieren.

GitHub-Tarife und Action-Jobs

Tarif	Speicherplatz	Minuten pro Monat
Free	500 MByte	2000
Pro	1 GByte	3000
Team	2 GByte	3000
Enterprise	50 GByte	50000

Der erste Workflow

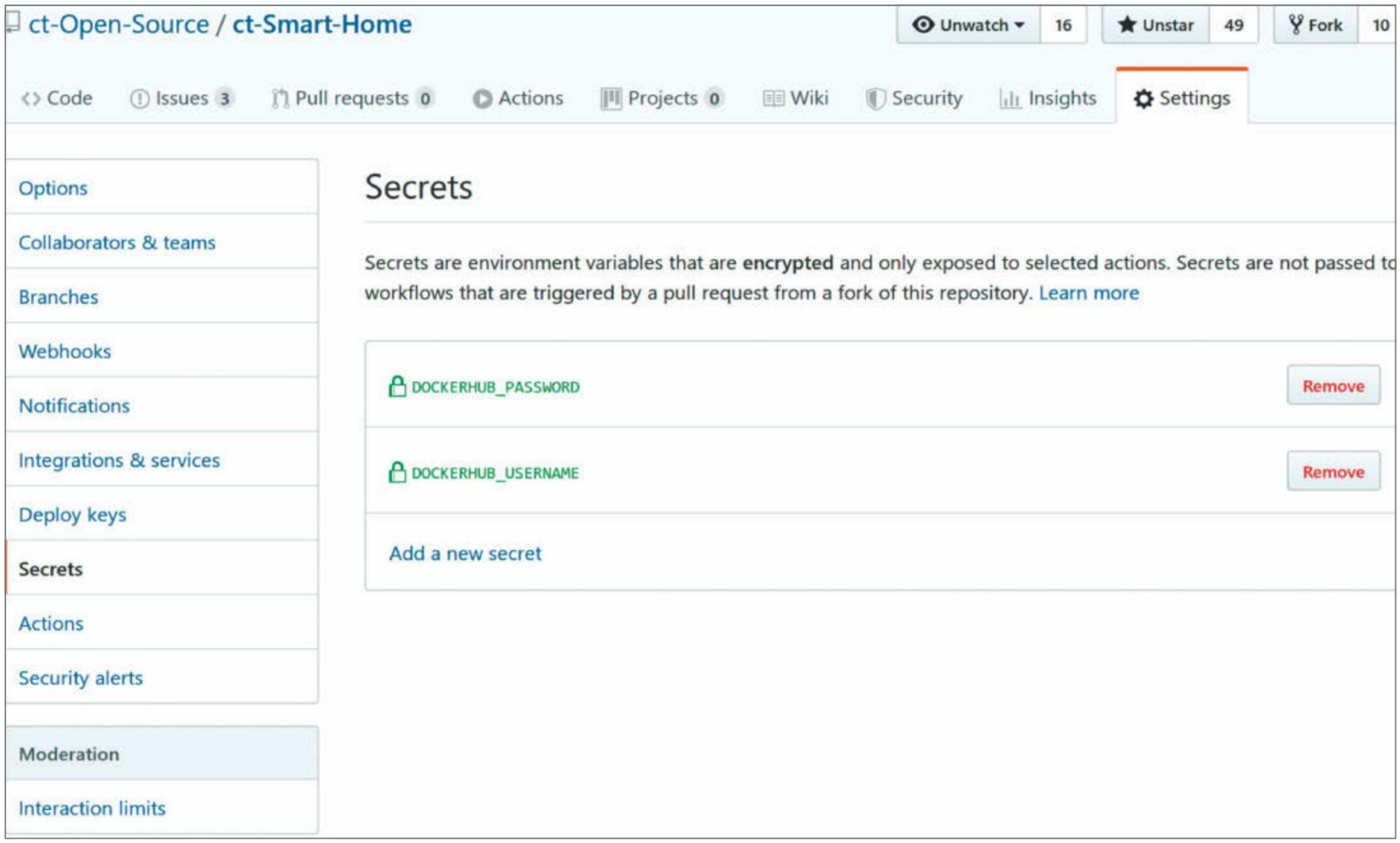
Um Ihren ersten Workflow zu erzeugen, klicken Sie in Ihrem Repository auf den Reiter „Actions“. Dort bekommen Sie allerhand Vorschläge für Workflows. Vermutlich ist bereits ein passender Vorschlag für Ihr Projekt dabei. Klicken Sie für den Nachbau der folgenden Schritte rechts oben auf „Set up a workflow yourself“, um einen simplen Beispiel-Workflow zu erzeugen.

Nun öffnet sich ein Editor mit der automatisch erzeugten Datei `main.yml`, die im Verzeichnis `.github/`

```yaml
name: CI
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v1
    - name: Run a one-line script
      run: echo Hello, world!
    - name: Run a multi-line script
      run: |

        echo Add other actions to build,
        echo test, and deploy your project.
```

GitHubs Einstiegsbeispiel für Actions sagt „Hallo, Welt!“.



Mittels Secrets können Sie Variablen und Zugriffsschlüssel, die im Code nichts verloren haben, an Workflows übergeben.

`workflows` liegt. Der Editor besitzt eine integrierte Autovervollständigung und hebt Fehler hervor. In `.github/workflows` liegen alle Beschreibungsdateien für die Actions-Workflows. Sie können dort beliebig viele Workflows hinterlegen. Die `main.yml` ist nicht leer. Sie enthält das Hello-World-Beispiel, das Sie im Kasten oben sehen können.

Das Schlüsselwort `name` definiert den im GitHub-Interface für den Workflow angezeigten Namen. Hier sollte man eindeutige Namen vergeben, um bei größeren Projekten den Überblick zu behalten. `on` bestimmt, was die Ausführung des Jobs anstößt. Hier ist das ein beliebiger Push in irgendeinen Zweig des Repositorys. Workflows können aber auch auf Pull-Requests, Issues, Forks und zahlreiche weitere Ereignisse reagieren. Die Ausführung zu einer bestimmten Zeit ist ebenfalls möglich. Eine komplette Übersicht findet sich in der Actions-Dokumentation (siehe ct.de/wbbc).

`jobs` definiert einen oder mehrere Aufgaben, die Teil des Workflows sind. Im Beispiel gibt es nur einen Job namens `build`. Jeder Job muss auf einer Betriebssystem-Plattform laufen (`runs-on`). Hier ist das die neueste Ubuntu-Version.

Ein Job umfasst wiederum mehrere Schritte (`steps`). Die Schritte können entweder durch `run` beliebige Kommandos innerhalb der Runner-Umgebung ausführen oder externe Actions starten, die `uses` definiert. Einigen Actions müssen Sie Parameter übergeben. Welche das sind, erklären deren Marketplace- oder GitHub-Seiten. Die Actions sind nicht nur auf Dienste von GitHub beschränkt. Sie finden Actions von anderen Entwicklern und Diensten in GitHubs Marketplace. Actions sind eigentlich nur

Nodejs-Module oder Docker-Container, die innerhalb des Jobs laufen. So können Sie sich bei fehlgeschlagenen Builds per Chat benachrichtigen lassen, vor dem Veröffentlichen noch mal den Code formatieren oder den Code per Druck auf einen Bluetooth-Knopf übersetzen lassen.

Der erste Schritt im Beispiel ist hier das Klonen des Repositorys. Dafür kommt die von GitHub bereitgestellte Action `actions/checkout@v1` zum Einsatz. Sie klont immer den Zustand des Repositorys, bei dem der Job ausgelöst wurde. So ist sichergestellt, dass ein Commit ein paar Minuten später nicht Ihr Testergebnis verändert. Fehlt der Aufruf von `checkout`, steht der Code innerhalb des Jobs nicht bereit.

Anschließend kommen zwei `run`-Aufrufe. Der erste gibt nur „Hello, world!“ aus. Auch hier legt `name` den Titel des Vorgangs fest. Der zweite Schritt ist etwas spannender: Wer mehrere Befehle innerhalb eines Schritts verketten muss, kann dies mit dem Voranstellen des Pipe-Symbols (|) tun. Die Befehle landen dann jeweils in einer Zeile.

Um den Workflow zu speichern, klicken Sie oben rechts auf „Start commit“ und geben, wenn Sie möchten, eine Commit-Beschreibung ein. Ein Klick auf „Commit new file“ speichert die Datei in Ihrem Repository. Die Workflows landen immer im Master-Zweig, auch wenn Sie Jobs auslösen, die für andere Zweige gedacht sind.

In den Workflows können Sie verschiedene Variablen nutzen, um deren Ablauf zu beeinflussen. So lassen sich Zugriffsschlüssel, Umgebungsvariablen oder Softwareversionen flexibel einbinden.

Einige Variablen werden von der Actions-Umgebung befüllt, andere können Sie mittels sogenannter Secrets für das Repository selbst definieren. Dabei muss man unterscheiden zwischen Umgebungsvariablen innerhalb des Jobs und Kontextvariablen, die vom Workflow bestimmt werden und den Aufbau des Jobs beeinflussen.

Umgebungsvariablen können Sie in `run`-Kommandos wie Shell-Variablen einsetzen, beispielsweise `$USER`. Eine Umgebungsvariable wie diese legen Sie mittels des Parameters `env` für einen Schritt fest:

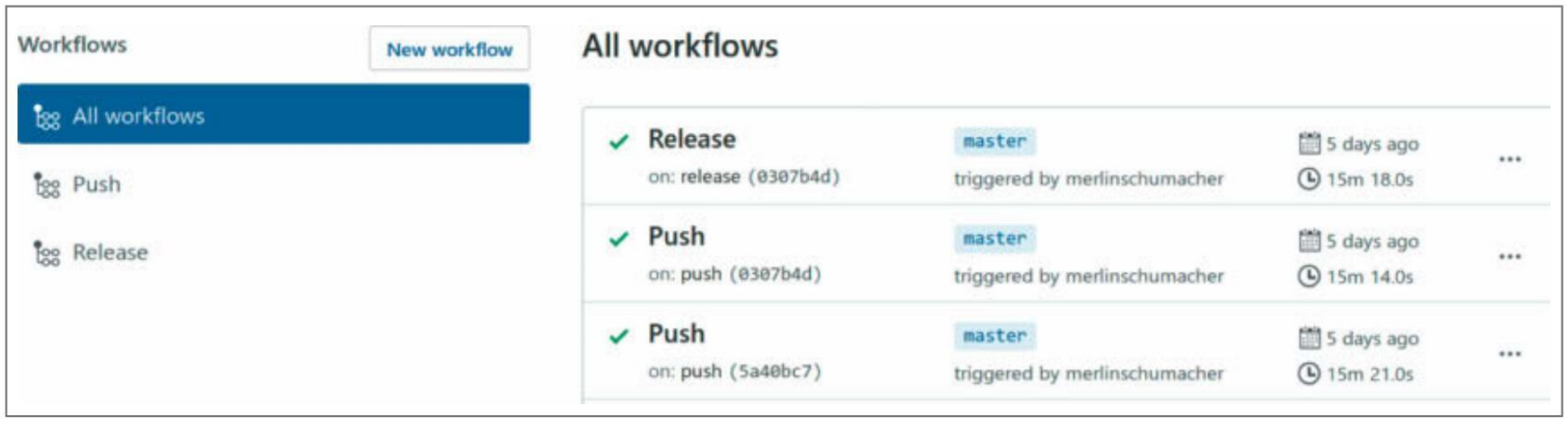
```
- name: Say hello
  run: echo Hello, $USER
  env:
    USER:  cttest
```

Die Variable steht dann innerhalb der Kommandozeilenumgebung des Jobs als Umgebungsvariable zur Verfügung oder `echo` gibt ihren Wert aus. Actions zeigt deren Wert innerhalb der Logs aus Sicherheitsgründen aber nicht an.

Kontextvariablen können Sie innerhalb der Workflow-Konfiguration einsetzen. Diese beginnen mit einem Dollar-Zeichen und werden in zwei geschweifte Klammern eingefasst. GitHub stellt diverse Kontextvariablen bereit, die zum Beispiel Informationen zum Repository und auch zum aktuellen Job liefern. Die Dokumentation aller vorbelegten Variablen finden Sie über ct.de/wbbc.

Für eigene Kontextvariablen kommen Secrets zum Einsatz. Das sind Variablen, die innerhalb von Actions verfügbar sein müssen, aber nicht Teil des für Dritte einsehbaren Codes auf GitHub sein dürfen. Festlegen können Sie die Secrets innerhalb eines Repositorys unter Settings/Secrets. Dort hinterlegen Sie zum Beispiel Zugriffsschlüssel für Paketregistrys oder Datenbanken.

```
-name: Greet ${{secrets.USER}}
 run: echo Hello, ${{secrets.USER}}
```



Ein Push und ein Release lösen unterschiedliche Actions-Workflows aus.

```yaml
strategy:
  matrix:
    os: [ubuntu-latest, windows-latest, macOS-latest]
    node: [8, 10, 12]
runs-on: ${{ matrix.os }}
steps:
  - uses: actions/checkout@v1
  - name: Setup node ${{ matrix.node }}
    uses: actions/setup-node@v1
    with:
      node-version: ${{ matrix.node }}
```

Aus eins mach neun: Mit Matrix-Builds lassen sich Jobs auf verschiedenen Plattformen ausführen.

Die Kontextvariable wird hier nicht nur innerhalb der Umgebung gesetzt. GitHub ersetzt sie bereits vor der Ausführung des Jobs.

Sie können Kontextvariablen aber auch in Umgebungsvariablen überführen, um diese innerhalb von Diensten oder Skripten zu nutzen:

```yaml
env:
  USER: ${{secrets.USER}}
```

Da die Umgebungsvariablen innerhalb des Jobs existieren, können Sie diese auch dort verändern. Die Kontextvariablen werden vor dem Ausführen des Jobs durch den passenden Wert ersetzt und sind dort statisch.

Matrix-Builds

Actions kann mittels Matrix-Builds mehrere Jobs in verschiedenen Umgebungen oder auf unterschiedlichen Betriebssystemen durchführen. So kann man den Kompiliervorgang seines Codes auf Windows, Linux und macOS durchführen und schauen, ob die neusten Änderungen Probleme machen.

Eine Matrix wird über das gleichnamige Schlüsselwort definiert und wie eine Kontextvariable innerhalb des Workflows verwendet. Will man etwa ein Nodejs-Modul auf Ubuntu, Windows und macOS mit den Nodejs-Versionen 8, 10 und 12 übersetzen, braucht man den Aufbau im Kasten unten.

GitHub erzeugt beim Start dieses Workflows neun Jobs: drei Betriebssysteme, auf denen drei Nodejs-Versionen laufen. Actions ersetzt die in doppelte geschweifte Klammern gesetzten Variablen durch die für den Vorgang notwendigen Werte. Bei `runs-on` fügt es die in `matrix.os` gelisteten Systeme ein. Bei `node-version` die Nodejs-Versionen 8, 10 und 12. Ein simples Beispiel zum Test eines Nodejs-Moduls mit Actions finden Sie über ct.de/wbbc. Sie können mittels weiterer Schritte die Test- und Paketiervorgänge anstoßen und bekommen Ergebnisse für neun Konstellationen.

Weiterdrehen

Dieser Artikel bietet nur einen kleinen Einblick in die Welt von Actions. Die Workflows lassen sich beliebig komplex ausgestalten. Die von GitHub bereitgestellte Dokumentation ist glücklicherweise sehr ausführlich und die Arbeit mit Actions recht einfach, wenn man die Grundkonzepte verinnerlicht hat. Die Fleißarbeit besteht darin, einen Workflow zu entwickeln, der den eigenen Code zuverlässig so verarbeitet, wie man das erwartet, und die gewünschten Ergebnisse produziert. Eine gute Inspirationsquelle ist eine Sammlung namens „Awesome Actions“ auf GitHub, die allerlei Strategien und Tipps zu Actions sammelt. In Kombination mit GitHubs Paket-Registry muss man die Plattform zum Entwickeln fast nicht mehr verlassen. (jam) c't

Dokumentation

ct.de/wbbc

storage2day

Die Heise-Konferenz für Speichernetze und Datenmanagement

3 x online im Herbst 2021

Auch in diesem Herbst bieten wir Ihnen wieder aktuelles Storage-Wissen. Besuchen Sie unsere Online-Events zu praxisrelevanten Themenschwerpunkten:

- **27. Oktober:**
  Der Storage Strategy Day
- **30. November:**
  Der Storage Security & Backup Day
- **07. Dezember:**
  Der Storage Technology Day

Neu: Zu jeder Veranstaltung stehen Freitickets zur Verfügung.

Jetzt **Freitickets** sichern!

www.storage2day.de

Veranstalter iX dpunkt.verlag

Goldsponsoren

actidata We care about data · CLOUDIAN · DATACORE · FAST LTA · FUJIFILM · FUJITSU · PoINT software & systems · PURESTORAGE · StarWind · Synology · TechData Advanced Solutions

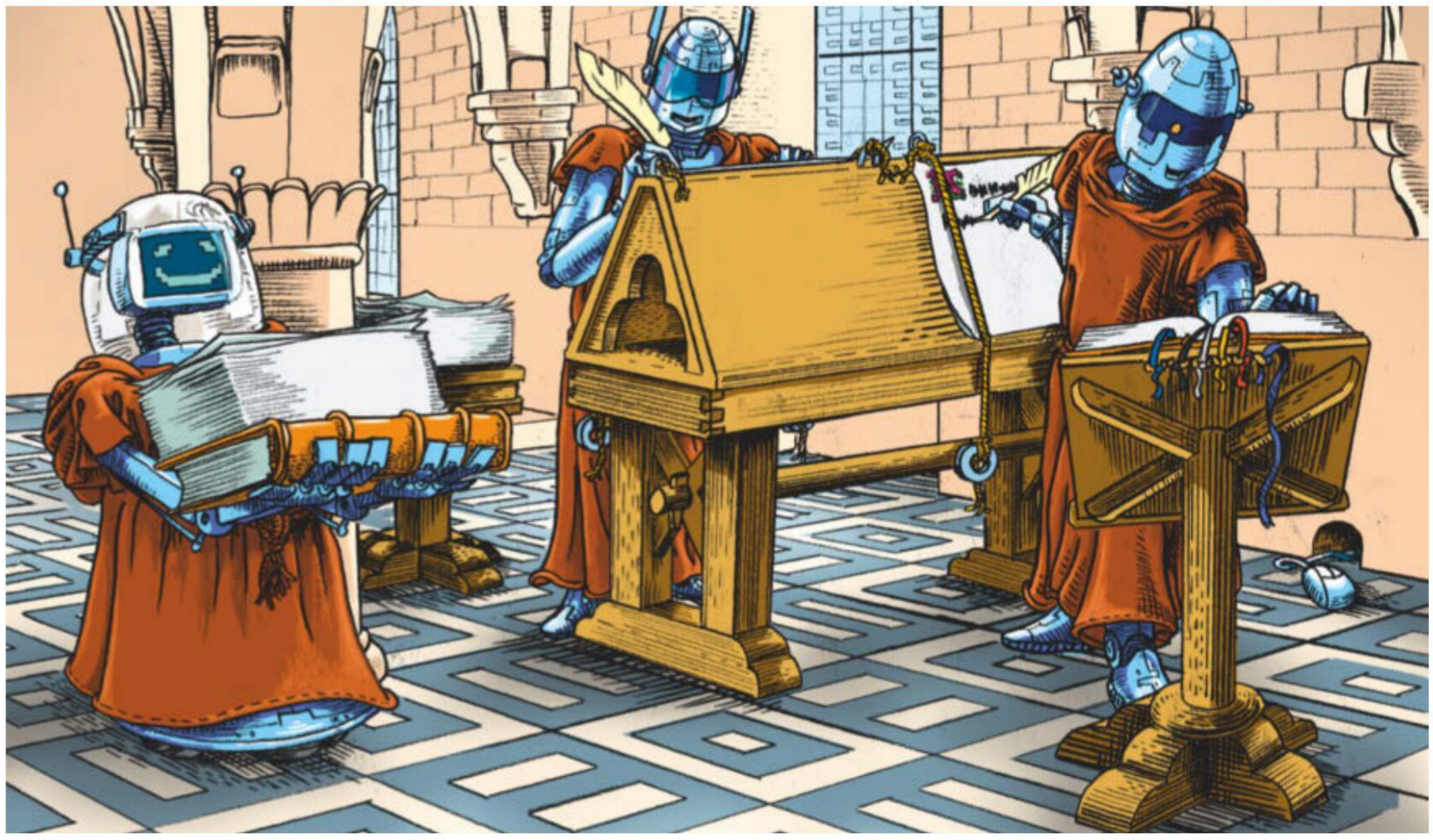


Bild: Rudolf A. Blaha

Online-Dokumentationen mit MkDocs

MkDocs unterstützt Softwareentwickler und Administratoren beim Dokumentieren und kümmert sich darum, dass die Inhalte ansprechend präsentiert werden. Das erhöht die Motivation, Inhalte zu verfassen – und sie zu lesen.

Von **Jan Mahn**

Eine gute Dokumentation ist mehr als nur lästiges Beiwerk. Administratoren sollten das Wissen über ihr Netzwerk für Kollegen und die Nachwelt festhalten und Softwareentwickler tun ihren Kunden einen großen Gefallen, wenn sie – am besten auf einer stets aktualisierten Website – eine ausführliche Dokumentation anbieten. Und selbst fürs Heimnetz, fürs Smart Home und die Bastelprojekte lohnt ein solches Handbuch, das stets aktuell im lokalen Netzwerk oder aus dem Internet erreichbar ist.

Schön auf Knopfdruck

Die Software MkDocs gehört zur Gruppe der statischen Seitengeneratoren – ein Schritt zurück zu den Ursprüngen des World Wide Webs. Autoren verfassen die Texte in einer leicht zu lernenden Auszeichnungssprache, in diesem Fall Markdown. MkDocs rendert aus diesen Texten und einer Vorlage fertige HTML-Seiten mit Menü, die man auf einem Webserver ausliefert.

MkDocs-Dokumentationen begegnet man recht häufig, sowohl bei der Dokumentation von Open-Source-, als auch bei kommerzieller Software. Nicht immer erkennt man sie auf den ersten Blick – je nach verwendetem Theme sehen die Ergebnisse sehr unterschiedlich aus. Über ct.de/wj6h finden Sie einige Beispiele von größeren Softwareprojekten. Viele nutzen das Material-Theme, das Googles Designsprache Material umsetzt. Unter den MkDocs-Themes ist es das mit Abstand ausgereifteste und sehr gut personalisierbar.

Einrichten

Ein eigenes MkDocs-Projekt ist schnell aufgesetzt und mit wenig Arbeit wird aus einem Markdown-Dokument eine hübsche Dokumentation im Browser. Ein Vorteil von statischen Seitengeneratoren ist es, dass man das Ausgangsmaterial, also Texte und Bilder, mit einer Versionsverwaltung wie Git genau wie Programmcode verwalten und versionieren kann – anders als zum Beispiel die Daten einer WordPress-Installation, die zur Hälfte in einer SQL-Datenbank und zur anderen Hälfte in Daten-Ordnern stecken.

Am besten erzeugen Sie schon für die ersten Versuche mit MkDocs ein Git-Repository – als Vorlage können Sie das Beispiel-Repository zu diesem Artikel auschecken (siehe ct.de/wj6h).

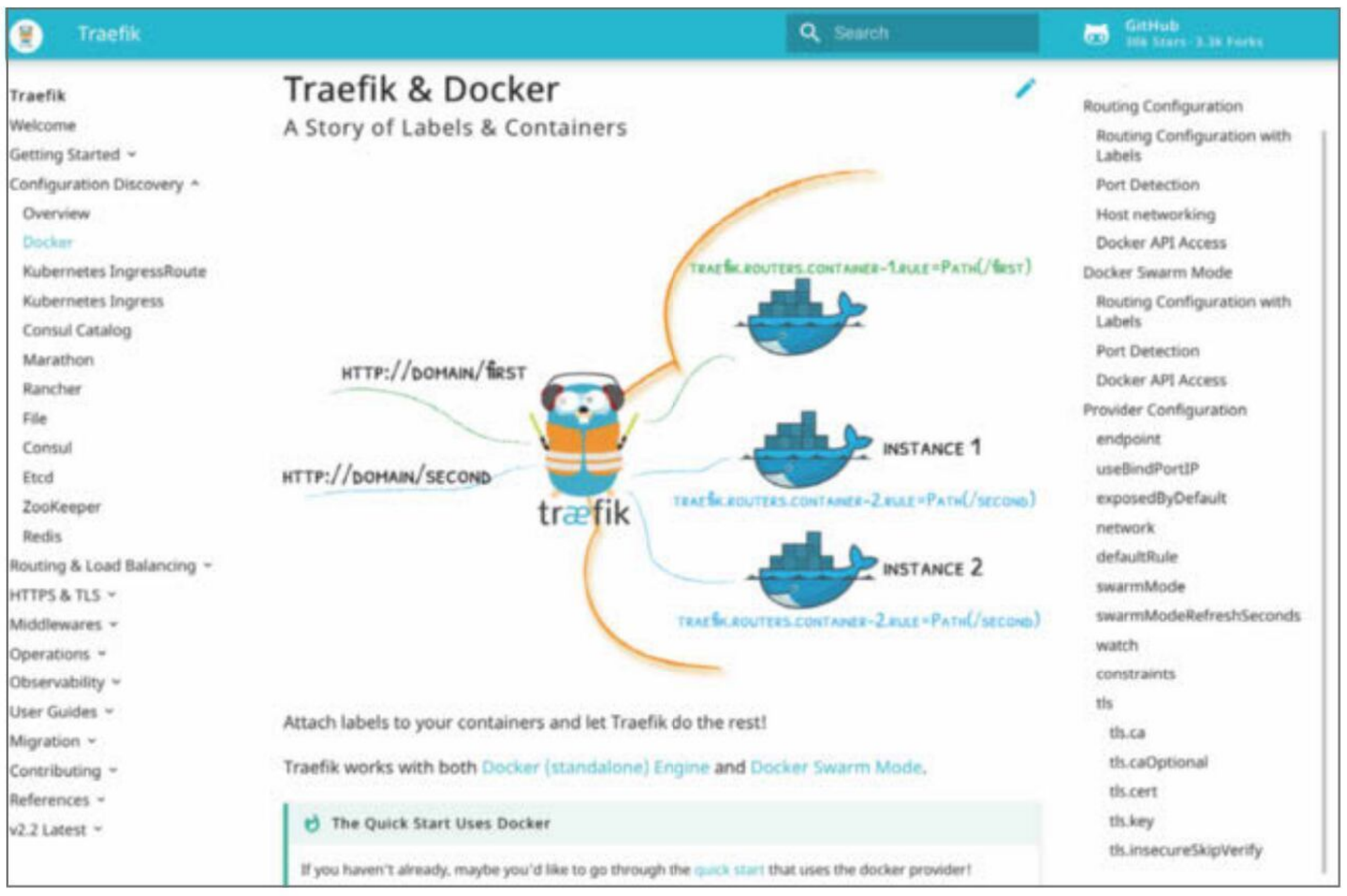
Damit MkDocs eine Dokumentation rendern und ein Menü erzeugen kann, braucht der Generator zunächst eine Konfigurationsdatei im YAML-Format mit dem Namen mkdocs.yml, die zum Beispiel so aussieht:

```
site_name: Beispiel-Doku
nav:
 - index.md
 - getting_started.md
 - legal/imprint.md
theme: material
```

Sie muss mindestens den `site_name`, das `theme` und einen Eintrag unter `nav` enthalten. Im Ordner, in dem die Konfiguration liegt, brauchen Sie einen Ordner mit dem Namen docs. In diesem landen alle Markdown-Dokumente mit Ihren Inhalten – bei der Gestaltung der Ordnerstruktur unterhalb von docs sind Sie recht frei. Die Ordner- und Dateinamen (ohne die Endung .md) werden beim Generieren zur URL der Seite verarbeitet. Das Impressum, das aus der Datei docs/legal/imprint.md entsteht, findet man später etwa unter „https://example.org/legal/imprint“. Eine Markdown-Datei muss nicht zwangsläufig in der Navigation eingebaut sein, damit MkDocs sie rendert. Alle Dateien im Ordner docs werden verarbeitet.

Legen Sie jetzt die Markdown-Datei index.md an und füllen Sie diese mit etwas Inhalt, etwa:

```
# Willkommen
Dies ist eine Dokumentation.
```



MkDocs ist bei großen Softwareprojekten im Einsatz. Die Dokumentation des HTTP-Routers Træfik zeigt, was mit der Software möglich ist.

```
![Ein Screenshot](screen.png)
Zum [Impressum](legal/imprint.md).
```

In der letzten Zeile sehen Sie einen internen Link, der beim Generieren automatisch zu einem vollständigen HTML-Link umgewandelt wird. MkDocs ist in der Lage, den Titel einer Seite anhand der ersten Überschrift erster Ebene (in Markdown mit # eingeleitet) auszulesen. In der Navigation erscheint die Seite also als „Willkommen“. Möchte man diesen Titel im Menü überschreiben, ergänzt man den alternativen Titel in der mkdocs.yml:

```
nav:
 - "Herzlich Willkommen": index.md
```

Einsetzen

MkDocs ist eine Python-Anwendung, vom Python-Code sieht man als Anwender aber nichts, wenn man nicht gerade selbst Plug-ins entwickeln möchte. Wer sich in der Python-Welt zu Hause fühlt und sowohl Python als auch den Python-Paketmanager Pip installiert hat, kann die Software als Paket beziehen:

```
pip install mkdocs
```

Wenn Sie, wie im Beispiel, das beliebte Material-Theme nutzen möchten, laden Sie auch dieses über pip:

```
pip install mkdocs-material
```

Anschließend navigieren Sie auf der Kommandozeile in den Ordner, in dem die Datei mkdocs.yml liegt und starten den Entwicklungsserver:

```
mkdocs serve
```

Nach etwa einer Minute erreichen Sie die Dokumentation unter der Adresse http://localhost:8000 im Browser. Der Entwicklungsserver bemerkt, wenn Sie eine Änderung vorgenommen haben und lädt automatisch die Seite neu. So kann man bequem die Dokumentation verfassen und die Ergebnisse fast in Echtzeit verfolgen.

Der Entwicklungsserver ist aber nicht dafür gedacht, die Seite auszuliefern. Stattdessen soll die Python-Anwendungen einen Ordner mit fertigen HTML-Dokumenten erzeugen, die man dann mit einem Webserver ausliefern kann. Den Generator starten Sie im Projektordner mit:

```
mkdocs build
```

Das Ergebnis landet im Ordner site. Wenn Sie ein klassisches Webhosting-Paket gebucht haben, auf das Sie zum Beispiel per SFTP Ihre Inhalte kopieren, können Sie diesen Ordner dort per Hand (auf der Kommandozeile oder mit Werkzeugen wie FileZilla) ablegen oder ein Skript schreiben, das die Doku lokal baut und dort hochlädt.

Eine Skriptsprache wie PHP, Python oder Perl ist auf dem Webserver nicht nötig – sogar die eingebaute Suche kommt ganz ohne aus. Beim Bauvorgang erzeugt MkDocs einen Suchindex und löst die Suche Client-seitig per JavaScript. Auch wenn Sie nur eine Seite geändert haben, sollten Sie daher immer den ganzen Ordner site auf den Server kopieren.

Standard aufbohren

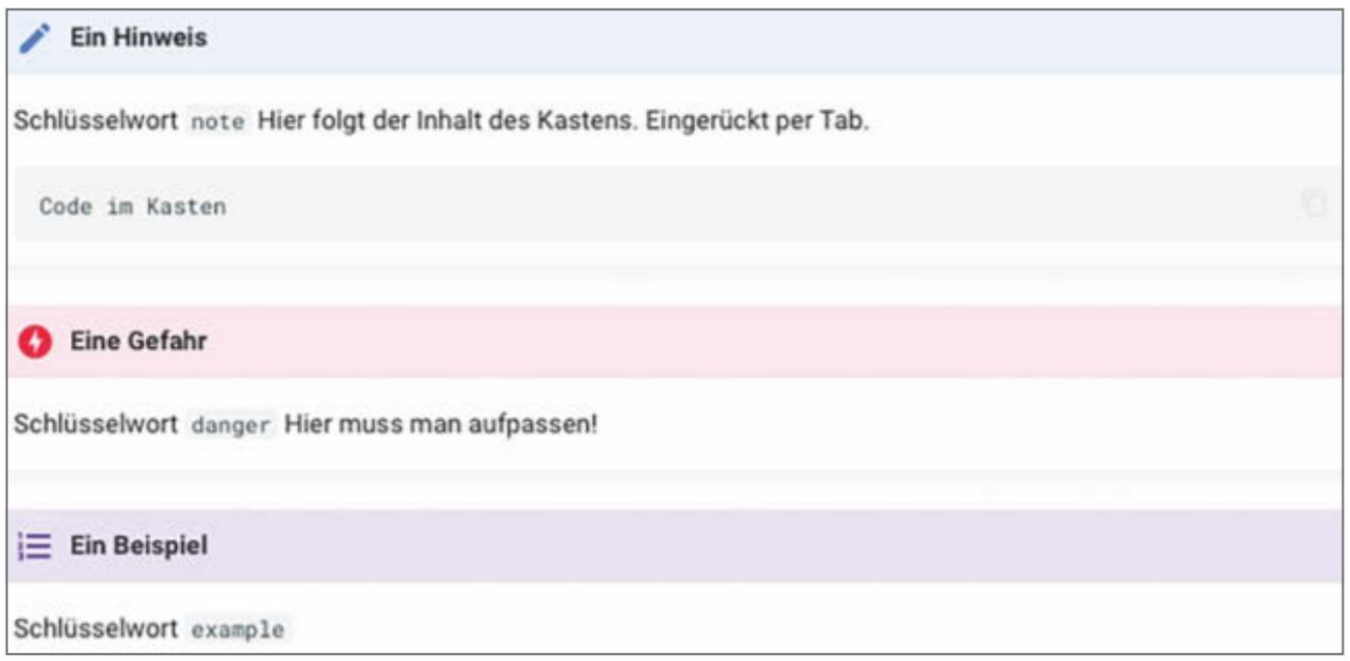
Bilder, Links, Tabellen und Überschriften und Listen und Listing-Abschnitte für Programmcode erstellen Sie mit handelsüblichem Markdown. Eine kleine Übersicht der gängigsten Formatierungen:

```
 # Überschrift erster Ebene

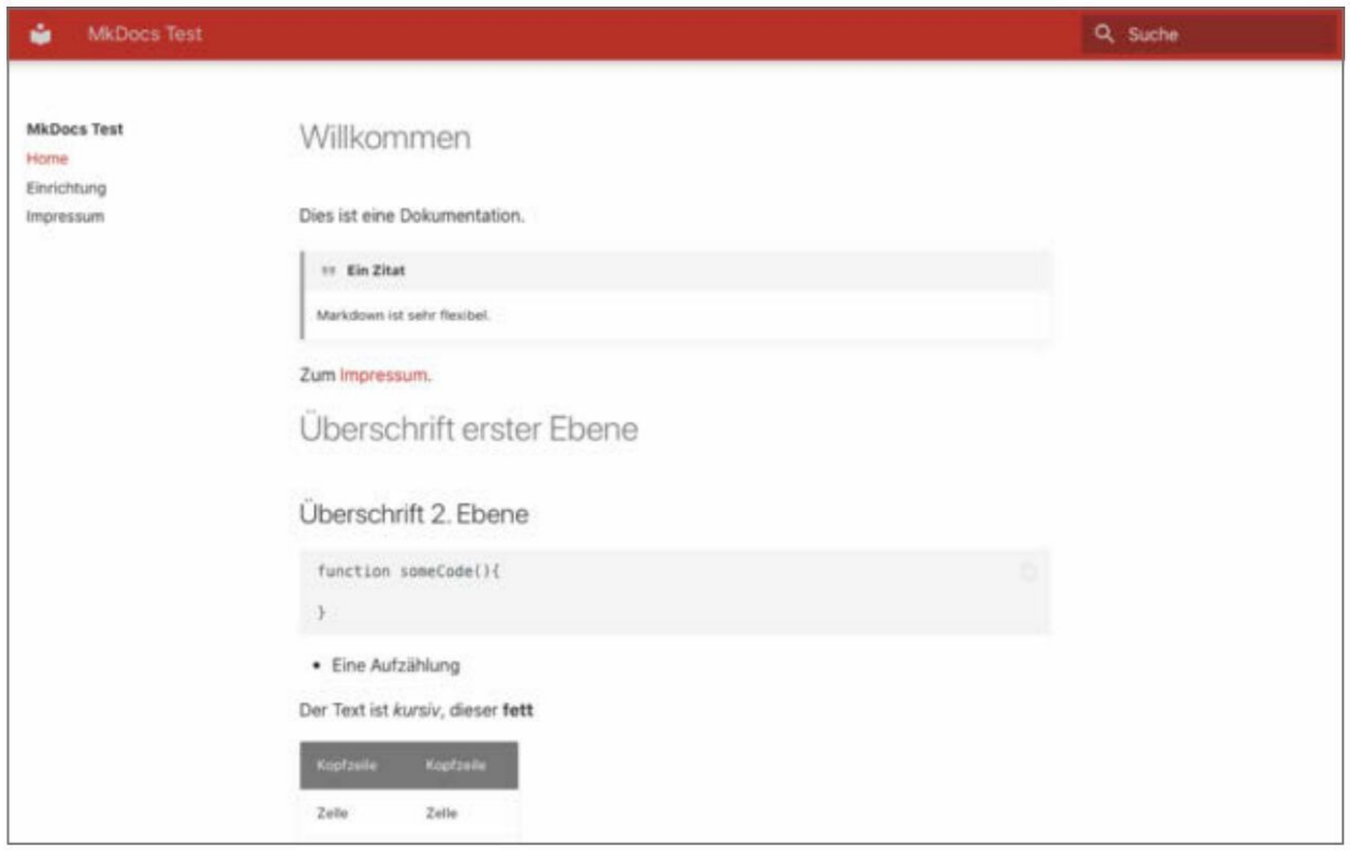
 ## Überschrift 2. Ebene

```
function someCode(){

}
```
```



Mit der Markdown-Erweiterung Admonition erzeugen Sie Textkästen, um Inhalte vom Fließtext abzuheben.



Mit CSS-Variablen passen Sie MkDocs-Dokumentation farblich an, damit Sie zu Ihrem Projekt passt.

```
 * Eine Aufzählung

 * mit zwei Punkten

Der Text ist *kursiv*, dieser **fett**
```

Darüber hinaus ist es hilfreich, Markdown mit sogenannten Extensions zu erweitern. In Dokumentationen kommt es recht häufig vor, dass man Abschnitte besonders hervorheben möchte – etwa eindringliche Warnungen oder gut gemeinte Hinweise. Mit der Erweiterung `admonition` erzeugen Sie Kästen, die im Fließtext auffallen. Die Erweiterung `superfences` kümmert sich darum, dass Sie in diesen Kästen Markdown-Formatierungen nutzen können. `details` macht die Kästen auf Wunsch ein- und ausklappbar. Die nötigen Pakete werden bereits mit dem Material-Paket heruntergeladen. Sie aktivieren die Erweiterungen durch eine kleine Ergänzung in der Datei mkdocs.yml:

```
markdown_extensions:
 - admonition
 - pymdownx.superfences
 - pymdownx.details
```

Sind die Erweiterungen aktiviert, erzeugen Sie einen Kasten in einer Markdown-Datei mit:

```
!!! note "Ein Hinweis"
 Hier folgt der Inhalt des Kastens.
 Eingerückt per Tab.
 ```

 Code im Kasten
 ```
```

Oben sehen Sie verschiedene Textkästen in ihrer ganzen Schönheit. Neben dem Schlüsselwort `note` für harmlose Hinweise unterstützt die Erweiterung auch `abstract`, `info`, `tip`, `success`, `question`, `warning`, `failure`, `danger`, `bug` und `quote`. Damit sind zum Beispiel auch FAQ-Bereiche oder Schulungsunterlagen für Mitarbeiter mit MkDocs kein Problem. Soll der Kasten ein- und ausklappbar sein, ersetzen Sie die `!!!` durch `???`.

Aller Wahrscheinlichkeit nach werden Sie mit den eingestellten Farben für Links und die Kopfleiste noch nicht zufrieden sein – als Unternehmen muss die Dokumentation zum Corporate Design passen. Dafür müssen Sie aber nicht von Grund auf ein neues Template entwickeln. Das Material-Design können Sie mit zusätzlichem CSS erweitern. Ersetzen Sie in der mkdocs.yml zunächst die Zeile `theme: material` durch folgenden Block:

```
theme:
 name: material
 language: de
 font: false
 favicon: favicon.png
 logo: logo.png
```

Mit `language: de` zeigt MkDocs die Navigationselemente auf Deutsch an. `font: false` verhindert das Einbinden von externen Schriftarten direkt von Google Fonts – der Datenschutzbeauftragte wird es Ihnen danken. Auch das Logo und das Favicon können Sie hier konfigurieren. Die Bilddateien gehören wie die Markdown-Texte in den Ordner docs oder einen Unterordner. Die vollständige Dokumentation des Material-Themes mit weiteren Optionen finden Sie über ct.de/wj6h.

Mit `extra_css` binden Sie eine zusätzliche CSS-Datei ein. In dieser können Sie mit sogenannten Custom Properties die Farben an Ihre Wünsche anpassen, zum Beispiel für eine rote Kopfleiste mit grauem Text:

```
:root {

 --md-primary-fg-color: #B51F10;
 --md-primary-fg-color--light: ...;
 --md-primary-fg-color--dark: ...;
```

```
 --md-primary-bg-color: #ccc;
 --md-primary-bg-color--light: #eee;
}
```

Die gesamte CSS-Datei finden Sie im Beispielprojekt über ct.de/wj6h.

Einfacher verpacken

Wer kein Python installiert hat, mit mehreren Entwicklern zusammen an einer Doku arbeiten möchte oder möglichst viel automatisieren will, kann sich mit Docker-Containern das Leben leichter machen – sowohl für die Entwicklung als auch für die Auslieferung der Seite. Steckt man die gesamte Doku in einen Container, kann man sicher sein, dass die Software auf allen Entwickler-Maschinen und auf dem Server identische Bedingungen vorfindet – identische Abhängigkeiten, MkDocs-Plug-ins und Python-Versionen zum Beispiel.

Für Entwicklung und Auslieferung brauchen Sie zwei unterschiedliche Dockerfiles. In der Entwicklungsumgebung soll das Neuladen bei Änderungen weiter funktionieren. Folgendes Dockerfile richtet eine Arbeitsumgebung im Container ein. Alle Dateien finden Sie im Beispielprojekt zum Artikel bei GitHub. Die Datei bekommt den Namen Dockerfile-dev:

```
FROM python:3-alpine
RUN apk add build-base
COPY ./mkdocs/ /mkdocs/
WORKDIR /mkdocs/
RUN pip install -
-upgrade pip && ↵
↳pip install mkdocs mkdocs-material
EXPOSE 8080
CMD ["mkdocs", "serve"]
```

Damit die Zusammenstellung funktioniert, müssen Sie das Repository etwas umstrukturieren: Auf der obersten Eben legen Sie den Ordner container an. In diesen kommt das oben stehende Dockerfile mit dem Dateinamen Dockerfile-dev. Außerdem ein Ordner mkdocs, der die oben angelegte Datei mkdocs.yml sowie den Ordner docs mit allen Inhalten enthält.

Um den Container für Entwicklungszwecke zu starten, sollten Sie sich eine kleine Docker-Compose-Datei mit dem Namen compose-dev.yml auf der obersten Ebene anlegen:

```
version: "3.7"
services:
 docs:
 build:
 context: ./container
 dockerfile: Dockerfile-dev
 ports:
  - 8080:8080
 volumes:
  - ./container/mkdocs/:/mkdocs
```

Navigieren Sie auf der Kommandozeile in den Projektordner und fahren Sie die Entwicklungsumgebung hoch:

```
docker compose -f \
docker-compose-dev.yml up
```

Docker baut den Container, startet ihn und hängt den Ordner mit den Inhalten als Volume ein. Auf der Entwicklermaschine muss nur Docker installiert sein (Docker-Compose ist in neuen Versionen direkt enthalten), Python, Pip und Wheel laufen im Container.

Für das Produktivsystem brauchen Sie ein weiteres Dockerfile und eine weitere Docker-Compose-Datei. Ein MkDocs-Projekt ist eine gute Gelegenheit für einen Multistage-Build. Die Technik hat Docker erfunden, um kleinere und sicherere Abbilder zu erzeugen.

In der ersten Stufe wird als Basis-Image ein Python-Abbild mit dem Paketmanager Pip verwendet. Dann werden die Daten ins Abbild kopiert und die Seite mit `mkdocs build` gebaut. Das Ergebnis des Bauprozesses, also der fertige Order, wird in die zweite Stufe kopiert, die nur den Nginx-Webserver enthält. Python, Pip und die Spuren des Bauprozesses landen nicht im fertigen Image. Das Dockerfile für den Produktivbetrieb sieht so aus:

```
#build stage
FROM python:3-alpine AS build
RUN apk add build-base
COPY ./mkdocs/ /mkdocs/
WORKDIR /mkdocs/
RUN pip install -
-upgrade pip && ↵
↳pip install mkdocs mkdocs-material
RUN mkdocs build

#web server stage
FROM nginx:alpine
COPY --from=0 /mkdocs/site/ ↵
↳usr/share/nginx/html
EXPOSE 80
```

Speichern Sie diese Datei unter dem Namen Dockerfile im Ordner container. Auf der obersten Ebene des Projekts können Sie eine Docker-Compose-Datei mit dem Namen docker-compose.yml für den Produktivbetrieb anlegen:

```
version: "3.7"
services:
 docs:
 build:
 context: ./container
 dockerfile: Dockerfile
 ports:
  - 80:80
```

Wenn Sie diese Zusammenstellung mit `docker compose up` starten, anwortet auf Port 80 die generierte Dokumentation, ausgeliefert vom Nginx-Webserver.

Aktuell ohne Arbeit

Beispielprojekt
ct.de/wj6h

Auf Grundlage von Containern können Sie die Automatisierung des Auslieferungsprozesses aber auch noch weiter optimieren. Wünschenswert wäre es, wenn niemand per Hand den Bau von Änderungen anstoßen müsste. Ein Push ins Repository bei GitHub oder GitLab soll ausreichen, damit eine Änderung wenige Minuten später online ist. Damit das funktioniert, müssen Sie den Container bei einem Push ins Repository automatisch bauen lassen. Im Folgenden zeigen wir diesen Schritt mit GitHubs Automationslösung GitHub Actions, mit GitLab funktioniert das Verfahren sehr ähnlich.

Damit GitHub den Container bei jedem Push baut und das Abbild in die eigene Container-Registry legt, brauchen Sie eine YAML-Datei im Ordner .github/workflows/ im Repository. Sie finden die Datei im Beispielprojekt über ct.de/wj6h. Nach einem Push liegt das Abbild in aktueller Version in der GitHub-Container-Registry unter dem Namen `docker.pkg.github.com/<Ihr Name>/<Ihr Repo>/my-mkdocs:latest`.

Auf dem Server, auf dem das Docker-Image laufen soll, müssen Sie sich zunächst an der Registry anmelden. Dafür sollten Sie nicht das Kennwort des Accounts nutzen, sondern mit einem Token arbeiten. Einen solchen erzeugen Sie unter https://github.com/settings/tokens. Geben Sie dem neuen Token einen Namen und wählen Sie die Punkte „repo“ und „read packages“ an. Kopieren Sie die erzeugte Zeichenkette und melden sich auf dem Server mit installiertem Docker an:

```
docker login docker.pkg.github.com
```

Sie werden nach einem Benutzernamen und dem Kennwort gefragt – als Kennwort nutzen Sie das Token. Unten sehen Sie eine Docker-Compose-Datei mit dem nützlichen Helfer `watchtower`. Er prüft regelmäßig, ob eine aktuelle Version des Doku-Containers in der Registry liegt, lädt die neue Version herunter und tauscht das Abbild selbstständig aus.

```
version: "3.7"
services:
 docs:
 image: docker.pkg.github.com/<Ihr Nutzername>/<Ihr Repo>/my-docs:latest
 restart: always
 labels:
  - "com.centurylinklabs.watchtower.enable=true"
 ports:
  - 80:80
 watchtower:
 image: containrrr/watchtower
 command: --cleanup --label-enable
 restart: always
 volumes:
  - /var/run/docker.sock:/var/run/docker.sock
  - ~/.docker/config.json:/config.json
```

Zwischen dem Push einer Änderung und der Veröffentlichung vergehen nur wenige Minuten.

Losdokumentieren

Die Hauptarbeit nimmt Ihnen MkDocs nicht ab – die Texte müssen Sie weiterhin selbst verfassen. Mit einem optisch ansprechenden Rahmen und einem automatischen Bauprozess sitzen Sie immerhin nicht mehr vor einem leeren Word-Dokument, wenn Sie die Doku tippen. Insbesondere für Entwickler ist der Arbeitsablauf attraktiv: Sie müssen zum Schreiben und Veröffentlichen nicht ihre gewohnten Werkzeuge verlassen. Markdown für MkDocs tippen Sie wie gewohnt in Ihrer Lieblings-Programmierumgebung. (jam) c't



Bild: Thorsten Hübner

Statusseiten für IT-Infrastruktur

Zum professionellen Betrieb von IT-Infrastruktur gehört auch der Umgang mit Fehlern. Weil Nutzer bei Problemen schnell nervös werden, sollte man sie zügig und transparent informieren. Mit der Software cState zeigen Sie übersichtlich, wo es gerade klemmt.

Von **Jan Mahn**

Kunden der Deutschen Bahn fühlen sich oft gleich doppeltem Ärger ausgesetzt: Erst kommt der Zug nicht oder zu spät, und dann sprechen automatische Lautsprecherdurchsagen und Apps schlicht von „Verzögerungen im Betriebsablauf“. So sieht eine professionelle Reaktion auf Störungen nicht aus. Kunden und Nutzer von Diensten erwarten heute – zu Recht – eine Erklärung des Problems, und viel Frust am Bahnsteig wäre vermieden, wenn zum Beispiel eine menschliche Stimme per Lautsprecher vermelden würde, dass Kriminelle soeben die Oberleitung gestohlen haben.

Aber nicht nur Bahnunternehmen, auch Betreiber von Websites und anderen IT-Dienstleistungen tun gut daran, ihren Nutzern eine Anlaufstelle bereitzustellen, bei der sie sich über Störungen informieren können. Einige der Großen machen vor, wie das aussehen kann. „Incident Management“ heißt diese Disziplin und informative Statusseiten sind ein Baustein. Über Probleme der IT-Infrastruktur offen zu sprechen ist heute keine Schande mehr – sie totzuschweigen schon eher. Unter der Adresse cloudflarestatus.com stellt zum Beispiel der DNS- und DDoS-Abwehr-Anbieter Cloudflare eine solche Übersicht über den Zustand seiner Systeme bereit. Für vernetzte Microsoft-Produkte wie Outlook.com, OneDrive und Skype gibt es admin.microsoft.com/servicestatus. Der Paketdienstleister DHL betreibt für

Nutzer seiner APIs die Seite status.api.dhl.com, GitHub hat githubstatus.com.

Kein Wunder, könnte man einwenden, dass solche Großunternehmen sich das leisten können, schließlich haben die auch eigene Teams mit Mitarbeitern, die sich ausschließlich um Ausfälle und Probleme kümmern. Doch auch in kleinen Umgebungen können sich solche Statusseiten lohnen. Wenn Sie etwa die Infrastruktur für ein Unternehmen betreuen, den Mitarbeitern Mail-Postfächer, VPN und weitere Dienste bereitstellen und sie dazu erzogen haben, bei vermeintlichen Ausfällen zunächst die Statusseite zu prüfen, sparen Sie sich eine Menge Mails im Support-Postfach mit redundanten Hinweisen auf den Ausfall.

Die gesparte Zeit können Sie im Störungsfall für die Problembehebung aufwenden. Die einmalige Einrichtungsarbeit der Statusseite sollten Sie in ruhigeren Zeiten erledigen. Auf eine solche Seite gehören aber nicht nur unvorhergesehene Störungen, sondern auch Ankündigungen von geplanten Wartungsmaßnahmen. Je genauer Sie den Nutzern erklären, was hinter den Kulissen vorgeht, desto zufriedener sind diese mit Ihrer Arbeit.

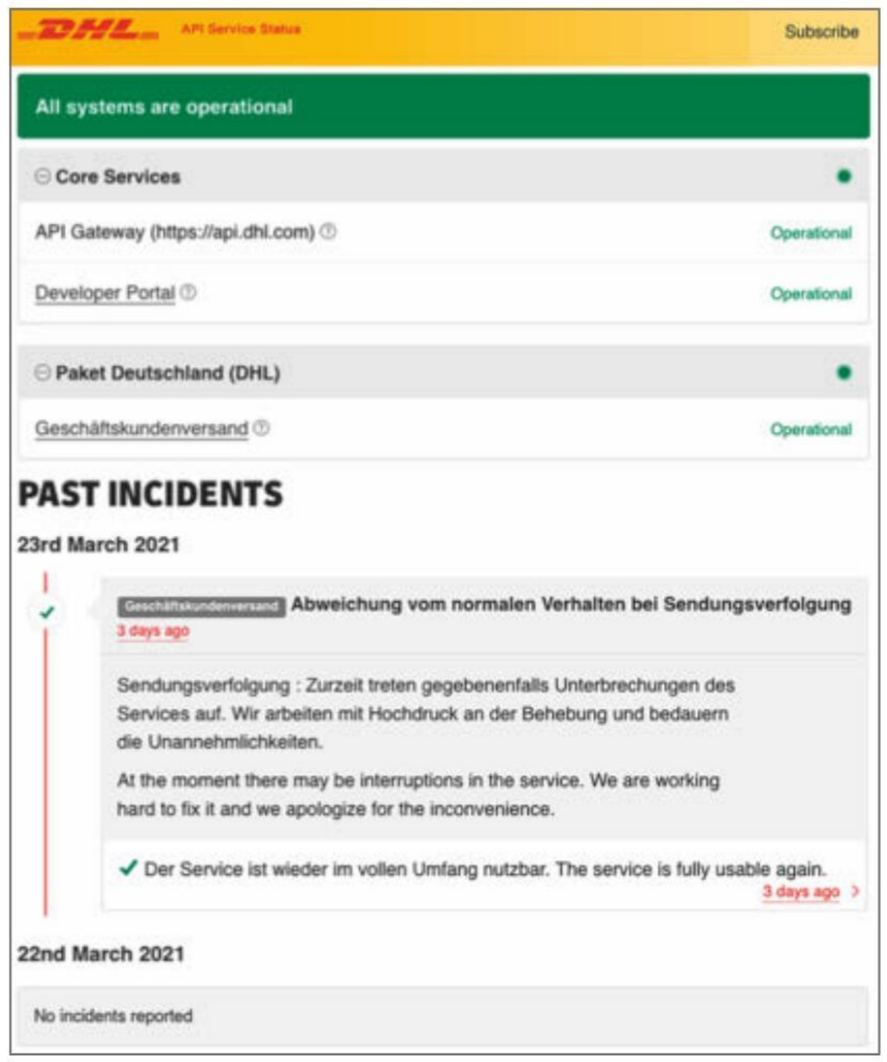
Die Voraussetzungen

Damit eine Statusseite ihren Zweck erfüllen kann, muss sie so unabhängig wie möglich von der restlichen Infrastruktur sein, über deren Status sie berichten soll. Wenn Sie Ihre Server im eigenen Rechenzentrum betreiben, sollten Sie für die Statusseite ein kleines Hosting-Paket bei einem Webhoster anmieten. Wenn Sie irgendwelche Dienste betreiben, die bei einem der großen Cloud-Provider wie AWS, Google oder Azure liegen (direkt oder indirekt über Produkte, die Sie bei Drittanbietern mieten), sollte die Statusseite definitiv woanders laufen – zum Beispiel bei einem kleineren Webhoster, der ein eigenes Rechenzentrum betreibt, gern auch außerhalb der Internetmetropole Frankfurt. So reduzieren Sie die Wahrscheinlichkeit, dass Sie sich einen „Single Point of Failure“ in Ihre Kommunikationsstrategie eingebaut haben.

Cloudflare geht sogar soweit, eine eigene Domain nur für seine Statusseite zu betreiben und vertraut diese nicht seinen eigenen DNS-Servern an, obwohl DNS das Kerngeschäft des Unternehmens ist. Die Domain cloudflarestatus.com wird von einem DNS-Server bei Google verwaltet. Ganz so weit müssen Sie nicht gehen. Viele Statusseiten liegen unter der Subdomain „status“, also etwa `status.example.org`. Solange Sie nicht selbst im DNS-Geschäft tätig sind und diese Aufgabe anderen überlassen, ist das ausreichend unabhängig. Versuchen Sie nicht, den Ausfall zu 100 Prozent auszuschließen, sondern vielmehr, offensichtlich verhängnisvolle Verkettungen zu verhindern.

Eine ganz simple Statusseite können Sie mit etwas HTML- und CSS-Erfahrungen selbst zusammenbauen. Es gibt aber auch Software, die noch schneller ansehnliche Statusseiten generiert und in ein Format bringt, das Nutzer von anderen Statusseiten gewohnt sind. Für solche Anwendungen kann man durchaus Geld ausgeben: Einige Unternehmen setzen auf das kommerzielle Produkt „Statuspage“ von Atlassian (in Aktion zu sehen bei redditstatus.com) und der Markt hat noch weitere kostenpflichtige Alternativen zum Selbsthosten oder als angemietete Lösung zu bieten.

Doch es gibt auch Open-Source-Alternativen: Lange war das Projekt Cachet populär (zu finden unter cachethq.io), das jedoch seit 2019 eingeschlafen zu sein scheint – obwohl auch große Unternehmen wie DHL noch darauf setzen (status.api.dhl.com).



Viele große Unternehmen stellen mittlerweile Statusseiten für ihre Dienste bereit. DHL setzt dabei noch auf das Open-Source-Projekt Cachet, dessen Entwicklung leider eingeschlafen ist.



Auf einen Blick sehen die Benutzer, wo es Probleme gibt. cState generiert statisches HTML, das jeder Webserver ausliefern kann.

Im Hintergrund arbeitet eine Art Content Management System nur für Störungen und Probleme, zum Betrieb ist eine Datenbank wie MySQL oder PostgreSQL nötig.

Wesentlich schlanker und aktiv entwickelt ist „cState“. Im realen Einsatz sehen Sie cState etwa beim deutschen IT-Dienstleister Proventa (status.proventa.io). Die Entwickler von cState haben das Projekt nicht bei Null begonnen, sondern den Static-Site-Generator Hugo als Basis eingesetzt: Hugo rendert aus Markdown-Dateien HTML-Seiten, die fertig sind zum Ausliefern durch einen Webserver. Die Bedingungen, um cState zu betreiben, sind entsprechend leicht zu erfüllen: Das günstigste Paket bei einem Webhoster Ihres Vertrauens reicht aus – der Server muss keine Datenbank bereitstellen und keine Skriptsprachen wie PHP oder Perl ausführen, sondern nur HTML ausliefern.

Einzige Herausforderung: Um eine Störung einzutragen, muss man das Kommandozeilenwerkzeug `hugo` ausführen und die gerenderten Dateien auf den Server kopieren. Je nach Umgebung kann man diese Aufgabe aber auch einer Automation anvertrauen.

cState ausprobieren

Wenn Sie cState ausprobieren wollen, brauchen Sie zunächst den Static-Site-Generator Hugo. Falls Sie den nicht installieren wollen und Docker benutzen, finden Sie später im Artikel Hinweise, wie Hugo im Container seine Arbeit verrichtet.

Eine vollständige Installationsanleitung für Windows haben wir über ct.de/ws9u verlinkt. Unter macOS kommen Sie am schnellsten mit dem Paketmanager Homebrew zum Ziel:

```
brew install hugo
```

Linux-Anwender finden ein Paket namens hugo in ihren Paketquellen. Unter Ubuntu und Debian installieren Sie es mit

```
sudo apt install hugo
```

Fedora-Nutzer installieren Hugo mit

```
sudo dnf install hugo
```

Sobald `hugo version` bei Ihnen eine Versionsnummer und keinen Fehler anzeigt, können Sie loslegen. Jetzt brauchen Sie einen Ordner, in dem Sie Ihre Statusseite ablegen und verwalten. Hier liegen die Konfiguration, Grafiken wie Ihr Logo und alle Störungsmeldungen als Markdown-Dateien. Sehr empfehlenswert ist es, diesen Ordner von Anfang an in einer Versionsverwaltung wie Git zu verwahren und bestenfalls über einen Git-Hoster wie GitLab oder GitHub mit Kollegen zu teilen.

Damit Sie nicht mit einem leeren Ordner beginnen müssen, haben wir ein Beispielprojekt vorbereitet. Es orientiert sich am offiziellen Beispielprojekt von cState, die Konfiguration ist aber schon auf die Bedürfnisse und Gewohnheiten von Mitteleuropäern angepasst und um Docker-Rezepte erweitert. Laden Sie unser Beispiel bei GitHub herunter:

```
git clone https://github.com/↵
↳jamct/cstate-example
```

Bewegen Sie sich in das Verzeichnis mit `cd cstate-example`. Hier müssen Sie zunächst das aktuelle cState nachladen, das als sogenanntes Git-Submodul hinterlegt ist:

```
git submodule update ↵
↳--init --recursive
```

Bevor Sie die fertige Seite rendern, sollten Sie mit Hugo einen lokalen Entwicklungsserver starten und Ihre Statusseite lokal anpassen:

```
hugo serve
```

Unter `http://localhost:1313` finden Sie jetzt die Beispielseite, die aussieht wie das Beispiel oben. Haben Sie Hugo nicht installiert und wollen die Docker-Variante nutzen, führen Sie im Verzeichnis einfach folgenden Docker-Compose-Befehl aus:

```
docker-compose -f dc-dev.yml up
```

Nötige Anpassungen

Läuft die Seite, können Sie cState an Ihre Bedürfnisse anpassen. Los geht die Arbeit in der Datei config.yml. Am Anfang ändern Sie den Seitennamen, dann folgt der große Abschnitt `params`, in dem cState personalisiert wird. Zunächst legen Sie `categories` für Ihre Infrastruktur an. Das sind Gruppen, in denen Sie mehrere Dienste zusammenfassen und oben auf der Statusseite gebündelt darstellen. Große Unternehmen brauchen vielleicht die Kategorien „Europa“ und „Asien“, andere vielleicht „Online-Dienste“ und „Lokale Infrastruktur“. Dann folgen in `systems` Ihre Systeme. Ein paar Beispiele haben wir schon angelegt, typische Einträge sind etwa „Website“, „Mail“ oder „VPN“. Störungen werden später diesen Systemen zugewiesen – cState zeigt dann oben direkt in Ampelfarben an, ob Störungen für ein System offen sind.

Alle weiteren Optionen in der Datei config.yml sind gut mit Kommentaren dokumentiert. Es lohnt sich, hier einmal bis zum Ende zu scrollen, die Kommentare zu lesen und die Werte an eigene Wünsche anzupassen. Anschließend sollten Sie die Datei static/logo.png durch Ihr eigenes Logo ersetzen und die Datei layouts/partials/custom/below-footer.html durch eine eigene Fußzeile austauschen.

Dann ist es an der Zeit, das erste Problem zu melden. Alle Störungen landen als eigene Dateien im Ordner content/issues. Zwei Beispiele haben wir bereits vorbereitet. Im Kopfbereich der Datei, eingerahmt von `---`, tragen Sie Metadaten der Störung im YAML-Format ein. Behalten Sie dabei das vorgegebene Datumsformat bei, cState rechnet es um und stellt es im europäischen Format dar. Außerdem rechnet es automatisch aus, wie lange ein Ereignis schon zurückliegt, wie lange die Beseitigung gedauert hat und sortiert das Ereignis in der Darstellung im richtigen Monat ein.

Ist ein Problem behoben, setzen Sie `resolved: true` und ein Datum für `resolvedWhen`. Sobald das eingetragen ist, springt die Ampel oben auf der Seite wieder auf Grün.

Im Bereich unter den `---` können Sie sich in der Auszeichnungssprache Markdown austoben und den Nutzern Status-Updates geben, während Sie am Problem arbeiten.

HTML, bitte

Der Hugo-Server ist nicht dafür gedacht, die Seite an die Kunden auszuliefern. Nutzen Sie stattdessen `hugo --minify`, um statisches HTML mit komprimiertem CSS und JavaScript zu erzeugen. Die fertige Seite landet im Ordner public. Docker-Nutzer bekommen mit unserer Docker-Compose-Datei ihre gerenderten Dateien:

```
docker-compose -f dc-build.yml
```

Wie Sie die fertigen Dateien am bequemsten auf den Webserver bekommen, hängt etwas von Ihren Vorlieben und Ihrer Umgebung ab. Wenn Sie es gewohnt sind, Daten per FTPS oder SFTP auf den Server Ihres Webhosters zu kopieren, schreiben Sie sich zum Beispiel ein kleines Skript, das den Bauprozess startet und den Upload mit `scp` auslöst. Wenn Sie Erfahrung mit CI/CD-Lösungen wie GitHub Actions (siehe Seite 74) haben, können Sie das Rendern auch dorthin auslagern – unser Beispiel-Repository enthält eine Workflow-Datei für GitHub Actions. Bei jedem Push ins Repository wird automatisch mit Hugo gebaut. So kann man die Seite bequem und sogar kostenlos mit GitHub Pages oder dem Serverless-Dienst „Zeit Now“ ausliefern lassen [1, 2].

Besser integriert

Weil alle Störungen als Textdateien in einem maschinenschreibbaren Format im Ordner content/issues liegen, sind nützliche Automationen denkbar – mit etwas Skript-Erfahrungen können Sie zum Beispiel Ihr Monitoring-Werkzeug dazu bringen, bei größeren Problemen schon mal eine mögliche Störung einzutragen. Sobald sich ein Mensch des Problems angenommen hat, ergänzt er einen Kommentar und beruhigt so die nervösen Nutzer. Zur Einführung einer solchen Statusseite gehört daher nicht nur die technische Komponente – damit das Projekt erfolgreich wird, gilt es auch, Nutzer und Admin-Kollegen von den Vorteilen zu überzeugen und daran zu gewöhnen. (jam) c't

Literatur

[1] Merlin Schumacher, **Hosting frei Haus**, Websites mit und ohne eigener Domain auf GitHub hosten, c't 11/2020, S. 162

[2] Jan Mahn, **Serverfrei**, Serverless: Webanwendungen ohne Server?, c't 3/2019, S. 156

cState und Beispielprojekt

ct.de/ws9u

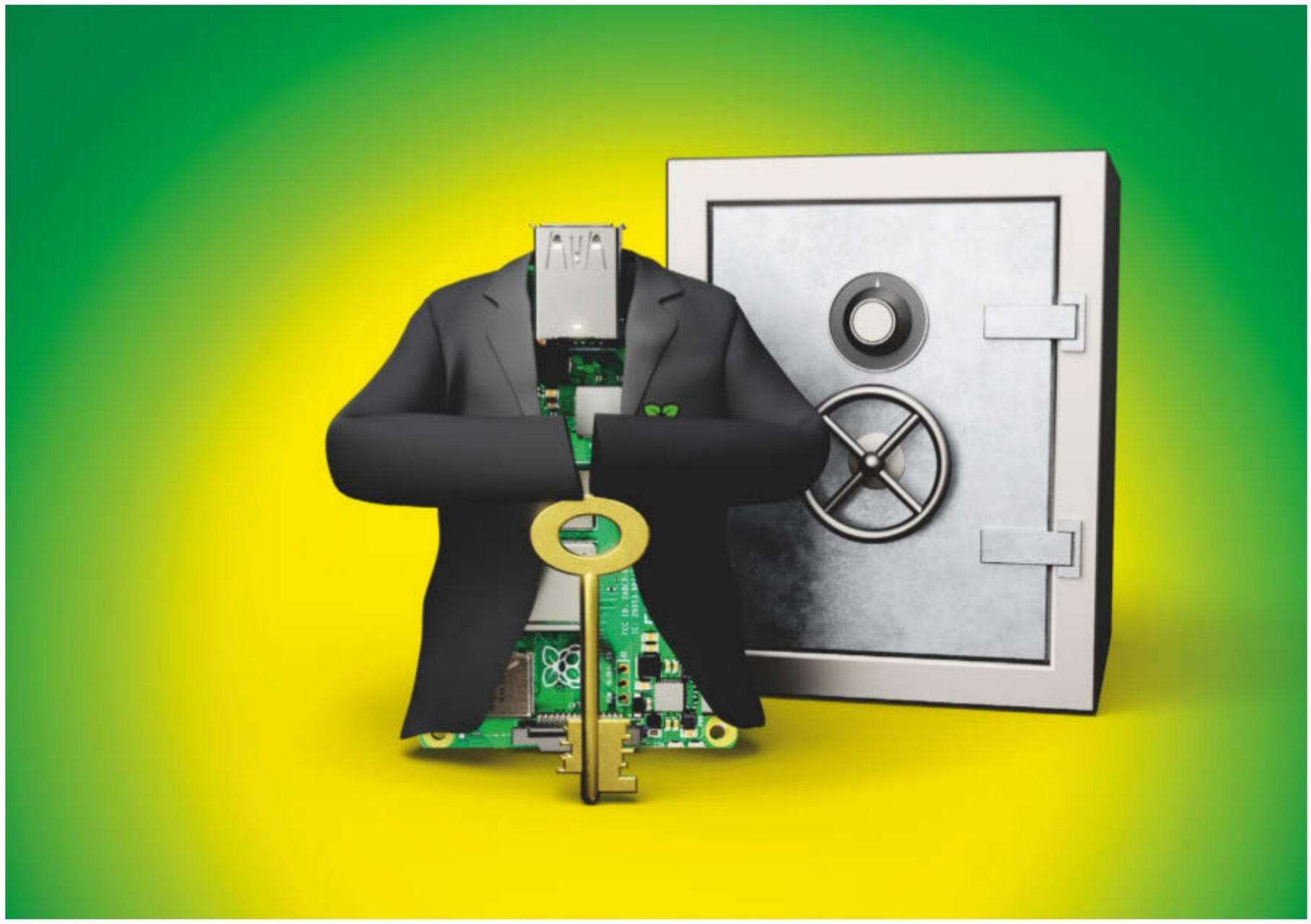


Bild: Andreas Martini

Passwortmanager in Eigenregie hosten

Mit Vaultwarden betreiben Sie Ihren eigenen Passwort-Tresor und -Server. Es synchronisiert Zugangsdaten zwischen mehreren Geräten, ohne Ihre Datenbank in fremde Hände zu geben. Das klappt auf einem (Heim-) Server oder mit dem Raspberry Pi.

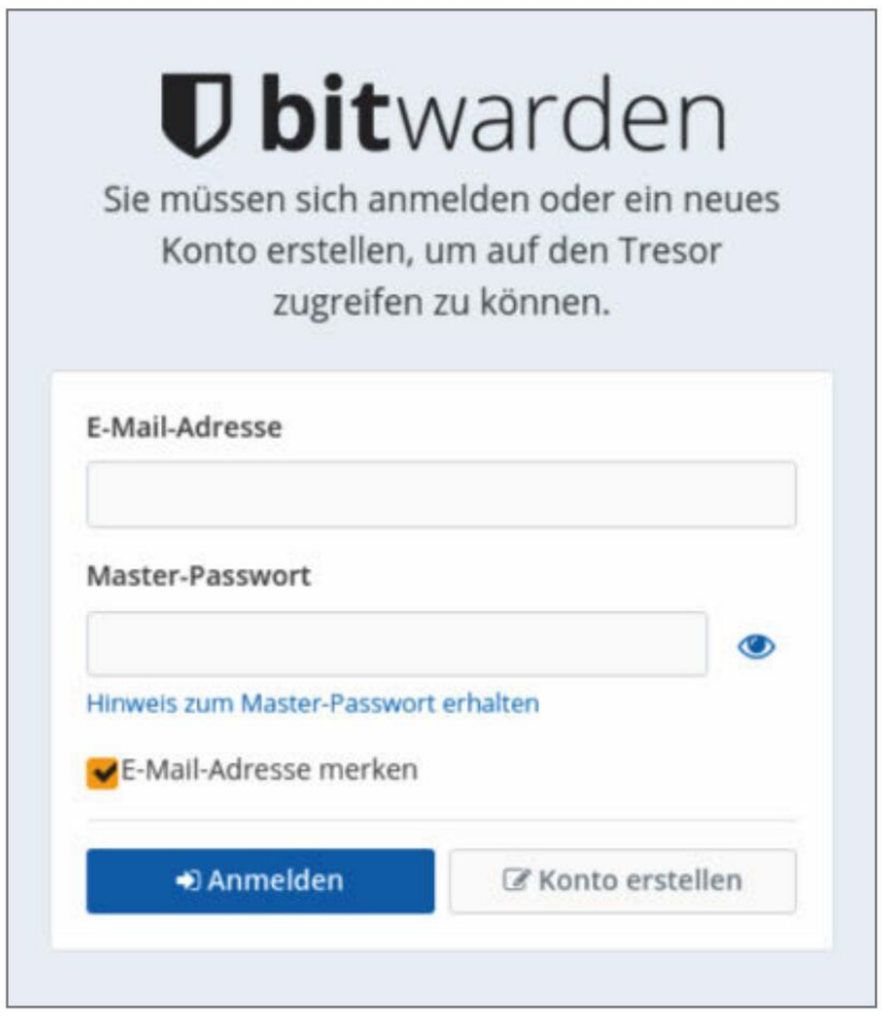
Von **Niklas Dierking**

Ein neues Kundenkonto hier, ein Social-Media-Login da. Bei vielen Internetnutzern türmen sich Zugangsdaten zu Portalen und Diensten zu einem bedrohlichen Haufen auf. Bei jeder Registrierung ein einzigartiges und sicheres Passwort zu überlegen und jedes einzelne zu merken, das ist eine schier unlösbare Aufgabe. Passwortmanager versprechen Abhilfe, aber sie hängen üblicherweise an der Cloud des Anbieters und synchronisieren die Passwortsammlung zwischen mehreren Geräten

nur, wenn man einen monatlichen Obolus zahlt. Außerdem brauchen Nutzer ein hohes Maß an Vertrauen in den Betreiber: Der virtuelle Schlüsselbund liegt immerhin in einem fremden Schließfach.

Den Open-Source-Passwortmanager Bitwarden können Sie in Eigenregie hosten. Bitwarden besteht aus zwei Teilen: dem Server, der Passwörter verwaltet und bereitstellt, und den Clients, die sie abrufen. Das Community-Projekt Vaultwarden hat eine schlankere Variante des offiziellen Bitwarden-Servers geschaffen, die das Bitwarden-API implementiert. Damit kann ihr (Heim-)Server oder ein Raspberry Pi die Rolle des Passwort-Servers übernehmen, der zu den offiziellen Bitwarden-Anwendungen kompatibel bleibt. Nach der Installation können Sie mit mehreren Accounts über die offiziellen Bitwarden-Clients oder mittels Weboberfläche („Web-Vault“) auf den Server zugreifen. Die Lösung eignet sich damit auch für Familien, Freunde und kleine Teams.

Für Vaultwarden empfiehlt sich eine Docker-fähige Linux-Distribution, beispielsweise Ubuntu 20.04 LTS oder Raspberry Pi OS . Falls Sie den Passwort-Server auf dem Raspi aufsetzen, brauchen dazu mindestens einen Raspberry Pi 3. Ob Sie das System mit grafischer Oberfläche an einem Monitor mit Maus und Tastatur bedienen oder sich mittels SSH von einem anderen Gerät aufschalten, bleibt Ihnen überlassen.



Der Login-Bildschirm der Bitwarden-Weboberfläche („Web-Vault“). Nach der Installation von Vaultwarden genügt beispielsweise ein Browser, um jederzeit Zugriff auf Ihre Passwörter zu haben. Um den Passwortmanager zu benutzen, sind keine Spezialkenntnisse mehr nötig.

Für die Installation braucht es lediglich den Zugriff auf die Kommandozeile. Nach der Einrichtung können Sie Vaultwarden mit einem Web-Browser von einem beliebigen Gerät aus konfigurieren. Die Weboberfläche enthält eine Web-App („Vault“), in der Nutzer ihren Account verwalten, und stellt Werkzeuge zur Administration über ein Admin-Panel bereit. Die Entwickler von Vaultwarden empfehlen, den Server als Docker-Container aufzusetzen. Diese Lösung hat den Vorteil, dass der Passwortmanager anderen Diensten auf dem System nicht in die Quere kommt.

Als Zutaten für eine solche Installation haben sich die folgenden Komponenten bewährt: Ein Reverse-Proxy ist der bequemste Weg, die Weboberfläche und das API verschlüsselt ins Web zu bekommen. Der Proxy nimmt Anfragen entgegen und liefert die Daten über eine HTTPS-Seite aus. In den nachfolgenden Beispielen wird angenommen, dass die TCP-Ports 80 und 443 des Systems nicht bereits belegt sind. Für diesen Zweck eignet sich der containerisierte Edge-Router Traefik, der sich die nötigen TLS-Zertifikate selbst besorgt und automatisch erneuert.

Docker-Compose erspart lange `docker run`-Befehle. Die Konfigurationen der Container sind dabei in einer Compose-Datei zusammengefasst, die durch eine .env-Datei mit Umgebungsvariablen ergänzt wird. Beispieldateien stellen wir in einem GitHub-Repository zur Verfügung, das Sie über ct.de/w7gs einsehen können.

Wenn Sie Vaultwarden im Heimnetz betreiben und von außen erreichen wollen, benötigen Sie einen festen Hostnamen, der auf die WAN-IPv4-Adresse des Routers verweist. Hier hilft ein DynDNS-Dienst. Dieser leitet Anfragen, die beim konfigurierten DynDNS-Hostnamen eingehen, an die jeweils aktuelle WAN-IPv4-Adresse des Routers weiter. Eine Anleitung für den kostenlosen DynDNS-Anbieter FreeDNS finden Sie online im Artikel „Mit DDNS auf’s Heimnetzwerk zugreifen“ (ct.de/w7gs). Sie haben die Wahl, welchen DynDNS-Dienst Sie verwenden. In dieser Anleitung dient die Adresse bitwarden.example.com als Beispiel.

Das Betriebssystem vorbereiten

Im ersten Schritt bringen Sie das System auf den neuesten Stand und installieren mit einem Skript die Docker-Engine für die Container-Laufzeitumgebung. Wenn es sich um eine frische Raspberry-Pi-OS-Installation handelt, dann starten Sie das

System nach dem Einspielen der Updates neu, bevor Sie Docker installieren.

```
sudo apt update
sudo apt upgrade
sudo reboot
curl -fsSL https://get.docker.com \
-o get-docker.sh
sudo sh get-docker.sh
```

Falls auf Ihrem System die Apache2-Utilities und das Versionsverwaltungswerkzeug Git nicht vorhanden sind, dann installieren Sie die Pakete ebenfalls.

```
sudo apt install apache2-utils
sudo apt install git
```

Docker-Compose beziehen Sie am besten aus dem Python-Paketmanager „pip“.

```
sudo apt install python3-pip
sudo pip3 install docker-compose
```

Eine Einführung in die Funktionsweise von Docker-Compose finden Sie im Artikel „Container-Komponist“ [1]. Um mit dem Docker-Daemon zu interagieren, brauchen Sie Systemverwalterrechte. Wenn Sie Docker-Befehle ohne vorangestelltes `sudo` verwenden möchten, müssen Sie Ihren Benutzer zur Gruppe „docker“ hinzufügen und sich erneut einloggen.

```
sudo usermod -aG docker user
passwd
```

Die nötige Software ist jetzt installiert und Docker einsatzbereit. Bevor es jedoch an das Einrichten der Container geht, muss sichergestellt werden, dass ihr Server – und damit Weboberfläche sowie API – auch aus dem Internet erreichbar ist. Wenn Sie Vaultwarden auf einem angemieteten Server betreiben wollen, überspringen Sie den nächsten Schritt.

Netzwerkverkehr dirigieren

Damit Anfragen zuverlässig beim Passwort-Server und nicht bei einem anderen Netzwerkgerät landen, sollte eine feste IP-Adresse vergeben werden. Handelt es sich um einen Raspi, dann tragen Sie diese für das aktive Netzwerk-Interface in der Datei /etc/dhcpcd.conf ein. Wenn der Raspi via Ethernet mit dem Netzwerk verbunden ist, dann wählen Sie das Interface eth0 (bei WLAN entsprechend wlan0). Passen Sie in der Datei die folgenden vier Zeilen an Ihr lokales Netzwerk an, etwa so:

```
interface eth0
static ip_address=192.168.0.184/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```

Starten Sie das Systen mit `sudo reboot` im Anschluss neu, damit die Änderungen wirksam werden. Richten Sie nun im Web-Interface Ihres Routers eine Portfreigabe (Port-Forwarding) ein. Anfragen von außen, die an TCP-Port 80 oder 443 eingehen, sollen an den Server durchgereicht werden und beim Reverse-Proxy Traefik landen. Als Protokoll wählen Sie TCP und tragen Port 80 ein. Der Name, der für den Dienst vergeben wird, ist beliebig. Den gleichen Vorgang wiederholen Sie für den Port 443.

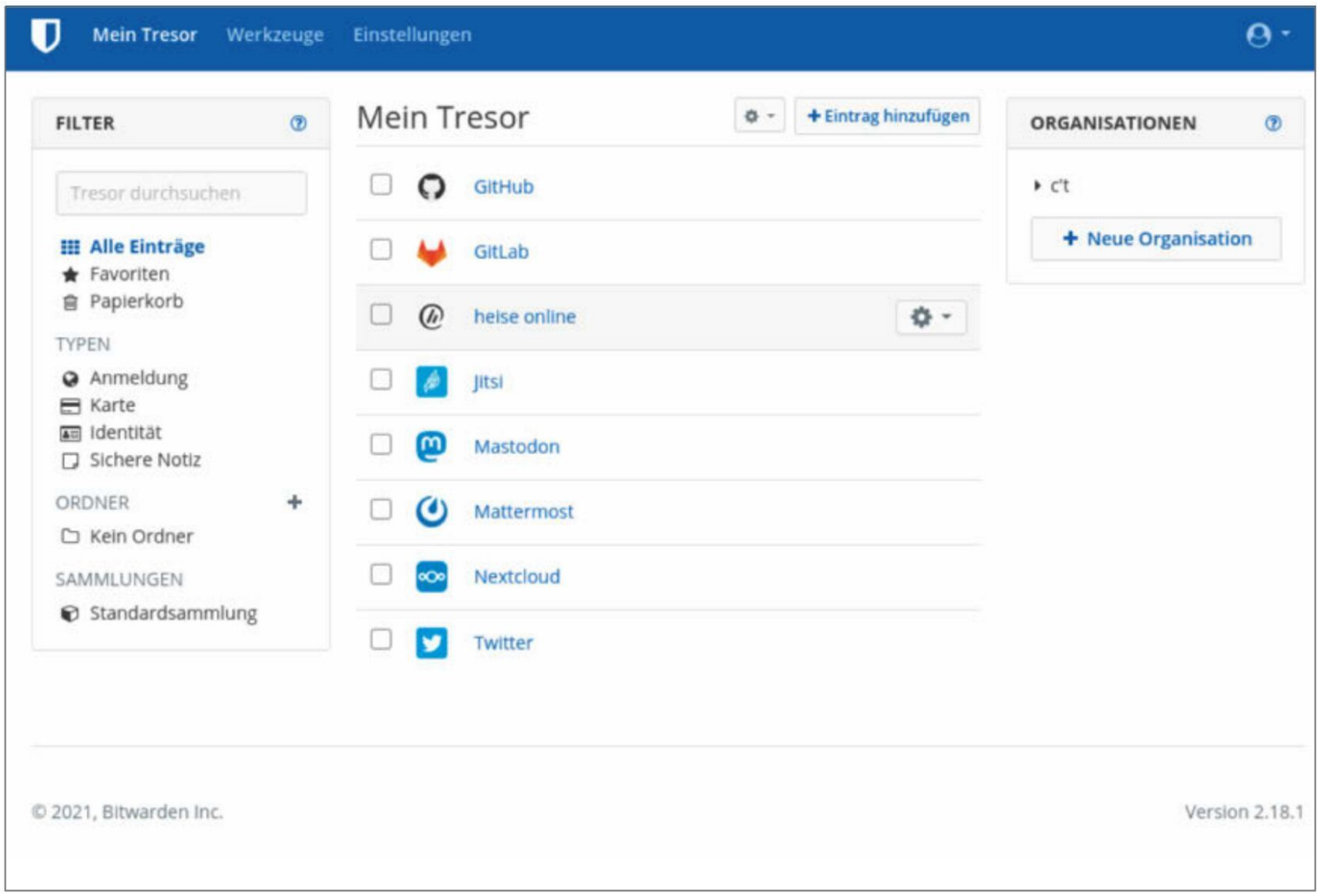
Damit der DynDNS-Dienst Ihrer Wahl in der Lage ist, auf die aktuelle WAN-IPv4-Adresse des Routers zu verweisen, benötigt er einen Gegenpart, der ihn mit dieser Information versorgt. Diese Aufgabe können die meisten modernen Router übernehmen. Die Konfiguration erledigen Sie ebenfalls im Webinterface des Routers. Wählen Sie im entsprechenden Menü den Anbieter aus und tragen Sie die abgefragten Daten, meist Login-Informationen und eine Update-URL, in die Maske ein. Wird Ihr DynDNS-Anbieter nicht gelistet, dann setzen Sie einen benutzerdefinierten Eintrag.

Transportverschlüsselung dank Traefik

Nach diesen Vorbereitungen geht es nun an die Container-Architektur. Als Speicherort für Container-Konfigurationen und Compose-Files auf dem Linux-Dateisystem hat sich das Verzeichnis /srv bewährt. Wenn Sie dort Dateien (be)schreiben wollen, müssen Sie mit `sudo` arbeiten oder sich Systemverwalterrechte einräumen. Klonen Sie das GitHub-Repository nach /srv und wechseln Sie danach in den Ordner.

```
sudo git clone \
https://github.com/ndi-ct/raspi-pwm \
/srv/raspi-pwm
cd /srv/raspi-pwm
```

Als Container-Blaupause steht im heruntergeladenen Repository eine YAML-Datei bereit. Die schreibt



Willkommen im Vault: In der Web-App können Sie Ihre Passwörter verwalten und Einstellungen vornehmen, beispielsweise Zwei-Faktor-Authentifizierung aktivieren.

fest, welche Container-Images aus dem Docker Hub geladen werden und wie diese konfiguriert sind. Umgebungsvariablen sind in die zugehörige .env-Datei ausgelagert. Diese wird standardmäßig nicht angezeigt. Bei Bedarf verschaffen Sie sich mit `ls -a` einen Überblick. Der Reverse-Proxy benötigt darüber hinaus noch die Dateien traefik.toml und dynamic.yml, die Anweisungen zur Beschaffung von Zertifikaten und zur Regelung des Netzwerkverkehrs geben. Diese finden Sie im Ordner config-files.

Öffnen Sie traefik.toml mit dem Texteditor Ihrer Wahl und tragen Sie in der Zeile `email = "mail@example.com"` eine gültige Mail-Adresse ein (die Anführungszeichen bleiben bestehen). An diese Adresse sendet Let's Encrypt Warnungen, falls es Probleme mit dem erteilten Zertifikat gibt. Traefik legt die Zertifikate, die es automatisiert mittels ACME-Protokoll besorgt, in der Datei acme.json ab. Diese darf nur für Root lesbar sein. Erstellen Sie im Ordner config-files eine leere Datei mit dem Namen acme.json und passen Sie die Rechte an.

```
sudo touch acme.json
sudo chmod 600 acme.json
```

Danach legen Sie ein virtuelles Netzwerk für den Reverse-Proxy an.

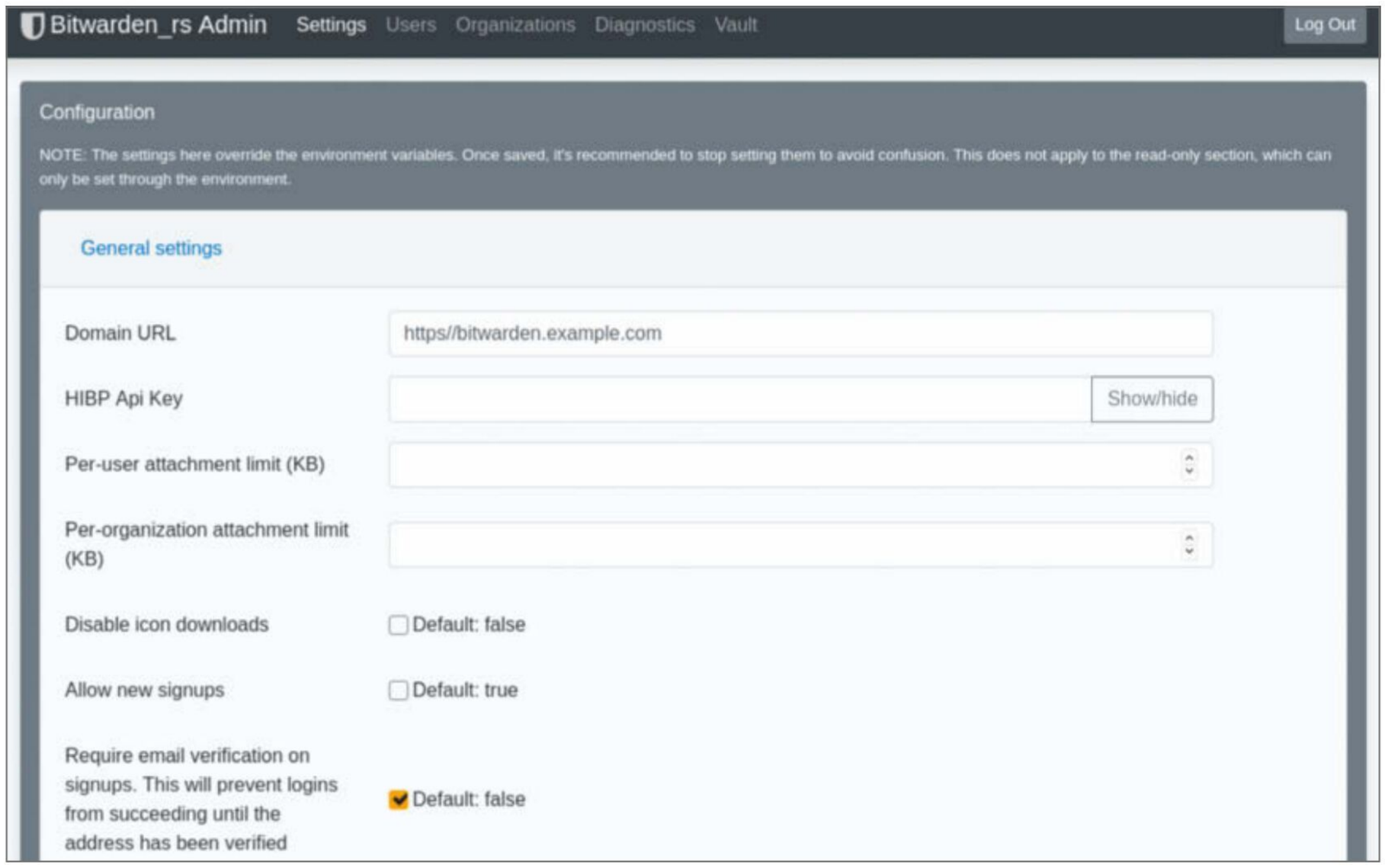
```
sudo docker network create \
traefik_reverse_proxy
```

Tresor aufstellen

Wechseln Sie jetzt in das Verzeichnis /srv/raspi-pwm. Um sich einen Überblick über den Container-Aufbau zu verschaffen, können Sie sich den Inhalt des Compose-File ausgeben lassen:

```
cat docker-compose.yaml
```

Die Images „vaultwarden/server“ und „traefik“ sind mit Versionsnummern versehen, um Ihnen unschöne Überraschungen zu ersparen. Ohne dieses Tag



Das Admin-Panel ist die Schaltzentrale von Vaultwarden. Hier haben Sie Zugriff auf wichtige Einstellungen und können die Nutzer verwalten.

lädt Docker stets das neueste Image vom Docker Hub herunter. Das kann dazu führen, dass der Container plötzlich nicht mehr startet, weil die neue Version Änderungen an den Konfigurationsdateien erfordert. Prüfen Sie regelmäßig, ob für Ihre Images (Sicherheits-)Updates vorliegen. Wenn das der Fall ist, dann ersetzen Sie das ausgediente Image durch ein neueres. Die verwendeten Images haben wir unter ct.de/w7gs verlinkt.

Der Parameter `restart: unless-stopped` sorgt dafür, dass die Container auch nach einem Neustart des Raspi wieder anlaufen, es sei denn, Sie haben zuvor einen Container manuell gestoppt. Der `volumes`-Abschnitt legt fest, wo der Bitwarden-Container persistente Daten speichert:

```
volumes:
  - /var/docker/bitwarden:/data
```

Das Verzeichnis /data im Container-Dateisystem entspricht /var/docker/bitwarden im Host-Dateisystem. Hier liegt auch die SQLite-Datenbankdatei db.sqlite3. Sie sollten das Verzeichnis /var/docker/bitwarden regelmäßig sichern.

Im Compose-File finden Sie die Variablen `${BITWARDEN_URL}`, `${WEBSOCKET_ENABLED}` und `${ADMIN_TOKEN}`. Die Werte holt Docker-Compose sich aus der .env-Datei. Öffnen Sie die Datei mit einem Texteditor.

```
BITWARDEN_URL=bitwarden.example.com
WEBSOCKET_ENABLED=true
ADMIN_TOKEN=
```

Tragen Sie hinter `BITWARDEN_URL=` Ihre DynDNS-Adresse ein. Die Adresse wird als Host an Traefik übergeben. Der Eintrag `WEBSOCKET_ENABLED=true` sorgt dafür, dass Passwörter und andere Einträge im Vault bei jeder Änderung synchronisiert werden, sofern der entsprechende Client dies unterstützt („Live-Sync“). Wir empfehlen, Vaultwarden über den Admin-Bereich der Weboberfläche zu konfigurieren. Zum Aktivieren des Admin-Panel setzen Sie hinter `ADMIN_TOKEN=` ein Token ein. Erzeugen Sie dazu auf der Kommandozeile einen langen String mit folgendem Befehl.

```
openssl rand -base64 48 | tr -d /=
```

Mit dem eingetragenen Token können Sie sich später im Admin-Panel der Weboberfläche authentifizieren. Es steht Ihnen frei, die Einstellungen, die das Admin-Panel bietet, manuell vorzunehmen, indem Sie in der .env-Datei weitere Umgebungsvariablen setzen und diese in der Datei docker-compose.yaml referenzieren. Die Entwickler von Vaultwarden stellen im GitHub-Wiki des Projektes eine umfangreiche Beispieldatei zur Verfügung, die Sie über ct.de/w7gs finden. Löschen Sie in diesem Fall die Zeile `ADMIN_TOKEN=` in der .env-Datei und die zugehörige Referenz aus dem Compose-File.

In unserem Beispiel haben wir das Admin-Panel zusätzlich mit einer Basisauthentifizierung, bestehend aus Username und Passwort, abgesichert. Dazu dient eine Middleware des Reverse-Proxy Traefik, die bereits in der Compose-Datei angelegt ist. Wer im Browser bitwarden.example.com/admin ansteuert, muss zunächst die Login-Daten eingeben, um die Seite zu sehen und kann sich erst im zweiten Schritt mit dem Token als Admin authentifizieren. Die nötigen Zugangsdaten erzeugen Sie mit folgendem Befehl:

```
htpasswd -nb user password\
 | sed -e s/\\$/\\$\\$/g
```

Dieser Befehl erzeugt das Zugangsdaten-Paar „user:password“, wobei „password“ als Hash ausgegeben wird. Ersetzen Sie „user“ und „password“ durch sichere Zugangsdaten. Öffnen Sie dann die Datei docker-compose.yaml mit einem Texteditor und ersetzen Sie in der folgenden Zeile „user:password“ durch die Ausgabe des Befehls.

```
- "traefik.http.middlewares.admin-auth.basicauth.
users=user:password"
```

Speichern Sie im Anschluss die Datei docker-compose.yaml und starten Sie die beiden Container:

```
sudo docker-compose up -d
```

Docker lädt nun zunächst die Images aus der Container-Registry. Der Reverse-Proxy kümmert sich um das Zertifikat von Let‘s Encrypt und liefert Web-Vault und Admin-Panel von Vaultwarden über die konfigurierte DynDNS-Adresse aus. Geben Sie den Containern dabei etwas Zeit zum Starten. Auf einem Raspberry Pi 4 mit 4 GByte Arbeitsspeicher nimmt der erste Startvorgang einige Minuten in Anspruch.

Den Status der Container können Sie mit dem Befehl `docker ps` einsehen. Sobald der Vaultwarden-Container den Status „up x minutes (healthy)“ ausgibt, können Sie das Web-Vault in einem Browser unter der Adresse bitwarden.example.com aufrufen. Für das Admin-Panel hängen Sie /admin an die URL an. Sollte Ihnen irgendwo ein Fehler unterlaufen sein, dann hilft der Befehl `docker logs` gefolgt vom Namen des Containers bei der Diagnose.

Tresortür abschließen

Registrieren Sie einen neuen Nutzer und vergeben Sie dabei ein sicheres Master-Passwort. Jetzt können Sie sich in das Web-Vault einloggen und Ihr Nutzerkonto einrichten. Wer zuvor einen anderen Passwortmanager genutzt hat, findet sich hier schnell zurecht. Die Kategorie „Mein Tresor“ listet Einträge wie gespeicherte Anmeldedaten und sichere Notizen auf. Diese können Sie als Favoriten markieren oder in Ordner sortieren. In „Werkzeuge“ finden Sie einen Passwortgenerator, Berichte über schwache oder kompromittierte Passwörter und die Funktionen zum Importieren und Exportieren von Passwort-Sammlungen. Bitwarden beherrscht die meisten geläufigen Formate (.csv, .json, .xml, .txt). Hier können Sie auch die eigene Passwortdatenbank exportieren. Wählen Sie dabei vorzugsweise das Format „.json (encrypted)“, sonst exportiert Bitwarden die Zugangsdaten im Klartext.

Sie sollten für den Login unbedingt einen zweiten Faktor verlangen. Andernfalls könnte es Angreifern potenziell gelingen, sich mit Brute-Force-Angriffen Zugriff auf Ihre Passwortdatenbank zu verschaffen. Die nötigen Handgriffe nehmen Sie in den Einstellungen im Untermenü „Zwei-Faktor-Authentifizierung“ vor. Der Passwortmanager bietet Zwei-Faktor-Authentifizierung mittels 2FA-Apps wie Authy, Duo oder Microsoft Authenticator, via E-Mail und mit FIDO-kompatiblen Hardware-Sicherheitsschlüsseln an.

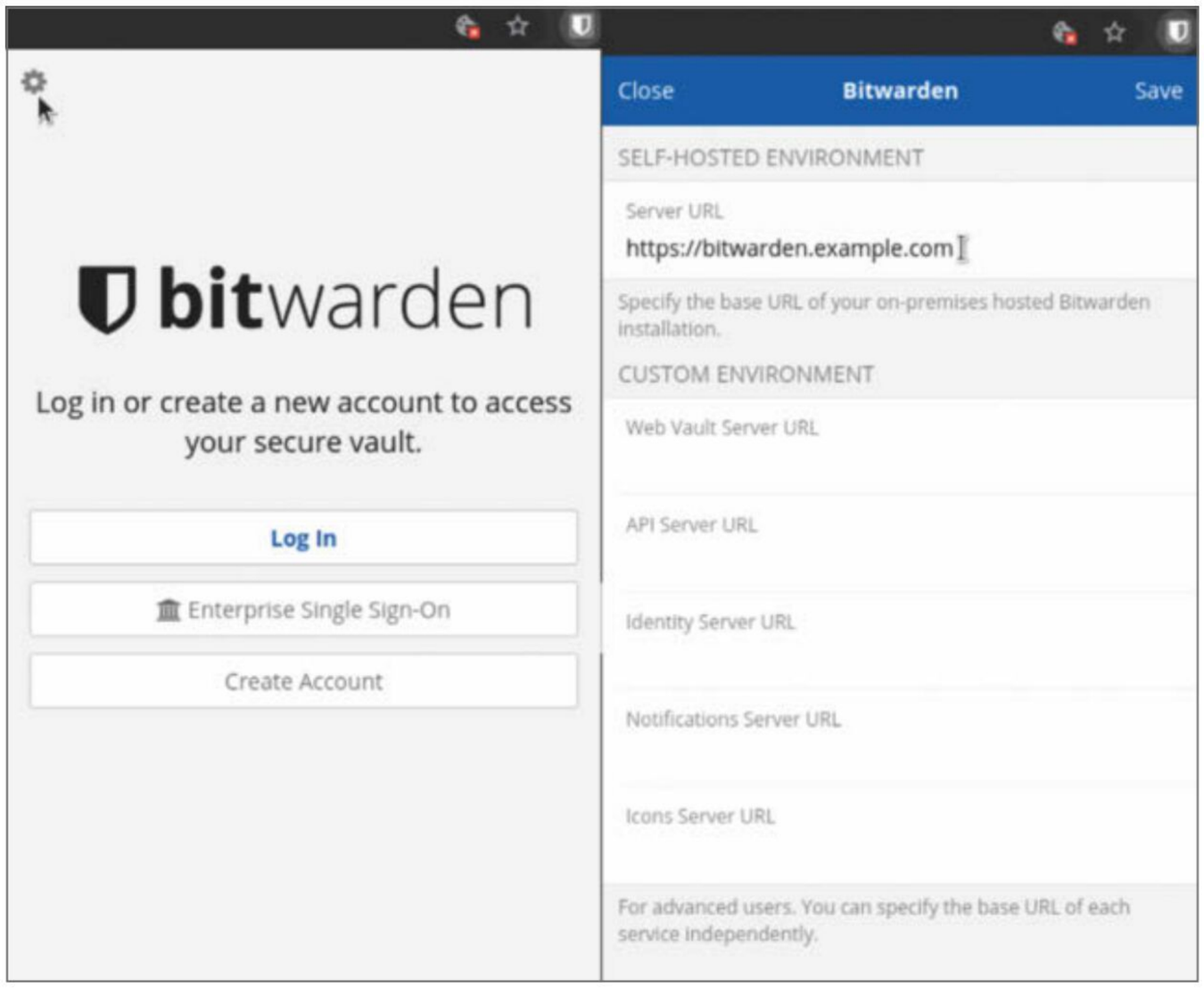
Sobald ein zweiter Faktor eingerichtet ist, können Sie im Menü „Zwei-Faktor-Authentifizierung“ einen Wiederherstellungsschlüssel anzeigen lassen. Falls Sie beispielsweise Ihr Mobiltelefon und damit Zugriff auf den zweiten Faktor verlieren, ist der Wiederherstellungsschlüssel die letzte Möglichkeit, erneut Zugang zum Vault zu bekommen. Notieren Sie den Schlüssel und verwahren Sie ihn an einem sicheren Ort. Tritt das Szenario ein, dann rufen Sie im Browser die Adresse bitwarden.example.com/#/recover-2fa/ auf und folgen Sie den Anweisungen. Wenn Sie direkt mehrere Methoden zur Zwei-Faktor-Authentifi-

zierung einrichten, dann kommen Sie erst gar nicht in diese Situation.

Aktuell kann noch jede Person, die die URL Ihrer Web-Vault kennt, ein Konto anlegen. Wer also keinen Gratis-Passwortmanager ins Internet hängen möchte, tut gut daran, neue Registrierungen zu deaktivieren. Rufen Sie das Admin-Panel im Browser unter der Adresse bitwarden.example.com/admin auf. Wir empfehlen, im Admin-Panel einige Einstellungen vorzunehmen, um die Angriffsfläche Ihres Passwort-Servers zu verringern.

Im Bereich „General“ müssen Sie den korrekten Wert im Feld „Domain URL“ eintragen, damit automatisch versandte Mails die richtigen Links enthalten. Verbieten Sie Registrierungen auf der Hauptseite, indem Sie den Haken beim Eintrag „Allow new signups“ entfernen. Die Schaltfläche und das Eingabeformular zum Registrieren bleiben zwar weiterhin sichtbar, Registrierungen sind aber nicht mehr möglich. Im Bereich „SMTP Email Settings“ sollten Sie einen SMTP-Postausgangsserver konfigurieren, damit der Passwort-Server Mails verschicken kann, etwa, um Accounts zu verifizieren oder zu informieren, falls eine Anmeldung in Ihrem Account von einem zuvor unbekannten Gerät erfolgt ist. Im Admin-Panel können Sie auch dann weitere Nutzer via Mail einladen, wenn die Registrierung auf der Hauptseite deaktiviert ist. Teilen Sie den Passwort-Server beispielsweise mit Freunden, sollten Sie darüber nachdenken, ihnen zu verbieten, weitere Nutzer einzuladen. Dazu entfernen Sie den Haken bei „Allow Invitations“ im Bereich „General“.

Wem das nicht genügt, der deaktiviert das Web-Vault mit der Umgebungsvariable WEB_VAULT_ENABLED=false oder betreibt Vaultwarden ausschließlich im Heimnetz mit selbsterzeugten Zertifikaten. Um dann trotzdem von unterwegs auf den Passwort-Server zuzugreifen, müssten Sie sich mit einer VPN-Lösung behelfen. Die Entwickler raten davon ab, die integrierte Methode ROCKET_TLS für HTTPS zu nutzen. Stattdessen empfehlen sie, private Instanzen mit dem Reverse-Proxy Caddy und Let's-Encrypt-Zertifikaten abzusichern. Eine Anleitung für das nicht ganz triviale Verfahren steht auf GitHub im Wiki des Projekts zur Verfügung, das wir unter ct.de/w7gs verlinkt haben.



Bitwarden-Clients existieren für diverse Plattformen, hier sehen Sie das Browser-Add-On in Chromium. Damit der Client den Passwort-Server ansteuern kann, braucht er die Server-URL.

Den Passwortmanager nutzen

Der Passwort-Server ist jetzt einsatzbereit. Vaultwarden bietet viele Features, die für gewöhnlich einen Bitwarden-Premium-Account voraussetzen. Einige Funktionen fehlen jedoch, etwa Push-Benachrichtigungen in iOS und Android, Single-Sign-On oder Notfall-Vollmachten (Emergency Access). Welche Unterschiede zum offiziellen Bitwarden-Server bestehen, haben die Entwickler im Wiki des Projekts in einer Liste festgehalten, die Sie in der Linksammlung unter ct.de/w7gs abrufen können.

Sie können zum Speichern, Verwalten und Synchronisieren von Passwörten neben dem Web-Vault auch die Bitwarden-Browser-Erweiterungen sowie die Apps für Android und iOS nutzen. Auch Desktop-Clients für Windows, Linux, macOS und sogar Tools für die Kommandozeile sind verfügbar. Eine vollständige Liste aller Clients finden Sie auf der offiziellen Bitwarden-Website, die wir unter ct.de/w7gs verlinkt haben. Damit die Anwendungen mit der Server-API kommunizieren können, müssen Sie die URL Ihrer Vaultwarden-Instanz eintragen. Das zuständige Menü verbirgt sich hinter dem Zahnrad in der linken oberen Ecke des Anmeldefensters.

Doch Vorsicht: Die Schaltfläche „abmelden“ in den Clients ist irreführend und bedeutet „vom Server abmelden“. Wenn Sie sich vollständig abmelden und die Server-API nicht erreichbar ist, dann verweigert der Bitwarden-Client einen neuen Login. Um sich dann erneut anzumelden, muss eine Verbindung zum Server bestehen. Wählen Sie darum stets die Option „Jetzt sperren“, um Ihr Nutzerkonto zu sperren, es sei denn, Sie beabsichtigen, das Nutzerkonto zu wechseln oder den Client an einem anderen Server anzumelden.

Sie können jetzt loslegen und den Passwort-Tresor befüllen. Wenn Sie zukünftig im Web eine Login-Seite ansteuern und Anmeldedaten eingeben, bietet beispielsweise die Bitwarden-Browsererweiterung an, die Login-Daten im Vault zu speichern. Für neue Logins kann der integrierte Passwortgenerator zufällige, sichere Passwörter erzeugen. Falls bereits Anmeldedaten im Vault gespeichert sind, kann Bitwarden Login-Formulare automatisch ausfüllen. Das Auto-Fill-Verhalten können Sie für einzelne Seiten an ihre Vorstellungen anpassen. Gespeicherte Passwörter werden von nun an automatisch synchronisiert.

Fazit

Mit Vaultwarden verwandeln Sie ihr System im Handumdrehen in einen Passwortspeicher mit Cloud-Funktionalität, müssen dafür jedoch Ihre Passwortsammlung nicht aus der Hand geben. Mit dem Zugewinn an Freiheit kommt allerdings auch Verantwortung: Nehmen Sie regelmäßig Backups vor und prüfen Sie die Container-Images auf Sicherheitsupdates. Richtig angewendet ersetzt Vaultwarden einen kostenpflichtigen Passwortmanager und eignet sich optimal für die Nutzung in kleinen Teams oder gemeinsam mit Freunden oder der Familie. (ndi) c't

Literatur

[1] Jan Mahn, **Container-Komponist**, Docker-Container mit Docker-Compose einrichten, c't 26/2018 S. 148

Linksammlung zum Projekt

ct.de/w7gs





Bild: Thorsten Hübner

Mitgliederversamm-lungen virtuell

Die Mitgliederversammlung des Sportvereins könnte in diesem Jahr wieder etwas anders ausfallen: Statt im Vereinsheim trifft man sich per Videokonferenz. Damit alle Teilnehmer Anträge, Wahlen und die Tagesordnung im Blick haben, hilft die Versammlungssoftware OpenSlides.

Von **Jan Mahn**

Mitgliederversammlungen von Vereinen und Verbänden, aber auch Hauptversammlungen von Aktiengesellschaften und Parteitage bestehen aus den immer gleichen Bausteinen: Es gibt Tagesordnungen, Berichte, Anträge und Wahlen. Damit alle Teilnehmer bei solchen Veranstaltungen den Überblick behalten, dem Ablauf folgen und ihn selbst mitgestalten können, hat ein Team deutscher Entwickler 2011 das Open-Source-Projekt OpenSlides ins Leben gerufen. Die Webanwendung war zunächst für klassische Versammlungen vor Ort gedacht – so kann man etwa mehrere Beamer mit Tagesordnung

und Rednerliste über den Browser fernsteuern. Zum Einsatz kam die Software seitdem schon in großen Hallen bei Zusammenkünften von Unternehmen, Parteien und Gewerkschaften – die Liste der Referenzen wird immer länger.

In diesem Jahr kommt ein weiterer Einsatzbereich hinzu: Wer eine Versammlung über ein Videokonferenzsystem seiner Wahl durchführen muss, kann den Teilnehmern mit OpenSlides vor, während und nach der Versammlung die benötigten Unterlagen bereitstellen. Anträge müssen nicht mehr per Rundmail eingereicht werden, sondern haben einen Platz in der Oberfläche. OpenSlides ist kein weiteres Videokonferenztool, sondern eine eigenständige Anwendung, lässt sich aber gut in eine Versammlung per Videokonferenz integrieren, indem ein Teilnehmer einen Bildschirm mit der Projektor-Ansicht freigibt.

Für große Organisationen und Unternehmen bieten die Entwickler Hosting- und Supportpakete an (siehe ct.de/wfph), sowohl für Vor-Ort-, als auch für virtuelle Versammlungen. Wenn Sie das System nicht selbst administrieren wollen oder können, bekommen Sie eine fertige Instanz zur Miete und unterstützen mit dem Geld die Entwicklung.

Sie können die Open-Source-Software aber auch selbst betreiben. Wie immer führen mehrere Wege zum Ziel. Die Anwendung ist in Python geschrieben – wenn Sie Python-Nutzer und mit den Werkzeugen vertraut sind, laden Sie das Paket mit:

```
pip install openslides
```

Anschließend starten Sie das Programm mit `openslides`. Damit läuft ein lokaler Webserver auf Port 8000. Unter der Adresse localhost:8000 können Sie über den Browser darauf zugreifen und OpenSlides erkunden.

Soll die Instanz auch über das Internet erreichbar sein, brauchen Sie einen Server, der mit Port 443 im Internet hängt und eine Domain oder Subdomain, die auf dessen Adresse zeigt – ein kleiner Linux-Mietserver reicht aus. Weil man eine Plattform, an der sich Nutzer anmelden sollen, nicht ohne Verschlüsselung ins Internet hängt, müssen Sie sich auch um die Beschaffung eines Zertifikats mit Let's Encrypt Gedanken machen. Um Ihnen all diese Schritte abzunehmen, haben wir OpenSlides in Container verpackt und in unsere Container-Zusammenstellung Team-Container gesteckt, die wir in Ausgabe 9/2020 vorgestellt haben [1]. Die Zusammenstellung selbst finden Sie bei GitHub (siehe ct.de/wfph).

Laden Sie das Repository auf dem Server herunter und installieren Sie zunächst das Grundgerüst, das unter anderem die Zertifikate abholt, wie im angegebenen Artikel oder in der Readme beschrieben. Öffnen Sie dann die Datei `values-openslides.yaml` und passen Sie den Namen Ihrer Subdomain (zum Beispiel `versammlung.example.org`) an. Anschließend fahren Sie Ihre OpenSlides-Sitzung hoch:

```
helm install openslides ⏎
↳team-openslides --values ⏎
↳values-openslides.yaml
```

Nach wenigen Minuten hat Ihre OpenSlides-Instanz ein Zertifikat und antwortet unter https://versammlung.example.org. Bei Problemen hilft ein Blick in die GitHub-Issues oder den zugehörigen Artikel.

Grundeinstellungen

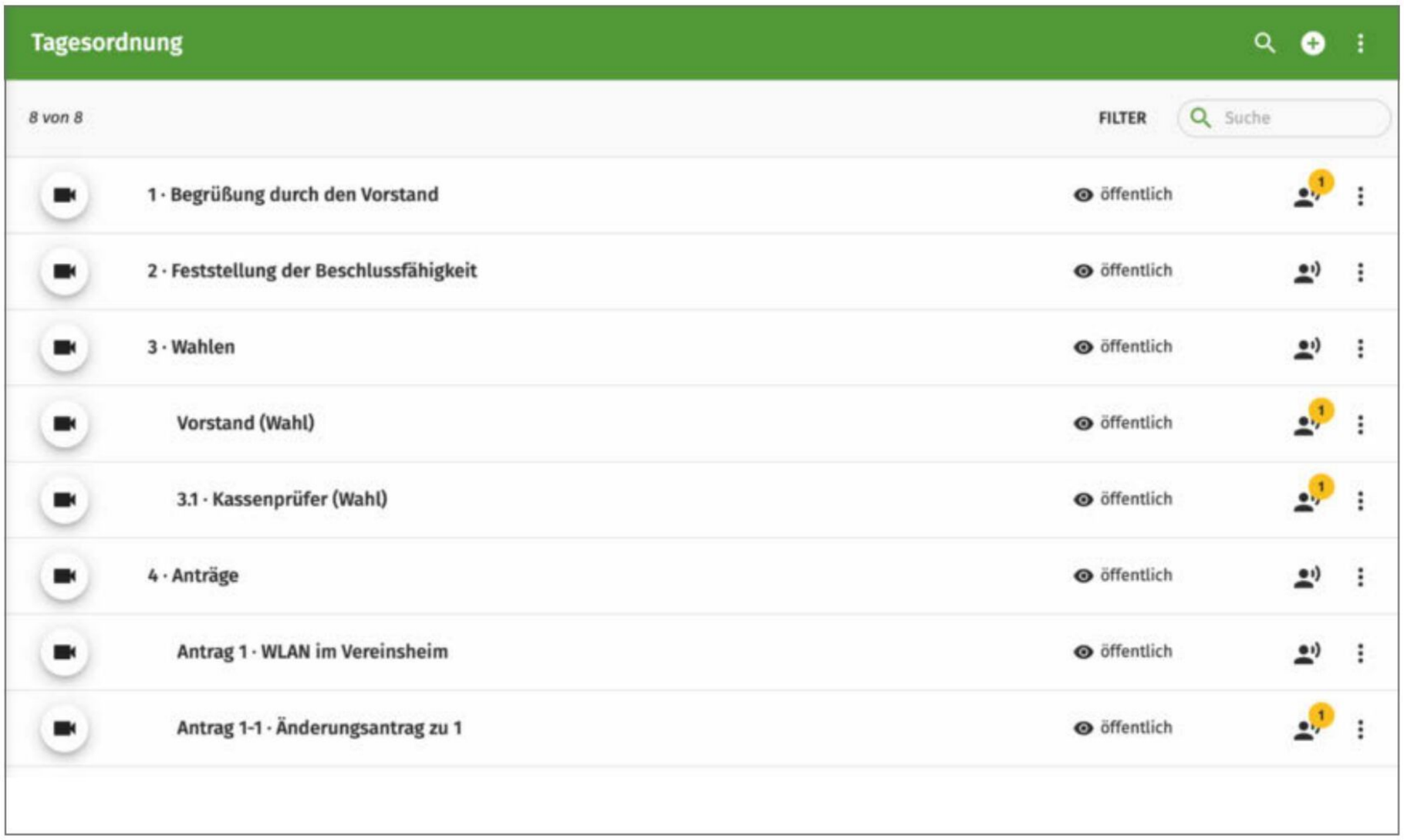
Wenn die Plattform läuft, muss man sie zunächst für die eigene Veranstaltung einrichten und alle Beispieltexte austauschen. Beim ersten Besuch müssen Sie sich mit dem Admin-Account (Nutzername und Kennwort: „admin“) einloggen. Anschließend sollten Sie das Kennwort direkt ändern: Links im Menü mit einem Klick auf „Administrator“ finden Sie den Dialog dafür.

Dann beginnt die Anpassungsarbeit: Auf der Startseite finden Sie oben rechts ein Stift-Symbol, um den Begrüßungstext zu ändern. Anschließend sollten Sie über „Einstellungen“ den Punkt „Allgemein“ öffnen und dort die Daten Ihrer Sitzung eintragen und passende Texte formulieren. Oben rechts speichern Sie die Änderungen.

Anschließend können Sie mit der Tagesordnung beginnen, um die Funktionsweise der Software kennenzulernen. Die Verwaltung der Tagesordnungspunkte finden Sie links im Menü. Über das Plus-Symbol oben rechts erstellen Sie neue Punkte – jeder Eintrag kann einen beschreibenden Text enthalten und mit Anhängen erweitert werden. Der Vorstand kann hier zum Beispiel seinen Vorstandsbericht schon vorab veröffentlichen und auf die Rundmail verzichten.

In der Übersicht finden Sie die wichtigsten Funktionen hinter den drei Punkten oben rechts. Über „Nummerierung“ fügen Sie Nummern für die Tagesordnungspunkte hinzu, mit „Sortieren“ ändern Sie die Reihenfolge.

Über den Punkt „Projizieren“ veröffentlichen Sie die Tagesordnung auf dem Projektor – eine der nütz-



Die Tagesordnung verwalten und sortieren Sie in der Oberfläche. Änderungen erscheinen sofort auf dem Projektor.

lichsten Funktionen der Software. Um einen Projektor zu betreiben, öffnen Sie die Adresse `<IP-Adresse Ihrer Instanz>/projector/1` in einem Browserfenster und versetzen Sie dieses in den Vollbildmodus. Im Vergleich zur klassischen PowerPoint-Präsentation ist diese Ansicht deutlich komfortabler: Wenn Sie die Tagesordnung ändern, erscheint die Anpassung sofort auf den Projektoren.

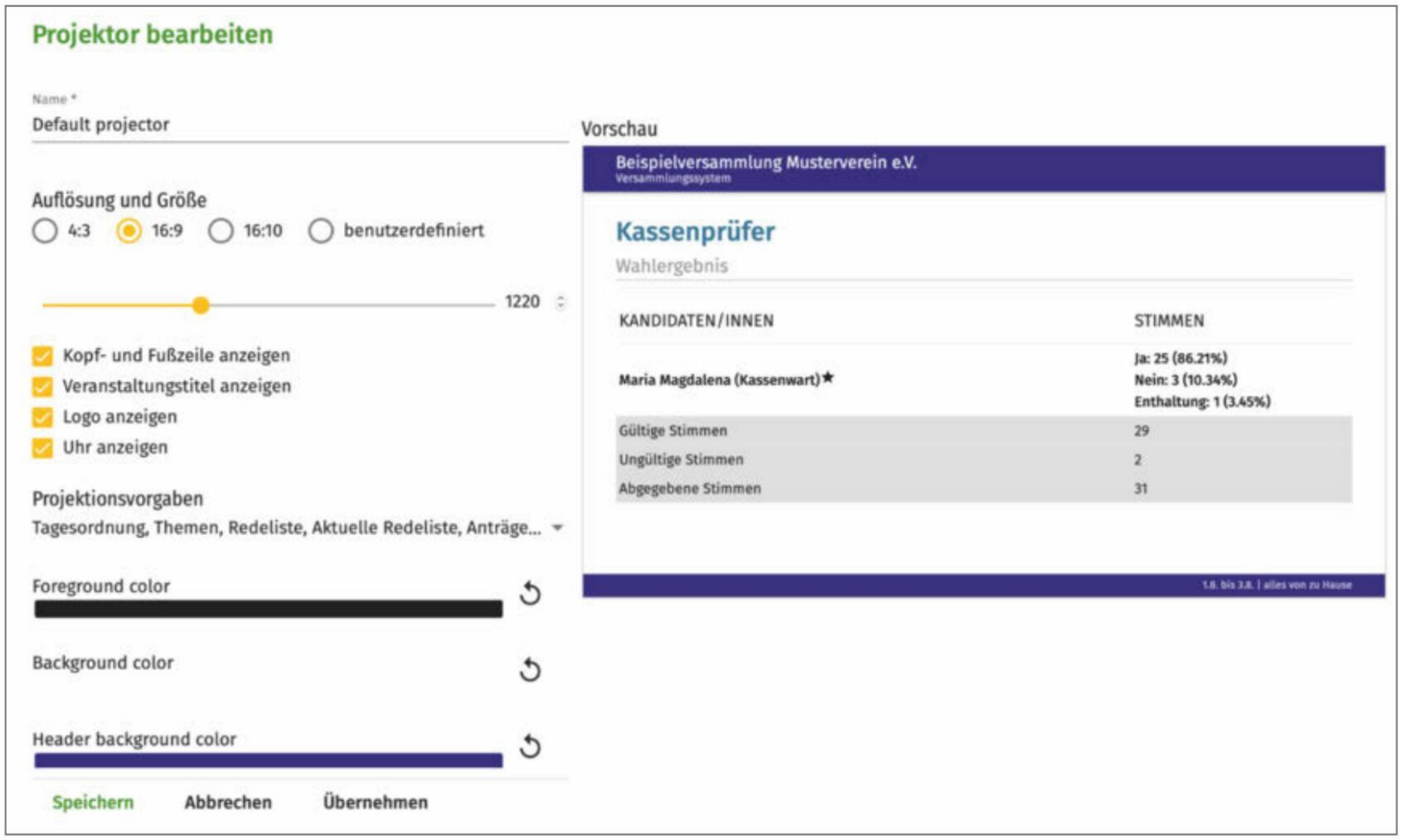
Bei einer Vor-Ort-Konferenz könnte man auch beliebig viele Projektoren mit demselben Inhalt gleichzeitig betreiben und im Gebäude aufstellen – mit klassischen Präsentationen wäre das nur mit teurer Video-Hardware und langen Kabeln möglich. Voraussetzung für einen OpenSlides-Projektor ist ein Computer mit Browser (mit Chrome haben wir gute Erfahrungen gemacht) im gleichen Netzwerk. Mit vergleichsweise wenig Aufwand könnte man zum Beispiel einen Monitor im Foyer aufstellen (viele Fernseher haben sogar einen eingebauten Browser, ansonsten reicht ein Raspberry Pi). Auch unterschiedliche Projektoren unterstützt OpenSlides: Links im Menü unter „Projektor“ können Sie als Admin einen weiteren anlegen, etwa einen für die Rednerliste. Dort können Sie die Projektoren auch fernsteuern, etwa zoomen und scrollen, wenn Inhalte nicht auf eine Bildschirmseite passen. Über das Stiftsymbol über dem Monitor können Sie weitere Einstellungen vornehmen und zum Beispiel die Farben durch Ihre Vereinsfarben austauschen oder die Ansicht auf 16:9 ändern (für Videokonferenzen und die meisten Beamer eine sinnvolle Idee).

Immer, wenn Sie in der Oberfläche das Projektor-Symbol an einem Objekt sehen, können Sie es auf einen der Projektoren werfen. Das funktioniert zum Beispiel auch mit Dateien (Bildern und PDF-Dateien), die Sie unter dem Menüpunkt „Dateien“ hochgeladen haben.

Teilnehmerliste

Um Teilnehmer hinzufügen zu können, muss man Mitglied der Gruppe „Administratoren“ sein, wie zum Beispiel der voreingestellte Admin-Account. Links im Menü unter „Teilnehmende“ sehen Sie alle Accounts. Über das Plus oben rechts fügen Sie neue hinzu – eine mühsame Aufgabe, wenn der Verein größer ist. Es geht aber leichter: Hinter den drei Punkten finden Sie den Punkt „Importieren“. Der Import-Dialog nimmt Text- oder CSV-Dateien entgegen. Die CSV-Datei muss im richtigen Format aufgebaut sein, eine leere Beispieldatei finden Sie im Dialog zum Download.

Auf virtuellen Projektoren präsentieren Sie Tagesordnung, Rednerlisten und Anträge.



Grundsätzlich kommen zwei Szenarien infrage: Entweder veröffentlichen Sie die Instanz von OpenSlides im Internet, an der sich jeder Teilnehmer anmelden und in der Oberfläche arbeiten kann, oder Sie nutzen die Oberfläche „nur“ als erweitertes Präsentationswerkzeug. Dann reicht eine lokale Instanz auf dem PC eines Veranstalters. Dieser öffnet die Projektoransicht im Browser, gibt diesen Bildschirm in der Videokonferenz-Software frei und steuert sie über ein weiteres Browserfenster (mehrere Bildschirme sind dann unbedingt nötig). Auch in diesem Szenario sollte man aber alle Teilnehmer im System anlegen – dann kann man die komfortable Rednerlistenfunktion mit Auto-Vervollständigung nutzen. Ein Moderator beobachtet, wer seine virtuelle Hand hebt (diese Funktion haben fast alle Videokonferenz-Werkzeuge) und setzt die Teilnehmer auf die Liste zum aktuellen Tagesordnungspunkt. Hinter fast allen Elementen in der Software findet man das Sprecher-Icon, über das man zur Rednerliste gelangt.

Hat man es mit einem technikaffinen Publikum zu tun, kann man den Teilnehmern Zugangsdaten geben und die Instanz ins Internet hängen. Am besten verschickt man die Zugangsdaten mit dem initialen Kennwort schon mit der fristgerechten Einladung zur Veranstaltung – die geplante Tagesordnung verschickt man zur Abwechslung mal nicht als PDF-Anhang, sondern legt sie direkt im System an. Die Zugangsdaten können Sie zum Beispiel als PDF-Dateien oder als gesammelte CSV-Liste herunterladen. Vorher sollten Sie unter „Einstellungen / Teilnehmende“ die externe Adresse des Systems und einen Begrüßungstext hinterlegen. Diese Informationen werden auf die PDF-Seiten gedruckt.

Haben alle Besucher ihre Zugangsdaten schon vorab, kann die inhaltliche Arbeit schon vor der eigentlichen Konferenz beginnen. Jeder Nutzer kann sein eigenes Profil anpassen, ein paar Sätze über sich schreiben und über den Menüpunkt „Teilnehmende“ die Profile der anderen Teilnehmer einsehen. Wenn die Sitzung losgeht, unterstützt OpenSlides die Ausrichter bei der Feststellung der Beschlussfähigkeit. Nützlich ist dafür der Dialog, um die Anwesenheit festzuhalten. Öffnen Sie den Dialog „Einstellungen / Ansicht zur Teilnehmeranwesenheit aktivieren“. Hinter den drei Punkten in der Ansicht „Teilnehmende“ finden Sie jetzt den Punkt „Anwesenheit“. In dieser Ansicht kann man die Teilnehmernummer (diese generiert OpenSlides beim Anlegen) eintippen und damit eine Person auf anwesend oder abwesend setzen. Das hilft bei großen virtuellen Sitzungen ungemein: In Videokonferenzsystemen

wie Zoom gibt es die Möglichkeit, dass neue Besucher zunächst in einem virtuellen Vorraum ankommen. Dort kann sie ein Moderator in Empfang nehmen und nach der Teilnehmernummer fragen. Schon bevor es losgeht, weiß die Sitzungsleitung, ob die Beschlussfähigkeit in Gefahr ist.

Will man sich die Arbeit erleichtern, fordert man alle Stimmberechtigten auf, sich immer per Chat des Konferenzsystems mit ihrer Teilnehmernummer abzumelden, wenn sie die virtuelle Versammlung verlassen. Ein Admin ändert dann den Status.

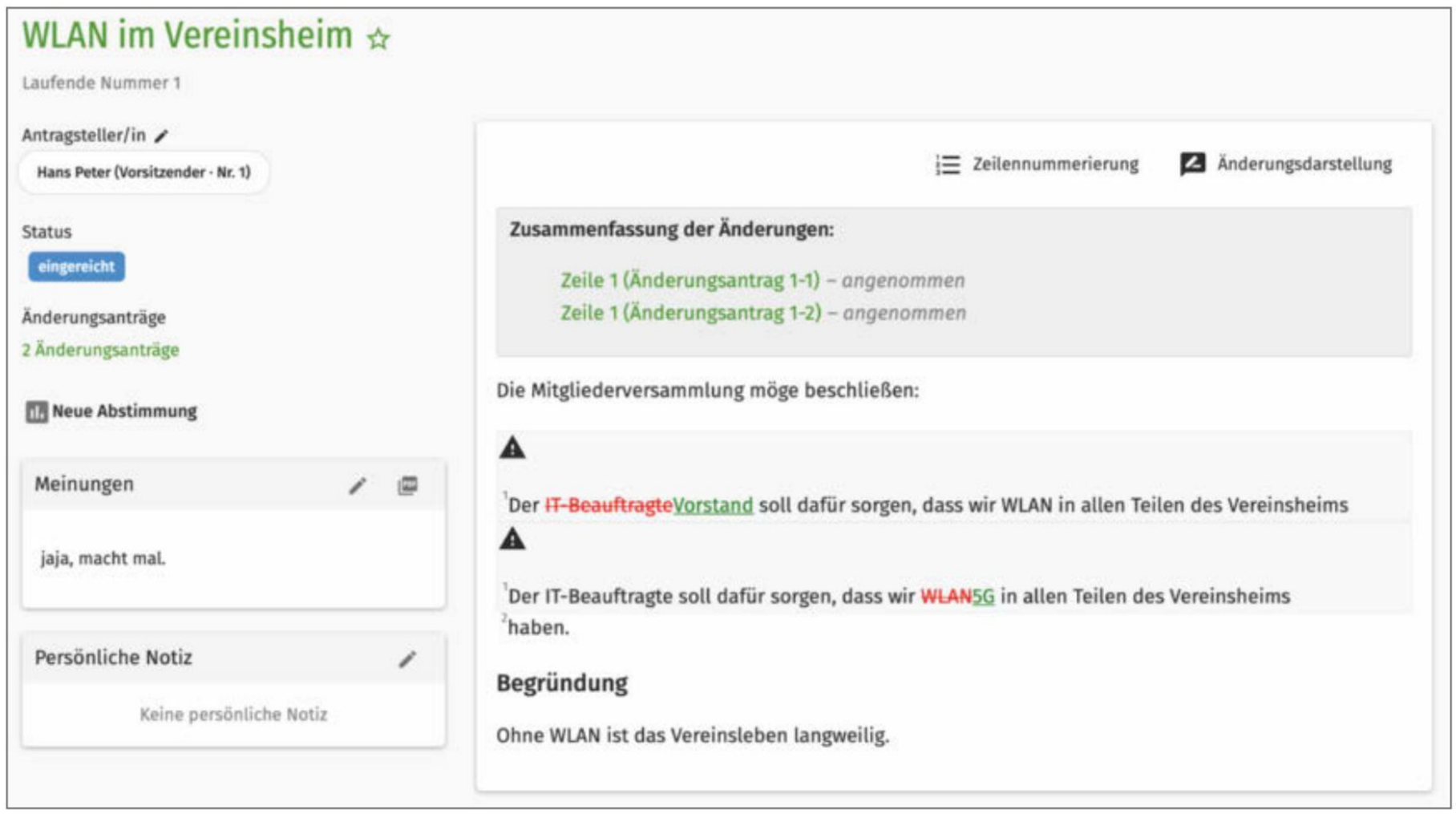
Anträge

Mitglieder der Gruppe „Delegierte“ dürfen in den Standardeinstellungen alle Aktionen ausführen, die Delegierte auf Versammlungen gewöhnlich dürfen (Details hängen natürlich von der Satzung Ihrer Organisation ab): Anträge stellen, für Wahlen kandidieren und sich auf Rednerlisten setzen. Aber der Reihe nach.

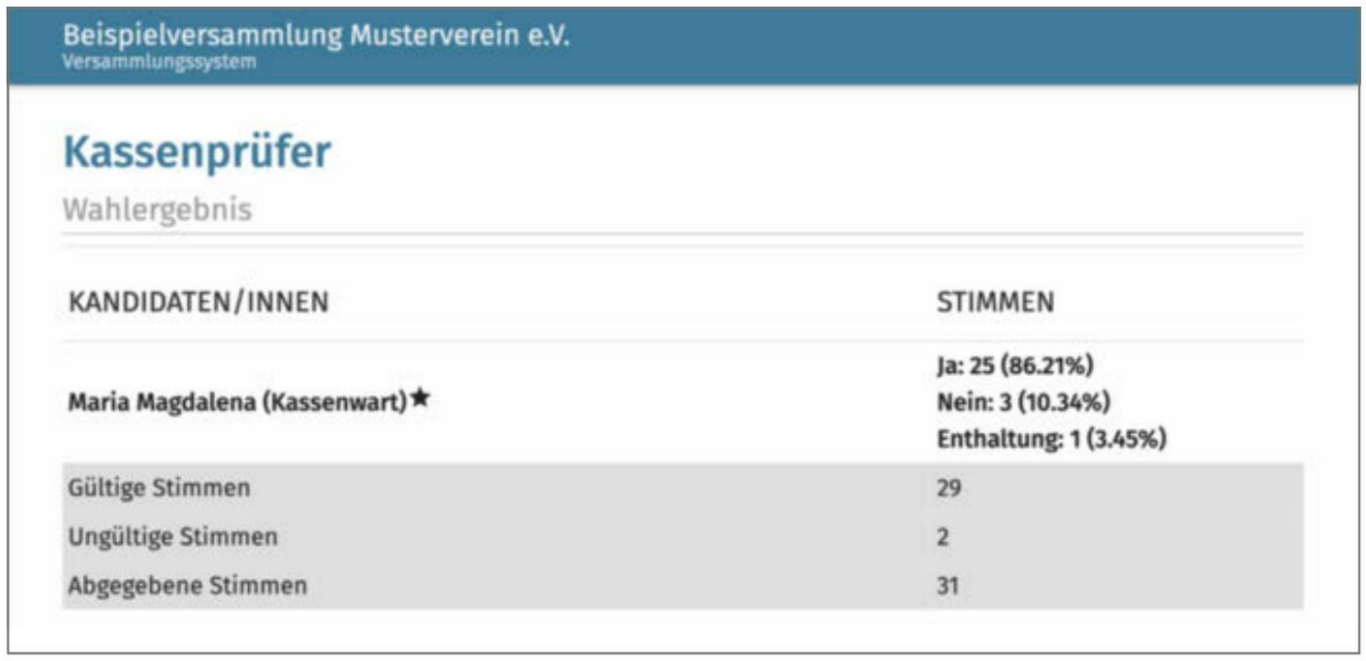
Anträge, Antragsdiskussionen und Änderungsanträge sind schon bei Vor-Ort-Versammlungen ein Quell ewiger Frustration. Wer schon mal dem Schriftführer dabei zugesehen hat, wie er reingerufene Änderungswünsche in ein Word-Fenster auf dem zu kleinen Beamerbild eingetippt hat, kennt die Ausmaße der Probleme. Per Videokonferenz wird das nicht unbedingt besser. OpenSlides macht diese lästige Arbeit etwas entspannter, vor allem wenn alle Mitglieder einen Account haben.

Damit ein Teilnehmer Anträge einreichen kann, muss sein Account Mitglied der Gruppe „Delegierte“ sein. Im Menüpunkt „Anträge“ finden Sie oben rechts das Plus-Symbol und können losformulieren. Ein neuer Antrag ist zunächst auf „intern“ geschaltet und gewöhnliche Delegierte können ihn nicht selbstständig in die Tagesordnung aufnehmen. Diese Funktionen sind Admins, also der Sitzungsleitung, vorbehalten. Es ist aber sinnvoll, sie schon vor der Versammlung freizugeben, damit sich andere vorab eine Meinung bilden und bei Bedarf auf die Rednerliste setzen können.

Unter „Einstellungen / Anträge“ kann man zum Beispiel die Einleitung „Die Versammlung möge beschließen …“ ersetzen. Große Organisationen mit Ausschüssen haben die Möglichkeit, dort den „komplexen Arbeitsablauf“ für Anträge zu aktivieren. Eine interessante Einstellung finden Sie in diesem Dialog weiter unten: Wenn Sie „Änderungsanträge aktivieren“, finden Delegierte in der Ansicht eines Antrags oben rechts die Möglichkeit, Änderungsanträge zu schreiben. Eingereichte Änderungen kann man sich anschließend in einer Diff-Ansicht mit roten und grü-



Anträge und Änderungen zeigt OpenSlides in einer übersichtlichen Diff-Ansicht an.



OpenSlides unterstützt bei der Durchführung von Wahlen bei Präsenz- und Online-Versammlungen.

nen Markierungen darstellen lassen. In der Versammlung würde man diese mit dem Projektor-Knopf auf den (virtuellen) Projektor werfen und darüber debattieren. Wenn die Änderung für gut befunden wurde, klickt ein Administrator im Änderungsantrag auf den Status und setzt ihn auf „angenommen“.

Als Medienbeauftragter oder Moderator der Versammlung sollte man diese Prozesse am besten vorab in einer Testumgebung durchspielen. Weil alles über das Netzwerk gesteuert wird, können sich auch mehrere Sitzungsleiter die Aufgaben teilen – einer verwaltet Anträge und Änderungen, einer bedient nur den Projektor.

Eine Abstimmfunktion ist in der Open-Source-Version nicht enthalten. Sehr wohl kann man aber Abstimmergebnisse erfassen. Ein Administrator kann die Anzahl der Ja- und Nein-Stimmen eintragen und die Zahlen veröffentlichen. Anschließend setzt er den Status des Antrags auf „angenommen“ oder „abgelehnt“. Für nicht-geheime Wahlen (wie bei Anträgen üblich) in einer Videokonferenz kann man zum Beispiel die Hand-heben-Funktion der Konferenzsoftware nutzen und so die Stimmen abfragen.

Wahlen

Auch bei Personenwahlen unterstützt OpenSlides die Versammlungsleitung. Admins müssen die Wahlen vorab anlegen, dann kann sich jeder Delegierte aufstellen. Wenn es während der Versammlung noch eine Phase gibt, in der alle Anwesenden andere Anwesende vorschlagen dürfen, kann ein Administrator die reingerufenen Vorschläge schnell per Autovervollständigung ergänzen. Leider gibt es hier keine Möglichkeit für Freitexteingaben. Wird ein Externer vorgeschlagen, muss ihn jemand schnell als Teilnehmer eintippen.

In der neuen Version von OpenSlides gibt es auch eine elektronische Stimmabgabe. Um die digitalen Wahlen zu aktivieren, müssen Sie in der Konfigurationsdatei settings.py eine Zeile anpassen:

```
ENABLE_ELECTRONIC_VOTING = True
```

Alternativ können Sie auch auf die klassische Briefwahl setzen. OpenSlides kann, wenn die Kandidatensuche abgeschlossen ist, Wahlzettel generieren. Der Menüpunkt ist etwas versteckt: Unter den Kandidaten sehen Sie als Admin den Abschnitt „Wahlergebnis“ und rechts drei Punkte. Dort verbirgt sich das Menü mit dem Punkt „Stimmzettel drucken“.

Als Sitzungsleitung müssen Sie diese ausdrucken und für alle Stimmberechtigten einen Brief vorbereiten: Er sollte Stimmzettel für alle Wahlen enthalten (am besten auf buntem Papier ausdrucken, eine Farbe pro Wahl). Außerdem sollte eine Anleitung beiliegen, sowie ein Formblatt, auf dem der Stimmberechtigte unterschreiben muss, dass er selbst abgestimmt hat. Zudem bekommen alle Teilnehmer zwei Umschläge. Einen kleinen, in den alle Stimmzettel kommen – ohne Unterschriften und weitere Beschriftungen. Zusätzlich einen größeren, in den das unterschriebene Begleitschreiben und der kleine Umschlag kommen. Begleitschreiben und Umschlag werden vor dem Auszählen getrennt, damit die Stimme wirklich geheim ist.

Nach dem Auszählen, einige Wochen nach der Versammlung, kann der Wahlleiter die Ergebnisse in der OpenSlides-Instanz eintragen und veröffentlichen. Nutzt man Online-Wahlen, kann er die Ergebnisse direkt veröffentlichen.

Forschungsarbeit

Alle Einstellmöglichkeiten der Software kann man unmöglich beim ersten Einsatz entdecken oder in einem Artikel beschreiben. Über die Jahre sind immer neue Funktionen, basierend auf Nutzerwünschen, hinzugekommen. Wenn Sie in Ihrem Verein die Aufgabe übernommen haben, die virtuelle Sitzung vorzubereiten, sollten Sie am besten rechtzeitig eine Testumgebung hochfahren und zum Beispiel mal zum Ausprobieren die nächste Vorstandssitzung damit begleiten. (jam) c't

Literatur

[1] Jan Mahn, **Privater Sammelplatz**, Dateiablage, Videokonferenz und Chat auf dem eigenen Server betreiben, c't 9/2020, S. 26

Weitere Infos

ct.de/wfph

Nextcloud als Bausatz

Nextcloud mit Dateisynchronisation, Kalender und Adressbuch sowie vielen weiteren als Apps eingebundenen Diensten bringt Sie der eigenen digitalen Souveränität näher. Unsere Installationsanleitung hilft, die folgenden Artikel rund um die Software nachzuvollziehen, und kann gleichzeitig als Blaupause für eigene Gehversuche mit Containern und Docker-Compose dienen.

Von **Jan Mahn und Peter Siering**



Bausatz für Nextcloud mit Docker-Compose	104
Chat-Server hinter Nextcloud	108
Nextcloud: Tipps für die Gruppenarbeit	112
Videokonferenzen mit Nextcloud Talk	118

Um heute Software bequem zu installieren und zu warten, bieten sich Container an. Das gilt besonders auch für Nextcloud, das fast schon als Gattungsbegriff für selbst betriebene Clouddienste gelten kann. Die unter vollem Namen ihrer Entwickler inzwischen Nextcloud Hub genannte Umgebung, die in PHP geschrieben ist, von einem Web-Server wie Apache ausgeliefert wird und üblicherweise eine Datenbank zur Seite gestellt bekommt, erhält regelmäßig Updates. In einer Container-Umgebung fällt das leicht.

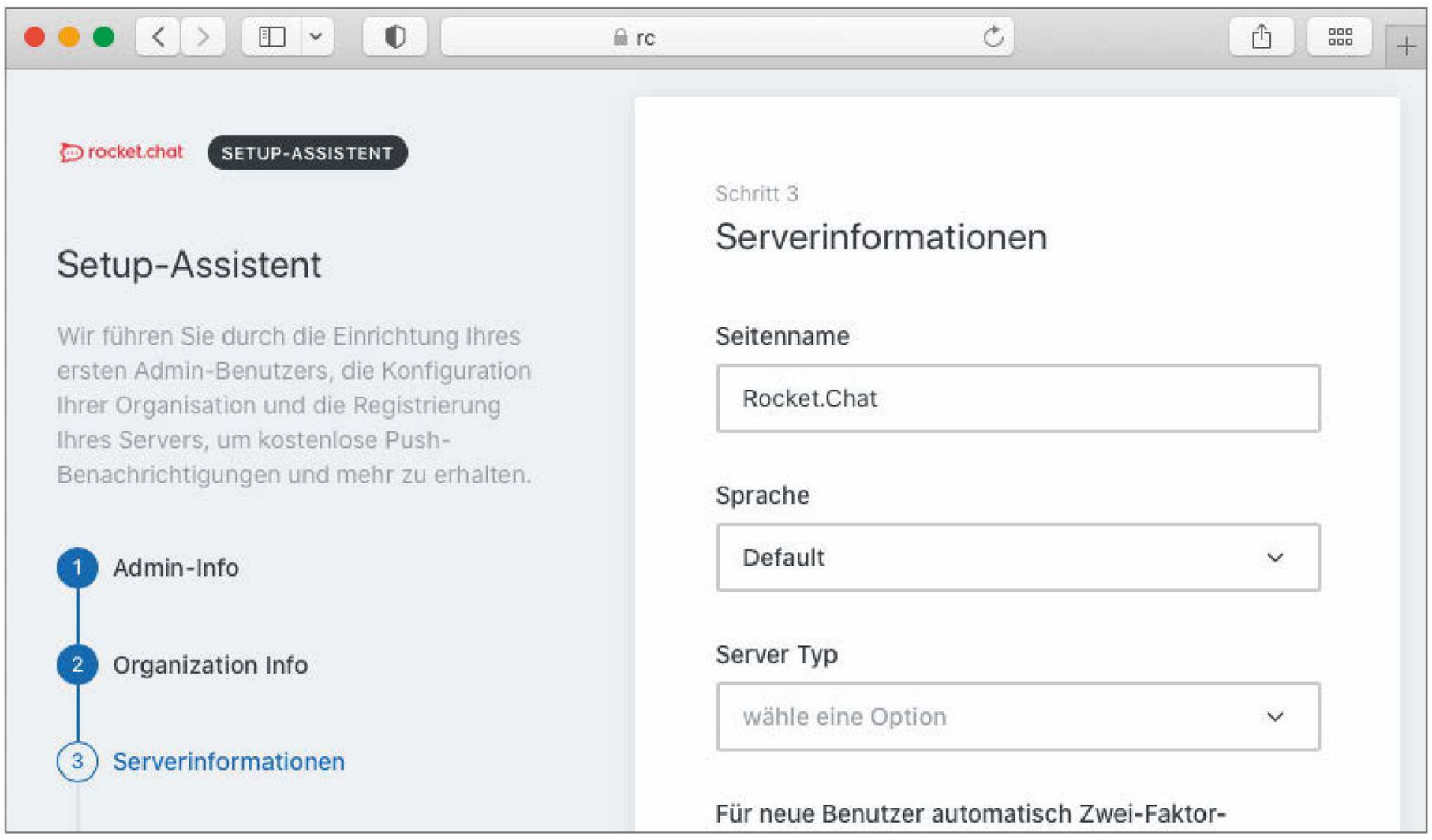
Die häufigen Updates sind Wohl und Wehe in einem. Da eine Installation nicht nur aus dem nackten Nextcloud besteht, führen darin aktivierte Apps und externe Komponenten für Web-Server und Datenbank in neuen Versionen gelegentlich zu Reibungsverlusten: Mal startet das System nicht mehr, mal fallen Komponenten aus. So störten zuletzt Updates die Freigabe von Ordnern, wenn in der Installation die Kanban-Funktionen (Deck-App) aktiv waren. Weitere Beispiele und Abhilfen liefern die folgenden Artikel.

Zutaten

Um unseren Bausatz zu verwenden, brauchen Sie ein Linux-System, auf dem Docker, Docker-Compose und Git in aktueller Version installiert sind – Hilfen dazu finden Sie unter ct.de/wkxt. Auf dem Server dürfen keine anderen Dienste laufen, die auf den Ports 80 und 443 lauschen. Das System sollte aus dem Internet direkt erreichbar sein. Ist das nicht der Fall, richten Sie in Ihrem Router Portweiterleitungen für die Ports 80 und 443 ein. Rund 8 GByte Speicherplatz sollten bereitstehen. Die Container-Images und die von ihnen angelegten Daten belegen ein wenig Platz.

Außerdem brauchen Sie eine eigene Internet-Domain, um unter der IP-Adresse des Systems Hosts wie cloud.heise.de und chat.heise.de zugänglich zu machen. Deren A- oder CNAME-Records sollten beide auf die IP-Adresse des Systems zeigen. Stellen Sie per nslookup sicher, dass Ihr Rechner zu Hause die neuen DNS-Einträge übernommen hat, bevor Sie mit der Einrichtung fortfahren.

Sind diese Voraussetzungen erfüllt, bestehen die nötigen Schritte nur noch darin, die Dateien aus unserem Repository zu klonen und ihre Konfigurationsdaten zu ergänzen, also Passwörter, Domain-Namen und E-Mail-Adresse einzutragen und den ganzen Stack mit einem Befehl zu starten. Nach einigen Minuten Wartezeit sollten alle im Bausatz vorhandenen Container laufen und sich grundsätzlich eingerichtet haben.



Unser Bausatz richtet Nextcloud und Rocket.Chat testbetriebsfertig ein und versieht die Admin-Zugänge für die beiden Dienste mit einem Passwort. Beim Anmelden läuft in Rocket.Chat der Setup-Assistent an.

Das sind im Einzelnen:

- **Traefik als Edge-Router** – es beschafft TLS-Zertifikate bei Let's Encrypt und nimmt eingehende Anfragen an die Web-Ports 80 und 443 entgegen, um sie an die anderen Container weiterzureichen, etwa Nextcloud oder Rocket.Chat
- **Watchtower als Update-Dienst** – er klappert regelmäßig den Docker-Hub ab, holt neue Images, wenn die aktualisiert worden sind, und startet die Container in gleicher Konfiguration mit dem neuen Versionsstand
- **Nextcloud als zentrale Cloud-Anwendung** – sie stellt die wesentlichen Dienste bereit und bildet auch die Basis, an die andere Cloud-Dienste anflanschen können, etwa der Chat-Server
- **MariaDB als Datenbank für Nextcloud** – hier lagert die Nextcloud-Instanz vor allem ihre Verwaltungsdaten. Dateien der Nextcloud-Dateiablage landen auf dem Host in einem benannten Volume
- **Rocket.Chat als eigenständiger Chat-Server** – der wendet sich, um Benutzer zu authentifizieren, an Nextcloud. Dafür nutzen beide Seiten das OAuth-Verfahren
- **MongoDB als Datenbank für Rocket.Chat** – in dieser Datenbank lagert Rocket.Chat die Nachrichten; die Docker-Compose-Datei enthält einen zweiten MongoDB-Container, der nur für die Initialisierung eingespannt wird

Mit `git clone https://github.com/psct/ctnextcloud2` erstellen Sie im aktuellen Verzeichnis eine Kopie aller Dateien aus unserem Repository. Wechseln Sie in das Verzeichnis ctcloud. Kopieren Sie dort Datei .env-example auf .env (`cp .env-example .env`). Bearbeiten Sie dann in einem Editor Ihrer Wahl die Datei .env. Alle dort definierten Variablen bis zur Leerzeile müssen Sie anpassen.

Das sind nicht nur die Namen Ihrer Domain, sondern auch die Kennwörter für die administrativen Zugänge zu Nextcloud und Rocket.Chat sowie Ihre E-Mail-Adresse. Für produktiven Betrieb sollten Sie auch die Namen der Administratorkonten ändern und die Passwörter für den Zugriff auf die Datenbankserver ändern – sicher ist sicher. (Obendrein sollten Sie dann die Traefik-Konfiguration für an aktuelle Erkenntnisse angepassten TLS-Betrieb korrigieren und in ctnextcloud2.yml die Adressen in TRUSTED_PROXIES weniger großzügig setzen.)

Ihre E-Mail-Adresse ist für das Ausstellen von Zertifikaten mit Let's Encrypt obligatorisch, landet aber nicht in den Zertifikaten, sondern dient vor allem dazu, Ihnen Informationen über das Ablaufen von Zertifikaten zukommen zu lassen. Das ist durchaus praktisch, weil Sie so erfahren, dass etwas bei der automatischen Erneuerung klemmt.

Stellen Sie mit `chmod 600 traefik/acme.json` sicher, dass die Dateirechte für die Datei zur Aufbewahrung der Zertifikate die richtigen Zugriffsrechte hat. Das Projekt trennt die Dateien, die der Edge-Proxy benötigt von den übrigen Daten und Konfigurationsdaten der anderen Container; die landen im Verzeichnis „data“ in der Projekthierarchie.

Aufsetzen

Wenn Sie jetzt mit `docker-compose -f ./ctnextcloud2.yml up -d` den ganzen Stack starten, wird es einige Minuten dauern, bis alle Dienste einsatzbereit sind. Traefik holt bei Let's Encrypt Zertifikate. Die Datenbank-Container initialisieren zunächst die grundlegenden Datenbanken, dann erst können Nextcloud und Rocket.Chat die von ihnen benötigten Strukturen anlegen und sich grundlegend konfigurieren.

Den Verlauf können Sie beobachten, indem Sie in die Logs einzelner Container hineinsehen, etwa mit `docker logs -f nextcloud` in die Ihrer Nextcloud. Das ist insgesamt übersichtlicher als den eingangs erwähnten docker-compose-Aufruf nicht im Hintergrund auszuführen (ohne die Option `-d`), sondern mit vollen Log-Ausgaben, denn dann werden Sie mit den Log-Meldungen aller Container überhäuft.

Sollte auch nach einigen Minuten Nextcloud oder Rocket.Chat nicht erreichbar sein, fahren Sie den ganzen Salat mit `docker-compose -f /ctnextcloud2.yml down` herunter und starten Sie ihn erneut. Manchmal reißt einem Container beim Initialisieren der Geduldsfaden. Ein solcher Neustart behebt das meist. Ansonsten müssen Sie sich auf die Fehlersuche machen: Mit den Logs von Traefik sollten Sie anfangen, dann die Datenbank-Container prüfen und dann schließlich die Logs der Dienste.

An einigen Stellen gibt unsere Bauanleitung bewusst Versionen vor, so bei Nextcloud Version 21, obwohl die schon etwas überholt ist: Version 22 hat Probleme mit OAuth, sodass der im folgenden Artikel geschilderte Trick die Nutzerdatenbank von Nextcloud für die Anmeldung an Rocket.Chat zu nutzen, nicht gelingt. Zum Redaktionsschluss genügten MongoDB 3.6, MariaDB 10.6, Traefik 2.5 und Rocket.Chat 3.18. Falls es, wenn Sie diese Bauanleitung konsultieren, neuere Versionen gibt und diese nicht harmonieren, stellen Sie dem Image-Namen einen Doppelpunkt und die genannte Version hintenan. (ps) c't

Einrichten von Docker

ct.de/wkxt

Verteidigen Sie Ihre Privatsphäre!

shop.heise.de/ct-datenschuetzen

Heft + PDF mit 29 % Rabatt



shop.heise.de/ct-datenschuetzen

In dieser neuen Auflage des c't-Sonderhefts „Daten schützen" erfahren Sie detailliert, wie Sie Ihre Privatsphäre vor Übergriffen bewahren können. **Darum geht's im Heft:**

- E-Mails und Telefone absichern
- Welche Messenger verschlüsseln wirksam
- Welche Angriffe bedrohen Ihre Privatsphäre on- und offline
- Passwörter für alle Systeme sicher verwalten
- Für Abonnenten portofrei
- Auch im Set mit Reiner SCT Authenticator zum Sonderpreis

Heft für 14,90 € • PDF für 12,99 € • Bundle Heft + PDF 19,90 €

Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 €.
Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise Shop

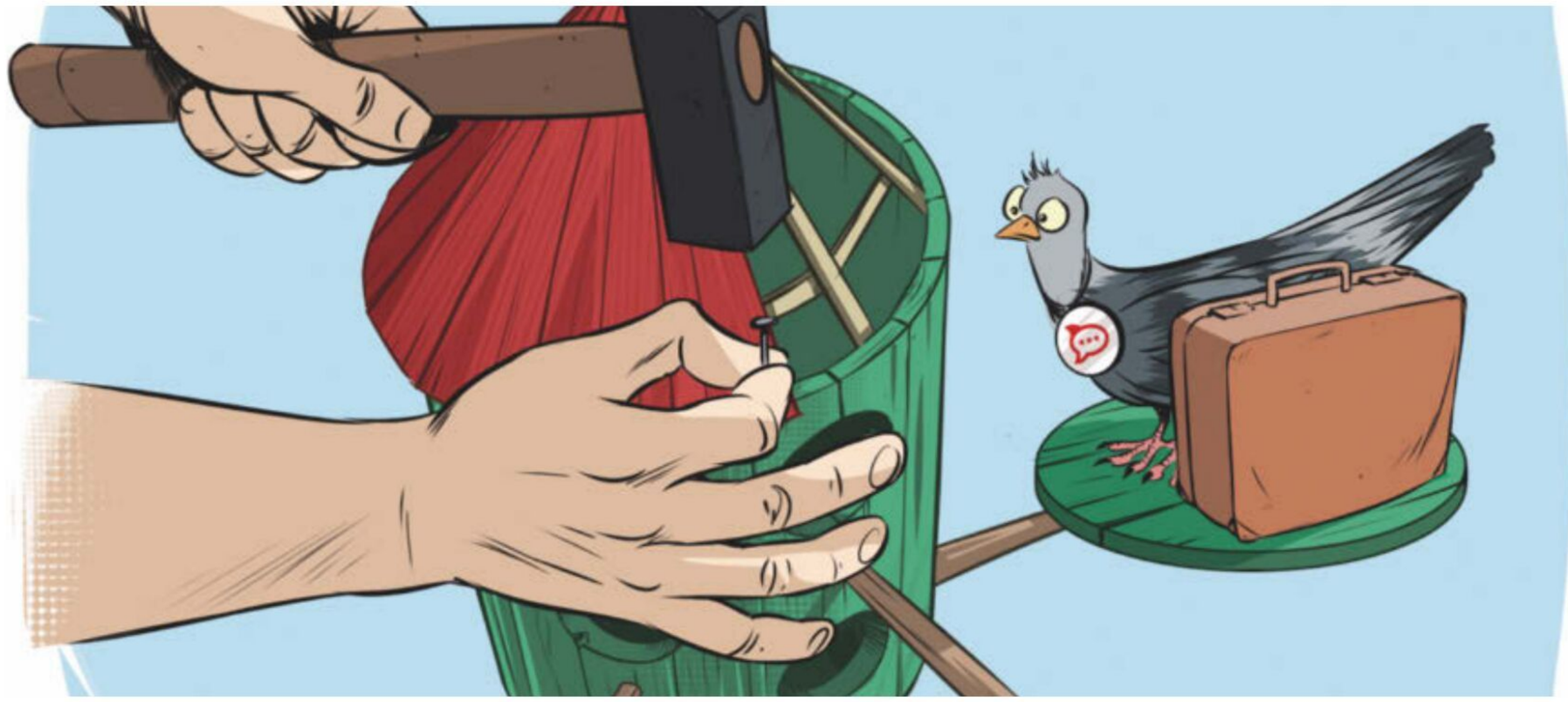


Bild: Albert Hulm

Chat-Server hinter Nextcloud

Ein selbst gehosteter Chat-Server ist für Vereine oder Unternehmen eine reizvolle Alternative zu WhatsApp und ähnlichen zentralen Diensten. Mit der Open-Source-Software Rocket.Chat hat man einen solchen schnell eingerichtet. Die Artikel zeigt, wie Sie mit OAuth eine Nextcloud-Installation als Anmeldeserver für Single-Sign-on verwenden.

Von **Jan Mahn**

Alternativen zu WhatsApp mit einem vergleichbaren Funktionsumfang gibt es viele. Die meisten Messenger-Apps greifen dabei auf eine zentrale Server-Infrastruktur zurück, die der jeweilige Anbieter bereitstellt. Über diese wickeln die Clients den gesamten Verkehr ab. Selbst wenn die Nachrichteninhalte dank Ende-zu-Ende-Verschlüsselung für den Betreiber nicht auszuwerten sind, vertraut man ihm die Meta-Informationen an, zum Beispiel, wer wann eine Nachricht an wen verschickt hat.

Wer für eine geschlossene Nutzergruppe, etwa einen Verein, eine Online-Spielergruppe oder ein Unternehmen eine Kommunikationslösung sucht, die sich beim Funktionsumfang nicht hinter WhatsApp & Co. verstecken muss, beim Datenschutz aber überlegen ist, kann einen Chat-Server auch selbst hosten – sofern man über Admin-Grundlagenwissen verfügt. Die Open-Source-Software Rocket.Chat ist für diese Aufgabe gut geeignet und als reiner Chat um Größenordnungen charmanter als Nextclouds eigene Talk-App.

Für die Nutzer gibt es kostenlose Clients für Windows, Linux, macOS, iOS und Android, außerdem eine Weboberfläche mit allen Funktionen. Den Server kann man auf einem Linux-System installieren oder als Docker-Container starten. Es reicht dafür ein günstiger vServer bei einem Hosting-Anbieter der Wahl. Selbst mit 500 MByte RAM können 50 Nutzer

problemlos chatten. Der Betrieb hinter der heimischen DSL-Leitung ist mit DynDNS ebenfalls kein Problem, wenn man Abstriche bei der Ausfallsicherheit machen kann.

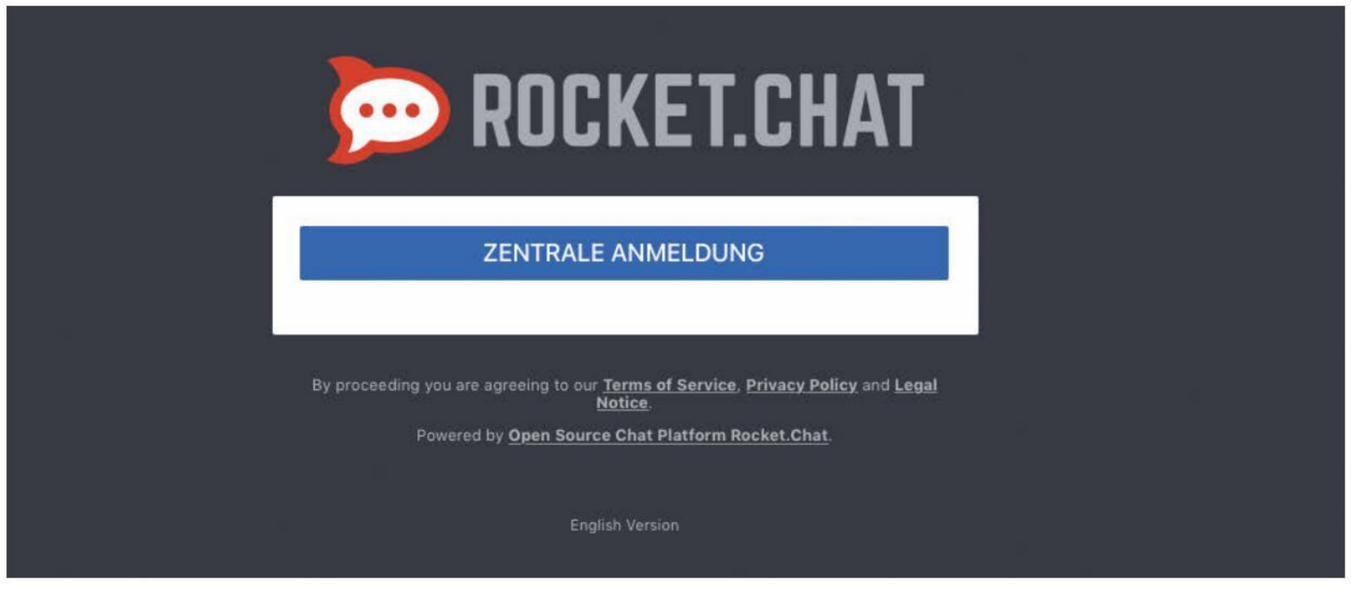
Drumherum

Unsere Beispielkonfiguration für Nextcloud, die der vorangehende Artikel im Detail vorgestellt hat, fährt von sich aus eine Rocket.Chat-Instanz hoch, die wir im Folgenden für die Zusammenarbeit mit Nextcloud konfigurieren. Die für die Anmeldung an Nextcloud und Rocket.Chat nötigen Administrationskonten müssen Sie vor dem Starten der Beispielkonfiguration ebenso in der Konfigurationsdatei ergänzen wie passende Domain-Namen, siehe Seite 106.

Um die OAuth-Verbindung zu Rocket.Chat in der Administrationsoberfläche von Nextcloud vorzubereiten, öffnen Sie dort die Einstellungen, zu finden über das Menü oben rechts (per Klick auf die Initialen). Als Administrator sehen Sie links im Menü die Abschnitte „Persönlich“ und „Verwaltung“. Öffnen Sie in Letzterem den Menüpunkt „Sicherheit“. Ganz unten unter „OAuth 2.0-Clients“ können Sie eine Anwendung hinzufügen, die sich per OAuth anmelden darf. Den Namen können Sie frei vergeben, zum Beispiel „chat“. Die „Weiterleitungs-URI“ muss, angepasst an Ihre Domain, so aussehen:

```
https://rocket.<DOMAIN>/_oauth/mychat
```

Das Ende der URL „mychat“ brauchen Sie gleich als Bezeichnung für Konfiguration auf Seiten von Rocket.Chat. Nach dem Speichern sehen Sie ein „Client-Identifikationsmerkmal“ und ein „Geheimnis“. Diese Werte benötigen Sie ebenfalls gleich für die Einrichtung von Rocket.Chat. Lassen Sie den Tab im Browser nebenbei geöffnet.



Rocket.Chat zeigt keine eigenen Felder zur Anmeldung an. Die Benutzer melden sich immer bei Nextcloud an.

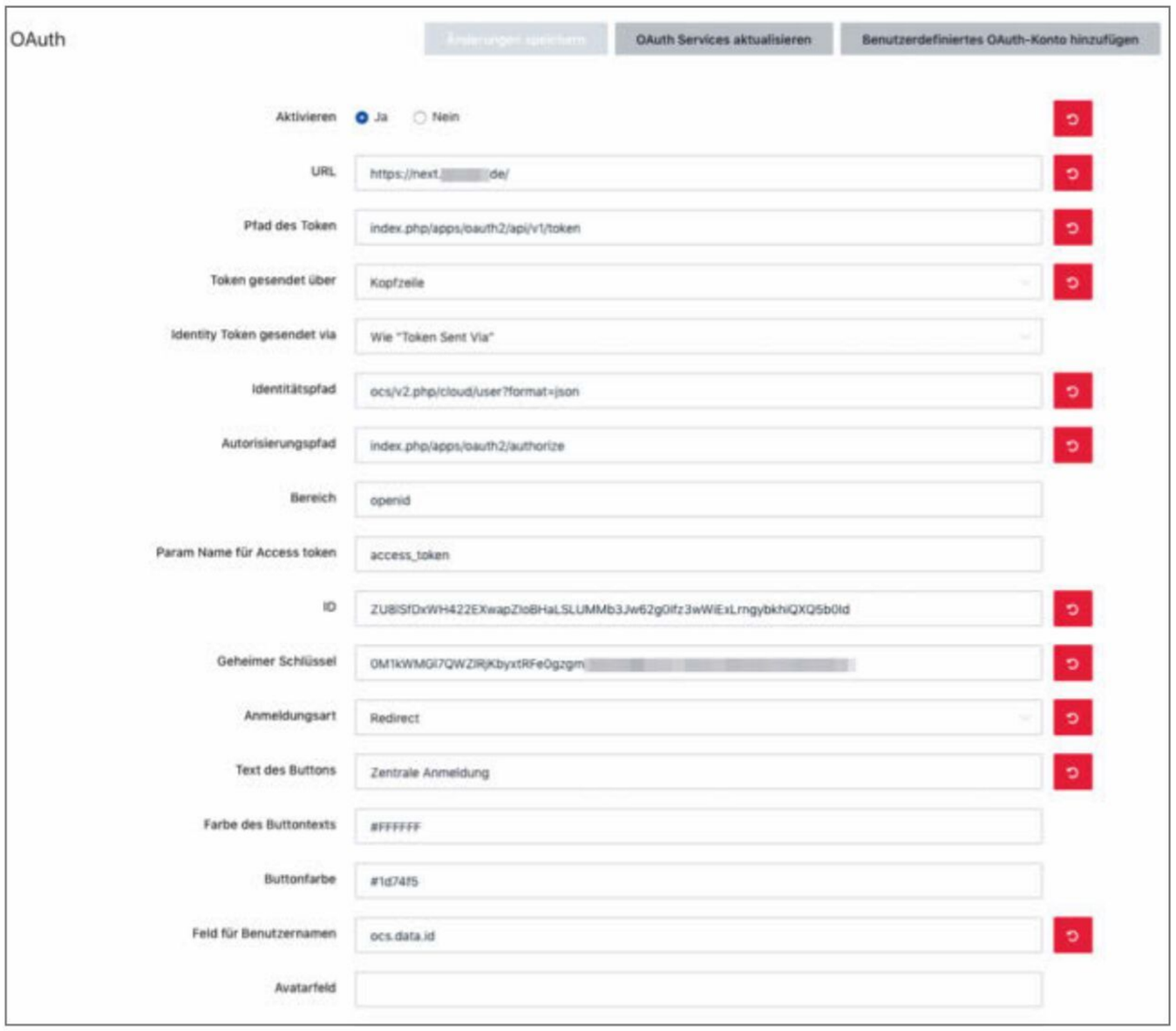
Chat-Server

Öffnen Sie jetzt die Adresse des Chat-Servers. Auch er möchte einen kleinen Assistenten mit Ihnen durchspielen. Neben Benutzername, Kennwort und Mailadresse können Sie den Entwicklern von Rocket.Chat statistische Angaben zu Ihrer Organisation hinterlassen.

Im nächsten Schritt sollten Sie einen „Seitennamen“ vergeben, die Sprache festlegen und den „Server Typ“ auf „Privates Team“ umschalten. Im dritten Schritt können Sie entscheiden, ob Ihr Server mit dem Push-Server von Rocket.Chat sprechen soll. Das hat den Vorteil, dass Sie die fertigen Apps aus den Stores verwenden können. Die mobilen Betriebssysteme lassen Push-Benachrichtigungen nicht von allen Servern zu. Für große Unternehmen ist es möglich, den Quellcode der Apps herunterladen, den Servernamen hartzukodieren, alles zu signieren und als eigene Chat-Apps in die Stores zu bringen. Für normale Anwender ist das definitiv zu viel Aufwand. Um einen Schlüssel für die Push-Dienste zu erlangen, müssen Sie sich später in der Rocket.Chat-Cloud registrieren.

Nach dem Assistenten begrüßt Sie ein sehr leerer Chat-Raum. Um Nutzer aus Nextcloud einzubinden, öffnen Sie die Administration. Die finden Sie als Menüpunkt, wenn Sie oben links das Symbol für den angemeldeten Nutzer anklicken (oder über die Adresse /admin/info). Öffnen Sie in der Leiste links den Punkt „OAuth“ ziemlich weit unten (nicht zu verwechseln mit dem Punkt „OAuth-Apps“ weiter oben). Hier sind bereits viele OAuth-Anmeldedienste voreingestellt. Die dort vorgefertigte Konfiguration für Nextcloud funktionierte in unseren Experimenten nicht rund: Sie konnte die Nextcloud-Nutzernamen nicht auslesen.

Unsere Empfehlung deshalb: Legen Sie Ihre eigene Anwendung mit „Benutzerdefiniertes OAuth-Konto hinzufügen“ an. Vergeben Sie als Namen den zuvor in Nextcloud am Ende der URL angegebenen Pfad, im obigen Beispiel „mychat“. Öffnen Sie dann das Formular für die benutzerdefinierte Konfiguration, die die Oberfläche in die vorgegebenen Konfigurationen einreiht. Stören Sie sich nicht daran, dass der gewählte Name groß geschrieben wird. Die dann zugänglichen Formular-



Auf den ersten Blick ist die OAuth-Einrichtung verwirrend – was auch an der fehlenden Dokumentation liegt. In der Tabelle auf Seite 111 finden Sie alle notwendigen Werte; für die nicht zu beschreibenden Felder übernehmen Sie die Vorgaben.

felder sind nicht selbsterklärend und weder die Dokumentation von Rocket.Chat, noch die von Nextcloud beantworten alle Fragen. Die Details finden sich teilweise in Fehlerbeschreibungen bei GitHub und Stackoverflow.

Diese Forschungsarbeit haben wir Ihnen aber bereits abgenommen. In der Tabelle auf Seite 111 sehen Sie alle Werte in der Übersicht. Ersetzen müssen Sie alle Angaben in „< >“. Hier kommen auch die Werte aus der Nextcloud-Oberfläche wieder zum Einsatz. Bei der URL darf der „/“ am Ende nicht fehlen; sonst misslingen die Weiterleitungen.

Auf keinen Fall dürfen Sie den Haken „Übernehme Rollen von SSO“ auf „Ja“ umschalten. Da das Übertragen der Rollen aus der Nextcloud nicht funktioniert, verliert Ihr Admin-Account sonst sofort seine Adminrechte, wenn er in beiden Systemen gleich heißt und Sie haben sich ausgesperrt. Nach der Abtippübung speichern Sie oben mit „Änderungen speichern“ und klicken danach auf „OAuth Services aktualisieren“.

Zum Test ist ein anonymes Browserfenster oder ein anderer Browser hilfreich, damit Sie sich nicht immer ausloggen müssen. Im Anmeldeschirm sehen Sie über den Formularfeldern den neuen Button „zentrale Anmeldung“ (oder einen beliebigen Text, den Sie für den Button vergeben haben). Er leitet Sie zur Anmeldeseite der Nextcloud. Geben Sie Benutzernamen und Kennwort ein und beobachten Sie, wie Sie zu Rocket.Chat zurückgeschickt werden. Sie sind authentifiziert und können mit Ihren Testbenutzern kommunizieren.

Ihren Nutzern möchten Sie sicher nicht erklären, dass sie immer den Button „zentrale Anmeldung“ verwenden sollen und ihren Benutzernamen nicht darunter eintippen sollen. Um den Ärger von Anfang an zu vermeiden, suchen Sie in der Rocket.Chat-Administration den Menüpunkt „Konten“. Setzen Sie hier „Standard Anmeldeformular zeigen“ auf „Nein“. Die beiden Eingabefelder verschwinden vom Login-Bildschirm und Ihren Nutzern bleibt nur die Anmeldung über die Nextcloud.

Im gleichen Dialog verbieten Sie das Ändern von Benutzernamen, Namen, Mailadresse und Passwort; all diese Angaben sollen nur zentral in der Nextcloud verwaltet werden und nicht nicht von Benutzern veränderbar sein. Damit sich nicht Fremde auf Ihrem Server Konten zulegen, setzen Sie am besten in „Konten“ unter „Registrierung“ das Feld Registrierungsformular auf „Deaktiviert“.

Literatur

[1] Oliver Lau, **Gottvertrauen**, Benutzer von Web-Anwendungen mit Hilfe von OAuth 2.0 authentifizieren, c’t 7/2014, S. 190

Dokumentation

ct.de/we61

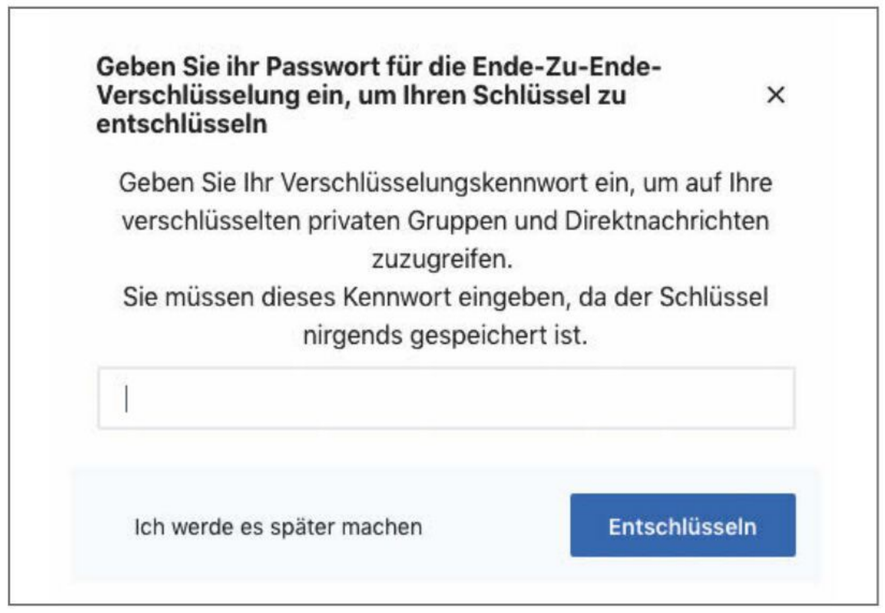
Client-Vielfalt

Der Pflichtteil ist erledigt und Sie können beginnen, die Möglichkeiten des Chat-Servers zu erkunden. Die Apps für iOS und Android, zu finden über den Suchbegriff „Rocket.Chat“ in den App-Stores, haben kein Problem mit der OAuth-Anmeldung. Beim ersten Start fragen sie nach einer Server-Adresse – hier geben Sie die Adresse des Chat-Servers an. Bei der Anmeldung sieht der Benutzer dann den Nextcloud-Dialog. Auch in allen gängigen Browsern funktioniert Rocket.Chat mit OAuth.

Bevor Sie Ihren Server für die Nutzer freigeben, sollten Sie sich die Zeit nehmen, alle Seiten in den Admin-Oberflächen durchzugehen und an Ihre Bedürfnisse anzupassen. Vieles von dem, was Nextcloud und Rocket.Chat können, lernen Sie bei der Gelegenheit kennen. Sinnvoll ist es zum Beispiel, einen Mailserver anzugeben, damit Cloud und Chat Benachrichtigungen verschicken können.

Hochsicherheit

In der Standardeinstellung sind alle Nachrichten transportverschlüsselt, der Server-Besitzer könnte die Nachrichten aber aus der MongoDB-Datenbank herausfummeln und auslesen. Wer seinen Nutzern auch Ende zu Ende verschlüsselte Nachrichten anbieten möchte, kann das noch als experimentell gekennzeichnete Feature in der Administration unter „Ende-zu-Ende-Verschlüsselung“ aktivieren. Alle Benutzer sehen einmalig am oberen Rand einen Hinweis, dass die Funktion aktiviert wurde. Anschließend können sie einen Schlüssel herauskopieren, der nicht auf dem Server gespeichert wird. Dieser muss auf jedem neuen Gerät eingetragen werden. Um ein Gespräch Ende zu Ende zu verschlüsseln, klicken Sie in einem Dialog oben rechts auf die drei Punkte und im aufklappenden Menü auf „E2E“. Ein grüner Schlüssel zeigt an, dass die Nachrichten verschlüsselt sind.



Ende-zu-Ende-Verschlüsselung ist noch im Beta-Zustand. Auf jedem Gerät muss der Nutzer ein Kennwort eingeben, das ihm die Oberfläche nur einmalig anzeigt.

Massen-Import

Hat man nur fünf Benutzer, kann man alle Namen, Benutzernamen und Mailadressen problemlos per Hand in der Nextcloud-Oberfläche eintragen und sie in Gruppen zuweisen. Wer dagegen einen Verein mit 50 oder 100 Mitgliedern zu versorgen hat, wünscht sich schnell einen Import über eine CSV-Datei. Nextcloud hat eine solche Funktion leider nicht eingebaut, lässt sich aber über ein API fernsteuern. Auf dieser Grundlage haben wir ein kleines Skript gebaut und in einen Container verpackt. Sie finden es mit einer kurzen Dokumentation im Ordner „importer“ im GitHub-Repository, das Sie bereits geladen haben. Fertig ist ein Server-Dienst nie. Die Zusammenstellung ist eine Grundlage für eine auf OAuth basierende private Cloud. Im nächsten Schritt sollten Sie sich eine Strategie zur Datensicherung überlegen. (jam) c't

Einstellungen für OAuth in Rocket.Chat

Feld	Inhalt
Aktivieren	Ja
URL	https://next.<IHRE_DOMAIN>/
Pfad des Token	index.php/apps/oauth2/api/v1/token
Token gesendet über	Kopfzeile
Identity Token gesendet via	wie „Token Sent Via“
Identitätspfad	ocs/v2.php/cloud/user?format=json
Token gesendet über	Kopfzeile
Autorisierungspfad	index.php/apps/oauth2/authorize
Bereich	openid
Param Name für Access Token	access_token
ID	ID <Client-Identifikationsmerkmal aus der Nextcloud-Oberfläche>
Geheimer Schlüssel	Geheimer Schlüssel <Geheimnis aus der Nextcloud-Oberfläche>
Anmeldungsart	Redirect
Text des Buttons	<Freitext, zum Beispiel „zentrale Anmeldung“>
Feld für Benutzernamen	ocs.data.id
Rollen / Gruppen Feld	roles
Übernehme Rollen von SSO	nein
Benutzer zusammenführen	ja

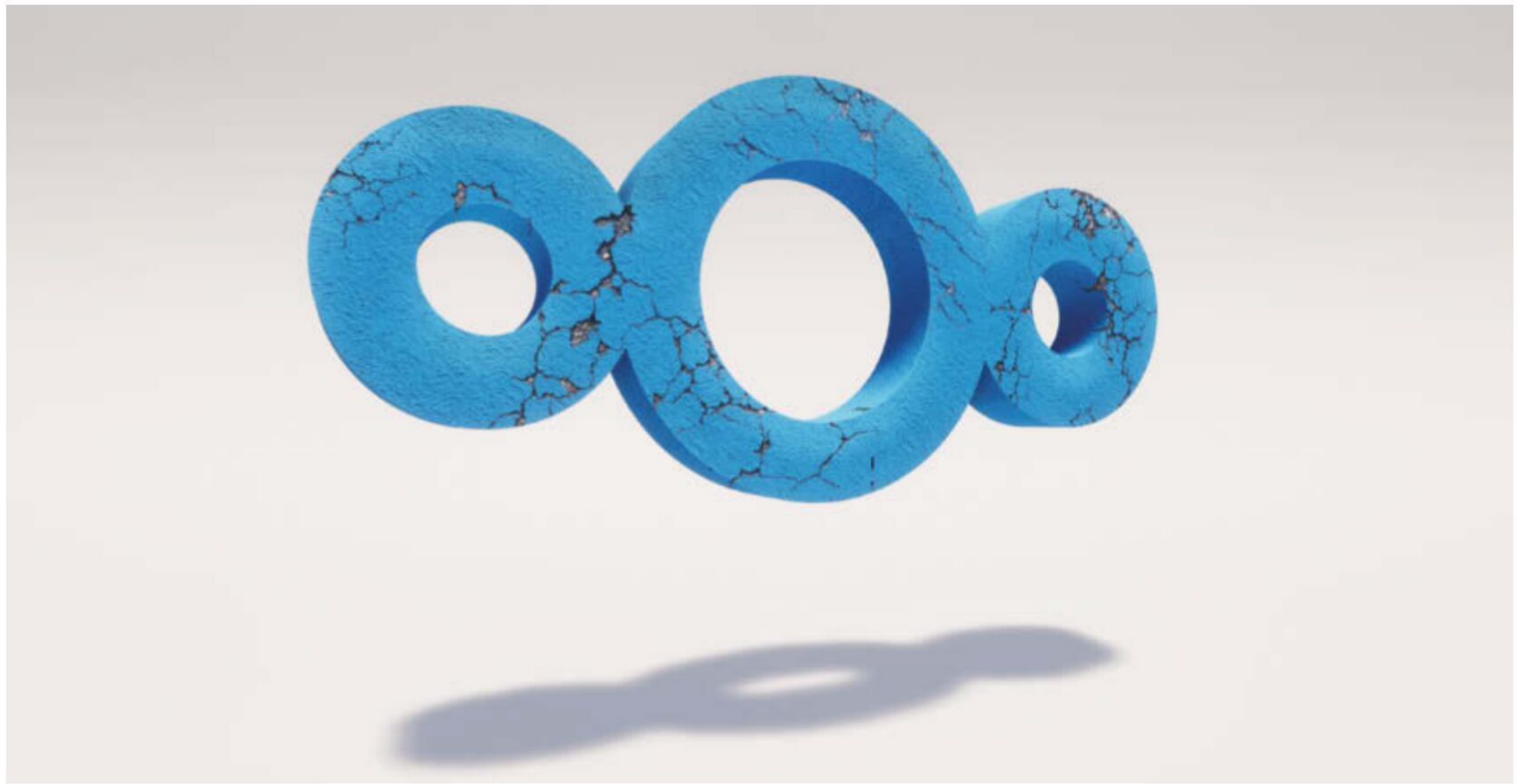


Bild: Hagen Kober

Nextcloud: Tipps für die Gruppenarbeit

Nextcloud bringt verteilt zusammenarbeitende Computer und Menschen einander näher. Doch die alternative Cloud ist manchmal spröde: Unsere Hinweise machen die neue Heimat schöner.

Von **Peter Siering**

Nextcloud kann beim Hoster, auf einem Mietserver oder auf dem hauseigenen NAS laufen. Dieser Beitrag ergänzt unsere Installationsanleitung um Tipps für frisch gebackene Nextcloud-Eigener und -Admins. Er hilft, die überbordenden Erwartungen zu justieren und gibt pragmatische Hilfestellungen zur geeigneten Nutzung.

Dreh- und Angelpunkt von Nextcloud sind Dateien: Nextcloud-Nutzer haben im Wesentlichen drei Möglichkeiten, Dateien darauf abzulegen. Sie können sie per Webbrowser dort hoch- und runterladen. Das Gleiche gelingt über das WebDAV-Protokoll ähnlich wie eine Freigabe im Netzwerk. Außerdem kann spezielle Client-Software Dateien speichern und abrufen.

Der Zugriff per Browser ist umständlich. Die zweite Methode, WebDAV, funktioniert wegen Unzulänglichkeiten des Protokolls [1] nur selten gut. Es möchte Dateien stets vollständig laden. Eine aktive Bearbeitung, wie man sie von Dateifreigaben kennt, erlaubt WebDAV nicht. Außerdem variieren die Fähigkeiten von WebDAV-Clients stark. Kurzum: Nutzen Sie die beiden Methoden nur, wenn sich der Zugang nicht anders realisieren lässt.

Die deutlich bessere Methode stellt die Nextcloud-eigene Client-Software dar. Die gleicht auto-

matisch Dateien zwischen lokalen Verzeichnissen und Nextcloud ab. Wenn sich die Datei auf einer Seite ändert, überträgt der Client sie auf die andere. Er nutzt dafür Funktionen, die im Hintergrund dafür sorgen, dass nicht nur die aktuelle Fassung der Datei erhalten bleibt, sondern auch die vorherigen Versionen – Nextcloud versioniert automatisch.

Wenn der Benutzer eine alte Fassung seiner Datei braucht, etwa weil er sich beim Bearbeiten eines Textes vergaloppiert hat, kann er über die Web-Oberfläche von Nextcloud alte Versionsstände der Datei abrufen. In die Web-Oberfläche bringt ihn ein Klick aus der Bedienoberfläche der Client-Software („Nextcloud im Browser öffnen“). Dort kann er im Kontextmenü einer Datei, das unter anderem über die drei Punkte erreichbar ist, unter „Details“ die „Versionen“ einsehen und alte wiederherstellen.

Dateiaustausch

Das klappt gut, wenn ein Benutzer die Ablage für seine eigenen Dateien verwendet, kriegt aber eine eigene Dynamik, wenn weitere Benutzer ins Spiel kommen und an den gleichen Dateien werkeln. Nextcloud erlaubt es nämlich, Dateien oder Verzeichnisse mit anderen Nutzern zu teilen. Die sehen die dann auch in ihrem Nextcloud-Client beziehungsweise ihrem synchronisierten Verzeichnis. Ändern die dort eine geteilte Datei, verpasst Nextcloud ihr ebenfalls eine neue Version.

Spannend wird es, wenn eine geteilte Datei gleichzeitig auf mehreren Clients geändert wird: Nextcloud legt dann von sich aus eine Kopie der Datei mit den lokalen Änderungen an und verpasst ihr einen Zeitstempel. Der Benutzer muss dann die Dateien vergleichen und die Daten von Hand zusammenpuzzeln – die Synchronisierung kann da also keine Wunder vollbringen. Für das gleichzeitige Arbeiten an ein und derselben Datei bieten sich Funktionen zur Online-Zusammenarbeit an – dazu gleich mehr.

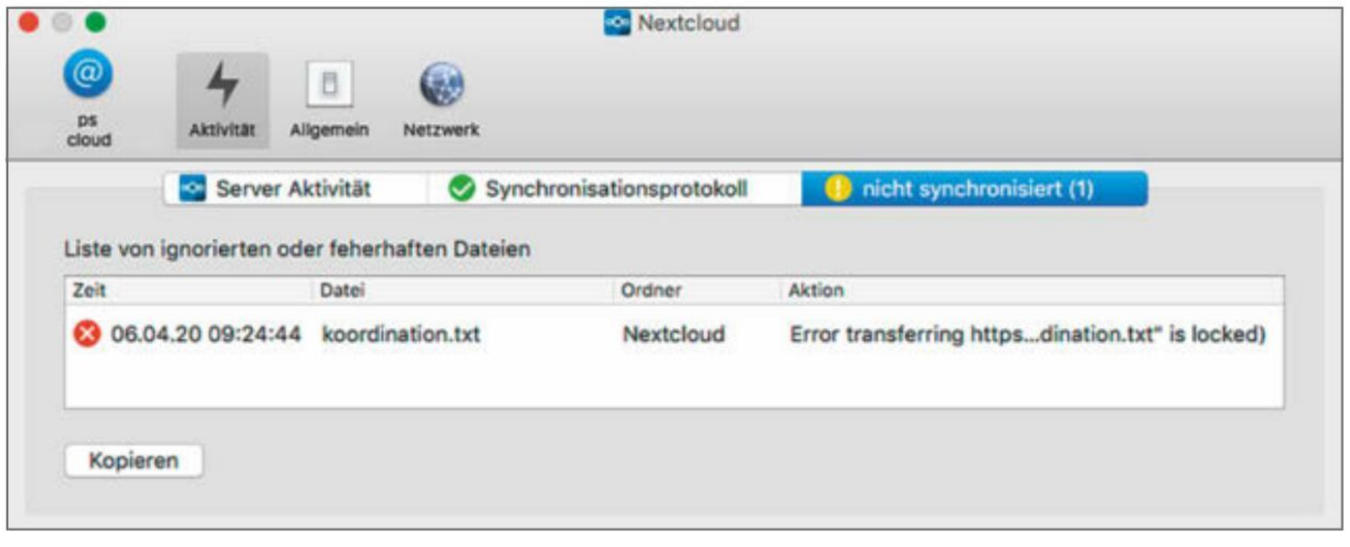
Zum Nutzungsalltag mit Nextcloud gehören allerdings auch hartnäckigere Synchronisierungsfehler, die damit enden, dass der Server eine Datei verriegelt („Locked“). Der Nextcloud-Client hat dann beim Versuch kapituliert, eine Datei zu synchronisieren. In einem solchen Fall kann sich der Nutzer allein nicht mehr helfen, sondern muss den Nextcloud-Admin um Hilfe bitten: Der versetzt das System in den Wartungsmodus und leert mit speziellen Befehlen eine Datenbanktabelle. Die einzelnen Schritte zeigt der Artikel später am Beispiel einer Container-Installation.

Gemeinsame Struktur

Subtilen Unterhaltungswert hat Nextcloud, wenn Mitglieder einer Arbeitsgruppe Dateien in geteilten Ordnern mit besten Absichten aufräumen. Sobald sie diese in einen privaten, nicht geteilten Ordner verschieben, verschwinden die Dateien aus dem Blickfeld der anderen – Nextcloud verhindert das nicht. Solange die Dateien aber in geteilten Ordnern verbleiben, betsteht keine Gefahr.

Soll eine Nextcloud-Instanz Benutzern eine einheitliche, statische Verzeichnisstruktur vorsetzen, so muss der Admin zunächst mithelfen: Er aktiviert in der Installation die Gruppenordner mit der App „Group Folders“. Anschließend tauchen in den „Einstellungen“ unter „Verwaltung“ die „Gruppen-Ordner“ auf. Dort lässt sich dann für jede benötigte gemeinsame Verzeichnishierarchie ein Gruppenordner anlegen. Die Ordner erhalten in der Web-Oberfläche ein für die Nutzer gut sichtbar um Personenprofile ergänztes Symbol.

Knifflig wird es, wenn innerhalb einer solchen Struktur unterschiedliche Zugriffsrechte gewünscht sind. Dann kann man einander überdeckende Grup-



Manchmal bleibt beim Synchronisieren eine Dateisperre hängen. Die muss dann der Admin im Wartungsmodus wegräumen.

penordner anlegen, etwa ebene1/ebene2a und ebene1/ebene2b und abweichende Rechte für ebene2a und ebene2b vergeben. Nextcloud blendet beide ein. Es funktioniert allerdings nur, wenn ebene1 auch als Gruppenordner angelegt ist. Das wird schnell unübersichtlich.

Gruppenordner überdecken die Ordner von Benutzern: Hat ein Nutzer etwa den Ordner „Texte“ angelegt und der Admin legt einen gleichnamigen Gruppenordner „Texte“ an, dann sieht der Benutzer seine in dem Ordner liegenden Dateien nicht mehr. In der Praxis bewährt es sich, Gruppenordnern deshalb ein spezielles Präfix zu verpassen. Nimmt der Admin dafür Nummern, tauchen diese Ordner in der Dateiübersicht im Browser gleich als erste auf.

Wer schon länger mit Gruppenordnern hantiert hat, dürfte eine frühe Einschränkung dieser Erweiterung noch kennengelernt haben: Die Versionierung und die Entlöschfunktionen, die den Charme von Nextcloud als Dateiablage ausmachen, waren dort nicht verfügbar. Doch das hat sich inzwischen geändert. Beides klappt in aktuellen Versionen auch in Gruppenordnern.

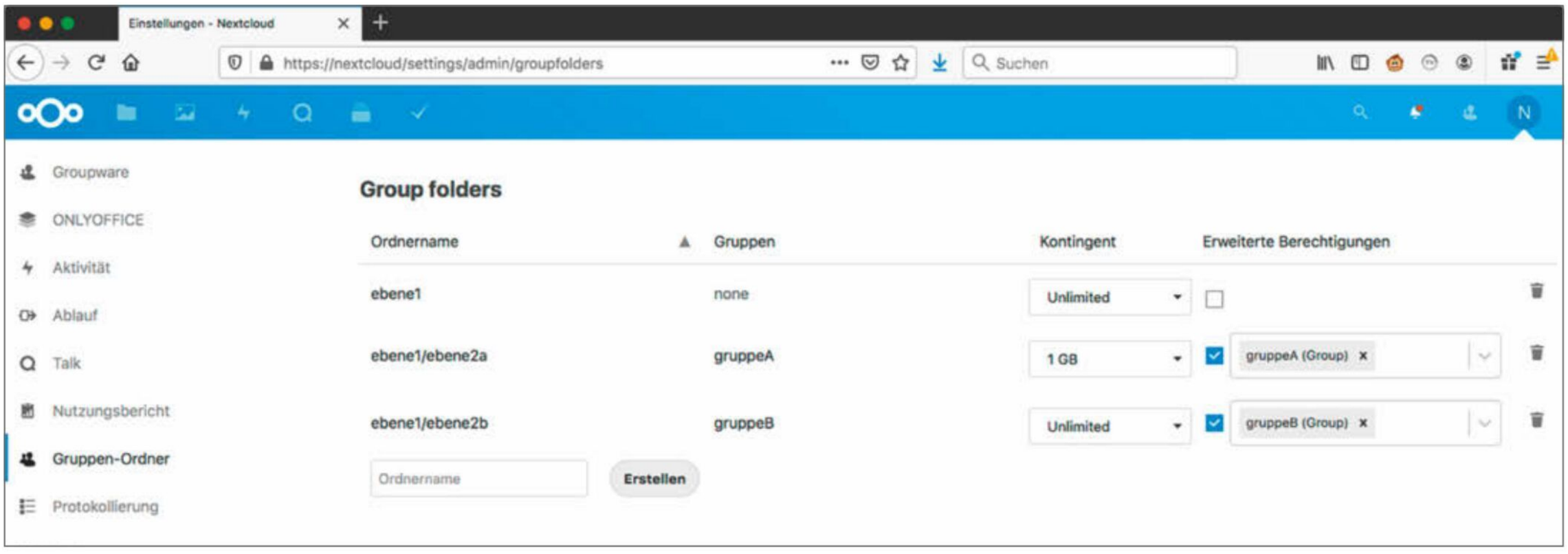
Gemeinsam arbeiten

Sollen mehrere Nutzer gleichzeitig an einem Dokument arbeiten, bieten sich zwei direkt in Nextcloud integrierbare Office-Pakete an: Collabora oder Onlyoffice. Letztlich sind das per Webbrowser bedienbare Office-Implementierungen, deren Integration technisch komplex ausfällt. Es wird neben Nextcloud ein weiterer Server hochgefahren und eine App stellt dann eine Brücke her. Das lässt sich mit Containern vergleichsweise geschmeidig einrichten, wenn einem das leicht von der Hand geht.

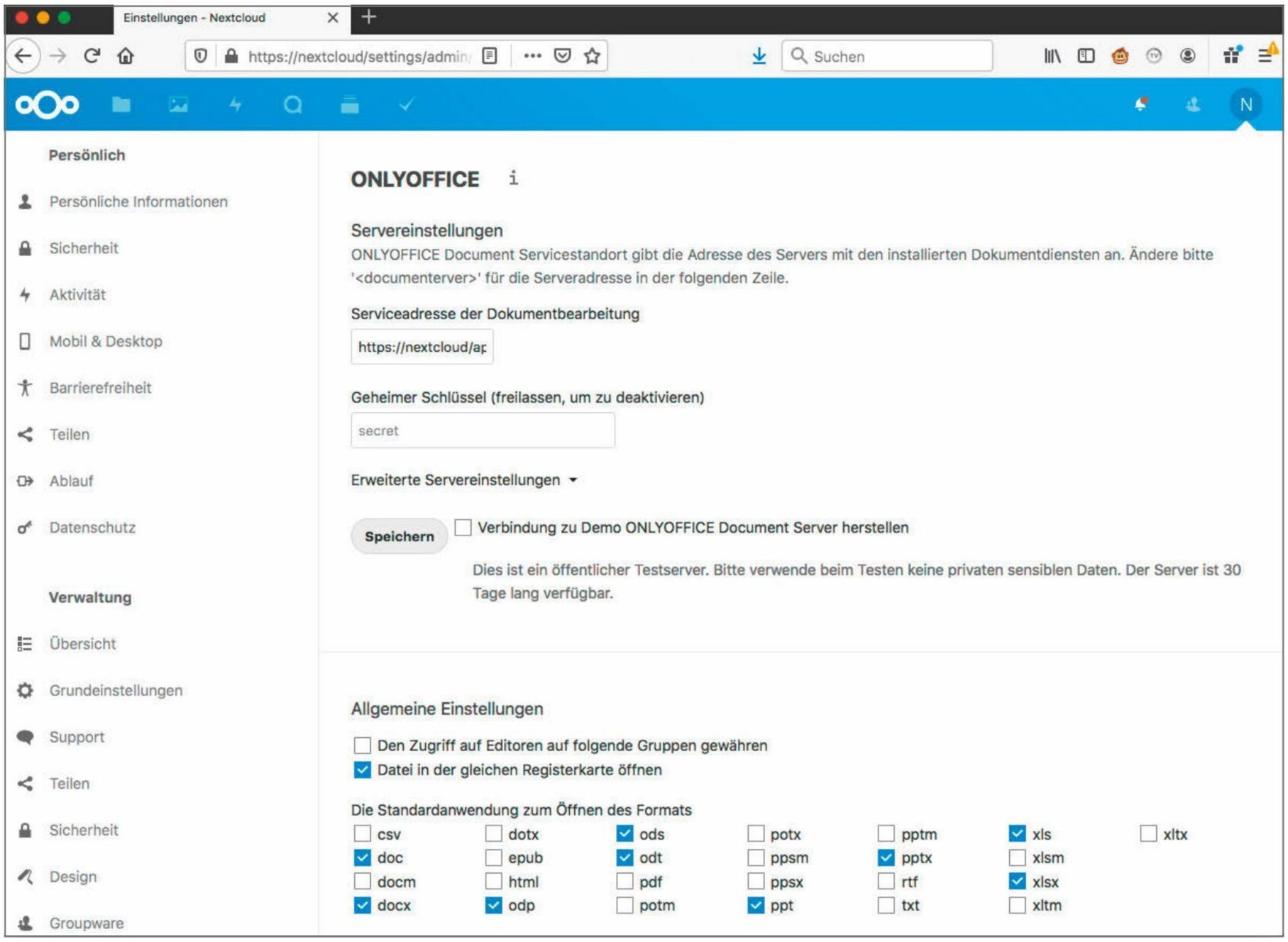
Die Nextcloud-Entwickler haben seit Nextcloud 18 einigen Aufwand investiert, um die Office-Integration noch viel einfacher zu gestalten. Was sich in der Ankündigung von Version 18 las, als hätten sie OnlyOffice eingebaut, ist dann doch etwas schwieriger. Grundsätzlich aber haben die Entwickler es erreicht, dass man mit Nextcloud 18 OnlyOffice verwenden kann, ohne parallel zu Nextcloud weitere Software einrichten zu müssen (was sonst der Fall war).

Der Admin muss, um OnlyOffice zu integrieren, zwei Apps seiner Nextcloud-Installation hinzufügen, die man am einfachsten über die Suche findet: den „Community Document Server“, der den für OnlyOffice bislang parallel einzurichtenden Server ersetzt, und „ONLYOFFICE“, das Nextcloud und den Server miteinander verbindet. In einer sauberen Nextcloud-Installation, die HTTPS-Zugriffe erzwingt, muss man gar nichts konfigurieren. Die zuletzt erwähnte App gibt die richtige URL für den Durchgriff auf die Server-Komponente vor.

Unter Umständen braucht das Einrichten des zur Zeit nur für x86-64 verfügbaren „Community Document Server“ eine Extraportion Geduld: Das Installieren der App zieht einen mehrere hundert MByte



Gruppenordner sind eigenwillig: Sie geben die Struktur einer Dateiablage für alle Benutzer vor. Wenn dort unterschiedliche Rechte gelten sollen, wird es schnell unübersichtlich.



Nach dem Hinzufügen der Server-Komponente für OnlyOffice ermittelt Nextcloud automatisch die Serviceadresse. Klappt der Zugriff, erscheinen weiter unten die „Allgemeinen Einstellungen“.

großen Download per Curl nach sich, der gern mal in Timeouts läuft. Bei einer lahmen Internet-Leitung kann es nötig sein, kurzzeitig den Timeout-Wert hochzusetzen (in der Zeile `RequestOptions::TIMEOUT => 30,` in .../lib/private/Http/Client/Client.php).

Leider zeigt sich der laut Nextcloud vor allem für Heimnutzer bereitgestellte Server nicht von seiner besten Seite: Änderungen an Office-Dokumenten, egal ob direkt mit OnlyOffice erzeugt oder von anderer Seite aufgespielt, landen nicht in den Nextcloud-Dateien, sondern werden verschluckt. Ein seit März 2020 noch immer offenes Ticket (siehe ct.de/weud) bringt etwas Klarheit in die Angelegenheit: Offenbar fehlt eine regelmäßig ausgeführte Aufgabe.

Das kann man automatisieren, etwa mit einem cron-Job, der über den Befehl

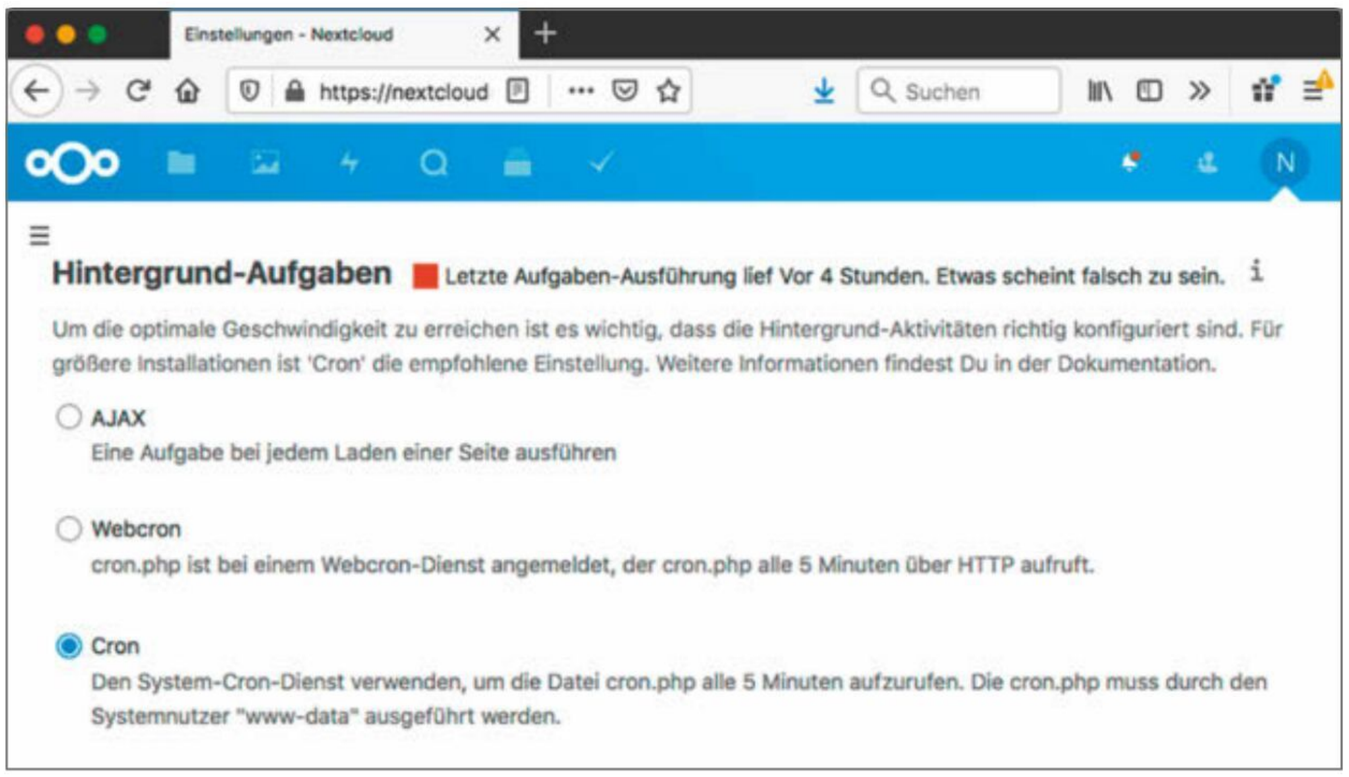
```
occ documentserver:flush
```

den Server auffordert, alle Änderungen aus seinen Caches preiszugeben. Dann landen in OnlyOffice ausgeführte Änderungen letztlich auch in den Dateien, die Nextcloud verwaltet. Dass das wirklich klappt, lässt sich einfach überprüfen: Für diese Dateien hält Nextcloud dann auch ältere Versionen bereit. Obendrein hilfreich ist es, einmal nächtlich Nextcloud neu starten zu lassen, um das Sichern offener Office-Dokumente zu erwzingen.

Regelmäßige Inspektion

Das Beispiel OnlyOffice, aber auch volllaufende Tabellen mit Dateisperren (Locks) untermauern, wie wichtig regelmäßig ausgeführte Reinigungsaufgaben für Nextcloud sind. Da es als PHP-Web-Anwendung daherkommt, ist es etwas schwieriger, solche auch regelmäßig auszuführen. Admins sollten sich deshalb vergewissern, dass das in ihrer Installation auch wirklich klappt.

Ein Blick in die Logs sollte regelmäßige Aufrufe von cron.php erkennen lassen. In üblichen Standardinstallationen wendet Nextcloud zum Ausführen von Hintergrundaufgaben Ajax als Methode an. Das heißt, dass es diese Aufgaben nur dann ausführt,



Die Nextcloud-Weboberfläche zeigt klar und deutlich, wenn das Ausführen regelmäßiger Aufgaben klemmt – in der Standardeinstellung löst aber genau der Abruf dieser Seite den Vorgang aus …

wenn Benutzer Nextcloud-Webseiten interaktiv abrufen. Alternativ bietet die Software den Rückgriff auf Webdienste an, die regelmäßig URLs abrufen – denen vertraut man dann die cron-URL seiner Installation an.

Wenn Nextcloud in einem Container läuft, tut es auch ein cron-Job auf dem Docker-Host. Entscheidend ist auch hier wieder zu überprüfen, ob die Hintergrundaufgaben auch ausgeführt werden. Nicht bei jeder Methode finden sich dafür eindeutige Spuren im Log, etwa der Aufruf von cron.php. In der Weboberfläche in den Grundeinstellungen bei den Hintergrundaufgaben verrät Nextcloud aber zuverlässig, wann die zuletzt ausgeführt worden sind, beziehungsweise warnt, wenn 30 Minuten lang nichts passiert ist.

Apropos wichtiger Blick: Als Admin sollten Sie sich durchaus die „Sicherheits- & Einrichtungswarnungen“ ansehen, die Sie in der Verwaltung unter „Übersicht“ finden. Nextcloud gibt nützliche Tipps, ob sich eine Installation noch verbessern lässt. In der Regel ist das der Fall. Zur Sicherheit können Sie auch den Dienst von Nextcloud anwenden, der von außen eine Installation auf die wesentlichen Probleme abklopft.

Versionen und Updates

Eine Nextcloud-Installation sollte schon wegen der Sicherheit stets aktuell gehalten werden. Je nach Installationsmethode sind dazu unterschiedliche Ansätze nötig. Dabei sind Admins nicht unbedingt gehalten, stets die allerneueste Fassung einzusetzen. Die Entwickler liefern immer auch noch Updates für mindestens einen Vorgänger.

Mit dieser Information im Hinterkopf verliert ein Aspekt beim Update seinen Schrecken: Oft sind beim Erscheinen einer neuen Major-Version längst nicht alle Apps bereits angepasst. Wer also bestimmte Apps stets braucht, tut gut daran, bei Updates etwas vorsichtiger zu Werke zu gehen und in einer getrennt laufenden Testumgebung zunächst zu prüfen, was nach dem Update nicht mehr bereitstehen würde. Davon ausgehend kann man dann qualifiziert einschätzen, wann der Wechsel der Major-Version möglich ist. Im Blindflug geht es auch, aber dann deaktiviert Nextcloud die nicht verträglichen Apps.

Im Container

Die aus unserer Sicht beste Methode, Nextcloud langfristig zu betreiben, ist der Einsatz als Container. Neue Versionen werden schnell als Image bereitgestellt. Auf einem NAS oder Linux-Host mit Docker und Watchtower laufen die Updates weitgehend störungsfrei automatisch. Man sollte nur darauf achten, einen parallel gestarteten Container mit MariaDB/MySQL auf eine konkrete Major-Version festzunageln. Sonst könnte ein Update händisches Einwirken erfordern.

Wenn Nextcloud auf diese Weise konfiguriert ist, sind ein paar Tricks hilfreich, wenn Sie sich in der Container-Welt noch nicht so auskennen. Um beispielsweise die Kommandozeile von Nextcloud zu erreichen, ruft man auf einem Docker-Host den Befehl

```
docker exec -it nextcloud bash
```

auf. Er startet eine Shell im nextcloud genannten Container. Die läuft dann allerdings mit root-Rechten, was für den Aufruf des Nextcloud-Admin-Befehls `occ` unpassend ist. Der sollte im Kontext des Benutzers aufgerufen werden, der auch bemüht wird, um den Webserver mit der Anwendung auszuführen. Um dessen ID herauszufinden, genügt es, in der root-Docker-Nextcloud-Shell

```
ls -ln
```

aufzurufen. Der Befehl gibt für die Dateien im Verzeichnis /var/www/html die Zugriffsrechte, Größe der Datei und die numerische ID des Nutzers gefolgt von weiteren Angaben aus. Schließen Sie die Shell (mit Strg+D) und rufen Sie sie erneut unter der ID des Nextcloud-Nutzers auf (meist 33 für www-data):

```
docker exec -it -u 33 nextcloud bash
```

Jetzt können Sie sich mit dem occ-Admin-Werkzeug in Ihrer Nextcloud-Installation umsehen, etwa mit

```
./occ list:users
```

eine Liste der bekannten Benutzer ausgeben lassen oder mit

```
./occ maintenance:mode --on
```

in den Wartungsmodus versetzen, um direkt in der Datenbank verriegelte Dateien zurückzusetzen. Wenn Sie die genauen Befehlszeilen nicht kennen, hilft das Anhängen von `--help` bei den meisten Optionen weiter.

Um im Wartungsmodus gesperrte Dateien beiseite zu räumen, muss der Admin im aktivierten Wartungsmodus direkt auf eine der von Nextcloud gepflegten Datenbanktabellen losgehen. Im Fall eines MySQL-Containers „clouddb“ könnte das wie folgt gehen:

```
docker exec -it clouddb bash
```

Anschließend verbindet er sich mit der MySQL-Instanz unter dem Nutzernamen und mit dem Passwort, das Nextcloud für die Zugriffe verwendet:

```
mysql -u nextcloud -p
```

Das Passwort fragt der Befehl interaktiv ab. Anschließend lässt sich mit

```
delete from file_locks where 1;
```

die Tabelle mit den Sperren leeren. Die MySQL-Shell verlassen Sie mit \q wieder und die Shell im Container mit Strg+D. Im Anschluss muss Nextcloud noch aus dem Wartungsmodus herausgeholt werden:

```
docker exec -it -u 33 nextcloud \
./occ maintenance:mode --off
```

Jetzt sollten der Client, der Probleme hatte, einzelne Dateien zu synchronisieren, wieder frisch ans Werk gehen. Wenn Sie Nextcloud in einer Hosting-Umgebung betreiben oder Owncloud verwenden, unterscheiden sich die Befehle geringfügig und die Datenbanktabellen sind mit einem Präfix anzugeben. Wie die Tabellen heißen, finden Sie in der MySQL-Shell mit `show tables;` heraus.

Mit diesem Wissen lässt sich mit einem cron-Job für den Benutzer root oder einen für Docker ermächtigten Benutzer auch die OnlyOffice-Macke ausbügeln:

```
1-59/1 * * * * docker exec -u 33 ↳
↲ nextcloud ./occ documentserver:flush

0 0 * * * docker restart nextcloud
```

Dann kann man auch gleich die Hintergrundaufgaben dort erledigen lassen, die passende cron-Zeile sähe folgendermaßen aus:

```
0-58/2 * * * * docker exec -u 33 ↳
↲ nextcloud php ./cron.php
```

Unterm Strich

Alles in allem ist Nextcloud sehr nützlich und arbeitet durchaus zuverlässig, selbst wenn es über mehrere Major-Versionen hinweg aktualisiert wird. Viele Schwierigkeiten resultieren aus unrunden Konfigurationen, etwa Container-Installationen mit vorgespannten Proxys: Die blockieren dann große Datei-Uploads, produzieren Timeouts oder lassen Nextcloud vermuten, dass jemand Schindluder treibt. Mit der Detailkonfiguration kann man Stunden, wenn nicht Tage vertun. Unsere Basisinstallation berücksichtigt hierbei den aktuellen Erkenntnisstand. (ps) c't

Literatur

[1] Matthias Withopf, **Das Grauen der Praxis**, Protokolle für die Internet-Festplatte: WebDAV & Co., c't 15/2010, S. 116

Erwähntes Ticket

ct.de/weud



Bild: Thomas Kuhlenbeck

Videokonferenzen mit Nextcloud Talk

Manchmal fühlt es sich seltsam an, Videokonferenzen über das System eines Dienstleisters zu führen. Wer kann mithören und -sehen oder geteilte Bildschirme beobachten? Mehr Privatsphäre verspricht ein selbstbetriebenes System wie Nextcloud. Seine Talk-App bietet Videokonferenzen, für die ein halbwegs aktueller Browser genügt.

Von **Johannes Endres**

Nextcloud Talk ist eine Server-App in der Open-Source-Cloud Nextcloud. Sie überträgt Bild und Ton direkt zwischen den Teilnehmern. Daher können alle Daten in Ende-zu-Ende-verschlüsselten Kanälen fließen. Dank der in modernen Browsern dafür integrierten Technik braucht man keine Extra-Software, nicht einmal fürs Teilen von Anwendungsfenstern oder des Desktops.

Durch die direkte Übertragung begrenzt der schwächste Upstream eines Teilnehmers die Zahl der Teilnehmer (siehe Kasten „Quadrat-Quatschen“). Nextcloud selbst gibt eine Obergrenze von rund

zehn Teilnehmern an, sofern alle an dicken Internetleitungen sitzen; wenn es eng wird, sollten einige ihr Video nicht senden, um Bandbreite zu sparen. Für Konferenzen mit mehr Teilnehmern muss eine zentrale Video-Bridge die Daten verteilen, um den Upstream zu schonen. Nextcloud bietet so etwas als Teil eines kompletten Videokonferenz-Backend inklusive Telefon-Gateway und Installationssupport an.

Die Server-App Nextcloud Talk hieß ursprünglich „Spreed“. Unter diesem Namen bietet die Stuttgarter Struktur AG Videokonferenzen an. Auch dort gibt es eine kostenlose Basis-Version, allerdings keine Möglichkeit, die Software selbst zu hosten. Der Name „Spreed“ taucht auch in Nextcloud noch an verschiedenen Stellen auf. Doch in Aussehen, Funktionsumfang und dem Grundverständnis unter-

Quadrat-Quatschen

Videokonferenzsysteme benutzen einen Server, der die Verbindung zwischen den Teilnehmern vermittelt. Beim Fluss der eigentlichen Bild- und Tondaten unterscheiden sie sich. Bei manchen, wie Nextcloud Talk, kommunizieren die Clients direkt untereinander; jeder sendet also an jeden anderen und empfängt von jedem. Deshalb sind die Anforderungen an den Server und seine Netzwerkverbindung sehr gering, für Nextcloud Talk genügt beispielsweise ein Raspberry Pi oder PHP auf gehostetem Webspace geringer Leistungsklasse.

Auch die Privatsphäre in der Videokonferenz lässt sich leicht schützen. Denn durch das Peer-to-Peer-Verfahren können jeweils zwei Clients Ende-zu-Ende-Verschlüsselung vereinbaren. Zudem ist die Gesamtheit der Daten recht schwer abzuhören, da die Datenströme verschiedene Wege nehmen.

Aber weil zur Übertragung UDP dient, fließen zwei getrennte Datenströme pro Teilnehmer-Paar, einer pro Übertragungsrichtung. Daraus entstehen zwei gewichtige Nachteile: Die Upstream-Bandbreite jedes Teilnehmers muss ausreichen, um allen anderen Teilnehmern parallel die Daten zu schicken. Wer an einem ADSL-Anschluss mit dem typischen schwachen Upstream sitzt, kann also nur bei Sitzungen mit sehr wenigen Teilnehmern mitmachen.

Außerdem steht oft die Network Address Translation (NAT) der Router im Weg. Sie funktioniert zuverlässig nur für ausgehende UDP-Ströme. Um die eingehenden Daten hindurchzuschmuggeln, müssen die Clients tricksen und UDP-Ports raten. Ob das funktioniert, hängt von der im Router implementierten NAT-Technik ab. Heim-Router lassen sich oft überlisten, aber manche Firmen- und vor allem Mobilfunknetze nutzen „symmetrische NAT“ und sind damit für die eingehenden Ströme undurchdringlich – wer dahinter sitzt, hört und sieht nichts von seinen Partnern. Hier hilft einer Nextcloud-Talk-Installation ein zusätzlicher TURN-Server auf die Sprünge.

Zentralismus

Die NAT- und Traffic-Probleme der Peer-to-Peer-Konferenzen betreffen nicht die Systeme, die Ton und Bild über einen zentralen Server leiten. Jeder Client muss Bild und Ton also nur einmal an den Server senden. Und auch wenn jeder die Daten der anderen Teilnehmer komplett erhält, passt das Profil „wenig senden, viel empfangen“ prima zu ADSL-Anschlüssen. Die Qualität des gesendeten Videos kann der Server pro Client an die Verbindungsqualität anpassen. So schaffen Systeme wie Zoom auch über wirklich schlechte Leitungen eine annehmbare Qualität.

Der Preis: die Privatsphäre. Denn wenn der Server die Videodaten optimieren soll, können sie nicht verschlüsselt bleiben. Statt Ende-zu-Ende-Sicherung kann nur die Übertragung zwischen Client und Server verschlüsselt laufen. Wer Admin-Zugriff auf den Video-Server hat, kann im Prinzip auch in die Videokonferenzen hineinsehen.

Dagegen hilft ein Server unter Ihrer Kontrolle. Doch die Anforderungen an den Server und vor allem seine Internetverbindung sind hoch. Schließlich muss er die Datenströme aller Teilnehmer empfangen und senden. Daher reichen ein Heimanschluss oder Hosting mit begrenztem Traffic für regelmäßige Konferenzen mit mehreren Teilnehmern nicht aus. Außerdem laufen Teile der Software außerhalb des Webservers, sodass ein einfaches Webhosting-Paket nicht genügt; ein Root-Server oder eine Container-Cloud muss es schon sein. Und die Video-Optimierung benötigt echte Rechenleistung, die ein Raspberry Pi nicht liefert.

scheidet sich Nextcloud Talk deutlich vom Stuttgarter Spreed.

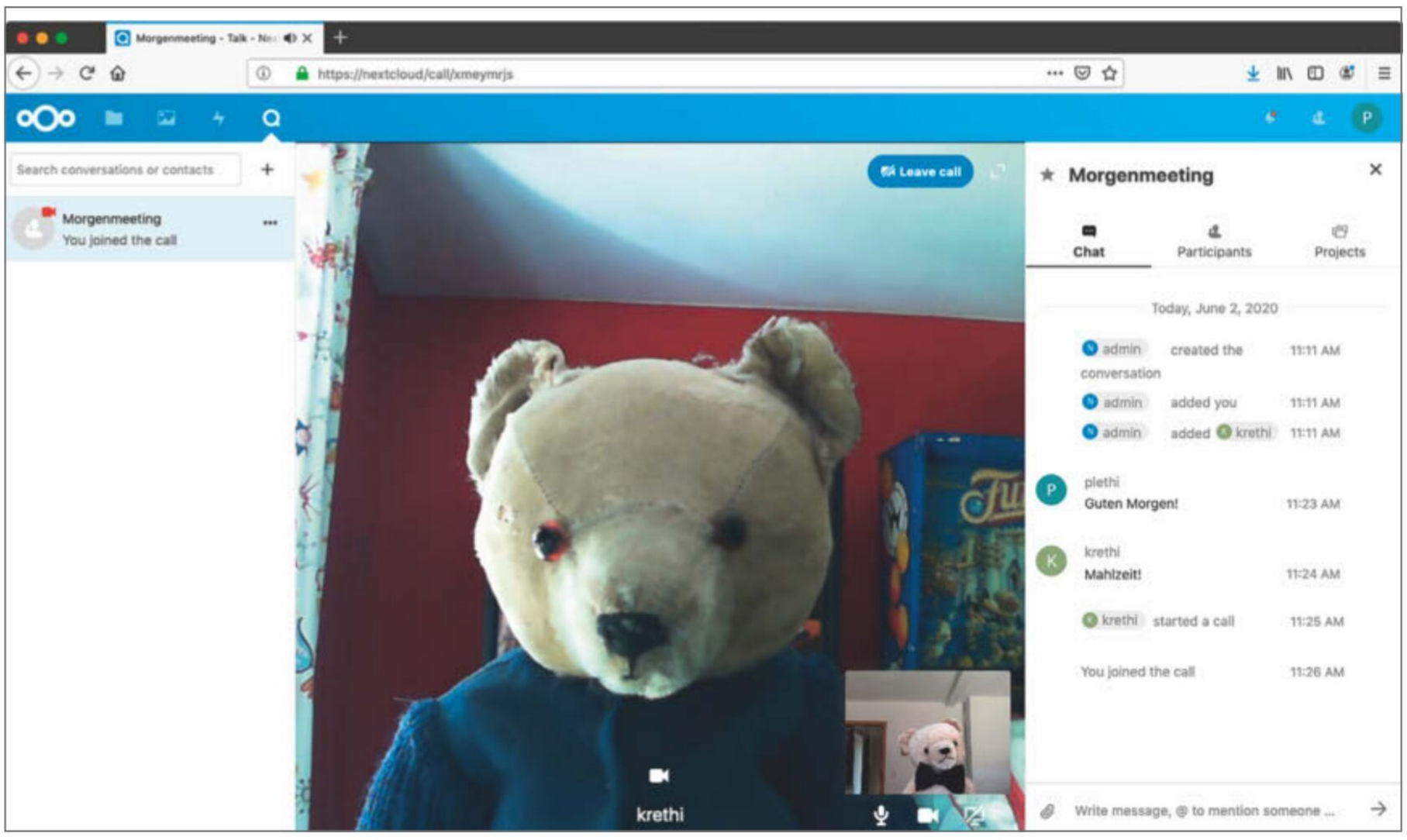
Denn während andere Videokonferenzsysteme eine mehr oder weniger ausgefeilte Chat-Funktion neben dem Video enthalten, folgt Nextcloud Talk der umgekehrten Logik: Es bietet einen klassischen Chat, den die Teilnehmer mit Bild und Ton weiterführen können. So gibt es keine von einem Moderator gestartete und beendete Konferenz (sofern der Nextcloud-Administrator nichts einschränkt). Stattdessen kann jeder Teilnehmer den Audio- und Video-Kanal öffnen. Moderatoren haben deshalb keine Möglichkeit, das Video zu beenden; wie jeder andere Teilnehmer können sie sich nur ausklinken oder den ganzen Chat abschalten. Und Anrufe klingeln nicht beim Angerufenen (wie bei Skype); denn wenn er in diesem Moment nicht am Chat teilnimmt, ist für ihn ja auch der Wechsel von Schrift zu Video nicht relevant.

Wenn man sich auf diese Sichtweise einlässt, erscheint die Bedienung logisch und man kann die Detail-Funktionen erkunden. Das ist auf jeden Fall nötig, denn die Nextcloud-Entwickler arbeiten derzeit heftig an Talk, sodass es oft Updates mit neuen Funktionen gibt.

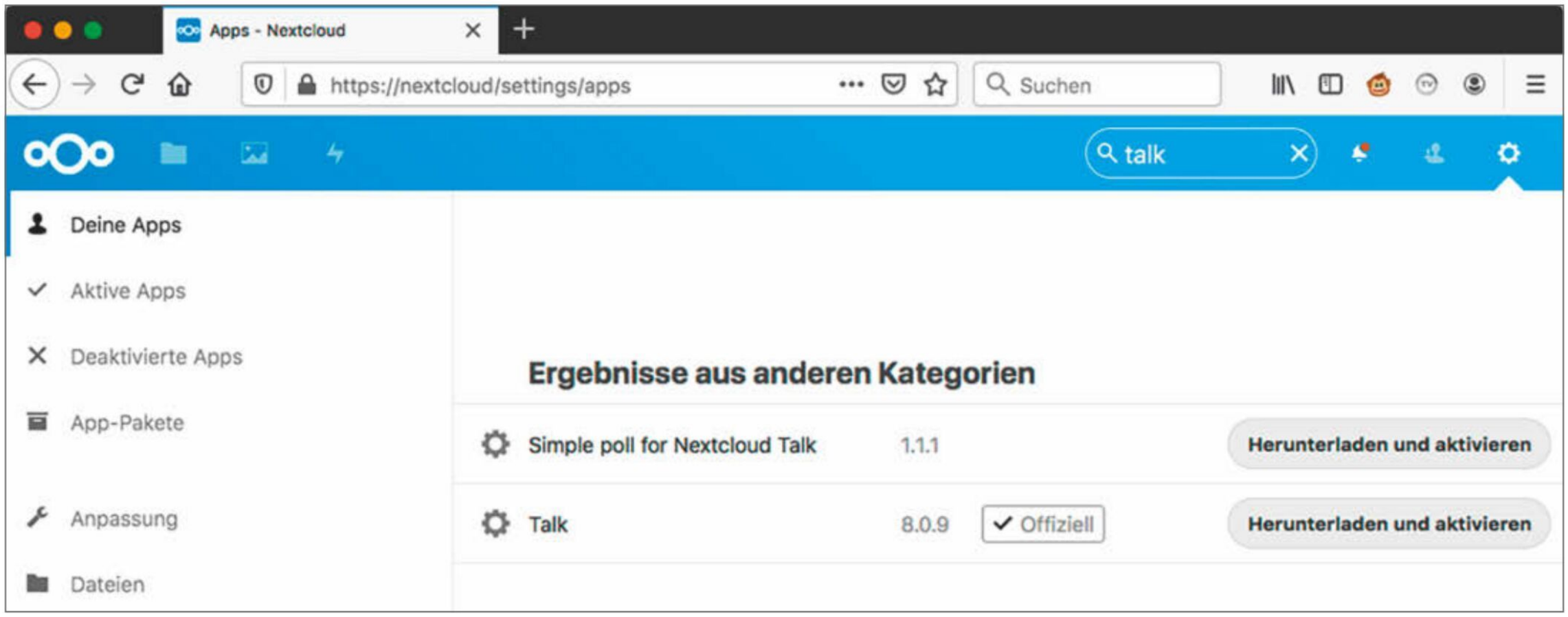
Einrichten

Bei Nextcloud Talk kommunizieren die Clients direkt per WebRTC miteinander [1]. Am Server bleibt also nur die anfängliche Verbindungsvermittlung hängen. Dafür genügt eine Nextcloud-Installation auf einem schwachen Server wie dem Raspberry Pi oder auf einer Hosting-Instanz. Er muss lediglich für alle potenziellen Teilnehmer erreichbar sein, notfalls per DynDNS und Port-Forwarding – allerdings zwingend per HTTPS, denn moderne Browser erlauben die Nutzung von Kamera und Mikrofon nur auf damit gesicherten Seiten.

Wenn die Talk-App in Ihrer Nextcloud-Installation fehlt, genügen wenige Klicks zur Installation: Als Administrator eingeloggt klickt man auf das Profil-Icon oben rechts und wählt aus dem Menü „Apps“ aus. Die App-Liste ist etwas unübersichtlich, doch die leicht zu übersehende Suchfunktion hinter dem Lupensymbol in der oberen Leiste hilft weiter. Nach



Etwas gewöhnungsbedürftig: Die Videotelefonie in Nextcloud verlängert den Chat mit anderen Mitteln.



Das Hinzufügen der Talk-App gelingt über die Suchfunktion ohne viel Herumklickerei.

einem Klick auf „Herunterladen und installieren“ wirft Nextcloud gelegentlich eine Fehlermeldung, die aber oft gar keine Grundlage hat. Es lohnt sich dann, unter „Deaktivierte Apps“ nachzusehen und Talk dort einfach zu aktivieren, sofern es trotz Fehlermeldung in der Liste steht.

In den Einstellungen des Administrator-Kontos gibt es nun einen Bereich „Talk“, in dem vor allem die Zugriffsrechte interessant sind. Diese sind ab Werk recht liberal, aber nicht gefährlich. Bevor der Administrator etwas daran umstellt, sollte er sich mit der Benutzung von Talk vertraut machen.

Losbrowsern

Die Video-Übertragung in Nextcloud Talk braucht auf dem Desktop keine eigene Software, sondern nur einen halbwegs aktuellen Browser, der WebRTC spricht. Chrome und Firefox funktionieren ohne Weiteres. Seit Talk 11 sollte auch Opera rudimentär funktionieren. Unter Windows steht manchmal noch eine systemweite Sperre im Weg. Dann muss man in den Einstellungen unter „Datenschutz“ auf den Seiten der Kamera und des Mikrofons jeweils ganz unten „Zulassen, dass Desktop-Apps ... zugreifen“.

Zur Talk-Zentrale gelangt man über das Q-Icon in der oberen Menüleiste der Weboberfläche. Über das Suchfeld und das Pluszeichen in der linken Spalte dieser Ansicht lassen sich zwei Arten von Chats starten, die sich durch die möglichen Teilnehmer unterscheiden: Das Suchfeld verspricht zwar, nach „Kontakten“ zu suchen; damit sind hier aber nur registrierte Benutzer der Nextcloud-Instanz gemeint. Der Klick auf ein Suchergebnis (Kontakt oder Gruppe) startet sofort den Chat. Das hat jedoch zwei Nachteile: In einen solchen Chat kann der Moderator keine weiteren Teilnehmer aufnehmen und nur der Admin darf den dort angezeigten Gruppenteilnehmer hinzufügen.

Lösungen für beides bringt das unscheinbare Pluszeichen neben dem Suchfeld. Es lässt Nextcloud eine „neue Gruppenunterhaltung erstellen“. Darin kann der Moderator flexibel Teilnehmer einladen – auch Externe ohne Benutzerkonto und auch nach dem Start des Chats. Auf diese Weise begonnene Gespräche zeigen in der rechten Spalte unter „Teilnehmer“ ein zusätzliches Suchfeld, über das der Moderator den Diskutantenkreis erweitert. Um Einzelne wieder auszuschließen, gibt es im Drei-Pünktchen-Menü rechts neben jedem Namen den Eintrag „Teilnehmer entfernen“.

Wer oft mit denselben Teilnehmern chattet, sollte sich die Nextcloud-App „Circles“ installieren (lassen). Damit kann jeder Nutzer eigene Listen anlegen, die in den Suchergebnissen erscheinen. Allerdings reduziert Circles nur die Klickarbeit beim Start eines

Angreifbares Drumherum

Wer sich um die Privatheit seiner Videokonferenzen sorgt, denkt zuerst an ungebetene Zuschauer. Doch drumherum gibt es allerhand weitere Ziele für den Lauschangriff, von denen einige interessanter und manche leichter zu knacken sind als das Online-Meeting selbst.

Da ist zunächst die Einladung: Wer die Zugangsdaten zur Konferenz – und vielleicht sogar das Meeting-Passwort – per E-Mail verschickt, verspielt jeden Schutz. Denn es ist wesentlich einfacher, unverschlüsselte Mails abzugreifen und auszuwerten, als Videokonferenzen im Netzwerk aufzufinden. Zumindest das Teilnahme-Passwort darf also nur über einen gesicherten Kanal gehen, beispielsweise ein persönliches Telefonat oder einen Ende-zu-Ende verschlüsselnden Messenger.

Besser vertraut man diesem auch die Einladung an, denn oft interessiert es Spione vor allem, wer mit wem worüber redet. Welche Details in der Konferenz zum Thema „Raketenteile für Nordkorea“ zwischen Vorstand und Vertrieb erörtert wurden, ist weniger relevant als die Tatsache, dass sie überhaupt stattfindet. Dies ist nur einer der Gründe, auch den Online-Kalender mit hoher Priorität zu sichern.

Ähnlich sieht es bei Systemen aus, die nur registrierte Teilnehmer zu einer Konferenz zulassen. Das wirkt auf den ersten Blick sicherer. Doch wenn diese Datenbank gehackt wird, lässt sich rekonstruieren, wer mit wem kommuniziert. Immer wieder patzen Online-Dienste bei der Sicherung ihrer Nutzerdatenbanken. Auch wer einen eigenen Konferenz-Server betreibt, muss selbst dann noch auf dessen Sicherheit achten, wenn alle Geheimgespräche längst geführt sind. Bei Konferenzen mit anonymer Teilnahme nur über Einladungs-Link und Meeting-Passwort fallen solche Daten gar nicht erst an.

Doch warum mühsam das Netzwerk oder den von einem Profi-Admin bewachten Server angreifen, wenn es ein einfacheres Ziel gibt: den PC eines Teilnehmers. Mit einem Trojaner auf einem beteiligten Rechner lässt sich die ganze Konferenz komfortabel und unbemerkt beobachten. Alle anderen Sicherungsmaßnahmen haben also keinen Zweck, wenn auch nur ein einziger Teilnehmer an einem ungesicherten PC sitzt.

Offensichtlich muss man die Teilnahme von ganz fremden Rechnern wie Business-Stationen im Hotel oder Internet-Cafés unterbinden. Aber auch der private PC, auf dem abends Pornos und illegale Streams aus dubiosen Quellen laufen, darf nie Teil einer Videokonferenz mit Sicherheitsanspruch sein. Wer Geschäftsgeheimnisse oder personenbezogene Daten zu schützen hat, sollte mobilen Mitarbeitern gut gesicherte und zentral verwaltete Geräte mitgeben.

Für Betriebsspione sind Videokonferenzen eigentlich kein besonders attraktives Ziel, denn die Informationsdichte ist meist genauso gering wie bei normalen Meetings. Warum soll sich ein Hacker das langatmige Selbstlob eines Junior Managers und die einschläfernden Belehrungen des Chefs antun, wenn er die relevanten Informationen an anderer Stelle viel kompakter bekommt? Der häufigste Patzer ist da sicher ein Meeting-Protokoll, das als unverschlüsselte Mail an alle geht. Oft werden in einer Konferenz auch Präsentationen gezeigt, die zuvor mit mehreren Personen abgestimmt werden müssen und dafür per Mail herumgehen oder auf unzureichend gesicherten Cloud-Diensten landen.

Für professionelle Lauscher sind Videokonferenzen also vor allem dann interessant, wenn es auf den genauen Wortlaut und Ablauf des Gesprächs ankommt oder wenn alle anderen Informationsquellen gut vernagelt sind. Genau darauf sollten sich Firmen-Admins konzentrieren. Um ihnen keine zusätzlichen Baustellen zu eröffnen, reicht es für viele Unternehmen, ihre Videokonferenzen über einen Dienstleister abzuwickeln, der die Sicherheit vertraglich zusichert. Wenn es dazu an Vertrauen oder juristischer Expertise fehlt, oder wenn der mögliche Schaden das vertraglich Absicherbare übersteigt, lohnt sich für Unternehmen die Pflege eines eigenen Videokonferenz-Systems.

Chats; Änderungen an einer Liste via Circles wirken sich nicht auf schon gestartete Chats aus.

Mit-Chatter ohne Benutzerkonto heißen bei Talk „Gäste“. Für sie schaltet der Moderator die Teilnahme per Link entweder schon beim Einrichten des Chats frei oder später über das Drei-Punkte-Menü des Chats ganz oben in der rechten Spalte. Ist diese Option aktiv, erscheint eine weitere zum Setzen eines Pass-

worts. Das ist überaus ratsam, um etwas mehr Kontrolle über die Teilnehmerschaft zu behalten. Denn anders als beispielsweise Zoom kann Talk einen Chat nicht für Neuanmeldungen sperren, wenn alle erwünschten Teilnehmer dabei sind. Der Moderator kann zwar auch Gäste aus der Runde entfernen. Doch ein einfacher Reload des Links genügt einem renitenten Gast dann, um wieder dabei zu sein. Link und Passwort gehören in zwei verschiedene, sichere Kanäle zu den Teilnehmern (siehe Kasten „Angreifbares Drumherum“). Seit Talk 11 kann der Moderator einen Chat immerhin für weitere Nachrichten sperren.

Gäste-Links mit Passwort sind deutlich sicherer, als bloß für Talk-Sitzungen zusätzliche Nextcloud-Benutzer anzulegen: Gäste haben keine weiteren Rechte auf der Cloud und sie können nicht vergessen, sich auszuloggen. Das passiert sonst bei Nextcloud oft, da Login-Sessions über viele Tage, über Browser-Neustarts und sogar Reboots bestehen bleiben.

Bei Zoom gab es Probleme mit Spaßvögeln, die einfach die recht kurzen Meeting-IDs durchprobierten und so in ungesicherte Meetings stolperten. Bei Talk machen das drei Faktoren ziemlich unwahrscheinlich: Die Skript-Kinder kennen die Server-Adresse einer selbst gehosteten Nextcloud nicht vorab und die Meeting-ID enthält nicht nur Ziffern, sondern auch Buchstaben, sodass der Suchraum hinlänglich groß ist. Zudem finden auf einer Nextcloud weitaus weniger Konferenzen statt, was die Trefferwahrscheinlichkeit weiter senkt.

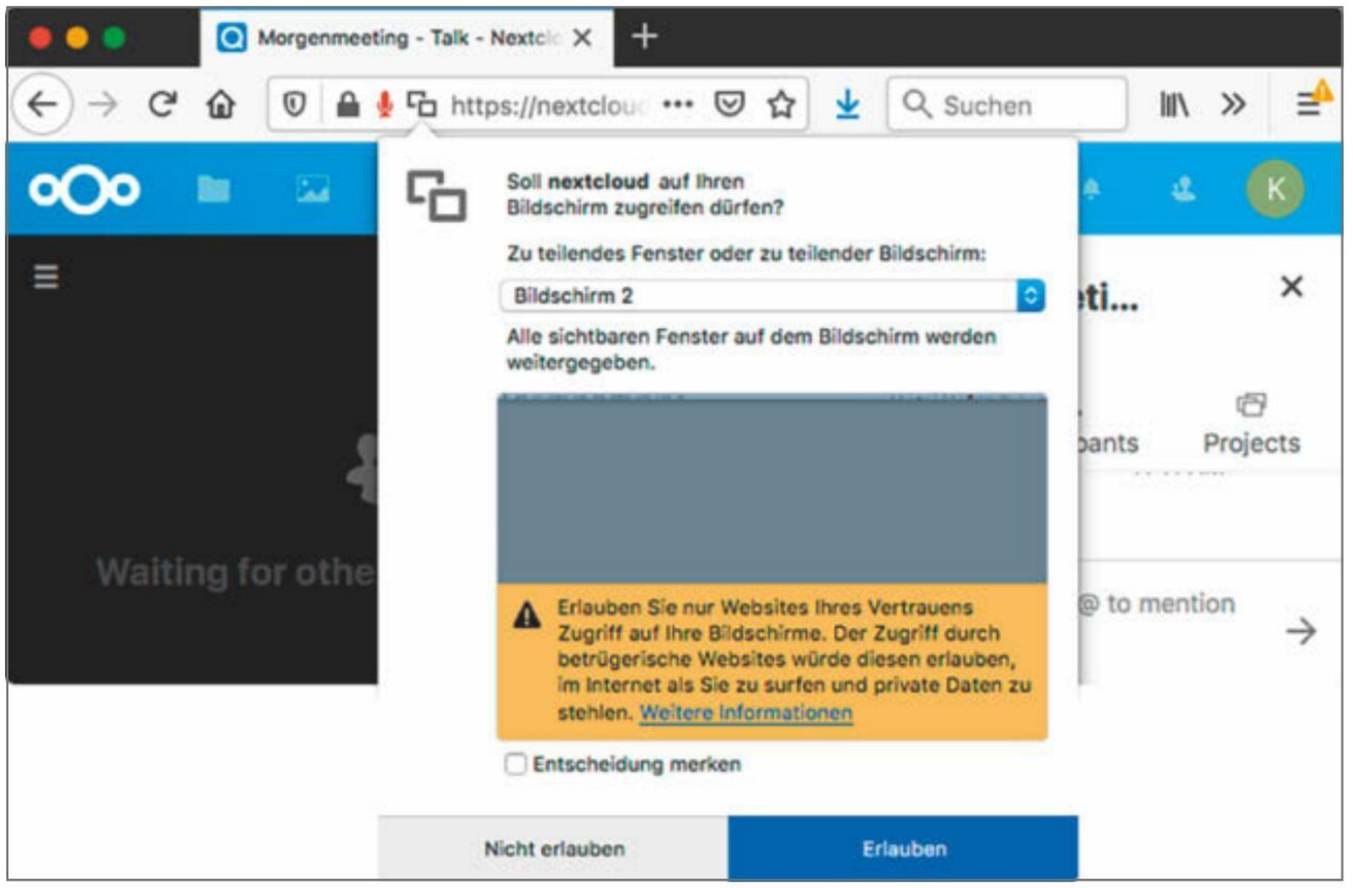
Loslabern

In der Talk-Logik schaltet man die Video-Konferenz als zusätzlichen Kanal im Chat ein. Der Knopf dafür heißt „Anruf starten“ und leuchtet oben im mittleren Chat-Fenster. Sofern der Admin das nicht eingeschränkt hat (siehe unten), darf jeder Teilnehmer des Chats den Knopf drücken. Bei den anderen heißt der Knopf „Anruf beitreten“.

Auch Gäste landen über den Freigabe-Link zunächst im Chat. In die Videokonferenz müssen sie sich selbst mit einem Klick auf den Knopf „Anruf beitreten“ einklinken. Das ist für Menschen, die Talk nicht regelmäßig nutzen, etwas ungewohnt und sollte daher in der Einladung erklärt werden.

In der Videokonferenz gibt es unspektakulär wenige Bedienelemente: Über drei Knöpfchen unten im Bild kann jeder sein Kamerabild und das Mikrofon ein- und ausschalten sowie Fenster oder den ganzen Bildschirm teilen. In den Fenstern mit den Bildern der anderen gibt es nur ein Kamerasymbol, mit dem sich das übertragene Bild lokal abschalten lässt. Die Auswahl von Video- und Audiogerät erledigt der WebRTC-Part des Browsers. Seit Talk 11 können Nutzer eine virtuelle Hand heben und der Moderator alle Nutzer mit einem Klick stummschalten.

Wenn während eines Videofonats die Bandbreite knapp wird, sollten einer oder mehrere Teilnehmer auf ihrer Seite das Senden des Bildes abschalten. Denn nur dann reduziert sich die übertragene Datenmenge.



In der Talk-App kann man während einer Videokonferenz einzelne Fenster oder einen ganzen Bildschirm teilen.

Rauswerfer

Weil das nicht zur Logik des Chats mit anderen Mitteln passt, gibt es keine Möglichkeit, einen Talk-Anruf zu beenden; er endet automatisch, wenn der letzte Teilnehmer den Anruf verlässt. Selbst ein Moderator kann eine Videokonferenz nicht schließen. Wenn er zwischenzeitlich den Call verlässt, läuft der Anruf weiter. Um dennoch für Ruhe zu sorgen, kann der Moderator in der Liste auf der linken Fensterseite im Drei-Punkt-Menü des Chats die „Unterhaltung löschen“. Damit geht allerdings auch der Chat-Verlauf verloren. Soll der Chat erhalten bleiben, kann der Moderator ihn sperren.

Auch wenn es etwas umständlich ist, sollte man eine Unterhaltung nach Ende unbedingt auf diese Weise sperren, weil sie aktiv bleibt, auch wenn kein Moderator mehr teilnimmt. Wer den Gast-Link auf einen Chat kennt, kann ihn weiter benutzen – auch mit ganz anderen Leuten. Außerdem bleibt die gesamte Chat-History für jeden sichtbar, der den Link kennt. Besser, man löscht die Unterhaltung ganz.

Lobbyismus

Spreed kennt einen Webinar-Modus, in dem die Teilnehmer lediglich den Moderatoren zusehen und lauschen dürfen, aber selbst weder Ton noch Bild übertragen. Zudem landen Sie in einer „Lobby“, wenn sie den Link aufrufen, bevor ein Moderator das Webinar gestartet hat.

Auch bei Talk ist etwas hochtrabend von „Webinar“ die Rede. Doch hier gibt es nur die Lobby-Funktion, und zwar im Menü des Chats im Kopf der rechten Spalte. Ein Haken bei „Lobby aktivieren“ setzt lediglich Zugangsrechte: Moderatoren können den Chat beliebig benutzen, alle anderen bekommen eine „Du bist in der Lobby“-Seite zu sehen. Es gibt keinen Knopf, um einzelne Teilnehmer aus der Lobby zu holen; stattdessen muss ein Moderator die Lobby dort deaktivieren, wo er sie eingeschaltet hat. Dann fallen alle wartenden Teilnehmer in den Chat ein, der sich anschließend nicht mehr von einem anderen Talk-Chat unterscheidet: Jeder kann das Video starten und wer dabei sein möchte, muss dem Anruf „beitreten“. Der Moderator kann ein „Webinar“ genauso wenig zentral beenden wie einen anderen Talk-Chat.

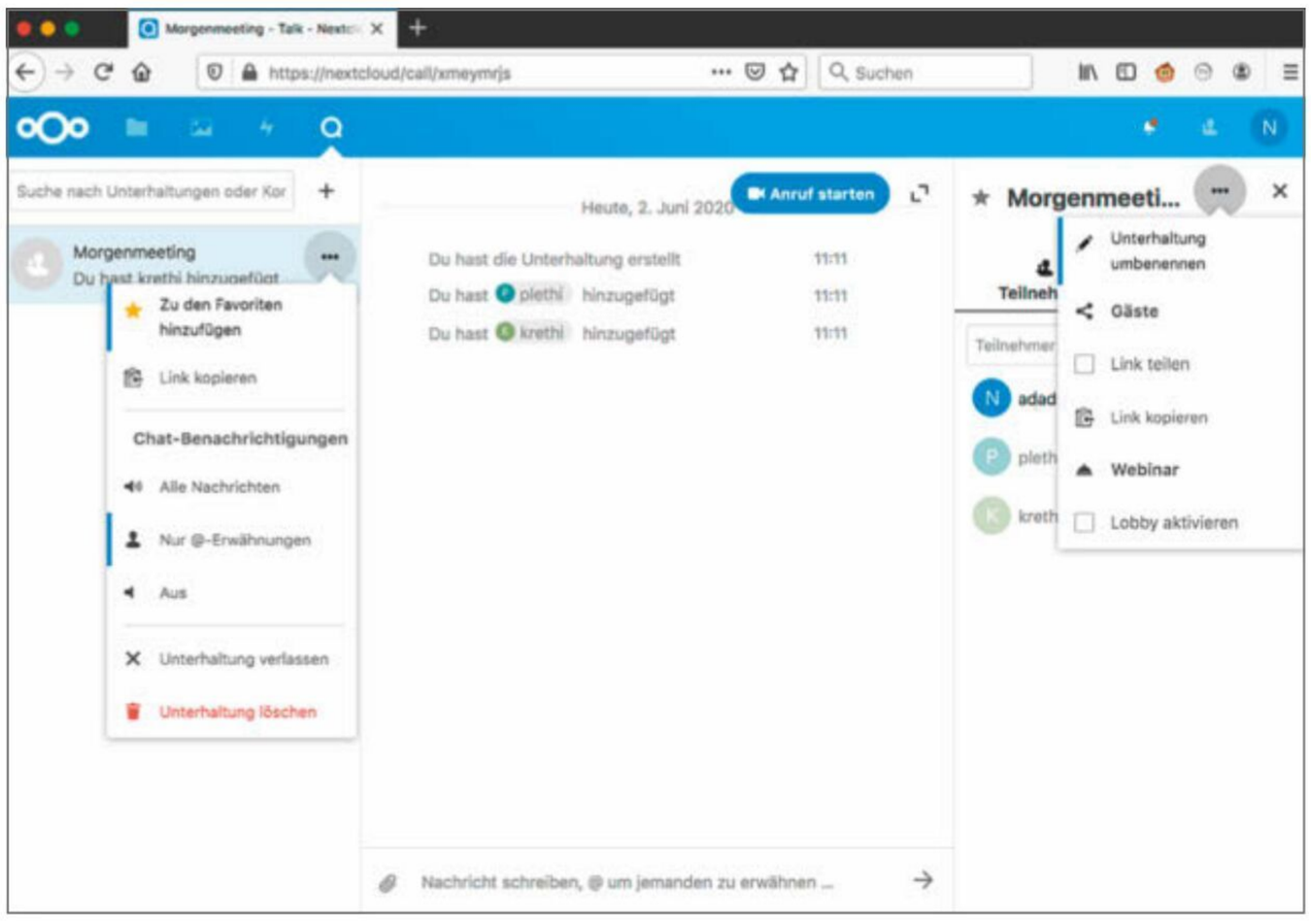
Setzt man beim Aktivieren der Lobby einen Zeitpunkt, dann purzeln in diesem Moment alle in der Lobby verharrenden Teilnehmer in den Chat und die Lobby wird automatisch deaktiviert. Nachzügler landen also direkt im Chat.

Anderswo schwätzen

Talk hängt sich an verschiedensten Stellen in die Nextcloud-Weboberfläche ein, zum Beispiel auch in die Suche nach anderen Benutzern über das Symbol in der Hauptleiste. Etwas unerwartet ist die Integration ins Datei-Sharing: Im Info-Bereich zu jeder Datei erscheint Talk als zusätzlicher Reiter. Sobald man die Datei teilt, entsteht ein Chat, dem jeder beitreten kann, der auch Zugriff auf die Datei hat. Und da es sich um einen Talk-Chat handelt, kann jeder Teilnehmer ein (Video-)Telefonat initiieren. Richtig nützlich wird das, wenn man eine Text-Datei im Online-Editor öffnet und Änderungen live diskutiert. Das soll prinzipiell auch mit anderen Online-Editoren funktionieren, aber unter anderem Onlyoffice verdrängt den Info-Bereich komplett, sodass der Chat nicht zugänglich ist.

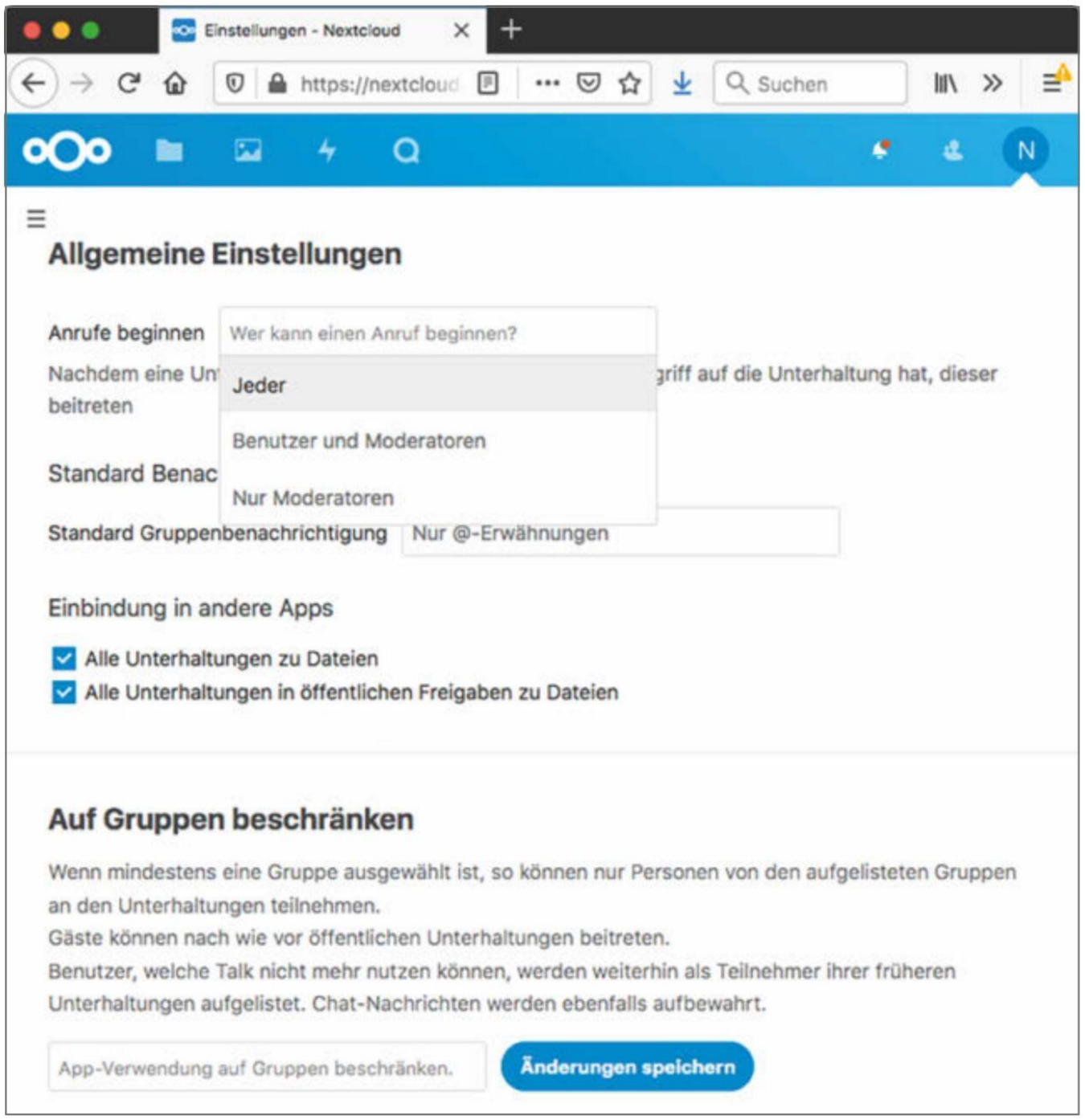
Admin-Einstellungen

Diese Datei-Chats gehören zu den wenigen Dingen, die der Admin an Talk konfigurieren kann. Auf der Einstellungsseite kann er die Integration nur für öffentlich geteilte Dateien oder gleich ganz deaktivieren. Hier kann der Admin auch bestimmen, dass



Die wesentlichen Funktionen, um Chats und somit Videokonferenzen zu verwalten, stecken in den Menüs hinter den Pünktchen.

Für den Admin sind von den wenigen Talk-Einstellungen vor allem die Zugriffsrechte interessant.



nur Benutzer mit Nextcloud-Konto oder Moderatoren in einem Chat den Anruf starten dürfen. Das ist sehr wörtlich zu verstehen: Der Admin beschränkt lediglich das Recht, einen Anruf zu starten; ist das einmal passiert, kann der Moderator ihn nicht beenden, sondern nur selbst austreten. Die anderen können dann munter weiterschwätzen und dem Gespräch auch beitreten. Erst wenn der letzte Teilnehmer den Anruf verlässt, wird er beendet. Will der Moderator den Anruf beenden, kann er also weiterhin nur wie oben beschrieben den Chat löschen oder alle Teilnehmer rauskicken.

Apps

Nextcloud Talk gibt es auch als App für Android und iOS. Sie erfragt zunächst die Anmeldung an einer Nextcloud und zeigt dann die Liste aller Talk-Chats, an denen der angemeldete Benutzer beteiligt ist. Neue Chats kann er darüber auch starten. Die Video- oder Ton-Teilnahme startet man im Chat mit den üblichen Icons am oberen Bildrand.

Allerdings schränkt die Android-App die Anonymität ein: An Konferenzen auf der Cloud, an der die App angemeldet ist, kann man nicht mehr als Gast teilnehmen. Der Server erkennt auch beim Pasten des öffentlichen Links den Benutzer und zeigt ihn mit Namen in der Teilnehmerliste. Bei Konferenzen auf anderen Cloud-Instanzen erscheint ein App-Benutzer zwar im Chat als Gast. Doch dort wollte in unseren Tests die Teilnahme am Video per App nicht gelingen.

App-Nutzer können unterwegs oft keine Anrufe starten oder ihnen beitreten, weil die NAT ihres Mobilfunk-Providers den Verbindungsaufbau verhindert. Auf ihrer Seite erscheint dann keine Fehlermeldung, aber im Talk-Web-Interface des Moderators ein Hinweis auf „TURN“. Womit Arbeit auf den Admin zukommt: Er kann durch Aufsetzen eines Servers für das „Traversal Using Relays around NAT“-Protokoll auch solchen Nutzern Zugang zu Talk verschaffen. (ps) c't

Literatur

[1] Herbert Braun, **Anruf aus dem WWW**, Echtzeit-Kommunikation zwischen Browsern mit WebRTC, c't 19/2013, S. 170

Downloads

ct.de/wxa6

E-Mail: Begriffe und Protokolle

E-Mails machen noch immer einen großen Teil elektronischer Kommunikation aus. Die uralten Protokolle wie SMTP und IMAP sind von Haus aus aber ziemlich unsicher. Admins, aber auch Nutzer, müssen deshalb ein paar Grundregeln über die Protokolle, Spam und Spoofing kennen.

Von **Jan Mahn**



Bild: Albert Hulm

Grundwissen E-Mail-Protokolle 126
Mailserver einrichten mit Mailcow 134

Ohne E-Mails wäre Büroarbeit in vielen Unternehmen kaum möglich. Und auch privates Online-Shopping und andere Alltagsaufgaben sähen ohne Mails definitiv anders aus. Daran haben auch Messenger und Videokonferenzen nichts geändert. Erfolgreich ist das System auch, weil es von Anfang an dezentral konzipiert war: Jeder Internetnutzer könnte seinen eigenen Mailserver betreiben und anderen darüber Mails in die Postbox legen. Dass viele in der Praxis auf einen eigenen Mailserver verzichten und auch Unternehmen gerne auf die Infrastruktur großer Anbieter zurückgreifen, liegt unter anderem an den großen Plagen, die diese Offenheit des Systems mitbringt: Spammer und Betrüger. Wer einen eigenen Mailserver betreiben will, muss ziemlich genau wissen, wie die Protokolle funktionieren und was es zu beachten gilt. Sonst wird er mit Spam überflutet oder sogar selbst zur Spam-Schleuder.

Dieser Artikel soll Administratoren, die einen Mailserver betreiben wollen, das nötige Hintergrundwissen liefern, damit das sicher und vor allem Spamarm gelingt. Aber auch für Anwender ist es wichtig zu verstehen, wie der Mailversand technisch funktioniert, wo es Probleme gibt und warum Spamming und Betrug so verbreitet sind. Wer bei der Einrichtung eines Mailkontos in Thunderbird oder Outlook unbedacht klickt, kann vielleicht mailen, riskiert aber unangenehme Sicherheitsprobleme und könnte schlimmstenfalls seine Zugangsdaten im Minutentakt unverschlüsselt durchs ungesicherte Hotel-WLAN funken.

Kernstück der Mailinfrastruktur sind Mailserver, die im Internet hängen und mit anderen Mailservern kommunizieren. Eine zentrale Instanz, bei der man sich als Serverbetreiber registrieren muss, um am weltweiten System teilzunehmen, gibt es nicht. Voraussetzungen sind lediglich eine Domain und passende DNS-Einträge für diese.

MX: Gefunden werden

Wer einen Mailserver betreiben will, der Nachrichten annimmt, braucht nur eine Mailserver-Software, die auf Port 25 im Internet horcht und eine passende Portfreigabe für diesen Port in der Firewall. Wenn ein anderer Mailserver Ihrem Server eine Mail zustellen will, wird er zunächst das DNS befragen, unter welcher Adresse Ihr Mailserver zu finden ist. Für empfangende Mailserver gibt es im DNS Einträge vom Typ MX (für „mail exchange“). Ein MX-Eintrag enthält eine Angabe für die Priorität – eine natürliche Zahl. Um Redundanz beim Empfang zu schaffen, ist es nämlich durchaus üblich, mehrere Mailserver als MX-Einträge zu veröffentlichen. Der versendende Mailserver ruft sie alle ab und probiert zunächst den Server mit dem niedrigsten Prioritätswert aus. Antwortet dieser nicht, versucht er es beim nächsten, dann beim übernächsten.

Ein MX-Eintrag enthält keine IP-Adresse, sondern einen vollständigen Namen (FQDN), zum Beispiel mail1.example.org. Um diesen Namen zu einer IP aufzulösen, ist also eine weitere DNS-Anfrage nach dem A- oder AAAA-Eintrag für eine IPv4- oder IPv6-Adresse nötig. Der MX-Eintrag für heise.de sieht bei Abfrage mit `nslookup -query=MX heise.de` beispielsweise so aus:

```
heise.de mail exchanger = 10 relay.heise.de.
```

Wenn Sie für eine Domain nur einen Server betreiben, auf dem nebenbei zum Beispiel noch die zugehörige Website auf den Ports 80 und 443 liegt, können Sie sich den MX-Eintrag auch komplett sparen. Findet ein versendender Mailserver keinen MX-Eintrag, versucht er, die Mail an den A- oder AAAA-Eintrag der Domäne zuzustellen, also zum Beispiel direkt an example.org.

Sobald man einen Server mit Port 25 ins Netz gehängt hat, kann er Mails empfangen – und wird das ziemlich sicher auch bald tun. Auch dann, wenn niemand die Domain kennt. Spammer haben nämlich nichts anderes zu tun, als das Internet zu scannen und Server mit offenem Port 25 mit ihrem Müll zu belästigen. Dass das so gut funktioniert, liegt am Aufbau des verwendeten Protokolls SMTP.

SMTP

SMTP, das Simple Mail Transfer Protocol, ist ein wahres Urgestein des Internets, erstmals spezifiziert im RFC 821 aus dem August 1982 (später mehrfach durch neuere Versionen ersetzt). Zwei Mailserver untereinander verwenden SMTP, um dem jeweils anderen eine Mail zuzustellen. Der Ablauf einer solchen SMTP-Sitzung ist sehr einfach und man kann ihn leicht auf der Kommandozeile mit dem Befehl `telnet` (Linux, macOS) nachspielen. Unter Windows muss man diese Grundfunktion als Windows-Feature aktivieren (siehe ct.de/wc2j). Öffnen Sie die Kommandozeile Ihres Vertrauens. Zunächst müssen Sie sich einen Mailserver zum Testen suchen, der Sie nicht sofort als Spammer identifiziert und abweist – Google lässt das Experiment zum Beispiel fast bis zum Ende zu. Verwenden Sie einen der vielen

MX-Einträge von Google und starten Sie eine SMTP-Sitzung wie folgt:

```
telnet gmail-smtp-in.l.google.com 25
```

Um Tippfehler zu vermeiden: Beim Buchstaben vor `google.com` handelt es sich um den Konsonanten L. Den folgenden Dialog mit dem Server wickeln Sie ab, indem Sie die Zeilen abtippen und mit Enter bestätigen. Zunächst müssen Sie sich als Absender vorstellen:

```
EHLO example.org
```

`EHLO` ist kein Tippfehler, sondern so im Standard festgeschrieben. Nach der Begrüßung können Sie das Abliefern einer Mail ankündigen:

```
MAIL FROM:<test@example.org>
```

Im nächsten Schritt übergeben Sie den Empfänger (Recipient) der Mail:

```
RCPT TO:<example@gmail.com>
```

Die spitzen Klammern müssen sein und ein Leerzeichen nach dem Doppelpunkt würde einen Syntaxfehler verursachen.

Jetzt leiten Sie die Übertragung des Inhalts mit folgender Zeile ein:

```
DATA
```

Der Betreff ist Teil des Inhalts und wird zuerst in einer Zeile übertragen:

```
Subject: Eine Test-Mail
```

Jetzt muss zwingend eine leere Zeile (Enter-Taste) gesendet werden. Erst dann darf der Inhalt der Mail folgen:

```
Moin, das ist ein Test.
```

Beendet wird die Übertragung mit einer weiteren Leerzeile, danach einer Zeile, die nur einen Punkt enthält. Dadurch weiß der Server, dass die Mail vollständig übermittelt wurde:

```
.
```

Damit kennen Sie den kompletten Dialog, den zwei Mailserver untereinander führen. Der empfangende Server würde die Mail jetzt weiterverarbeiten und dem richtigen Nutzer ins Postfach legen. Bei Google wird Ihnen das aber voraussichtlich nicht gelingen, denn das Unternehmen hat einige Anstrengungen unternommen, um unerwünschte Mails zu entfernen. Die Reaktion von Google beim Versuch von einer heimischen Adresse sieht zum Beispiel so aus:

```
Our system has detected that this
message is suspicious due to the very
low reputation of the sending IP
address. To protect our users from
spam, mail sent from your IP addresss
has been temporarily rate limited.
Please visit
https://support.google.com/mail/
answer/188131 for more information.
```

Googles Server haben also ein Problem mit Ihrer IP-Adresse und weisen Sie als Spam-Verdächtigen direkt wieder ab.

Spam und Spoofing

SMTP selbst kennt keinen Schutz vor Spam. Konzipiert war es als Kommunikationsmittel für staatliche Einrichtungen und Forscher. Wer spammte, bekam persönlich Ärger, weil der genervte Kollege im Büro auftauchte. Im öffentlichen Internet sieht das natürlich anders aus und die Schwächen von SMTP wurden schnell offenbar.

Im Beispiel-Dialog haben Sie sich vorgestellt mit `EHLO example.org`, obwohl Ihnen diese Domain nicht gehört. SMTP kennt hier keine verpflichtende Prüfung. Ein häufiger Irrtum ist, dass nur der Server senden darf, der mit dem A-Eintrag für example.org im DNS steht. Das stimmt nicht und wäre auch nicht erwünscht, schließlich müssen Website und Mailserver nicht auf demselben Server laufen oder am selben Anschluss im Internet hängen. Weil SMTP da so großzügig ist, können Sie beim `EHLO` auch vorgeben, der Server bundeskanzlerin.de zu sein und versuchen, anderen Servern Mails von a.merkel@bundeskanzlerin.de zu übermitteln. SMTP hält Sie davon nicht ab.

Solche Fälschungen werden als Spoofing bezeichnet. Als empfangender Server muss man zwingend Filtersysteme installieren, um sich vor solchen Versuchen weitestgehend zu schützen. Die wichtigste Information für Admins und Nutzer: Alle Spam- und Spoofing-Schutzmaßnahmen sind nur Näherungslösungen und nie perfekt! Sie basieren

auf Wahrscheinlichkeitsabschätzungen und ein guter Spam-Schutz aus mehreren Filterstufen. Zur Arbeit des Admins gehört es, diese Filter immer mal wieder anzupassen und so mit Gewichtungen zu versehen, dass möglichst viel Spam und möglichst wenig Wichtiges vernichtet wird. Eine Erkennungsrate von 100 Prozent und eine Fehlerrate von 0 Prozent sind Utopie.

Private Anschlüsse

Das Filtern beginnt bestenfalls schon auf SMTP-Ebene, also bevor der Absender mit `DATA` seine Mail ankündigt. Wenn Ihr Server vorher schon die Verbindung zu einem Spammer kappt, werden RAM, Festplatten und Leitung gar nicht erst mit Müll belastet. Zu den Anti-Spam-Maßnahmen, die die meisten Serverbetreiber schon auf SMTP-Ebene einsetzen, gehört eine grobe Prüfung der IP-Adresse. Eine Menge Spam kann ein Server nämlich schon vorm Einliefern entsorgen, wenn er schlicht keine Mails annimmt, die von einem Kundenanschluss mit dynamischer IP abgeschickt wurden. Solcher Spam entsteht vor allem durch Schadsoftware auf PCs oder gekaperten Smart-Home-Geräten. Übernommene Systeme werden von Angreifern aus der Ferne gern zu Spam-Schleudern umprogrammiert.

Fast alle Empfänger prüfen zuallererst den sogenannten Reverse-DNS-Eintrag (PTR-Eintrag). Das ist ein Texteintrag für eine IP-Adresse, den man als Internetkunde nicht selbst ändern kann. Bei einem Kabelanschluss von Vodafone sieht er zum Beispiel so aus: ip123456.dynamic.kabel-deutschland.de. Die Information „dynamic“ verrät, dass das keine typische Server-Adresse mit statischer IP ist, sondern eine, die dynamisch an Kunden vergeben wird. In dem Fall kann ein empfangender Server die Sitzung eigentlich schon beenden. Wenn Sie als Unternehmen eine Leitung mit statischer IP gebucht oder einen Server im Rechenzentrum gemietet haben, können Sie bei Ihrem Provider (bei Mietservern auch direkt online im Kundencenter) den PTR-Eintrag ändern, etwa auf mail1.example.org. Der PTR-Eintrag muss exakt den Namen enthalten, mit dem sich der Server bei einem `EHLO` vorstellt.

Auch hier gibt es einen häufigen Irrtum: Entgegen anderslautender Erzählungen muss der PTR-Eintrag nicht zwangsläufig mit Ihrer Domain korrespondieren, von der Sie verschicken wollen – ein Server kann problemlos von mehreren Domains verschicken. Die Server unseres Verlages versenden zum Beispiel von heise.de, ct.de und ix.de, können aber nur einen PTR-Eintrag pro IP-Adresse haben – als Admin muss man sich für eine Angabe im PTR-Eintrag entscheiden und diesen Namen auch in seinem Server für die `EHLO`-Begrüßung hinterlegen. Zusätzlich zum PTR-Eintrag verwenden viele Admins Blacklists wie die „Policy Block List (PBL)“ von Spamhaus (spamhaus.org). Sie enthält eine regelmäßig gepflegte Sammlung an IP-Adressen, die von den Providern selbst als Endkundenanschlüsse markiert sind. Selten enthält diese Liste auch mal Fehler.

Sollten Sie mit dem Gedanken spielen, einen Mailserver zu betreiben, der Mails versenden kann, brauchen Sie also zwingend eine statische IP-Adresse. Alle Versuche, einen versendenden Mailserver an einem Heimanschluss zu betreiben, sind zum Scheitern verurteilt. Was aber gut funktioniert, ist ein kleiner Mietserver im Rechenzentrum mit statischer IP.

Erfolgreich versenden

Doch wie kann ein gewöhnlicher Nutzer überhaupt eine Mail an einen anderen Mailserver schreiben, wenn dieser doch SMTP-Sitzungen von gewöhnlichen Internetanschlüssen blockiert? Der Mailclient des Absenders, zum Beispiel Outlook oder Thunderbird, nutzt ebenfalls SMTP zum Versand, kommuniziert aber nie direkt mit dem Server des Empfängers. Stattdessen liefert der Client die Mail beim eigenen Mailserver ab und überlässt es diesem, die Mail dem Empfänger zuzustellen. SMTP wird also auf zwei

Optionale Sicherheit mit STARTTLS

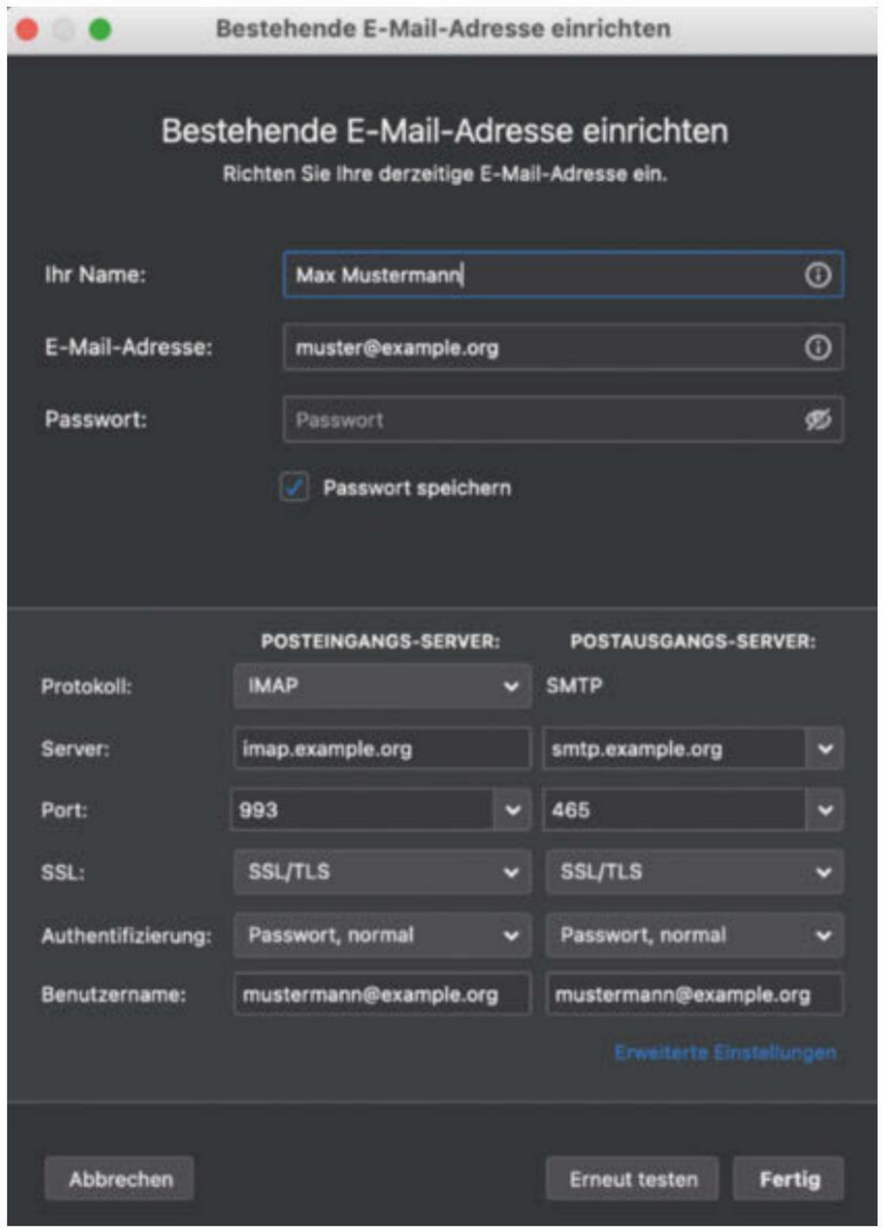
Viele Protokolle sind zu einer Zeit entstanden, als an Verschlüsselung noch nicht zu denken war – nicht nur solche aus dem E-Mail-Umfeld. Um Verschlüsselung nachzurüsten, kam das Verfahren STARTTLS auf. Eine Verbindung beginnt hier zunächst unverschlüsselt. Erst wenn der Client mit dem Befehl `STARTTLS` ein Upgrade der Sitzung anfordert, beginnt ein Schlüsselaustausch. Man spricht hier von expliziter Transportverschlüsselung. STARTTLS wurde auf viele Protokolle nachträglich aufgepflanzt – zum Beispiel auf LDAP, FTP oder eben SMTP und IMAP. Sinnvoll ist es immer dann, wenn auf einem Port verschlüsselte und unverschlüsselte Verbindung stattfinden soll. Als Alternative gibt es viele Protokolle auch in einer Variante, die von vornherein jede Sitzung mit TLS absichert.

Wegen genutzt: Zur Kommunikation von Servern untereinander und zum Einliefern von Mails für den Weiterversand durch den Client.

Damit nicht jeder Ihren Mailserver zum Auftragsversand nutzen kann, muss sich der Client zuerst anmelden – meist mit Benutzername und Kennwort, seltener mit Verfahren wie NTLM. Eine Authentifizierung war in der ursprünglichen Idee von SMTP noch nicht vorgesehen und wurde erst später eingebaut. Ein sauber konfigurierter Mailserver prüft bei solchen authentifizierten Sitzungen auch, ob der angemeldete Nutzer berechtigt ist, von der genannten Absenderadresse zu verschicken. Wer sich mit user@example.org eingeloggt hat, soll ja nicht zwangsläufig von admin@example.org senden dürfen.

Wie bei allen Protokollen aus den Anfangstagen des Internets ist auch bei SMTP Verschlüsselung nicht verpflichtend – auch nicht bei der Authentifizierung mit Benutzername und Kennwort. Unternimmt man nichts weiter, würde der Client diese sensiblen Zugangsdaten immer unverschlüsselt über das Netz schicken, was spätestens bei öffentlichen WLANs ein massives Problem darstellt. Für die Verschlüsselung gibt es gleich zwei Optionen: Weit verbreitet ist STARTTLS, ein Verfahren, das sowohl verschlüsselte als auch unverschlüsselte Kommunikation auf einem Port gestattet (siehe Seite 129). Die Entscheidung für STARTTLS basiert auf der Annahme, dass es noch gute Gründe für eine unverschlüsselte Verbindung beim Anliefern von Mails gibt. Mit dem RFC 8314 der IETF, der 2018 erschien, wurde das Umdenken in der Szene zu Papier gebracht: Er trägt den vielversprechenden Titel „Cleartext Considered Obsolete“ – plausible Gründe fürs unverschlüsselte Anliefern gibt es nämlich nicht. Empfohlen wird stattdessen, implizites TLS zu nutzen, jede Sitzung also sofort zu verschlüsseln. Per se unsicherer ist STARTTLS nicht, allerdings völlig unnötig, wenn es sowieso bei jeder Verbindung aufgerufen wird. Dann reicht es, auf einem separaten Port TLS sofort aufzubauen.

Rund um diese beiden Verschlüsselungsmöglichkeiten gibt es einige Verwirrung und gleich zwei Ports. Im IETF-Standard vorgesehen für die Anlieferungen von Mails war zunächst Port 587. Entgegen der Gepflogenheiten hatte die Internet Assigned Numbers Authority (IANA), die für das Reservieren von Standardports zuständig ist, zusätzlich Port 465 für SMTPS (also SMTP mit Verschlüsselung) freigehalten, das wieder zurückgenommen und den Port später ausdrücklich für das authentifizierte Einreichen von Mails über SMTP eingeplant. Welchen Port der Betreiber bereitstellt, ist seine Entscheidung – er muss sie nur seinen Nutzern mitteilen. Aktuell im RFC empfohlen wird, in den nächsten Jahren sowohl STARTTLS auf Port 587 als auch implizites TLS auf 465 anzubieten. Langfristig soll STARTTLS dann weichen. Wichtig für alle Serverbetreiber: Damit sich kein Nutzer versehentlich unverschlüsselt anmeldet, sollten Sie Ihre Serversoftware so einrichten, dass auf Port 587 STARTTLS erzwungen wird. Endanwender, die sich unsicher sind, ob sie ihren Mailclient richtig konfiguriert haben, sollten in den Kontoeinstellungen ihrer Software prüfen, ob SSL/TLS oder STARTTLS angehakt ist. Bei den großen Anbietern wird eine verschlüsselte Übertragung mittlerweile erzwungen. Bei kleinen Hostern ist eine unverschlüsselte Übertragung leider häufig noch möglich, auch wenn das nicht mehr zeitgemäß ist.



Beim Einrichten eines Mailprogramms wie Thunderbird sollte man sicherstellen, dass die Anmeldung beim SMTP- und IMAP-Server verschlüsselt geschieht.

Wie immer bei SSL/TLS muss man sich als Serverbetreiber auch bei einem Mailserver um ein Zertifikat kümmern, das die Clients akzeptieren. Das Zertifikat muss also mit einem vertrauenswürdigen Stammzertifikat von einer Zertifizierungsstelle signiert sein. Hier gelten dieselben Regeln wie auch bei einem Webserver, der über HTTPS veröffentlicht wird. Kostenlos kommen Admins bei Let's Encrypt an ein Zertifikat, das alle gängigen Clients (wie zum Beispiel Browser) akzeptieren.

STARTTLS findet übrigens nicht nur zwischen Client und Server beim Anliefern Verwendung. Immer mehr Serverbetreiber haben in Sachen Security dazugelernt und Transportverschlüsselung mit STARTTLS auch für die Server-Server- Verbindung auf Port 25 aktiviert. Damit ist eine Mail zwar noch nicht Ende-zu-Ende-verschlüsselt, aber immerhin auf dem größten Teil der Verbindung. Einige Empfänger, zum Beispiel die niedersächsischen Finanzämter, weigern sich sogar schon, Mails ohne STARTTLS anzunehmen.

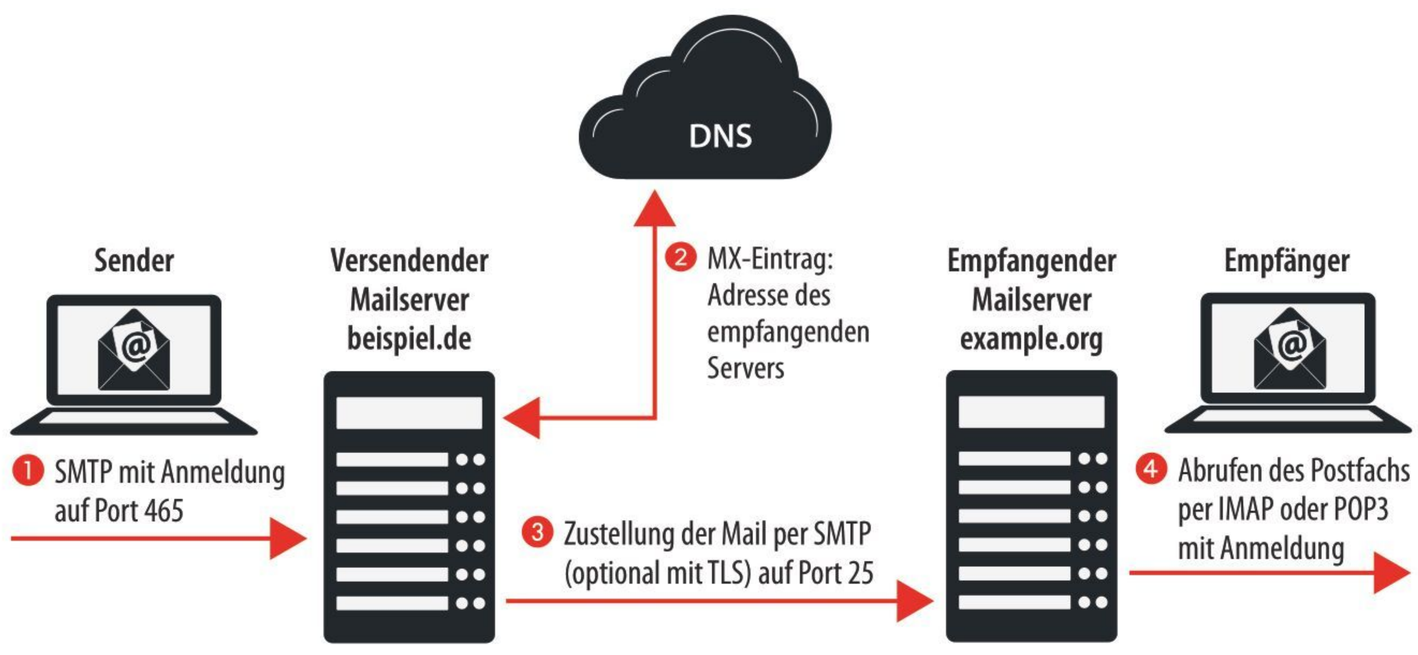
Open-Relays und Reputation

Als SMTP erfunden wurde, kam es schon mal vor, dass der Empfänger zwischendurch nicht erreichbar war. Daher war es durchaus erwünscht, dass jeder Server im Netzwerk als Open Relay arbeitet – wer gerade aktiv und erreichbar war, nahm jede Mail an und versuchte dann, diese ans eigentliche Ziel weiterzuleiten. So konnten befreundete Server-Betreiber sich zum Beispiel gegenseitig als MX-Eintrag setzen und so die Wahrscheinlichkeit erhöhen, dass Mails ankommen. Für ein Netzwerk unter Freunden war das eine brauchbare Idee, im heutigen Internet mit Berufs-Spammern sind Open-Relays eins der größten Probleme. Wer selbst einen Mailserver betreibt, muss unbedingt sicherstellen, dass dieser nicht versehentlich als Open-Relay arbeitet und unauthentifiziert auf Port 25 Mails für andere annimmt und sie weiterleitet.

Wenn Sie einen Server betreiben und einen Blick in dessen Logs werfen, werden Sie schnell sehen, dass Spammer Sie regelmäßig auf Open-Relay-Konfiguration prüfen. Sind Sie als solches identifiziert, beginnt der Sturm: Die Spammer laden ihre Müll-Nachrichten auf Ihrem Server ab und lassen ihn die Drecksarbeit machen, den Spam an die Server der Empfänger zuzustellen. Nach wenigen Stunden als Open Relay bricht Ihr Server entweder unter der Last zusammen oder landet schlagartig auf

Versand einer E-Mail per SMTP

Ein Nutzer, der eine Mail versenden will, liefert diese per SMTP bei seinem Mail-Server ab und meldet sich dafür mit seinen Zugangsdaten an. Dieser Mailserver schlägt die Adresse des empfangenden Servers im DNS nach und stellt die Mail ebenfalls per SMTP zu.



allen relevanten Spam-Blacklisten. Bei Mietservern schreitet außerdem der Rechenzentrumsbetreiber zügig ein.

Um all das zu verhindern, gehört es zur obersten Pflicht eines Mailadmins, sofort nach Einrichten des Servers einen Open-Relay-Test zu starten – zum Beispiel mit einem geeigneten Online-Werkzeug (siehe ct.de/wc2j). Ein richtig konfigurierter Server nimmt ohne Anmeldung nur Mails an, die für ihn selbst bestimmt sind.

Bei der Jagd nach Spam geht es vor allem um die Reputation des Absenders, also eine Abschätzung, wie vertrauenswürdig eine versendende IP-Adresse ist. Um Admins zu unterstützen, gibt es neben der Organisation Spamhaus noch weitere Betreiber von Listen, die Auskunft über die Reputation eines Anschlusses geben. Wer negativ auffällt, etwa als Open-Relay, landet mit seiner IP – oder noch schlimmer, der Domain – auf solchen Listen. Von daher ist es wichtig, den Server von der ersten Minute an sauber zu konfigurieren. Eine Adresse von allen Blacklists entfernt zu bekommen, ist nicht ganz einfach. Teilweise muss man nur Formulare ausfüllen, anderswo sogar Geld bezahlen. Wenn Sie einen neuen Anschluss mit statischer IP oder einen Mietserver mit IP übernehmen, sollten Sie vor der Inbetriebnahme mit Online-Werkzeugen (siehe ct.de/wc2j) prüfen, ob die IP-Adresse schon verbrannt ist, weil der Vorbesitzer als Spam-Schleuder aktiv war. Wenn die Adresse auf mehr als zwei Listen steht, kontaktieren Sie besser den Anbieter und lassen sich eine andere IP geben. Wenn gleich ein ganzes IP-Segment auf einer Blackliste gelandet ist, haben Sie ein ernsthaftes Problem. Wenn gar nichts hilft, kann man nur den Provider wechseln, oder diesen bitten, bei den Betreibern der Listen vorstellig zu werden.

Ein weiteres wichtiges Mittel gegen unangenehme Einträge auf Blacklists ist ein Limit für Mails pro Benutzer in einem Zeitraum. Besonders, wenn man viele Benutzer auf dem System hat, sollte man zu der Maßnahme greifen. Stellen Sie sich vor, ein Nutzer hat ein sehr einfaches Kennwort, das er auch anderswo nutzt. Wird es irgendwo geleakt, können sich Spammer damit an Ihrem Server anmelden und im Namen Ihres Nutzers andere vollspammen. Früher oder später landet dadurch Ihr Server auf Blacklists und Sie als Admin haben die Arbeit. Fast alle Mailserver haben eine Option, die Mails pro Nutzer und Stunde zu beschränken. Mehr als 50 Mails muss ein gewöhnlicher Mitarbeiter in 60 Minuten meist nicht versenden.

Checkliste für Mailserver-Betreiber

Damit Mails auf dem eigenen Server ankommen und man nicht zur Spam-Schleuder wird, muss man sich von der ersten Minute an die Spielregeln halten:

- MX-Eintrag für eingehende Mails gesetzt
- Statische IP-Adresse für den Versand eingerichtet
- Open-Relay-Test durchgeführt
- PTR-Eintrag über den Provider gesetzt und im Server hinterlegt
- Blacklists nach Einträgen für die IP geprüft
- Mails pro Benutzer pro Stunde limitiert (zum Beispiel auf 50)

Postfachzugriff mit POP3 und IMAP

Mit POP3 und IMAP haben E-Mail-Nutzer am meisten Kontakt. Schließlich wollen sie ihre Mails auch lesen. Beide stammen im Kern aus den 80ern und lösen das Problem recht unterschiedlich. POP3 hat den überschaubareren Funktionsumfang:

Ein Client meldet sich über POP3 mit den Befehlen `USER` und `PASS` an, kann danach mit `STAT` Anzahl der Mails und Größe des Posteingangs prüfen und mit `LIST` eine Übersicht abrufen. Eine einzelne Mail lädt er mit `RETR` herunter und löscht sie mit `DEL` auf dem Server. Alles Weitere passiert offline beim Nutzer. Wenn dieser zum Beispiel Unterordner anlegt und dort seine Mails sortiert, erfährt der Server davon nichts. Außerdem bekommt der Server mit POP3 nicht mit, ob man eine Mail bereits gelesen hat. Spätestens wenn man mehrere Clients (etwa auf dem PC und dem Mobiltelefon) nutzt, macht POP3 keinen Spaß mehr. Heute ist POP3 ein Protokoll für nostalgische Liebhaber und spezielle Anwendungen. Wenn beispielsweise ein Skript alle Nachrichten automatisch aus einem alten Postfach abrufen und in ein neues Postfach legen soll, reicht POP3 völlig aus.

Weil POP3 nicht sonderlich flexibel ist, hat sich IMAP zum Standardprotokoll entwickelt. Um dessen Funktionsumfang mit allen Optionen zu beschreiben, wären einige Seiten Text nötig. IMAP ist für alles verantwortlich, was Sie als Nutzer eines modernen Mailprogramms kennen und schätzen: Der IMAP-Server verwaltet Postfächer und ihre Verschachtelung

und alle Clients bilden das synchron ab. Per IMAP verschiebt man Mails in Ordner, markiert sie als gelesen, versieht sie mit Flags oder wirft sie in den Papierkorb. Auf allen Geräten, die per IMAP mit dem Server sprechen, erscheinen die Änderungen sehr zügig. Wenn Sie aus einem IMAP-Client eine Mail verschicken, passieren übrigens zwei Dinge nacheinander: Erst wird die Mail per SMTP beim Server abgeliefert, dann landet eine Kopie per IMAP in Ihrem Gesendet-Ordner.

Zu den größten Ärgernissen zählt, dass der Standard nicht streng genug ist: Die Namen der Gesendet-, Papierkorb- und Entwurfs-Ordner sind nicht standardisiert. So kann eine Client-Anwendung den Ordner „Sent“ nennen, eine andere „Gesendet“ oder „Gesendete Elemente". Fast immer kann man das aber per Hand angleichen. Wer sich wundert, dass ihm auf einigen Geräten Mails in diesen Ordnern fehlen, sollte tief in den Einstellungen seiner Software nach der passenden Option suchen und die Bezeichnungen ändern.

Auch POP3 und IMAP sind natürlich (weil historisch gewachsen) zunächst unverschlüsselt. POP3 arbeitet im Klartext auf Port 110 – optional mit STARTTLS nach der Begrüßung. Weil Kennwörter übertragen werden, verbietet sich auch hier unverschlüsselte Übertragung. STARTTLS muss man als Admin also verpflichtend machen. Noch konsequenter ist es, Port 110 direkt zu schließen und nur implizites TLS auf Port 995 zu erlauben. In vielen Umgebungen können Sie sich POP3 auch komplett sparen, weil nahezu alle Nutzer nur IMAP nutzen wollen.

Für IMAP gilt bei der Verschlüsselung dasselbe. Unverschlüsseltes IMAP auf Port 143 ist obsolet, verpflichtendes STARTTLS das Mindeste. Weil es ohnehin keine Alternativen zu Verschlüsselung gibt, reicht aber auch IMAP mit verpflichtendem TLS auf Port 993.

Auch bei IMAP und POP3 lohnt als Nutzer ein Blick in die Einstellungen Ihres Clients. Weil einige Anbieter noch unverschlüsselte Verbindungen erlauben, kann es gut sein, dass Sie seit Jahren versehentlich Ihre Zugangsdaten unverschlüsselt über das Netz schicken. Der Haken für SSL/TLS oder STARTTLS ist Pflicht!

Neben IMAP und POP3 gibt es noch proprietäre Protokolle, vor allem aus dem Hause Microsoft. Für den Unternehmens-Mailserver Exchange haben sich die Entwickler über die Jahre verschiedene Protokolle ausgedacht – allen voran MAPI und ActiveSync. SMTP, IMAP und POP3 sprechen Exchange-Server aber dennoch, damit Nutzer auch mit Clients wie Thunderbird arbeiten können.

Feinabstimmung

Hat man als Betreiber eines Mailservers die gröbsten Fehler vermieden und ist nicht direkt als Spammer negativ aufgefallen, muss man mit der Zeit Optimierungen an zwei Stellen vornehmen: Zunächst sollte man sich darum kümmern, dass möglichst viel Spam gefiltert wird. Neben weiteren Filtermaßnahmen, die auf SMTP-Ebene greifen und schon die Übertragung verhindern, gibt es nachgelagert noch Inhaltsfilter, die sich die Mail ansehen und zum Beispiel auf Schlüsselwörter („Casino“, „Viagra“) oder verdächtige verlinkte URLs reagieren. Solchen Spam verschiebt man meist in ein gesondertes Postfach, wo ihn der Nutzer im Notfall noch finden kann – während die Nutzer von Spam, der schon auf SMTP-Ebene gescheitert ist, nie etwas erfahren.

Außerdem müssen Sie sich stetig darum kümmern, dass Ihre Mails bei anderen zuverlässig ankommen. Wie Sie selbst Filter einrichten und mit vertrauensfördernden Maßnahmen wie SPF und DKIM erreichen, dass Sie von anderen nicht gefiltert werden, erfahren Sie in einer der nächsten Ausgaben.

Angst vor dem Betrieb eines eigenen Mailservers müssen Sie nicht haben, wenn Sie bereits Administrationserfahrungen mit anderen Servern haben – auch wenn diese Einführung zunächst vielleicht abschreckend wirken mag. An Spam-Abwehr und der Zustellung an andere Server muss man immer mal wieder kleine Optimierungen vornehmen (weil sich auch Spam immer wieder anpasst). Ein Vollzeitjob ist das aber nicht und ein eigener Mailserver kann auch für kleine Unternehmen oder Organisationen ein lohnenswertes Projekt sein: Sie haben die volle Kontrolle über die Daten und sind bei Zustellproblemen nicht auf die schnelle Reaktion eines fremden Unternehmens angewiesen. (jam) c't

IETF-Standards

ct.de/wc2j

Protokolle und Ports für E-Mail

Protokoll	Port	Einsatz	Verschlüsselung
SMTP	25	Kommunikation von Servern untereinander	optional über STARTTLS
SMTP	465	Einlieferung von Mails durch den Client	TLS verpflichtend
SMTP	587	Einlieferung von Mails durch den Client	optional über STARTTLS
IMAP	143	Synchronisieren von Postfächern	optional über STARTTLS
IMAP	993	Synchronisieren von Postfächern	TLS verpflichtend
POP3	110	Reines Abrufen von Postfächern	optional über STARTTLS
POP3	995	Reines Abrufen von Postfächern	TLS verpflichtend



Bild: Albert Hulm

Mailserver einrichten mit Mailcow

Mit dem Container-Komplettpaket Mailcow kommen Sie ohne fummelige Einrichtung schnell zu einem eigenen Mailserver. Er empfängt und versendet Mails, verwaltet Kalender, filtert Spam, beschafft automatisch Zertifikate und bietet Komfort, den man sonst nur bei Microsoft Exchange bekommt.

Von **Jan Mahn**

Der eigene Mailserver für die Familie, den Sportverein oder das Unternehmen ist eine Unabhängigkeitserklärung: Kein Mailprovider oder Webhoster ist dann mehr dafür verantwortlich, dass Mails im Postfach landen. Kein Spamfilter, den man nicht selbst kontrollieren kann, fischt versehentlich Ihre erwünschte Post ab. Als Unternehmen ist das auch aus Datenschutzsicht attraktiv, weil Kunden-Mails nicht auf Servern Dritter liegen. Im Gegenzug hat man als Admin aber auch selbst die Verantwortung, dass der Server läuft und auf einem sicheren Stand ist.

Wenn Sie den Betrieb eines eigenen Mailservers planen, sollten Sie sich vorab mit der Funktionsweise des Systems E-Mail und den eingesetzten Protokollen auseinandersetzen. Eine ausführliche Einführung in SMTP, IMAP und POP3, sowie in die Tücken von Spam und Spoofing, finden Sie ab Seite 126.

Diese Lektüre lohnt sich später bei der Einrichtung in jedem Fall.

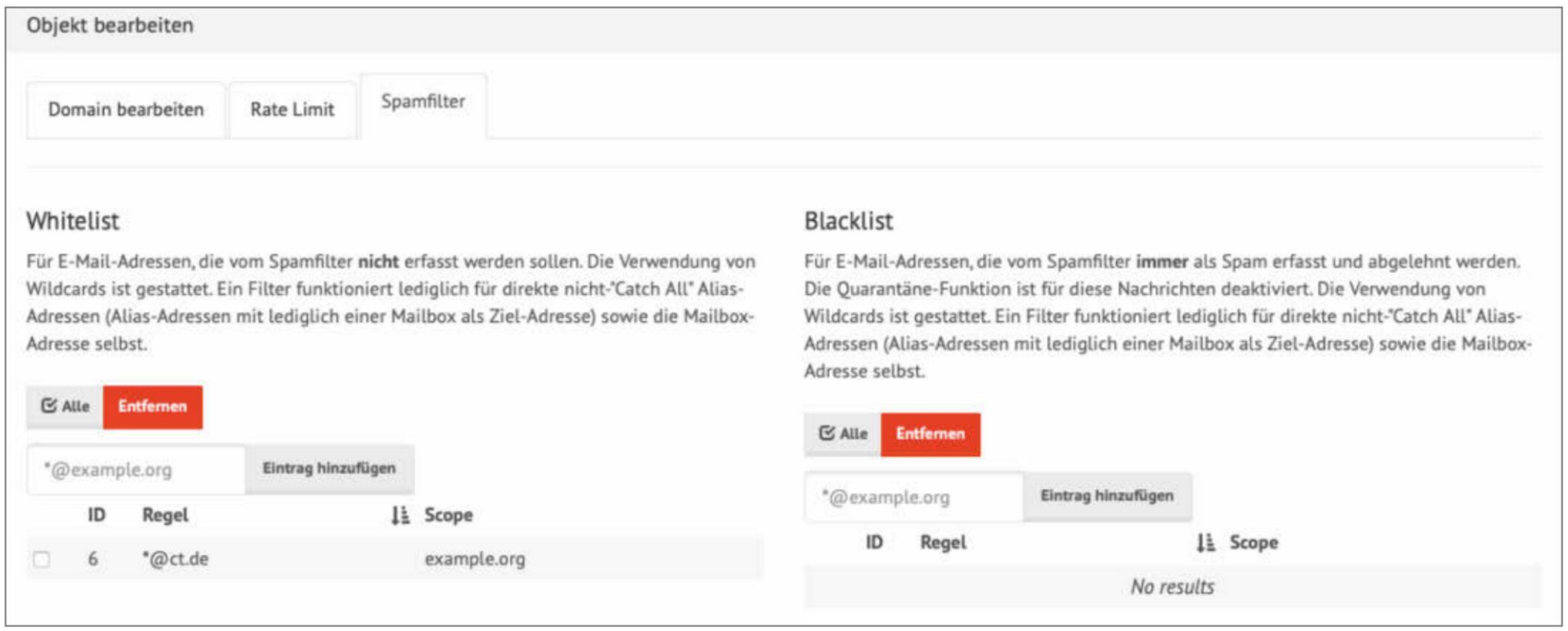
Lizenzkosten müssen Sie für einen eigenen Mailserver nicht einplanen. Große und etablierte Communities haben in den letzten Jahrzehnten robuste und erprobte Komponenten entwickelt: Namen wie Postfix, Dovecot oder Rspamd sind Ihnen womöglich schon begegnet, wenn Sie sich nach Open-Source-Mailservern umgesehen haben. Sie sind für unterschiedliche Rollen eines Mailservers verantwortlich und keines der Werkzeuge ist ein kompletter Mailserver. Eine Komponente nimmt Mails an und verschickt sie (auch als Message Transfer Agent, MTA bezeichnet), eine andere stellt dem Nutzer das Postfach per IMAP bereit (auch als Mail Delivery Agent, MDA bezeichnet), eine weitere entfernt Spam. Die Aufgabe des Admins bei der Einrichtung ist es also, geeignete Bausteine auszuwählen und über Konfigurationen miteinander zu verdrahten. Am Ende soll sich ein Benutzer mit einem Kennwort ja sowohl an der Webmail-Oberfläche, als auch an der SMTP- und IMAP-Komponente anmelden können.

Genau diese Arbeit, die verschiedenen Programme zu verbinden, nimmt Ihnen das Open-Source-Projekt Mailcow ab. Ergebnis der Arbeit ist eine Docker-Zusammenstellung aus populären E-Mail-Diensten, die aufeinander abgestimmt und vorkonfiguriert sind. Eine Weboberfläche zur Verwaltung rundet das Paket ab. Docker-Fachwissen ist nicht erforderlich. Wenn Sie eine Domain und einen Linux-Server zur Verfügung haben, der aus dem Internet mit einer statischen IP-Adresse erreichbar ist, kommen Sie mit Mailcow schnell zu einem eigenen Mailserver. Unter der Adresse servercow.de bekommen Unternehmenskunden darüber hinaus auch kommerziellen Support für die Open-Source-Software.

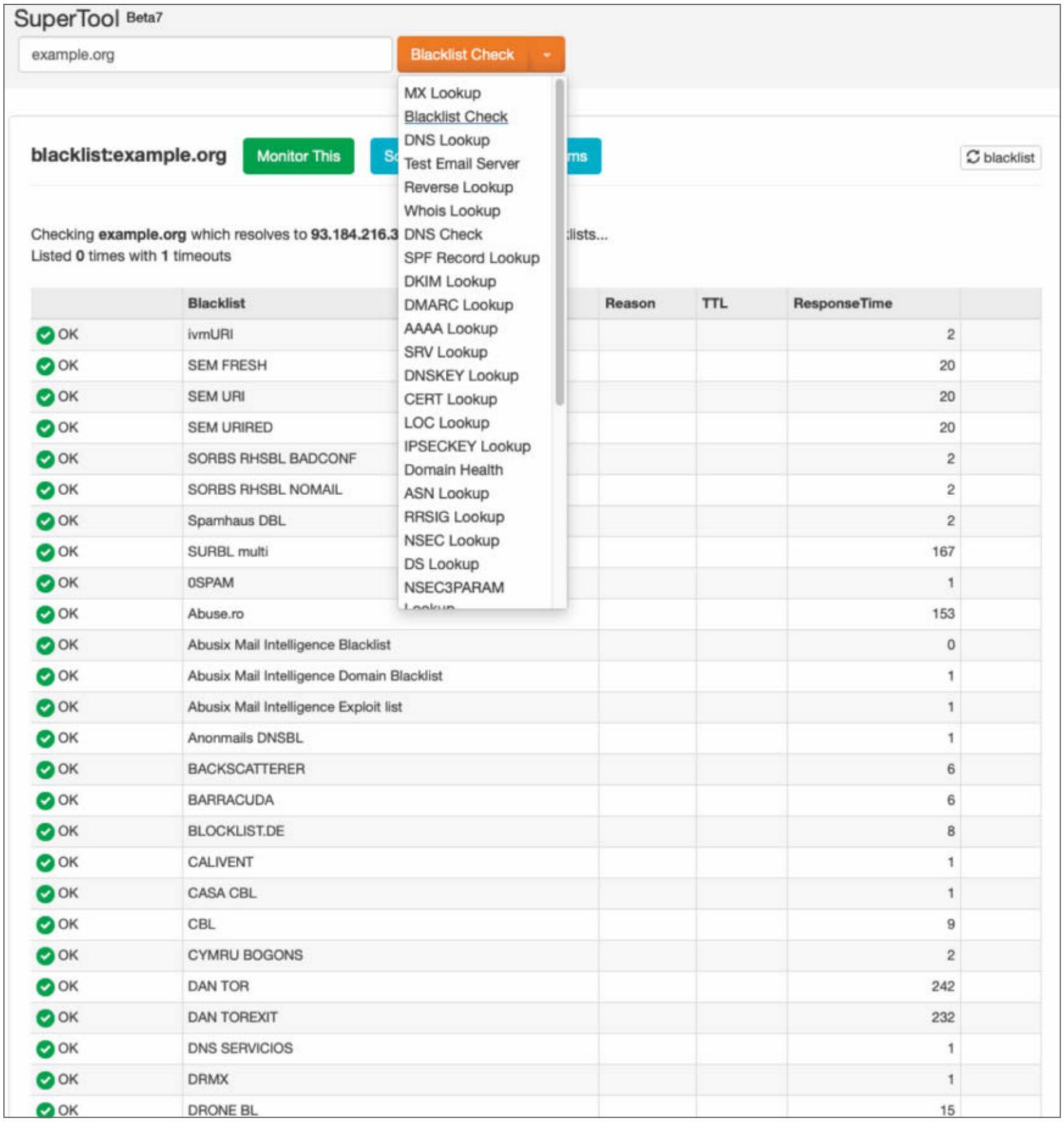
Grüne Wiese

Getestet haben wir die Zusammenstellung mit einem Ubuntu-Server (20.04) mit 4 GByte RAM und 40 GByte SSD, gemietet zum Stundentarif bei einem deutschen Cloud-Anbieter. Auf dem Server brauchen Sie Root-Rechte, ein Webhosting-Paket reicht also nicht aus. Etwa 5 Euro müssen Sie im Monat dafür einplanen, können den Server aber auch noch für andere Aufgaben einsetzen, etwa für Websites mit überschaubarem Nutzeraufkommen. Je nach erwarteter Nutzerzahl sollten Sie ausreichend Festplattenplatz für die Postfächer einkalkulieren. Mailcow unterstützt aber auch Speicherplatz-Limits, sodass Sie Ihre Nutzer irgendwann zum Aufräumen zwingen können.

Sollten Sie Ihren Mailserver in einem Unternehmen im eigenen Haus betreiben wollen und von



Sollen Mails von wichtigen Absendern nicht versehentlich im Filter landen, kann man sie zur Whitelist hinzufügen.



Der Online-Werkzeugkasten von mxtoolbox.com erleichtert dem Mail-Admin die Arbeit, stellt DNS-Einträge dar und findet Konfigurationsfehler.

diesem Mails verschicken, brauchen Sie zwingend eine statische IP-Adresse – von dynamischen IPs an Privatkundenanschlüssen können Sie keine Mails versenden. Viele Business-Internettarife haben gleich mehrere statische IP-Adressen. In der Firewall Ihres Unternehmens müssen Sie die für Mails relevanten Ports auf den Server durchlassen: 25, 465 und 587 für SMTP, 143 und 993 für IMAP sowie 110 und 995 für POP3 (falls gewünscht), außerdem 80 und 443 für die Weboberfläche. Die gleichen Freigaben brauchen Sie in der Firewall des Servers.

Zur Veranschaulichung verwendet dieser Artikel die Domain example.org – ersetzen Sie diese stets durch Ihre eigene. Die Einrichtungsarbeit beginnt mit passenden Einträgen im DNS – wo Sie die setzen,

hängt davon ab, wer Ihre Domain verwaltet. Die DNS-Einstellungen sind bei Webhostern leider oft tief im Kundencenter vergraben. Ein A-Record mit einem Hostnamen für den Server ist die erste Voraussetzung, etwa `mail.example.org`. Dieser Eintrag muss auf die externe IP-Adresse Ihres Servers zeigen. Wenn der Server auch per IPv6 erreichbar sein soll, legen Sie außerdem den gleichnamigen AAAA-Record an.

Damit fremde Mailserver wissen, wo sie Mails für die Domain example.org zustellen sollen, brauchen Sie einen Eintrag vom Typ MX. Als Priorität nehmen Sie zum Beispiel 10 – solange Sie nur einen empfangenden Server haben, ist das nicht entscheidend. Ein solcher Eintrag enthält keine IP-Adresse, sondern den Hostnamen, in diesem Beispiel `mail.example.org`. Der fremde Mailserver fragt das DNS also gleich doppelt ab: Erst nach dem MX-Eintrag, dann nach der IP für den Namen, der dort hinterlegt ist. Weil sich der Name fast nie ändert, kann der MX-Eintrag eine sehr lange Gültigkeit von 24 Stunden bekommen. Wenn Sie den Mailempfang vorerst nur testen wollen, können Sie auch zunächst auch nur einen MX-Eintrag für eine Subdomain anlegen – etwa `test.example.org`.

Zum Versenden von Mails brauchen Sie außerdem einen PTR-Eintrag, also einen Eintrag für Reverse-DNS. Diesen Eintrag setzen Sie nicht beim DNS-Anbieter für Ihre Domain, sondern beim Inhaber der IP-Adresse. Bei gemieteten Servern finden Sie die Einstellung im Kundenportal Ihres Hosters, beim Hosting im eigenen Haus beim Provider Ihrer Internetleitung. Den PTR-Eintrag nutzten Empfänger Ihrer Mails, um eine erste Abschätzung Ihrer Vertrauenswürdigkeit zu treffen. Tragen Sie auch hier `mail.example.org` ein. Mailcow hat kein Problem damit, mehrere Domains auf einem Server zu verwalten. Sie müssen sich aber für einen Namen entscheiden, der im PTR-Eintrag hinterlegt und später im Server als Name eingetragen ist.

Als angehender Mail-Admin sollten Sie sich spätestens jetzt mit einem Online-Werkzeug zur Fehlersuche vertraut machen – zum Beispiel mxtoolbox.com. Das verrät Ihnen viele Details, die man von außen über Ihren Server und die DNS-Konfiguration herausfinden kann. Rechts neben dem Suchfeld gibt es ein Aufklappmenü, über das Sie alle Funktionen des Werkzeugkastens finden. Wenn Ihre MX- und A-Einträge dort gefunden werden und Sie bei einem Klick auf die IP-Adresse den richtigen PTR-Eintrag sehen, sind Sie startklar. Lohnenswert ist auch eine Abfrage der wichtigsten Spam-Blacklisten für Ihre Domain und IP-Adresse. Mxtoolbox prüft automatisch die gängigen Listen ab und zeigt die Ergebnisse an. Steht die IP gleich auf mehreren Listen, sollten Sie sich sogleich eine andere beschaffen.

Software-Einrichtung

Mailcow basiert auf Docker-Containern. Alle Komponenten des Mail-Setups stecken in solchen und werden über ein internes Netz miteinander verdrahtet. Begeben Sie sich auf die Kommandozeile Ihres Servers und installieren Sie zunächst Docker und Docker-Compose. Eine regelmäßig aktualisierte Anleitung für diese Schritte finden Sie über ct.de/whfe. Wenn Sie die Befehle `docker version` und `docker-compose version` auf der Kommandozeile ohne Fehler absetzen können, sind Sie bereit für Mailcow. Stellen Sie vor den weiteren Arbeiten noch sicher, dass auf dem Server kein anderer Mail- oder Webserverdienst auf den oben genannten Ports lauscht.

Die Mailcow-Entwickler empfehlen, das Verzeichnis /opt zum Ablegen der Konfiguration zu verwenden. Navigieren Sie dorthin und laden Sie das Mailcow-Repository per Git herunter. Sofern Git nicht installiert ist, erledigen Sie das vorher über den Paketmanager Ihres Betriebssystems (zum Beispiel `apt install git` bei Debian oder Ubuntu).

Die Mailcow-Installationsroutine ist darauf ausgelegt, dass Sie die Skripte als Root ausführen. Die Ordner für die Konfigurationen bekommen dann passende Berechtigungen:

```
cd /opt

git clone https://github.com/↵
↳mailcow/mailcow-dockerized
cd mailcow-dockerized
```

Bevor Sie die Zusammenstellung starten, müssen Sie eine Konfigurationsdatei erzeugen. Den Assistenten starten Sie mit dem Befehl

```
./generate_config.sh
```

Auf die Frage nach dem FQDN (vollständiger Servername) antworten Sie mit dem Hostnamen Ihres Servers, der auch im PTR-Eintrag steht (etwa `mail.example.org`). Wenn Sie später Änderungen an der Konfiguration vornehmen wollen, bearbeiten Sie die Datei mailcow.conf, die im Verzeichnis angelegt wird. Für eine Standardkonfiguration sind keine Änderungen nötig.

Anschließend können Sie die Docker-Container starten:

```
docker-compose up -d
```

Für alle Docker-Einsteiger: `docker-compose up` stößt eine Container-Zusammenstellung an, der Parameter `-d` verhindert, dass alle Log-Ausgaben direkt auf Ihrer Kommandozeile landen. Download der Abbilder und Einrichtung der Container werden etwa fünf Minuten in Anspruch nehmen.

Willkommen zu Hause

Nach der Einrichtung finden Sie Ihre Mail-Umgebung im Browser über die Adresse https://mail.example.org. Sollten Sie zunächst mit einem Zertifikatsproblem vom Browser begrüßt werden, hilft nur etwas Geduld. Bis das Zertifikat von Let's Encrypt bereit ist, vergehen drei bis vier Minuten. Danach haben Sie mit Zertifikaten nie wieder Arbeit: Mailcow beschafft Ersatz, wenn ein Zertifikat ausläuft. Wenn alles geklappt hat, meldet sich die Seite mit einem Zertifikat, das Ihr Browser problemlos akzeptiert. Der Benutzername beim ersten Start ist `admin`, das passende Kennwort `moohoo`. Zunächst werden Sie gebeten, das Administratorkennwort zu ändern. Mailcow unterstützt außerdem einen zweiten Faktor per TOTP und eine kennwortlose Anmeldung per FIDO2-Gerät.

In diesem Bereich der Admin-Oberfläche können Sie Mailcow noch etwas personalisieren, eigene Logos hochladen und Texte ändern. Wenn Sie darauf Wert legen, navigieren Sie im Menü neben dem Reiter „Zugang“ auf „Konfiguration/UI-Anpassung“.

Um jetzt endlich mailen zu können, müssen Sie in Mailcow eine Domain anlegen. Den Dialog finden Sie im Menü ganz oben über dem Punkt Konfiguration/E-Mail-Setup. Der Wechsel zwischen den Ebenen ist in der Oberfläche nicht immer ganz intuitiv. Klicken Sie auf „Domain hinzufügen“ – die Standardwerte können Sie zunächst belassen. Sehr empfehlenswert ist es, das Feld „Rate limit“ im Formular auszufüllen. Es legt fest, wie viele Mails insgesamt in einem Zeitraum von dieser Domain verschickt werden dürfen. Diese Funktion kann Sie davor bewahren, auf allen Spam-Blacklisten dieser Welt zu landen, wenn ein Nutzer schlampig mit seinem Kennwort umgeht. Zusätzlich können Sie auch ein Rate-Limit pro Benutzer festlegen. Am Ende des Dialogs können Sie die Domain hinzufügen und „SOGo neustarten“. SOGo ist die Webmail-Oberfläche, die Mailcow mitbringt. In der Tabelle mit den Domains sehen Sie jetzt rechts den Button „DNS“. Dieser listet Ihnen jederzeit übersichtlich auf, ob Ihre DNS-Einstellungen schon für Mailversand und -empfang bereit sind.

Über den Reiter „Mailboxen“ verwalten Sie die Benutzer für diese Domain. Legen Sie einen Demo-Nutzer an und vergeben Sie ein Kennwort – der eingebaute Generator ist zwar eine nette Idee, erzeugt aber zu kurze Kennwörter. Öffnen Sie ein anderes Mailkonto und schicken Sie Ihrem Demo-Nutzer schon mal eine Mail.

Nutzerfreundlich

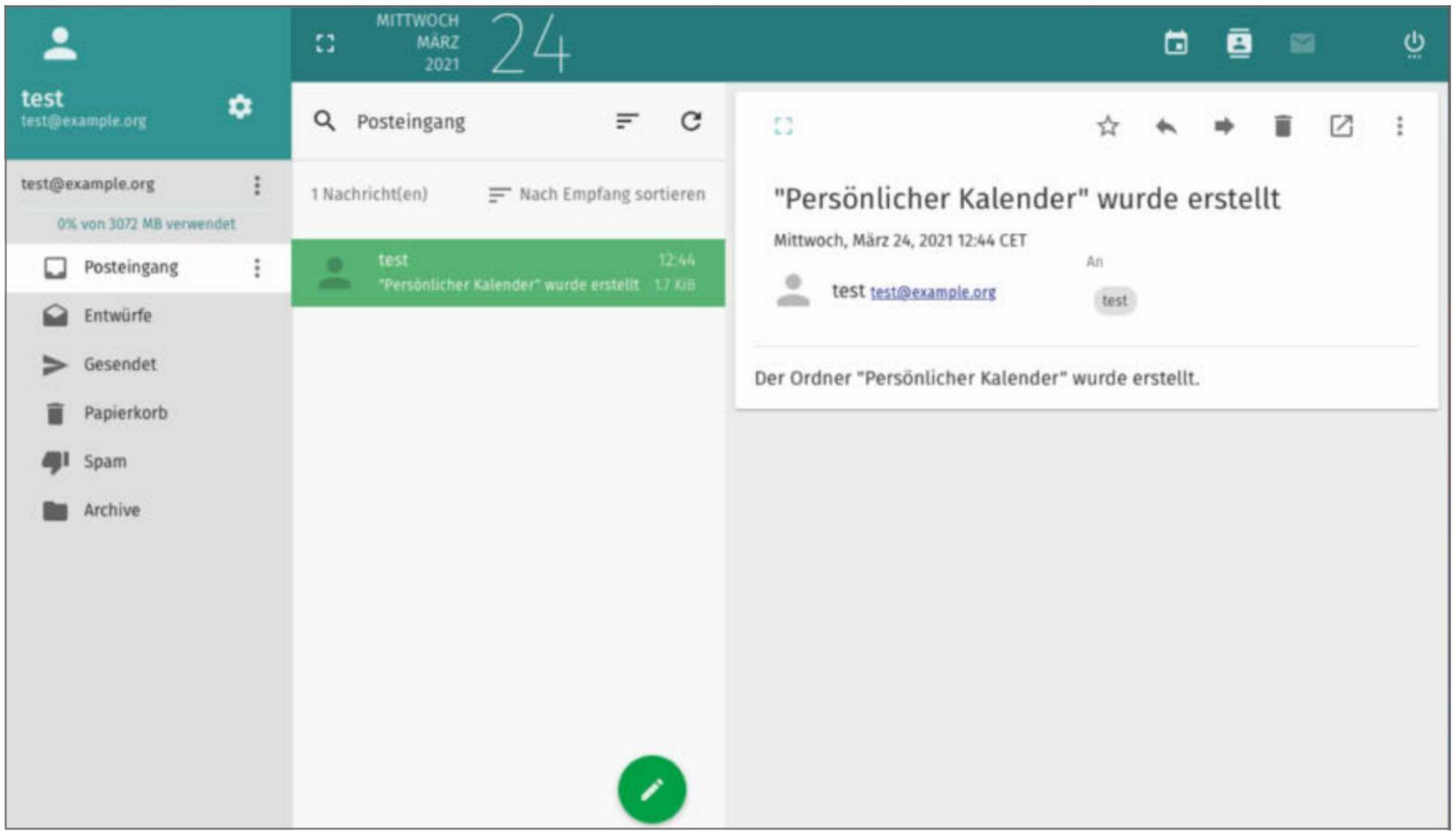
Die Webmail-Oberfläche finden Sie unter der Adresse mail.example.org/sogo. Anmelden müssen Sie sich immer mit der Mailadresse (nutzer@example.org) als Benutzername. Die Oberfläche kann schließlich für mehrere Domains zuständig sein.

SOGo ist optisch recht schlicht, stellt neben Postfächern aber auch Adressbücher und Kalender dar. Wenn man eine Mail an Nutzer derselben Domain über die Oberfläche schreibt, profitiert man von Vorschlägen aus dem globalen Adressbuch – ist das nicht gewünscht, klemmen Sie dieses in der Admin-Oberfläche für die Domain ab.

Auch die Verwaltungsoberfläche mail.example.org, die Sie bisher nur aus Admin-Perspektive gesehen haben, gibt es für Nutzer. Die sehen hier aber andere Menüpunkte, können zum Beispiel ihr Kennwort ändern oder sich temporäre Alias-Adressen einrichten (auch diese Funktion können Sie als Admin unterbinden).

Nutzer, die nicht per Webmailer arbeiten möchten, richten Outlook, Thunderbird, Apple Mail oder einen anderen Mailclient ihres Vertrauens schnell manuell ein, indem sie im Feld für IMAP und SMTP jeweils den Servernamen mail.example.org eintragen. Wichtig auch hier: als Benutzernamen immer die vollständige Mailadresse verwenden.

Aber es ist noch mehr Komfort möglich: Mailcow kann ActiveSync emulieren und sich gegenüber Clients als Microsoft-Exchange-Server ausgeben. Die Nutzer bekommen dann neue Nachrichten per Push, Adressbücher und Kalender können direkt übernommen werden und auch die Einrichtung klappt ohne Eintippen des Servernamens – Autodiscover macht es möglich. Damit diese Komfortfunktionen laufen, brauchen Sie zwei weitere Einträge im DNS für Ihre Domain: Ein `CNAME`-Eintrag für die Subdomain `autodiscover.example.org` muss auf `mail.example.org` zei-



Die Webmail-Oberfläche SOGo ist optisch minimalistisch, zum Lieferumfang gehören aber auch ein Kalender und ein Adressbuch.

gen. Außerdem brauchen Sie einen Eintrag vom Typ `SRV` für die Domain mit dem Inhalt `_autodiscover._tcp.example.org`, der auf mail.example.org auf Port 443 zeigt. Je nach DNS-Anbieter kann es etwas fummelig sein, einen `SRV`-Eintrag zu setzen. `_autodiscover` wird als Dienst bezeichnet, `_tcp` ist das Protokoll. Ob der Eintrag passt, sehen Sie wie die anderen DNS-Einträge in der Admin-Ansicht von Mailcow.

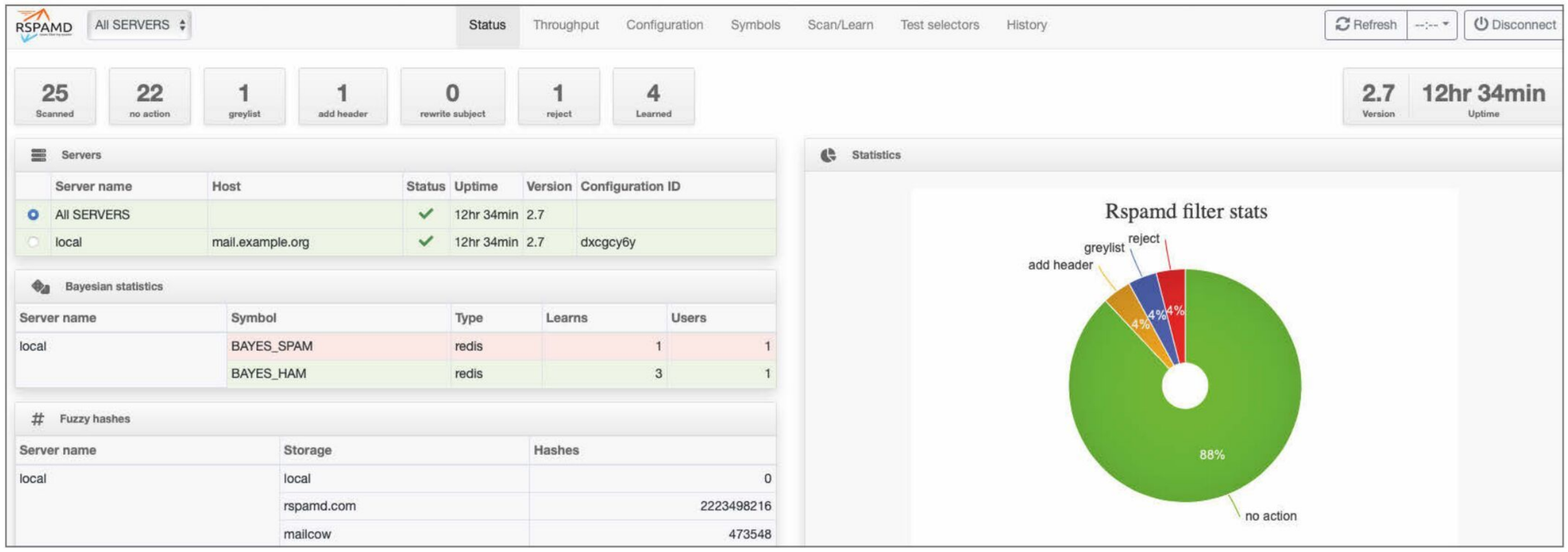
Ist das erledigt, sind etwa Outlook auf dem Desktop oder Apple Mail für macOS oder iOS rasend schnell einsatzbereit: Kontotyp Exchange, Mailadresse und Passwort eingeben, weiter, fertig.

Spam-feindlich

Während Sie es Ihren Nutzern leicht machen wollen, den Server zu nutzen, wollen Sie Spammern das Leben möglichst erschweren. Die werden schon nach kurzer Zeit testen, wie sie Ihnen effektiv auf die Nerven gehen können. Zum Schutz bringt Mailcow die Allzweckwaffe Rspamd in Stellung. Deren Oberfläche erreichen Sie über mail.example.org/rspamd. Um sich anmelden zu können, müssen Sie vorher aber ein Kennwort festlegen. Das ist etwas versteckt: Melden Sie sich mit Ihrem Admin-Account an der Konfigurationsseite an und scrollen Sie nach ganz unten. Dort vergeben Sie das Kennwort und finden auch gleich einen Link zu Rspamd.

Die Oberfläche wirkt auf den ersten Blick etwas überladen, Sie sollten sich aber nicht abschrecken lassen und sich mit der Funktionsweise anfreunden. Lohnenswert ist immer ein Blick auf den Reiter „History“ oben im Menü. Dort sehen Sie alle Mails, die Rspamd angenommen und untersucht hat. Rechts finden Sie die Entscheidung, die die Filter getroffen haben. „no action“ bedeutet, dass es keinen Spam-Verdacht gab, „reject“ heißt, dass die Mail nicht im Postfach gelandet ist. Wenn Sie auf eine Mail klicken, sehen Sie im Detail, welcher Filter angesprungen ist und wie viele „Strafpunkte“ er vergeben hat. Standardmäßig braucht eine Mail 15 Punkte, um sofort vernichtet zu werden.

Am besten lassen Sie den Server zunächst etwas laufen und werfen dann immer mal wieder einen Blick in diese Liste. Spam erkennt man oft schon am Betreff. Wurde eine verdächtige Mail nicht als Spam klassifiziert, schauen Sie sich die Details an und überlegen Sie, welchen Filter Sie höher gewichten



Rspamd ist Ihr treuer Helfer beim Kampf gegen Spammer. Wenn Sie viele Nutzerkonten betreuen, können Sie sich darauf einstellen, dass bis zu 90 Prozent der eingehenden Mails in die Tonne wandern.

könnten. Auf unserem Testserver bekamen wir sehr schnell viel deutschsprachigen Spam mit Büromöbeln und FFP2-Masken. Ihnen war gemein, dass die Verursacher wohl mehrere offene Relays gefunden hatten und über verschiedene Adressen von eigentlich seriösen Domains verschickten. Als Reply-to-Adresse gaben sie stets Adressen bei Freemail-Anbietern an. Für genau diesen Fall hat Rspam den Filter FREEMAIL_REPLYTO eingebaut, diesen aber standardmäßig mit 0 Punkten gewichtet. Der Filter schlug bei allen Mails an, war aber wirkungslos, wie wir der Übersicht entnehmen konnten. In solchen Fällen wechselt man oben auf „Symbols“, sucht den Filter und gibt ihm ein Gewicht. Auf unserem Server war schlagartig Ruhe, als wir den Filter mit 5 Punkten gewichteten. Zusammen mit anderen Filtern reichte es dann für eine Vernichtung.

Universell einsetzbare Tipps, welche Filter Ihnen helfen, kann Ihnen niemand geben. Spam ist immer sehr individuell. Manchmal reicht es aus, dass eine einzelne Adresse eines Nutzers in einer Spammer-Datenbank landet – und plötzlich kommen die Spammer aus allen Ecken und bombardieren Sie. Einige über offene Relays, andere über gehackte Freemail-Accounts, wieder andere über Schadsoftware auf privaten PCs. Was man in jedem Fall sagen kann: Je mehr Nutzer Sie haben, desto eher müssen Sie die Filter immer mal wieder nachjustieren.

Nett zu anderen

Weil auch andere Serverbetreiber mit Spam zu kämpfen haben, müssen Sie nett zu ihnen sein, wenn Sie Ihre Mails bei diesen einliefern wollen. Der PTR-Eintrag war schon der erste Schritt zu etwas Vertrauen. Eine weitere vertrauensbildende Maßnahme ist DKIM [1]. Das Verfahren basiert auf asymmetrischer Kryptografie. Auf Ihrem Server liegt ein privater Schlüssel, im DNS hinterlegen Sie den zugehörigen öffentlichen Schlüssel. Jede ausgehende Mail signiert der Server mit dem privaten Schlüssel, der Empfänger kann die Gültigkeit anhand des DNS-Eintrags prüfen. Mailcow hat es geschafft, die Einrichtung von DKIM sehr einfach zu gestalten, hat den Dialog nur gut versteckt. Öffnen Sie die Admin-Oberfläche und navigieren Sie im Menü ganz oben auf „Konfiguration/Server-Konfiguration“, dann im Reiter auf „Konfiguration/ARC/DKIM-Keys“. In der Mitte der Seite tragen Sie alle Domains ein, für die Sie ein Schlüsselpaar brauchen. Als Länge sollten Sie 2048 Bit wählen. Mit einem Klick auf „Hinzufügen“ erscheint oben der öffentliche Schlüssel richtig formatiert in einer Zeichenkette, die Sie genau so in einen TXT-Eintrag für die Domain im DNS hinterlegen. Um den privaten Schlüssel müssen Sie sich nicht kümmern – Mailcow nutzt ihn ab jetzt und signiert die Mails.

Ebenfalls zu den vertrauensfördernden Techniken zählt SPF, das durchaus umstritten ist. Ein SPF-Ein-

trag im DNS enthält eine Auflistung aller Server, die zum Versand berechtigt sind. Probleme können entstehen, wenn auch externe Dienstleister im Namen Ihrer Domain versenden sollen und diese nicht richtig im SPF hinterlegt sind. Außerdem können Sie sich per SPF die Möglichkeit verbauen, Mails automatisch weiterzuleiten. Wenn Sie SPF nutzen wollen, müssen Sie einen Eintrag zusammenbauen. Dabei hilft zum Beispiel mxtoolbox mit einem SPF-Generator. Ein SPF-Eintrag beginnt mit der Zeichenkette `v=spf1`. Dann folgt eine Auflistung von Servern, die zum Versand berechtigt ist. Der Server, der als MX im DNS steht, gilt automatisch als berechtigt. Wenn auch der Webserver von example.org senden darf, ergänzen Sie einfach ein `a` (das steht für den A-Eintrag Ihrer Domain). Weitere Server hängen Sie mit Leerzeichen getrennt an. Abgerundet wird der SPF-Eintrag von einer Regel für alle anderen Server, die nicht aufgeführt wurden. Mit dem Abschluss `-all` empfiehlt man dem Empfänger, bei anderen Servern die Zustellung als Spoofing einzustufen und abzulehnen. `~all` empfiehlt einen Soft-Fail – der Empfänger kann zum Beispiel ein paar Strafpunkte in seinem Spam-Filter vergeben. Genau beeinflussen können Sie das nicht. `?all` ist eine Einstellung für Feiglinge und empfiehlt eine neutrale Beurteilung. Wer Angst vor Zustellungsproblemen durch SPF hat, kann den Eintrag auch ganz weglassen.

Ein sehr strenger SPF-Eintrag, der nur Versand vom eigenen Mailserver erlaubt, sieht folgendermaßen aus:

```
v=spf1 -all
```

Soll etwa ein externer Dienstleister Newsletter versenden dürfen, verwenden Sie zum Beispiel

```
v=spf1 a:externe-domain.com -all
```

Haben Sie sich für einen Eintrag entschieden, hinterlegen Sie den String als `TXT`-Eintrag im DNS für die Domain. Die Oberfläche von Mailcow oder Mxtoolbox.com verraten Ihnen anschließend, ob SPF- und DKIM-Einträge richtig gesetzt sind. Korrekt eingerichtetes SPF und DKIM bringen bei einigen Empfängern negative Strafpunkte, führen also zu weniger Filterung.

Ständige Pflege

Ihr eigener Mailserver ist damit gut gerüstet für den Alltag. In den Oberflächen für Nutzer und Admins gibt es aber noch einiges zu entdecken – klicken Sie sich am besten durch die Menüs und konsultieren Sie bei Bedarf die Dokumentation. Das führt schneller zum Erfolg als ein Studium der gesamten Dokumentation am Stück.

Auch an Updates haben die Mailcow-Entwickler gedacht. Rufen Sie einfach regelmäßig im Projektverzeichnis das Skript `./update.sh` auf, zum Beispiel per Cronjob. Veraltete Mailserver, die im Internet hängen, werden sonst zu dankbaren Zielen für Angreifer.

Weitere regelmäßige Arbeit ist die Überwachung des eingehenden Spamfilters sowie gelegentliche Zustelltests an große Anbieter. Als Mail-Admin lohnt es sich, eine Handvoll Test-Accounts bei Outlook.com, GMX und Google zu haben. Diese großen Fische bieten für Versender auch eigene Dokumentationen für Zustellprobleme (siehe ct.de/whfe). Wenn sich Ihre Nutzer beklagen, dass Mails an einen Anbieter nicht ankommen, müssen Sie Fehlerprotokolle studieren und sich vielleicht an die Wünsche der Empfänger anpassen. (jam) c't

Literatur

[1] Patrick Koetter, **Wagenburg**, Wie SPF, DKIM und DMARC gegen Phishing und Spoofing helfen, c't 9/2019, S. 174

Mailcow

ct.de/whfe



Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 €.
Nur solange der Vorrat reicht. Preisänderungen vorbehalten.
heise Shop

Go-Einstieg: Für die Cloud entwickeln

Um in eine Programmiersprache wie Go einzusteigen, reicht es nicht, die Syntax in der Theorie zu lernen. Am besten macht man sich schon früh mit Werkzeugen und Compiler vertraut und lernt nicht unter Laborbedingungen. Ein praktischer Einstieg für Programmier-Erfahrene.

Von **Jan Mahn**

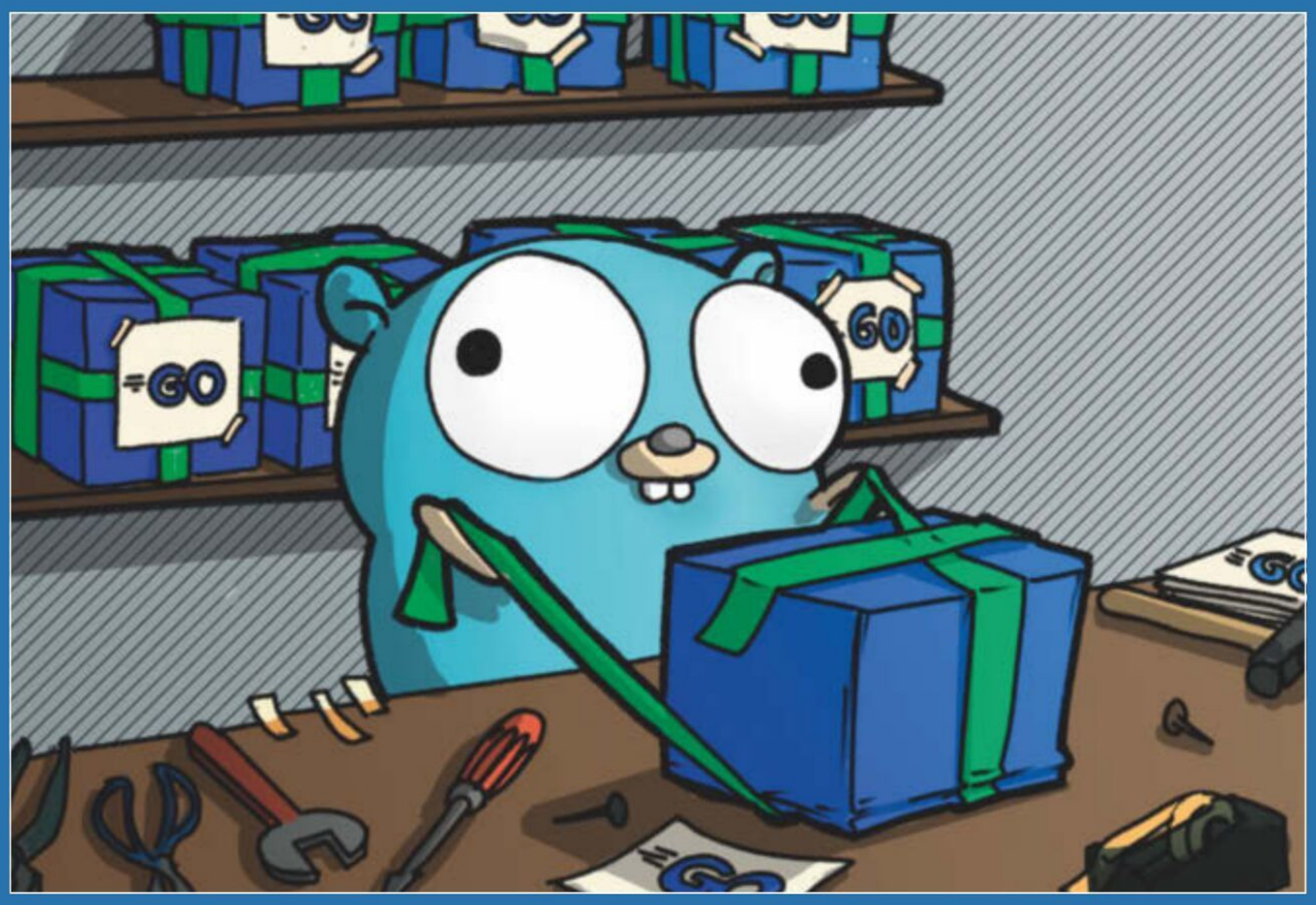


Bild: Thorsten Hübner

Module und Konzepte	142
Kommandozeilen	148
Unit -Tests	154
Server-Dienste	160
Zugangsdaten	166

Die Programmiersprache Go gehört zu den erfolgreichsten Sprachen, die in den vergangenen Jahren erfunden wurden. Go ist mehr als eine akademische Spielerei von Google-Mitarbeitern (siehe Kasten „Go: Steckbrief für Entwickler“). Die größten Anwendungen im Cloud-Umfeld wurden in Go entwickelt.

Der Compiler übersetzt den Code für fast jede erdenkliche Prozessorarchitektur und jedes gängige Betriebssystem und ist dabei so schnell, dass man fast das Gefühl bekommt, eine Skriptsprache vor sich zu haben.

Besonders für Kommandozeilenprogramme und Serverdienste, die in der Cloud laufen sollen, ist Go eine gute Wahl, aber auch manches Bash- oder Perl-Skript kann man gut durch ein Go-Programm ersetzen.

Die Macher von Go haben sich für Einsteiger das Online-Tutorial „A Tour of Go“ ausgedacht, zu finden über ct.de/w8k5. Im Browser kann man Codebeispiele eintippen, die dann auf einem Server ausgeführt werden – das Ergebnis landet wieder im Browser. Diese Einführung ist in jedem Fall empfehlenswert für Go-Neulinge. Am Ende hat man eine grobe Vorstellung davon, worauf man sich beim Go-Programmieren einlässt.

Dieser Artikel richtet sich an alle, die die Sprache in freier Wildbahn ausprobieren wollen. Als Beispielprojekt soll ein kleines Kommandozeilenwerkzeug entstehen, das einen Atom-Newsfeed aus dem Internet herunterladen und die ersten Meldungen auflisten kann. Bevor Sie mit dem Programmieren anfangen können, ist ein wenig Einrichtungsarbeit auf Ihrer Entwicklungsmaschine nötig.

Programmier-Grundkonzepte wie Variablen, `if`, `else` und `for` sollten Ihnen aus einer anderen Programmiersprache bereits vertraut sein – Go ist aus unserer Sicht nicht die optimale Wahl für absolute Programmier-Einsteiger. Entwickeln können Sie in Go auf jedem Desktop-Betriebssystem: Den Compiler gibt es für Windows, Linux und macOS. Alle Downloads finden Sie über ct.de/w8k5.

Unter Windows sollten Sie zur angebotenen MSI-Datei greifen. Sie entpackt Go in den Ordner C:\Go. Starten Sie nach der Installation eine Kommandozeile und führen Sie `go version` aus. Der Befehl sollte mindestens Version 1.14 zurückgeben. Unter macOS laden Sie die PKG-Datei herunter und starten den Einrichtungsassistenten. Er richtet Go im Ordner /usr/local/go/ ein. Öffnen Sie eine Kommandozeile und prüfen Sie mit `go version` die Installation. Sollten Sie die macOS-Paketverwaltung Homebrew nutzen, bekommen Sie die aktuelle Software auch mit `brew install go`.

Linux-Nutzer bekommen Go über die Paketquellen ihrer Distribution. In diesem Artikel kommt das moderne Konzept zum Verwalten von externen Paketen („Go Modules“) zum Einsatz. Das ist erst seit Go 1.14 aus der Beta-Phase entlassen, aktuelle Version ist 1.16. Nach der Installation starten Sie eine neue Sitzung in Ihrer Kommandozeile und prüfen Sie mit `go version`, ob Go richtig installiert ist.

Projektstart

In vielen Anleitungen werden Sie die Anweisung finden, dass sämtliche Programmierprojekte im oben angelegten Go-Pfad, also bei Windows zum Beispiel unter C:\Go liegen müssen. Diese Einschränkung galt früher, also vor Version 1.14, und ist zum Glück überholt. Legen Sie für Ihr erstes Go-Projekt einfach einen Ordner dort an, wo Sie gewöhnlich Ihre Arbeiten ablegen. Wenn Sie Git und einen Git-Hoster wie GitHub nutzen, ist es empfehlenswert, gleich ein Repository anzulegen und den Code von Anfang an zu versionieren.

Entwickeln können Sie Go-Code in einem beliebigen Editor. Wenn Sie noch keinen Lieblingseditor haben oder offen für Neues sind, ist die Open-Source-IDE Visual Studio Code (VSC) eine gute Wahl. Empfehlenswert ist für sie die Erweiterung mit dem schlichten Namen „Go“, die von einem Team bei Microsoft entwickelt wird. Sie bringt Funktionen wie Autovervollständigung mit, rückt Code automatisch richtig ein und kann Code auf Wunsch beim Speichern sofort kompilieren.

Öffnen Sie den leeren Ordner für das erste Projekt in der IDE Ihres Vertrauens und eine Kommandozeilensitzung im gleichen Ordner.

Ein modernes Go-Programm wird vom Compiler als Modul behandelt. Ein solches Modul besteht aus sogenannten Paketen – solche, die Sie selbst schreiben, und solche, die Sie von anderen Entwicklern importieren. Damit aus einem Ordner ein Go-Modul wird und Go seine Paketverwaltungsmagie aktiviert, müssen Sie ein Modul mit `go mod init` initialisieren und ihm einen Namen geben. Den Namen übergeben Sie dem Befehl als zusätzlichen Parameter, zum Beispiel `web-cli`. Wenn Sie Ihr erstes Projekt gleich bei GitHub verwalten, sollten Sie die Adresse des Repos als Namen angeben, also zum Beispiel:

```
go mod init github.com/jamct/web-cli
```

Go: Steckbrief für Entwickler

Die Programmiersprache Go entstand als Privatprojekt von Google-Mitarbeitern, die vor allem vom langsamen C-Kompilieren genervt waren. Nicht nur dank eines schnellen Compilers stieg Go schnell zur beliebtesten Sprache bei Cloud- und Container-Entwicklern auf. Populärstes Go-Projekt ist Kubernetes mit 1,4 Millionen Zeilen Go-Code.

Ein C-Nachbau ist Go aber nicht: Die Syntax erinnert höchstens entfernt an C und seine Verwandten. Stattdessen überrascht sie mit ungewohnten Konzepten, die sich nach kurzer Umgewöhnung als erstaunlich durchdacht entpuppen. So ist die Sprache zum Beispiel objektorientiert, verwendet aber keine Klassen.

Variablen sind typsicher und der Compiler beklagt sich mit aussagekräftigen und verständlichen Fehlermeldungen, wenn man nachlässig mit Typen umgegangen ist. Eine der Stärken der Sprache ist die Umsetzung von Nebenläufigkeit – mit einem vorangestellten `go` wird ein Befehl nebenläufig ausgeführt. Weil die Umsetzung so einfach ist, kommt man als Entwickler gar nicht auf die dumme Idee, etwa mit Netzwerkaktivitäten den Haupt-Thread zu blockieren.

Go-Code wird kompiliert – und im Compiler stecken viele gute Ideen: Cross-Compiling ist seit 2015 eingebaut, ausführbare Binärdateien für Windows, Linux und macOS für gängige und seltenere Prozessorarchitekturen erzeugt man ohne Einrichtungsaufwand von jeder Plattform. Die fertigen Binarys haben keine externen Abhängigkeiten – wer einen Serverdienst oder ein Kommandozeilenprogramm entwickeln muss, wird Go schätzen. Adapter für GUI-Frameworks gibt es auch, für grafische Desktop-Programme ist Go aber nicht die erste Wahl.

Für Programmier-Einsteiger ist Go nicht unbedingt empfehlenswert. Sie sollten die Konzepte lieber in einer Skriptsprache wie Python oder JavaScript lernen und erst dann auf Go umsteigen. Go-Entwickler mögen oft gute Programmierer sein, von systematischen Dokumentationen halten sie meist nichts oder haben schlicht keine Zeit dafür. So passiert es häufig, dass man eine Bibliothek findet, die das Problem lösen könnte. Aber statt eine hilfreiche Dokumentation mit einem Beispiel zu schreiben, verweisen die Macher nur auf die trockene Klassendokumentation. Konkrete Beispiele muss man sich in Foren oder GitHub-Issues zusammensuchen. Für Umsteiger eine entmutigende Arbeit. Dass fast alle Dokus und Blogs über Go kein Syntax-Highlight beherrschen, ist ebenfalls nicht hilfreich.

Die größte Schwachstelle von Go war aber eine architektonische. Die Erfinder haben den Fehler mittlerweile eingesehen: Ursprünglich gab es auf jeder Entwicklermaschine einen Pfad (den GOPATH), in dem alle Projekte zusammen mit dem Go-Compiler und nachgeladenen Paketen lagen. Probleme mit unterschiedlichen Versionen von Abhängigkeiten waren an der Tagesordnung. Seit Version 1.14 (erschienen Ende Februar 2020) ist das neue Konzept von „Go Modules“ aus der Testphase entlassen und Projekte können unabhängig von anderen auf Ihrer Festplatte liegen. Wenn Sie Go lernen, sollten Sie darauf achten, gleich mit „Go Modules“ zu arbeiten.

Für wen?

Go ist eine Sprache für Fortgeschrittene, nicht nur aus dem Cloud- und Containerumfeld. Wenn Sie selbst Dienste entwickeln wollen, die in Containern laufen, können Sie mit Go viel Freude haben. Die Syntax ist zunächst ungewohnt, wenn man C und seine Verwandten gewohnt ist. Schmutzige Tricks beim Programmieren gewöhnt Ihnen der Compiler knallhart ab und zwingt Sie zu gutem Code. Im Gegenzug für etwas Umgewöhnung bekommen Sie schnelle und leicht zu verteilende Programme sowie Nebenläufigkeit mit vergleichsweise wenig Schmerzen – und als Go-Entwickler sind Sie gefragter Experte.

GitHub ist die inoffizielle Paket-Registry für Go-Pakete geworden. Anders als etwa bei npm aus der JavaScript-Welt müssen Pakete nicht speziell verpackt und in einer Registry veröffentlicht werden – es reicht aus, ein Modul in einem Repo bei einem Git-Hoster abzulegen. Ist das Repo öffentlich, können andere es einbinden. Durch die Initialisierung hat Go die Datei go.mod angelegt, die zunächst nur den Namen Ihres Moduls und die Go-Version enthält. Später kommt hier eine Liste der verwendeten externen Pakete hinzu.

Jetzt können Sie mit dem Programmieren beginnen. Legen Sie die Datei main.go an. Die Datei darf auch anders heißen – der Name ist nur eine Konvention, der viele Go-Programmierer folgen. Wichtig ist nur, dass Ihr Modul ein Paket mit dem Namen `package main` enthält. Nur dann kann Go aus Ihrem Modul ein ausführbares Programm kompilieren. In diesem Paket wird der Compiler die Funktion `main()` suchen. Das ist der Einstiegspunkt ins Programm. Das minimale Kommandozeilenprogramm in Go kommt mit sieben Zeilen aus. Es gibt die Zeile „Eine Zeile Text“ aus:

```
package main
import (
 "fmt"
)
func main() {
 fmt.Println("Eine Zeile Text")
}
```

Nachdem das Paket den Namen `main` bekommen hat, wird ein von Go mitgeliefertes Paket mit dem Namen `fmt` importiert. Darin sind Funktionen zur Ein- und Ausgabe auf der Kommandozeile enthalten. Die Funktion `main()` ruft `Println()` aus dem Paket `fmt` auf. Der Methodenname beginnt nicht zufällig mit einem Großbuchstaben. Go kennt anders als etwa C++ keine Zugriffsmodifikatoren wie `private` oder `public`. Stattdessen achtet der Compiler auf die Schreibweise: Beginnt der Name einer Methode oder einer Variablen mit einem Großbuchstaben, ist sie automatisch `public`, sonst `private`.

Den Kompiliervorgang starten Sie auf der Kommandozeile mit

```
go build
```

Bisher besteht Ihr Programm nur aus einer Datei. Wenn Sie dem Go-Compiler keine weiteren Parameter übergeben, wird er den gesamten Ordner und alle Unterordner beim Kompilieren einbeziehen. Wenn das Projekt größer wird, müssen Sie nichts am Befehl ändern.

Der Compiler verarbeitet alle Dateien mit der Endung .go und baut daraus ein Binary. Im Ordner landet eine ausführbare Datei mit dem Namen, den Sie beim Initialisieren des Moduls angegeben haben, im Beispiel „web-cli“. Arbeiten Sie unter Windows, heißt die Datei „web-cli.exe“. Sie können Ihr Ergebnis direkt ausprobieren: `./web-cli` (oder `.\web-cli.exe`).

Beim Entwickeln wollen Sie möglicherweise nicht immer eine ausführbare Datei bauen, sondern nur schnell das Ergebnis sehen. Mit `go run ./...` startet das kompilierte Programm direkt, wird aber nirgends gespeichert.

Kreuzweise kompilieren

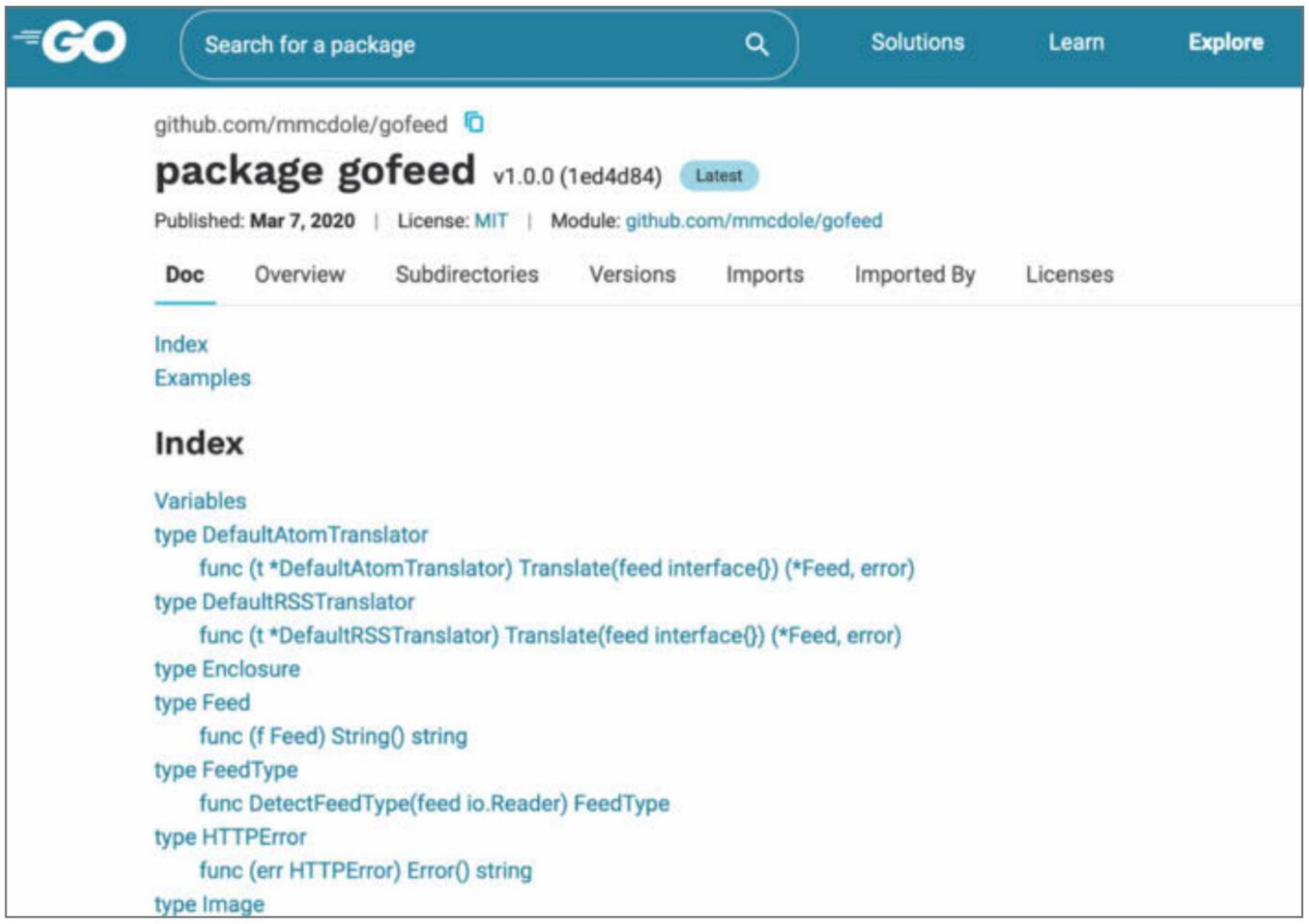
Sollten Sie sich für Go entschieden haben, um Software für unterschiedliche Betriebssysteme und Prozessoren zu entwickeln, können Sie direkt die Fähigkeiten des Cross-Compilers ausprobieren. Die Einrichtung ist denkbar einfach. Standardmäßig kompiliert Go immer für die Plattform, auf der der Compiler läuft. Ändern kann man das mit je einer Umgebungsvariable für die Prozessorarchitektur (`GOARCH`) und das Zielbetriebssystem (`GOOS`).

Die Werte für die gängigsten Betriebssysteme sind `windows`, `darwin` (für macOS) und `linux`. Die gängigsten Architekturen sind `386` (für 32-Bit-x86-Prozessoren), `amd64` (für 64-Bit-x86-Prozessoren) und `arm64` (für 64-Bit-ARM-Prozessoren wie den des Raspberry Pi).

Beachten Sie, dass nicht alle Kombinationen sinnvoll sind und funktionieren – so hat Apple zum Beispiel den Support für 32-Bit-Anwendungen unter macOS eingestellt und Go unterstützt die Kombination ebenfalls nicht mehr. Wenn Sie für exotische Betriebssysteme oder Prozessoren entwickeln müssen (zum Beispiel für FreeBSD oder MIPS64-Prozessoren), finden Sie über ct.de/w8k5 alle erlaubten Kombinationen in der Dokumentation.

Unter Windows setzen Sie Umgebungsvariablen mit `set`. Bauen Sie am besten eine Batch-Datei zum Build zusammen:

```
set GOOS=darwin
set GOARCH=amd64
go build
set GOOS=linux
go build
```



Die Paketverwaltung von Go kommt ohne Registry aus. Der Code landet bei einem Git-Hoster wie GitHub. Die Dokumentation findet man über pkg.go.dev.

Als Ergebnis bekommen Sie eine ausführbare Datei für den Mac und eine für Linux. Unter unixoiden Systemen arbeiten Sie mit `export` statt `set`. Das Beispielprojekt zum Artikel, das bei GitHub liegt, geht noch einen Schritt weiter: Es überlässt das Bauen der Binärdateien der CI/CD-Lösung GitHub Actions und legt sie bei jedem Release zum Download bei GitHub ab.

Fremdpakete

Bisher kann das Kommandozeilenprogramm nicht viel, das aber immerhin auf allen Plattformen. Einen Parser für Atom-Newsfeeds müssen Sie nicht selbst erfinden. Diese Aufgabe haben andere bereits gelöst – eine gute Gelegenheit, ein fremdes Paket einzubinden. Zu den umfangreichsten und am besten dokumentierten Bibliotheken für das Problem gehört das Projekt „mmcdole/gofeed“, das bei GitHub liegt.

Um den Parser einzubinden, müssen Sie ihn nur im `import()`-Block eintragen. Folgender Code reicht aus, um den Newsfeed von heise.de auszulesen und die letzten fünf Überschriften auszugeben:

```
package main
import (
  "fmt"
  "github.com/mmcdole/gofeed"
)
func main() {
  fp := gofeed.NewParser()
  feed, _ := fp.ParseURL("https://
↳www.heise.de/rss/heise-atom.xml")
  for i := 0; i < 5; i++ {
    fmt.Println(feed.Items[i].Title)
  }
}
```

Wenn Sie das Programm kompilieren, können Sie das Paketmanagement von Go Modules bei der Arbeit beobachten: Auf der Kommandozeile erfahren Sie, dass `gofeed` geladen wurde und selbst weitere Abhängigkeiten hat. Die Datei go.mod wurde automatisch um direkte und indirekte Abhängigkeiten ergänzt. Außerdem ist die Datei go.sum hinzugekommen. Sie nagelt die aktuellen Versionsstände aller Abhängigkeiten fest, damit das Ergebnis des Kompilierens konsistent ist, auch wenn sich in einer Bibliothek eine Kleinigkeit geändert haben sollte. Das Konzept dürfte etwa JavaScript-Entwicklern vom Paketmanager npm bekannt vorkommen, der die Datei package-lock.json mit dem gleichen Zweck anlegt. Anders als zum Beispiel in der Java- oder

.NET-Welt, wo Closed-Source-Pakete als vorkompilierte Binarys geladen werden können, lädt Go immer den Quelltext von externen Paketen und kompiliert ihn frisch.

Die Go-Entwickler empfehlen, im Alltag Abhängigkeiten wie oben beschrieben zu platzieren. Dabei wird immer die aktuelle Version heruntergeladen. Möchte man mehr Kontrolle über die verwendete Version haben, kann man jederzeit den Befehl `go get` einsetzen. Der Feed-Parser ist zum Zeitpunkt, als der Artikel fertig wurde, in der zweiten Beta-Version „v1.0.0-beta2“ verfügbar. Wenn Sie im Projekt beispielsweise unbedingt die erste Beta testen wollen, wechseln Sie mit folgendem Befehl:

```
go get github.com/mmcdole/gofeed@v1.0.0-beta
```

In der Datei go.mod wird sofort die ältere Version eingetragen. Weil Go sich einfach die Möglichkeiten von Git zunutze macht, können Sie auch ein Modul auf dem Stand eines speziellen Commits einbinden – das ist hilfreich, wenn man in einem Open-Source-Projekt die Auswirkungen einer Änderung prüfen will. Dazu reicht es, nach dem `@` den sechsstelligen Hash-Namen des Commits einzutragen.

Zum Umgang mit nicht abwärtskompatiblen Änderungen gibt es in der Go-Gemeinschaft klare Vereinbarungen: Releases werden nach den Regeln semantischer Versionierung benannt, beginnend mit einem `v`, also zum Beispiel `v1.2.5`. Minor-Versionen (mit einer neuen Nummer an der zweiten Stelle) dürfen nur Funktionen ergänzen, nicht entfernen. Patches (letzte Stelle) verändern nichts an der Funktion und beheben nur Fehler. Ist eine inkompatible Änderung, also ein Major-Release nötig, eröffnet man ein neues Modul (mit `go mod init`) und fügt an den Namen `/v2` an. Wer das Modul in Version 2 nutzen will, bindet es mit seinem vollen Namen, also inklusive `/v2` ein. Go behandelt Module mit unterschiedlichen Major-Versionen unabhängig voneinander. Es ist also problemlos möglich, beide Versionen im gleichen Projekt zu nutzen.

Es geht weiter

Das Beispielprogramm ist bisher alles andere als perfekt: Fehler werden nicht abgefangen, der Download des Newsfeeds geschieht nicht nebenläufig und typische Elemente eines vollwertigen Kommandozeilenwerkzeugs fehlen, beispielsweise ein Hilfetext oder Parameter. Auf der nächsten Seite wird das Projekt direkt erweitert. (jam) c't

Dokumentation und Beispiele

ct.de/w8k5

heise Academy

Neue Online-Kurse für IT-Professionals



Serverless Applications mit AWS Lambda

Der Programmierkurs für Node.js, Python und Go auf AWS

▸ *85 Videos, 10 Std. Spielzeit*



Azure Active Directory im hybriden Kontext

Der Einstiegskurs für Planung und Betrieb in Azure und On-Premises

▸ *60 Videos, 6:30 Std. Spielzeit*



Data Engineering: Überblick in Microsoft Azure

Big Data und Business Intelligence in der Microsoft-Cloud nutzen

▸ *57 Videos, 5:30 Std. Spielzeit*

Gratis-Lektionen aus **allen Kursen** anschauen unter:

heise-academy.de

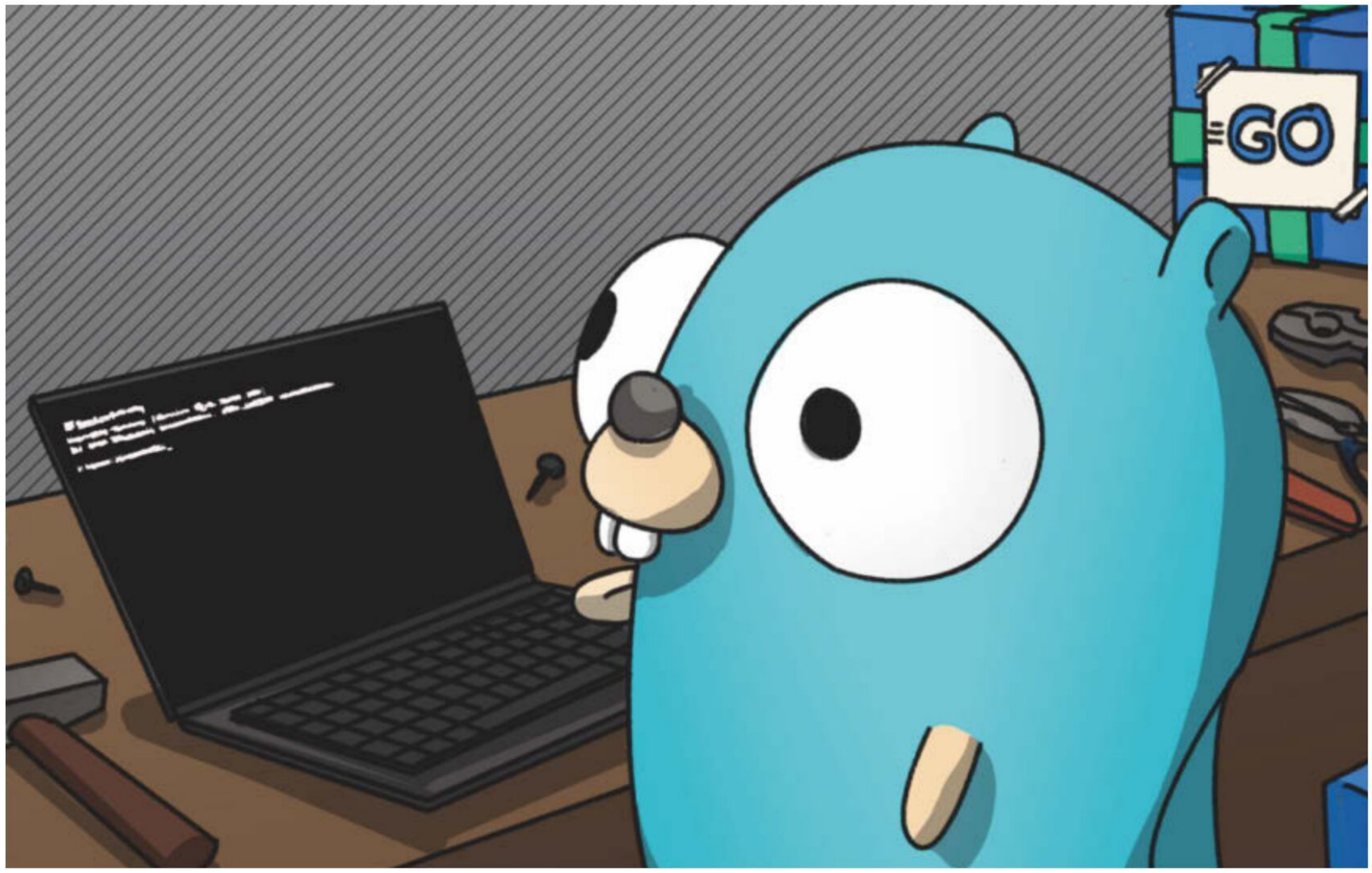


Bild: Thorsten Hübner

Go-Einstieg: Kommandozeile

In Go entstehen effiziente Kommandozeilenprogramme und Serverdienste für alle gängigen oder exotischen Betriebssysteme. Im zweiten Teil unserer Reihe lernen Sie Go-typische Sprachkonstrukte und Fehlerbehandlung kennen.

Von **Jan Mahn**

Wie Sie auf Ihrer Entwicklermaschine eine Umgebung schaffen, um Ihre ersten Anwendungen in Go zu entwickeln und zu kompilieren, haben wir ab Seite 142 vorgestellt. Neben den Fähigkeiten des Compilers stand vor allem das moderne Paketmanagement „Go Modules“ im Mittelpunkt – darüber können Sie schnell Bibliotheken anderer Entwickler in eigene Projekte einbinden.

Um den Code selbst ging es aber nur am Rande. In diesem zweiten Teil der Reihe soll die Beispielanwendung, ein einfacher News-Reader für die Kommandozeile, etwas dazu lernen. Auf dem Weg begegnen Sie typischen Sprachkonstrukten, klassenloser Objektorientierung und Fehlerbehandlung.

Den gesamten Code finden Sie über ct.de/wf36 zum Download bei GitHub. Damit Sie diese Reihe leichter nachvollziehen können, gibt es für diesen

zweiten Teil einen eigenen Branch mit dem Namen „part-2“. Wechseln Sie bei GitHub oben im Aufklappmenü auf diesen Branch, um den zugehörigen Code zu sehen.

Kommandozeilenprofi

Das Beispielprogramm hat den Namen `web-cli` bekommen und spuckt bisher schlicht die ersten fünf Überschriften aus dem Heise-Newsfeed aus. Von einem guten Kommandozeilenprogramm (Command line interface, CLI) erwartet man heutzutage aber noch ein paar Grundfertigkeiten: Mit dem Aufruf von `web-cli help` sollte es zum Beispiel Informationen zu seiner Benutzung ausgeben.

Schön wäre es, wenn das Programm außerdem mehrere Unterkommandos verstünde: `web-cli ls` soll alle Einträge mit einer ID tabellarisch darstellen und `web-cli describe <ID>` den Inhalt einer Nachricht anzeigen. Docker- oder Kubernetes-Nutzern wird diese Syntax mit Unterkommandos von der Docker-CLI, `kubectl` oder `helm` durchaus vertraut vorkommen.

In Go können Sie leicht auf Parameter zugreifen, die einem Programm übergeben werden. Das Paket `os` übernimmt die Kommunikation mit dem Betriebssystem. Was Sie in die Kommandozeile eingegeben haben, finden Sie im Slice `os.Args` – Slices heißen in Go alle Arrays mit dynamischen Längen. Der eingegebene Befehl wird an den Leerzeichen zerteilt und ins Slice gelegt. An nullter Stelle liegt der Name des Programms selbst, dann folgen eventuelle Parameter.

Ein übergebenes Unterkommando, das ja direkt auf den Namen des Binarys folgt, bekommen Sie mit diesem Befehl:

```
fmt.Println(os.Args[1])
```

Mit dem Slice `os.Args` und zahlreichen `if`- oder `switch`- und `case`-Konstrukten sowie der Funktion `len()`, die die Länge eines Strings auswertet, könnten Sie sich den Rahmen für die Anwendung zusammenstellen – inklusive eines manuell verfassten Hilfetextes und viel Fehlerbehandlung für alles, was der Nutzer so fälschlich eintippen könnte.

Wesentlich mehr Spaß macht es dagegen, von Beginn an ein CLI-Framework einzusetzen und sich dann zügig mit dem eigentlichen Problem zu befassen. Nicht nur für Webentwickler ist der schnelle Griff zum Framework sinnvoll – auch ein vermeintlich überschaubares CLI-Projekt, das nur ein kleines Bash-Skript ersetzen soll, kann schnell anwachsen, wenn der Nutzerkreis größer wird.

Das verbreitetste Open-Source-CLI-Framework heißt `spf13/cobra` und wird von fast allen großen CLI-Werkzeugen im Cloud-Umfeld eingesetzt: zum Beispiel von Docker, Kubernetes, etcd oder CockroachDB. Veröffentlicht ist der Code unter Apache License 2.0. Mit Cobra sind Sie also bestens gerüstet, wenn Ihr Programm unversehens zur Standardanwendung für eine ganze Industrie werden sollte.

Klassenlose Objekte

Beim Einrichten von Cobra lernen Sie nebenbei, wie man Go-Projektordner strukturiert und wie Objektorientierung ohne Klassen funktioniert. Die Datei main.go mit dem Einstiegspunkt enthält nur noch das Nötigste:

```
package main
import (
 "github.com/jamct/web-cli/cmd"
)
func main() {
  cmd.Exec()
}
```

Importiert wird das Subpackage `cmd`, das sich unterhalb von `github.com/jamct/web-cli` befinden soll. Den Namen des übergeordneten Pakets müssen Sie an Ihren Paketnamen anpassen (diesen haben Sie mit `go init` festgelegt und finden ihn in der Datei go.mod). Go erkennt, dass es sich um ein Subpackage des aktuellen Pakets handelt, und sucht es im Ordner „cmd“. Aus dem Paket `cmd` wird die Methode `Exec()` aufgerufen – Go-Programmierer bevorzugen, anders als ihre Kollegen aus der Java-Szene, kurze Namen für Methoden und Variablen. Nennen Sie Ihre Methode also bitte nicht `ExecuteCobraCliRuntimeInstanceOrFail()`.

Die Methode `Exec()` implementieren Sie im Ordner „cmd“. Legen Sie dort eine Datei an – bei der Benennung von Dateien innerhalb eines Pakets gibt es keine Regeln, Sie können den Code nach Belieben so auf Dateien aufteilen, wie Sie es für übersichtlich halten. Alle Dateien in diesem Ordner werden in das Paket einbezogen. Wichtig ist nur, dass sie alle mit der Zeile `package cmd` beginnen. Die Cobra-Entwickler empfehlen, die Instanziierung der Anwendung in einer Datei mit dem Namen root.go abzulegen:

```
package cmd
import (
 "fmt"
 "github.com/spf13/cobra"
 "os"
 "strings"
)

var rootCmd = &cobra.Command{
 Use:   "web-cli",
 Short: "Read Atom feeds",
 Long: `Just a small CLI application.
        Read Atom feeds`,
}

func Exec() {
 err := rootCmd.Execute()
 if err != nil{
  fmt.Println(err)
  os.Exit(1)
 }
}
```

Wie in anderen Sprachen gehört es auch in Go zur Konvention, Variablen vor den Methoden zu deklarieren. Das geschieht mit dem Schlüsselwort `var`. Die Variable `rootCmd` wird damit innerhalb des Pakets bekannt. `&cobra.Command` legt fest, womit die Variable befüllt werden soll: mit einem Zeiger (signalisiert durch das `&`) auf ein sogenanntes Struct mit dem Namen `Command` aus dem Paket `cobra`. Wie C kennt auch Go das Konzept von Zeigern. Dabei handelt es sich um einen Verweis auf die Speicheradresse, an der eine andere Variable abgelegt ist.

Structs sind die interessanteste Datenstruktur in Go. Mit ihr lassen sich verschachtelte Daten darstellen. Möchte man zum Beispiel Daten einer News-Meldung verarbeiten, definiert man ein passendes Struct mit:

```
type News struct {
 Id     int
 Title  string
 Text   string
 Author string
}
```

Um eine solche Hülle für Daten zu nutzen, befüllt man eine Variable und übergibt die Werte in `{ }`:

```
myNews := News{
 1, "Something happened",
 "Lorem ipsum",
 "Karla Kolumna",
}
```

Die Datenstruktur liegt jetzt in `myNews` und man kann auf die Inhalte mit der aus anderen objektorientierten Sprachen bekannten Syntax zugreifen. So gibt `myNews.Author` beispielsweise den Autor zurück.

Auch die Entwickler von Cobra haben ein solches Struct für die Befehle einer CLI-Anwendung definiert. Einen solchen Befehl haben Sie bereits erstellt – bei den Texten handelt es sich um den Namen des Befehls (`Use`), einen kurzen (`Short`) und einen langen Hilfetext (`Long`). Umrahmt man einen String mit Backticks, darf er auch Zeilenumbrüche enthalten.

In der Methode `Exec()` wird dann die Methode `Execute()` auf dem Struct aufgerufen – das mag jetzt etwas merkwürdig klingen, haben Sie Structs soeben als reine Datenstruktur kennengelernt. In Go kann man aber Methoden auf einem Struct definieren, indem man der Methodendefinition das Struct in Klammern voranstellt – das ist praktisch, um etwas mit den Daten im Struct anzustellen. Die folgende Methode für das News-Beispiel gibt etwa die ersten 100 Zeichen des Inhalts zurück. Das Struct selbst wird in der Methode als `n` bekannt gemacht:

```
func (n News) Abstract() string {
 return n.Text[:100]
}
```

Damit sind Sie bereits mit dem Grundprinzip der klassenlosen Objektorientierung in Go vertraut. Schon im ersten Teil dieser Reihe haben Sie erfahren, dass Groß- und Kleinschreibung darüber entscheidet, ob eine Variable oder Methode `public` oder `private`, also von außerhalb erreichbar ist. Dieses Prinzip gilt auf allen Ebenen: auf Paketebene genauso wie auf Ebene eines Structs. So könnten Sie einen Variablennamen in der Struct-Definition auch kleinschreiben und zum Anlegen und Auslesen eine Getter- und Setter-Funktion programmieren, die die Werte beispielsweise vorab validiert.

Wenn es schiefgeht

An dem Aufruf von `rootCmd.Execute()` in der Methode `Exec()` sehen Sie ein weiteres Konstrukt, das Sie in Go-Programmen häufig finden werden: Die Fehlerbehandlung wird über den Rückgabewert einer Funktion erledigt. In Go gibt es keine `throw`- und `catch`-Konstruktionen, es wird also nicht mit Excep-

tions umhergeworfen. Stattdessen geben Funktionen, die scheitern können, im Fehlerfall einen Wert vom Typ `error` zurück. Im Erfolgsfall ist dieser Wert leer, der zugehörige Nullwert für Fehler heißt `nil`. Um den Fehler abzufangen, prüft man per `if`-Statement, ob der Fehler `!=nil` ist, und reagiert dann entsprechend.

Da die meisten Funktionen einen Wert zurückgeben sollen, macht man sich die Fähigkeit zunutze, mehrere Werte gleichzeitig zurückzugeben. Angenommen, man braucht eine Funktion, die zwei Zahlen dividiert, und möchte den typischen Fehler abfangen, dass jemand versucht, durch null zu teilen. Die beiden Rückgabewerte der Funktion `divide()` sind ein `float64` und ein `error`:

```
import(
 "errors"
)

func divide(first float64, second float64)
(float64, error) {
 if second == 0 {
  return 0, errors.New("Div. by zero")
 }
 return first / second, nil
}
```

Um einen Fehler zu erzeugen, muss man das Paket `errors` importieren und dessen Methode `New()` aufrufen. Man könnte die Funktion wie folgt sicher nutzen, ohne dass das Programm mit einem „Division durch null“-Fehler abbricht:

```
value, err := divide(1, 0)
if err != nil {
 fmt.Println(err)
} else {
 fmt.Println(value)
}
```

Um den Programmierer zur Sparsamkeit zu erziehen, beklagt sich der Compiler, wenn er eine Variable anlegt, später aber nicht drauf zugreift. Der Versuch,



```
Read atom feeds (e.g. heise online newsfeed)
and show details on the command line.

Usage:
  web-cli [command]

Available Commands:
  describe    Show details for an article
  help        Help about any command
  ls          List articles

Flags:
  -h, --help   help for web-cli

Use "web-cli [command] --help" for more information about a command.
subcommand is required
```

Nutzt man ein CLI-Framework wie Cobra, bekommt man schön formatierte Hilfetexte ohne viel Programmieraufwand.

`err` erst zu befüllen, die `if`-Konstruktion aber wegzulassen, geht also schief. Möchte man während der Entwicklung eine Funktion schnell ausprobieren und sich erst später um die Fehlerbehandlung kümmern (explizit nicht empfohlen für fertigen Code), kann man den Fehler statt in eine Variable auch ins Nirwana umleiten. Dafür sieht Go den Unterstrich vor:

```
value,_ := divide(10, 5)
fmt.Println(value)
```

News abrufen

Zurück zum Kommandozeilenprogramm und dem kleinen Newsreader. Wenn Sie den oben stehenden Code kompilieren, haben Sie schon ein funktionierendes Programm. Der Aufruf von `./web-cli` zeigt die Hilfe zum Programm an, enthält aber noch keine Funktionalität. Es ist an der Zeit, die Subcommands `ls` und `describe` zu implementieren:

```
var cmdLs = &cobra.Command{
 Use:   "ls",
 Short: "List news",
 Long:  `List first 5 news`,
 Run: func(cmd *cobra.Command,
  args []string) {
  list()
 },
}
var cmdDescribe = &cobra.Command{
 Use:   "describe [id]",
 Short: "Show details for an article",
 Long:  `Details for an article`,
 Args:  cobra.MinimumNArgs(1),
 Run: func(cmd *cobra.Command, args []string) {
  describe(args[0])
 },
}
```

Um die neuen Befehle `cmdLs` und `cmdDescribe` als Unterkommandos zu registrieren, müssen Sie nur zwei Zeilen in der Methode `Exec()` vor `rootCmd.Execute()` einsetzen:

```
rootCmd.AddCommand(cmdLs)
rootCmd.AddCommand(cmdDescribe)
```

Auf diese Weise könnten Sie auch weitere Verschachtelungsebenen hinzufügen und dem Unterkommando ein Unterunterkommando spendieren. Legen Sie dazu einfach ein weiteres Struct vom Typ `cobra.Command` an und führen Sie `AddCommand()` auf `cmdLs` aus.

Anders als der Root-Befehl `rootCmd` haben die Befehle `cmdLs und cmdDescribe` das Attribut Run. Das enthält jeweils die Funktion, die ausgeführt werden soll, wenn man den Befehl aufruft.

Die Funktion `list()` sollten Sie in der Datei list.go im Ordner „cmd“ implementieren, der Ordner „describe“ bekommt ebenso eine eigene Datei. Die Funktion `list()` besteht aus dem Code, den Sie schon im ersten Teil mithilfe von `mmcdole/gofeed` umgesetzt haben. Hinzu kommt nur die Ausgabe der eindeutigen ID:

```
fmt.Println(item.Title + ": " + item.GUID)
```

Führen Sie `./wp-cli ls` aus, bekommen Sie die Titel der ersten fünf Nachrichten mit den IDs. Wer die Darstellung gern etwas übersichtlicher hätte, sollte sich nach einer Go-Bibliothek für hübsche Tabellenausgaben umsehen – davon gibt es viele.

Die Funktion `describe()` nimmt eine ID entgegen, lädt den Atom-Feed erneut herunter und iteriert über alle Einträge. Zum Einsatz kommt dabei eine `for`-Schleife mit dem Schlüsselwort `range`.

Dieses Konstrukt können Sie immer dann einsetzen, wenn Sie durch ein Slice oder Array laufen müssen. Zurück kommen der Index und das Element. Der Index wird hier nicht gebraucht und mit _ verworfen:

```
func describe(id string) {
 fp := gofeed.NewParser()
 feed, _ := fp.ParseURL(

  "https://www.heise.de/rss/heise-atom.xml")
 for _, element := range feed.Items {
  if element.GUID == id {
   fmt.Println(element.Title)
   fmt.Println(" ")
   fmt.Println(element.Description)
   return
  }
 }
 fmt.Println( id + " not found.")
}
```

Zur Perfektion

Damit ist der Workflow für den News-Reader fertig: Rufen Sie die Artikelliste mit `web-clils` auf. Kopieren Sie eine ID und zeigen Sie die Details mit `web-cli describe <ID>` an.

Ein paar Details sind noch verbesserungswürdig. So landen zum Beispiel mögliche Fehler des Atom-Parsers noch im Nirwana:

```
feed, _ := fp.ParseURL
```

Hier könnten Sie noch etwas Go-typische Fehlerbehandlung ergänzen und eine aussagekräftige Fehlermeldung auf der Kommandozeile ausgeben. Im dritten Teil dieser Reihe (ab Seite 154) wird der Code Unit-getestet und ab Seite 160 um einen kleinen Serverdienst erweitert, der im Hintergrund Nachrichten abrufen kann. (jam) c't

Beispielprojekt

ct.de/wf36



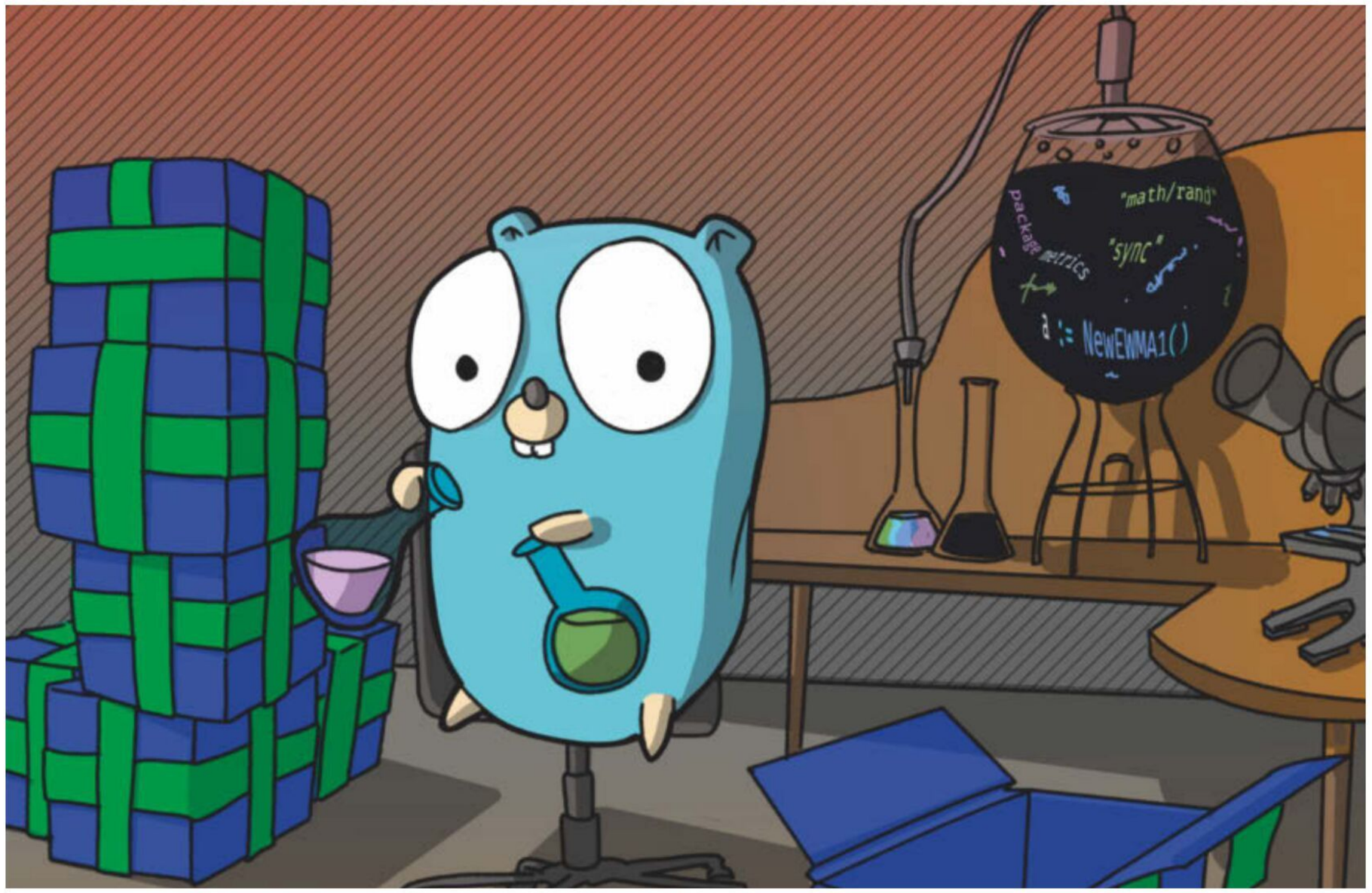


Bild: Thorsten Hübner

Go-Einstieg: Unit Tests

Damit aus gutem Go-Code sehr guter wird, sollte man Unit-Tests schreiben und regelmäßig ausführen. Go bringt die Voraussetzungen dafür schon mit und macht das Testen einfacher als viele andere Sprachen.

Von **Jan Mahn**

Unit-Tests sparen Zeit. Zugegeben, nicht direkt in der Anfangsphase eines Programmierprojekts, sondern erst, wenn die Software auch genutzt wird oder die Entwicklung vorangeschritten ist. Getestete Software enthält meist weniger leicht vermeidbare Fehler, und Tests verhindern vor allem, dass man denselben Fehler später wiederholt. Solche Unit-Tests enthalten Erwartungen, welchen Wert eine Funktion mit bekannten Eingabeparametern zurückgeben wird – die Testumgebung prüft diese systematisch und beschwert sich, wenn eine Funktion anders reagiert als im Test erwartet. Solche Tests kann der Entwickler direkt nach dem Verfassen der Funktion schreiben, er sollte die Liste der Tests aber vor allem immer dann ergänzen, wenn er gerade einen Fehler beseitigt hat.

In den meisten Programmiersprachen muss man sich als Entwickler eine Denkweise angewöhnen, um solche Unit-Tests zu verfassen. Die Testumgebungen kamen häufig erst später zur Sprache hinzu oder werden von einem externen Test-Framework erzeugt, das erst Jahre nach der Sprache erdacht wurde. Go als relativ junge Programmiersprache entstand dagegen zu einer Zeit, in der Unit-Tests schon gängige Praxis waren – daher konnten sich die Go-Erfinder von Anfang an Gedanken machen, wie sie Tests in die Go-Bordmittel integrieren können und wie Entwickler ohne zusätzliche Pakete und Hilfsmittel auskommen.

Dieser Artikel ist Teil 3 unserer Reihe für Go-Einsteiger. Der erste Teil beginnt auf Seite 142. Zum Nachvollziehen dieses dritten Teils ist es empfehlenswert, die ersten beiden Teile gelesen und nachgebaut zu haben. Den gesamten Code des Beispielprogramms finden Sie bei GitHub über ct.de/wrx6. Wechseln Sie für diesen Artikel im GitHub-Repository auf den Branch „part-3“.

Testkandidat

Eine wesentliche Ausgabe fehlt dem Beispiel-Reader noch, und die benötigte Funktion eignet sich gut, um die Vorgehensweise beim Unit-Testen zu veranschaulichen. Mit dem Befehl `web-cli describe <id>` kann man bereits den Titel und die Einleitung eines Artikels anzeigen. Es fehlt aber das Datum der Veröffentlichung. Diese Information wird im Newsreader-Format Atom in einem unansehnlichen Datumsformat nach ISO 8601 ausgeliefert, zum Beispiel:

```
2020-06-19T14:35:00+02:00
```

Übersetzt bedeutet das: 19.6.2020 um 14:35 nach Mitteleuropäischer Sommerzeit (UTC+2). Die Übersetzung von Zeit- und Datumsangaben geht Go anders an als viele andere Programmiersprachen. In Java, C und Verwandten gibt es Buchstaben, die je einen Teil des Datums darstellen, etwa y für das Jahr oder `HH:mm` für Stunden und Minuten.

Die Go-Funktion `time.Parse()` verwendet statt dieser Platzhalter eindeutige Werte für die einzelnen Datumskomponenten: Die Go-Entwickler haben 2006 für das Jahr, 1 für den Monat, 2 für den Tag und so weiter festgelegt. Für eine nach ISO 8601 formatierte Zeitangabe sieht der Format-String folgendermaßen aus: `2006-01-02T15:04:05-07:00`. Wollte man eine in Deutschland übliche Datumsangabe parsen, würde er `2.1.2006` lauten.

Mit diesem Wissen ist eine Funktion, die das ISO-8601-Datum annimmt und als deutschen String zurückgibt, schnell entwickelt:

```
func formatDate(date string) string {
 ref := "2006-01-02T15:04:05-07:00"
 t, _ := time.Parse(ref, date)
 return t.Format(
   "02.01.2006 um 15:04")
}
```

Die Methode `time.Parse()` macht aus dem String ein Datumsobjekt und `Format()` erstellt daraus einen neuen String. Als Zwischenschritt bietet es sich noch an, das Datumsobjekt auf die mitteleuropäische Zeitzone anzupassen – schließlich interessiert einen europäischen Leser internationaler Nachrichten, wann die Nachricht nach seiner Zeit veröffentlicht wurde. Das funktioniert mit einem `Location`-Objekt, das `LoadLocation()` erzeugt:

```
func formatDate(date string) string {
 ref := "2006-01-02T15:04:05-07:00"
 t, _ := time.Parse(ref, date)
 loc, _ := time.LoadLocation("Europe/Berlin")
 return t.In(loc).Format("02.01.2006 um 15:04")
}
```

Die Funktion `time.In(loc)` übernimmt die Übersetzung der Zeitzone. Platzieren Sie die Funktion `formatDate()` in der Datei cmd/describe.go, in der sich bisher nur die Funktion `describe()` befindet, die Details zu einer Newsfeed-Nachricht ausspuckt. Damit das Paket `time` geladen wird, ergänzen Sie es in der Liste der importierten Pakete am Anfang der Datei.

Zum Einsatz kommen soll die neu erstellte Funktion innerhalb von `describe()`, zum Beispiel vor der Zeile, die den Titel ausgibt:

```
fmt.Println(formatDate(element.Published))
```

Um die Funktion in Aktion zu sehen, müssen Sie das Projekt kompilieren (`go build .`), anschließend eine Liste der Artikel aufrufen (`./web-cli ls`) und einen der Einträge genauer untersuchen (`./web-cli describe <id>`). Das Programm gibt ein passend formatiertes Datum aus.

Der Test

Dass diese Umwandlung auch wirklich zuverlässig funktioniert, soll jetzt ein Unit-Test beweisen. Go-Ent-

wickler legen großen Wert darauf, dass Tests zum Code gehören. Es ist daher üblich, sie nicht in einem separaten Ordner von der Anwendung zu trennen. Legen Sie für die Tests, die die Datei describe.go betreffen, die Datei describe_test.go im gleichen Ordner mit folgendem Inhalt an:

```
package cmd
import "testing"
```

Das Paket `testing` ist Voraussetzung, um Tests zu verfassen. Bei der Benennung einer Testfunktion gibt es ebenfalls strikte Vorgaben: Der zu testenden Funktion wird `Test` vorangestellt und der erste Buchstabe gegebenenfalls großgeschrieben. Die passende Testfunktion zu `formatDate()` heißt demnach `TestFormatDate()`:

```
func TestFormatDate(t *testing.T) {
 in := "2011-11-11T11:11:11+01:00"
 want := "11.11.2011 um 11:11"

 out := formatDate(in)
 if out != want {
  t.Errorf("Date was wrong, got:↵
          ↳ %s, want: %s.", out, want)
 }
}
```

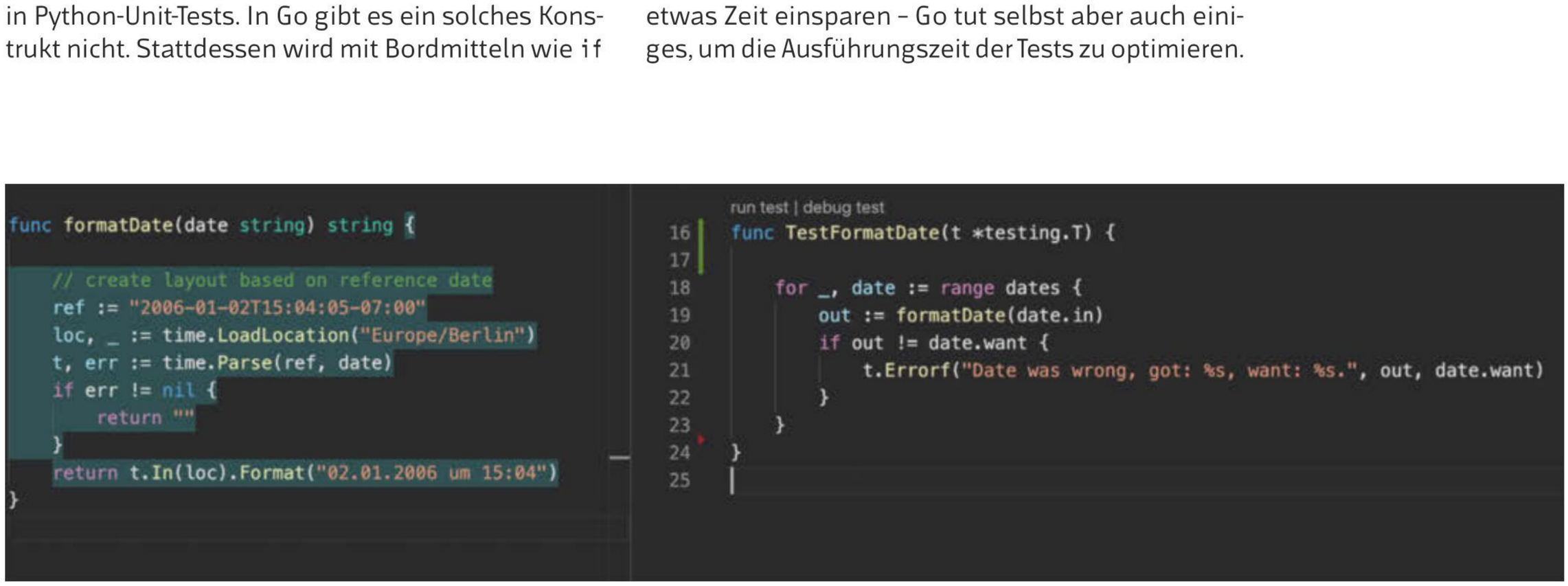
Innerhalb der Funktion kommt nur gewöhnlicher Go-Code zum Einsatz. In anderen Sprachen gibt es spezielle Befehle, zum Beispiel `self.assertEqual()` in Python-Unit-Tests. In Go gibt es ein solches Konstrukt nicht. Stattdessen wird mit Bordmitteln wie `if` geprüft, ob die Funktion wie geplant arbeitet. Weicht das Ergebnis von der Erwartung ab, lässt man die Funktion mit einem Fehler scheitern. `t.Error()` eignet sich für einfache Texte, mit `t.Errorf()` kann man den Text parametrisieren. Am besten ist die Fehlermeldung so aussagekräftig, dass man das Problem sofort erfassen kann.

Anstoßen können Sie den Test auf unterschiedlichen Wegen. Wenn Sie sich auf der Kommandozeile im Hauptverzeichnis des Projekts befinden, testen Sie das Paket `cmd` mit `go test ./cmd`. Das Testwerkzeug beginnt mit der Arbeit und vermeldet mit `ok`, dass keine Fehler aufgetreten sind. Alle Pakete und Unterpakete in einem Ordner testet `go test ./...` auf der Kommandozeile.

In den kompilierten Code der Anwendung fließen Tests nicht ein. Der Compiler ignoriert die Dateien und die Binärdatei wird nicht größer, wenn Sie viele Tests schreiben.

Besser mit IDE

Richtig Spaß macht die Arbeit mit Tests aber erst mit einer Entwicklungsumgebung. Bereits in den ersten Teilen der Reihe haben wir Visual Studio Code (VSC) mit der Erweiterung „Go“ empfohlen, die direkt von Googles Go-Team stammt. Ist sie installiert und aktiviert, finden Sie direkt über jeder Testfunktion die kleine Schaltfläche „run test“. Am Anfang der Datei erscheinen zusätzlich die Schaltflächen „run package tests“ und „run file tests“. Wenn man gerade an einem Problem herumlaboriert, kann man damit etwas Zeit einsparen – Go tut selbst aber auch einiges, um die Ausführungszeit der Tests zu optimieren.



Die Go-Erweiterung für Visual Studio Code zeigt, welche Zeilen die Tests aufgerufen haben.

Gibt es keine Änderungen an Code und Testfunktion, wird das Ergebnis eines Tests aus dem Cache geholt.

Die VSC-Erweiterung kann eine weitere Funktion der Testwerkzeuge nutzen und die Abdeckung des Tests veranschaulichen. Wenn Sie das gesamte Paket über VSC testen lassen, wird ein sogenanntes Cover Profile erzeugt. Um zu sehen, welche Zeilen Ihres Codes vom Test erfasst wurden, öffnen Sie die Befehlspalette von Visual Studio Code (Strg+Umschalt+P) und beginnen zu tippen: „Go: Toggle Test Coverage“. Die Autovervollständigung schlägt die richtige Funktion vor. Vom Test erfasste Zeilen in der Datei describe.go werden jetzt grün hinterlegt, alle anderen rot. Die prozentuale Abdeckung zeigt `go test` direkt auf der Kommandozeile an.

Was schiefgehen kann

Eine Testfunktion sollte alle aussagekräftigen Fälle prüfen. Was prüfenswert ist, muss sich der Entwickler gut überlegen und die Liste immer dann anpassen, wenn in der Software ein Problem aufgetreten ist. Die Datumsfunktion könnte man zum Beispiel mit einem Datum aus der Vergangenheit, einem Datum aus der fernen Zukunft, einem Datum aus dem Sommer und einem aus dem Winter auf die Probe stellen.

Die bisherige Testfunktion ist für solche Testreihen aber noch nicht vorbereitet und kann nur einen Wert testen. Ein häufiges Muster, das man auf viele Testfunktionen anwenden kann, ist eine Testtabelle, die Eingaben und erwartete Ausgaben enthält. Über diese Tabelle lässt man die Testfunktion iterieren. Im Listing unten finden Sie eine solche. Das `struct` mit dem Namen `dates` liegt außerhalb der Testfunktion – kommt später eine weitere Funktion mit ähnlichem Aufgabenbereich hinzu, könnte sie ebenfalls auf diese Testfälle zurückgreifen.

Die Liste der Testfälle enthält auch ein unvollständiges Datum. Es könnte ja durchaus passieren, dass der Anbieter des Newsfeeds einen Fehler macht.

```
package cmd
import "testing"

var dates = []struct {
   in   string
   want string
}{
  {"2006-01-02T15:04:05-07:00", "02.01.2006 um 23:04"},
  {"2020-01-01T10:00:00+01:00", "01.01.2020 um 10:00"},
  {"2020-05-31T10:04:06+02:00", "31.05.2020 um 10:04"},
  {"2111-11-11T11:11:11+01:00", "11.11.2111 um 11:11"},
  {"2020-05-31T", ""},
}

func TestFormatDate(t *testing.T) {
 for _, date := range dates {
  out := formatDate(date.in)
  if out != date.want {
   t.Errorf("Date was wrong, got: %s, want: %s.", out, date.want)
  }
 }
}
```

Ein Struct mit Testdaten und erwarteten Rückgabewerten deckt alle interessanten Fälle für die zu testende Funktion ab.

Erwartet wird ein leerer String – ist das Datum im Datensatz beschädigt, soll der Nutzer nicht mit einer Fehlermeldung oder gar einem falschen Datum behelligt werden, sondern lieber eine leere Zeile sehen.

Wenn Sie diese Testfunktion ausführen (über die Kommandozeile oder in VSC), wird sich das Testwerkzeug beschweren. Der letzte Fall, also ein ungültiges Datum, verlief nicht zufriedenstellend. Der Code der Funktion `formatDate()` muss repariert werden – das typische Vorgehen, wenn man mit Tests arbeitet. Eine kleine Fehlerbehandlung löst das Problem:

```
t, err := time.Parse(ref, date)
if err != nil {
 return ""
}
```

Wirft `time.Parse()` einen Fehler, wird jetzt der leere String erzeugt und der Test läuft erfolgreich durch.

Geschmackssache

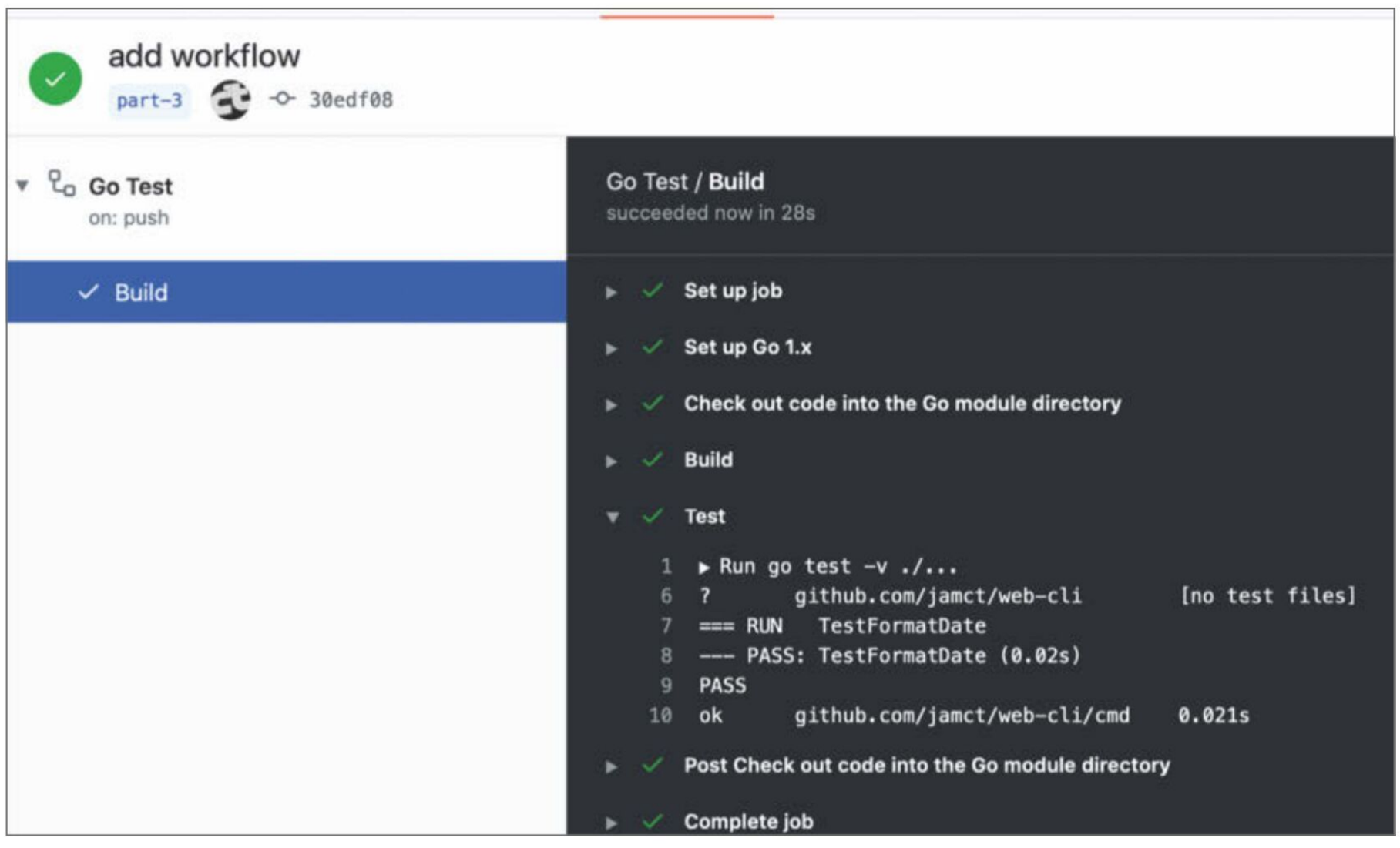
Wie Sie mit dem Wissen über Tests in Go umgehen, hängt von den persönlichen Vorlieben ab. Test-Puristen, die Test-Driven-Development lieben, verfassen erst die Tests und implementieren dann so lange die Funktionen, bis alle Tests durchlaufen. Die meisten Entwickler schreiben aber erst die Funktionen und konzipieren dann Tests, die möglichst jede Zeile Code aufrufen. Schafft man es nicht, einen Testfall zu konstruieren, der eine bestimmte Stelle im Code aufruft, ist das ein guter Hinweis darauf, dass die Code-Stelle womöglich überflüssig ist.

In jedem Fall ist es sinnvoll, die Tests so oft wie möglich aufzurufen, um Gewissheit zu haben, dass Änderungen ursprünglich lauffähigen Code nicht zerstört haben. Am besten gewöhnen Sie sich an, Ihren Code nach jeder erfolgreichen Kompilierung auch gleich testen zu lassen. Liegt der Code bei einem Git-Hoster wie GitLab oder GitHub, sollte man dessen CI-Lösung nutzen, um die Tests immer dann automatisch aufzurufen, wenn man Code ins Repository pusht. Ein Beispiel zur Einrichtung einer solchen Automation finden Sie im Repository zum Artikel.

Hat man sich an die Arbeitsweise mit Unit-Tests gewöhnt, wird das Testen schnell zur Sucht – insbesondere die Jagd nach höherer Code-Abdeckung ist reizvoll. Belohnt wird man durch stabilere und bessere Software. Jetzt ist es an der Zeit, Ihre kleine Anwendung mit einer Server-Komponente auszustatten. Wie das gelingt, erfahren Sie auf den nächsten Seiten. (jam) c't

Beispielprojekt

ct.de/wrx6



Integriert man die Unit-Tests in eine Automation wie GitHub Actions, kann man fehlerhaften oder unvollständigen Code rechtzeitig erkennen, bevor er im fertigen Programm landet.

IMPRESSUM

Redaktion

Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-300
Telefax: 05 11/53 52-417
Internet: www.ct-special.de

Leserbriefe und Fragen zum Heft: sonderhefte@ct.de

Die E-Mail-Adressen der Redakteure haben die Form xx@ct.de oder xxx@ct.de. Setzen Sie statt „xx“ oder „xxx“ bitte das Redakteurs-Kürzel ein. Die Kürzel finden Sie am Ende der Artikel und hier im Impressum.

Chefredakteur: Jobst-H. Kehrhahn (keh)
(verantwortlich für den Textteil)

Konzeption: Jan Mahn (jam), Peter Siering (ps)

Koordination: Ann-Sophie Eikermann (ase),
Angela Meyer (anm)

Redaktion: Holger Bleich (hob), Niklas Dierking (ndi),
Jan Mahn (jam), Angela Meyer (anm), Peter Siering (ps)

Mitarbeiter: Johannes Endres, Götz Rieger, Merlin Schumacher

Assistenz: Susanne Cölle (suc), Tim Rittmeier (tir),
Christopher Tränkmann (cht), Martin Triadan (mat)

DTP-Produktion: Lara Bögner, Beatrix Dedek, Madlen Grunert,
Lisa Hemmerling, Cathrin Kapell, Kirsten Last, Martina Lübke,
Steffi Martens, Marei Stade, Matthias Timm, Ricardo Ulbricht

Fotografie: Andreas Wodrich, Melissa Ramson

Produktion: Christine Kreye (Ltg.),
Melanie Becker, Kevin Harte, Martin Kreft,
Pascal Wissner

Illustration: Rudolf A. Blaha, Frankfurt am Main;
Sven Hauth, Schülp; Thorsten Hübner, Berlin;
Albert Hulm, Berlin; Hagen Kober, Nagold;
Thomas Kuhlenbeck, Münster; Andreas Martini, Wettin

Titel: Steffi Martens, www.flaticon.com, www.freepik.com,
www.stock.adobe.com

Verlag

Heise Medien GmbH & Co. KG
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-0
Telefax: 05 11/53 52-129
Internet: www.heise.de

Herausgeber: Christian Heise, Ansgar Heise, Christian Persson

Geschäftsführer: Ansgar Heise, Beate Gerold

Mitglieder der Geschäftsleitung: Jörg Mühle, Falko Ossmann

Anzeigenleitung: Michael Hanke (-167)
(verantwortlich für den Anzeigenteil),
www.heise.de/mediadaten/ct

Anzeigenverkauf: Verlagsbüro ID GmbH & Co. KG,
Tel.: 05 11/61 65 95-0, www.verlagsbuero-id.de

Leiter Vertrieb und Marketing: André Lux (-299)

Service Sonderdrucke: Julia Conrades (-156)

Druck: Firmengruppe APPL Druck GmbH & Co. KG,
Senefelder Str. 3–11, 86650 Wemding

Vertrieb Einzelverkauf:
DMV DER MEDIENVERTRIEB GmbH & Co. KG
Meßberg 1
20086 Hamburg
Tel.: 040/3019 1800, Fax: 040/3019 145 1815
E-Mail: info@dermedienvertrieb.de
Internet: dermedienvertrieb.de

Einzelpreis: € 14,90; Schweiz CHF 27.90;
Österreich € 16,40; Luxemburg € 17,10

Erstverkaufstag: 19.10.2021

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden. Kein Teil dieser Publikation darf ohne ausdrückliche schriftliche Genehmigung des Verlags in irgendeiner Form reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfaltigt oder verbreitet werden. Die Nutzung der Programme, Schaltpläne und gedruckten Schaltungen ist nur zum Zweck der Fortbildung und zum persönlichen Gebrauch des Lesers gestattet.

Für unverlangt eingesandte Manuskripte kann keine Haftung übernommen werden. Mit Übergabe der Manuskripte und Bilder an die Redaktion erteilt der Verfasser dem Verlag das Exklusivrecht zur Veröffentlichung. Honorierte Arbeiten gehen in das Verfügungsrecht des Verlages über. Sämtliche Veröffentlichungen in c’t erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes.

Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Hergestellt und produziert mit Xpublisher:
www.xpublisher.com

Printed in Germany.

Alle Rechte vorbehalten.

© Copyright 2021 by
Heise Medien GmbH & Co. KG

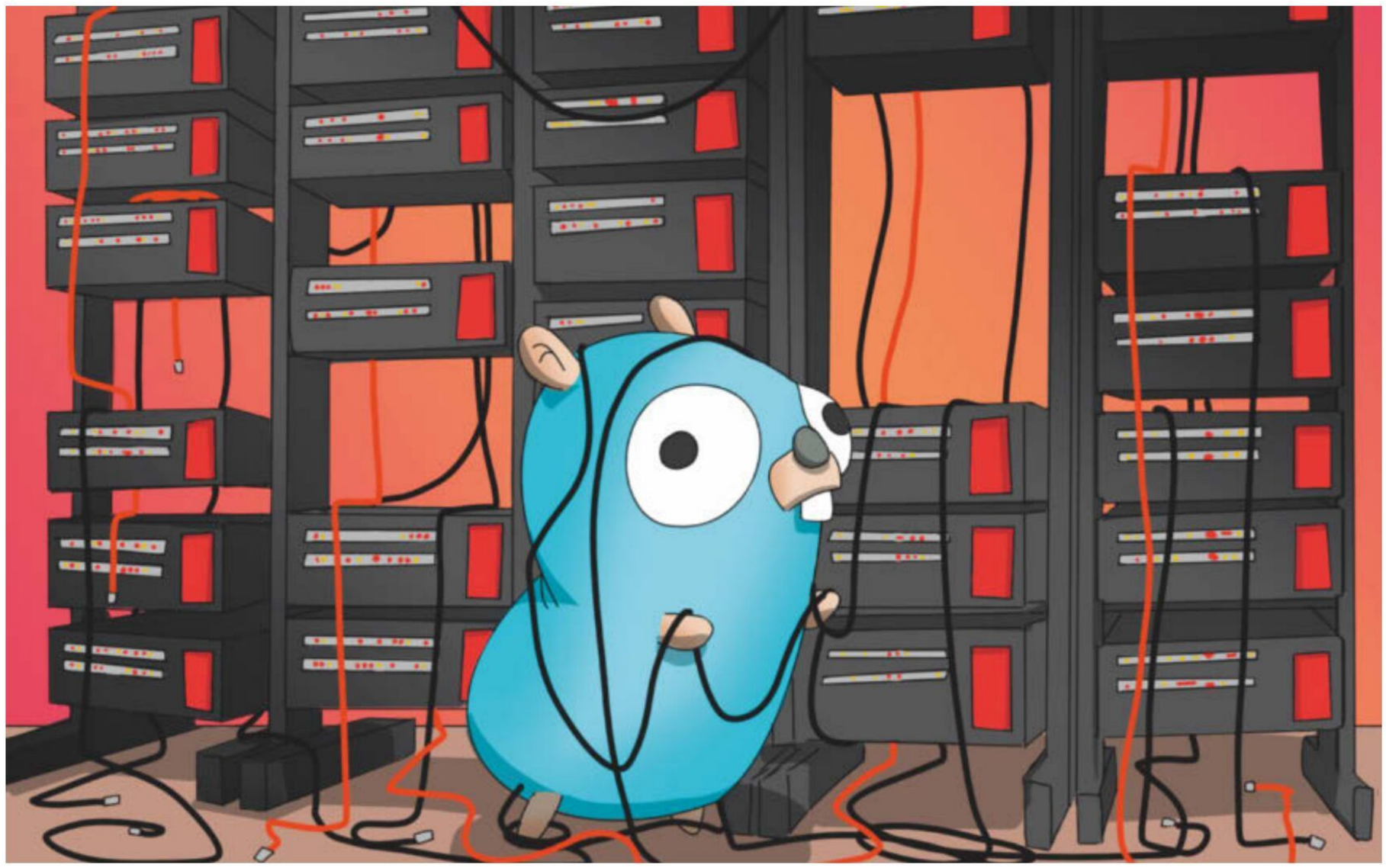


Bild: Thorsten Hübner

Go-Einstieg: Server-Dienste

Ein Server, der Webseiten oder ein JSON-API ausliefert, ist in Go schnell eingerichtet. Bei der Arbeit mit JSON-Objekten muss man die Datenstruktur aber sorgfältig modellieren.

Von **Jan Mahn**

Die Programmiersprache Go eignet sich für plattformübergreifende Kommandozeilenwerkzeuge, aber auch Serverdienste programmieren Sie in der Sprache effizient und zügig. Auf den vorangegangenen Seiten haben Sie das Handwerkszeug für einen kleinen News-Reader in Go kennengelernt. Dieses Programm soll jetzt um eine Server-Funktion erweitert werden, die eingelesene Nachrichten als formatierte HTML-Seite und per JSON-API ausgibt. Den Code finden Sie im GitHub-Repository zur Artikelreihe (siehe ct.de/wjfw). Wechseln Sie für diesen Artikel auf den Branch `part-4`.

Stiller Diener

Verantwortlich für das Lauschen auf einem Port und das Beantworten von HTTP-Anfragen ist das

Go-Standardpaket `net/http`. Einen HTTP-Server starten Sie mit nur einer Zeile:

```
http.ListenAndServe(":8080", nil)
```

Den Port übergeben Sie nicht als Zahlwert, sondern als String mit einem Doppelpunkt am Anfang – dann lauscht der Server auf allen Netzwerkschnittstellen. Sie können auch eine IP-Adresse mit Port übergeben, etwa `"127.0.0.1:8080"`.

Wenn Sie diese Zeile in einem leeren Go-Programm ausführen, antwortet bereits ein Webserver – allerdings stets mit dem Statuscode 404, weil noch keine Antworten definiert sind. `ListenAndServe()` gibt nur bei schweren Fehlern einen Rückgabewert zurück, und zwar vom Typ `Error`. Ansonsten läuft die Funktion so lange, bis jemand das Programm beendet. Häufig hüllt man den Listener in eine Log-Funktion, die einen Absturz des Programms protokolliert:

```
log.Fatal(http.ListenAndServe(":8080", nil))
```

Damit der Server mehr als nur „not found“ zurückgeben kann, müssen Sie vor dem Start von `ListenAndServe()` sogenannte Handler registrieren. Der Handler bekommt eine eigene Funktion:

```
func handleTest(w http.ResponseWriter,
r *http.Request) {
 title := r.URL.Path[1:]
 fmt.Fprintf(w, "<h1>%s</h1>", title)
}
```

Dem Handler übergibt der HTTP-Server den `ResponseWriter` in der Variable `w`. Diesen Kanal muss man nutzen, um Inhalte als HTTP-Antwort zurückzuschicken. Außerdem bekommt er in `r` einen Zeiger auf die HTTP-Anfrage. Im Beispiel wird ihr der Pfad aus der URL entnommen und als Titel ausgegeben. `fmt.Fprintf()` verhält sich wie `fmt.Printf()`, schreibt aber nicht auf die Kommandozeile, sondern in einen Ausgabekanal, in diesem Fall in den HTTP-Kanal.

Vor `http.ListenAndServe()` müssen Sie den Handler noch registrieren. Alle Pfade unterhalb von „localhost:8080/test/“ sollen von diesem Handler behandelt werden:

```
http.HandleFunc("/test/", handleTest)
```

Diese Route landet automatisch im Standard-Router `DefaultServeMux`, der im Hintergrund angelegt und von `http.ListenAndServe()` genutzt wird.

Den Gorilla loslassen

Für einfache Szenarien ist das mitgelieferte Routing ausreichend. Innerhalb des Handlers könnte man den Pfad zerlegen und mit `if` und regulären Ausdrücken an weitere Funktionen verweisen. Mit einem HTTP-Router aus einem Go-Web-Framework machen Sie sich das Leben aber einfacher. Er reicht eingehende Anfragen zum passenden Code weiter. Einer der beliebtesten Router im Go-Universum heißt `gorilla/mux`. Er kommt in vielen großen Projekten zum Einsatz und erspart viel Arbeit beim Zuteilen von Anfragen an den passenden Handler.

Der Newsreader soll vier Anfragen unterscheiden. Unter der Adresse /news soll er eine Liste aller Nachrichten ausliefern, unter /news/{id} die Detailansicht einer Nachricht als HTML-Webseite. Zusätzlich soll es zwei API-Endpunkte geben, die unter /api/news und /api/news/{id} mit einem JSON-Objekt antworten.

Der Code für die Server-Funktionen gehört in die Datei cmd/serve.go. Im Listing auf der nächsten Seite sehen Sie das Grundgerüst der Server-Funktion. `serve()` ist der Einstiegspunkt – diese Funktion wird vom CLI-Framework Cobra aufgerufen, das im zweiten Teil dieser Reihe vorgestellt wurde. Wenn Sie das Programm kompilieren und `./webcli serve` ausführen, startet der Server. Der Code erzeugt zunächst eine Gorilla-Router-Instanz und speichert sie in der Variablen `r`. Die Funktion `StrictSlash(true)` löst ein häufiges Problem ganz ohne reguläre Ausdrücke: Der Nutzer könnte entweder /news oder /news/ in den Browser eintippen – der Slash am Ende soll immer entfernt werden, damit die Route gefunden wird.

Eine der Stärken von Gorilla ist die Parametrisierung von Pfadangaben. Mit `/news/{id}` landet die ID im Handler. Über `mux.Vars(r)` greifen Sie auf diese Pfadparameter zu.

Die Umsetzung der Funktionen für die HTML-Ausgabe ist reine Fleißarbeit und für die Serverfunktionen von Go nicht von Interesse. Sie finden den vollständigen Code im GitHub-Repository.

Spannender sind die JSON-Endpunkte des API. Um JSON auszugeben, muss man ein solches Objekt erst erzeugen. Eine dumme Idee ist es, `{ }`, `[ ]` und die Daten als Strings per Hand zu JSON zusammenzuflicken. Stattdessen lohnt ein Exkurs in die JSON-Funktionen von Go. Diese funktionieren in zwei Richtungen: Entweder hat man, wie in diesem Fall, strukturierte Daten und möchte einen JSON-String erzeugen. Oder man hat einen solchen irgendwo

abgerufen (etwa von einem anderen API oder aus einer Textdatei) und möchte ihn zu einem Objekt umwandeln.

Arbeiten mit JSON

Im besten Falle kennt man die Struktur der Daten, die man verarbeiten möchte, und kann sie als `struct` oder als Verschachtelung solcher modellieren:

```
type Example struct{
  Name string
  ID   int
  Tags []string
}
```

Alle Elemente des `struct` müssen von außen zugänglich sein, also mit einem Großbuchstaben beginnen. Mit dieser Vorbereitung ist die Umwandlung von und nach JSON sehr einfach. Importieren müssen Sie das Paket `encoding/json`. Das Umwandeln eines Objekts in JSON wird als „Marshalling“ bezeichnet – auf Deutsch etwa „ordnen“ oder „aufstellen“:

```
package main
import  (
 "fmt"
 "encoding/json"
)
func main() {
 ex := Example{
  Name:  "Beispiel",
  ID:    1,
  Tags:  []string{"Test", "123"},
 }
 bytes, err := json.Marshal(ex)
 if err != nil {
  fmt.Println("error:", err)
 }
 fmt.Println(string(bytes))
}
```

Erst wird `ex` mit einem `struct` vom Typ `Example` angelegt. Es enthält einen String, einen Integer und ein Slice aus Strings, also ein Array dynamischer Länge. Mit `json.Marshal(ex)` wird daraus ein JSON-Objekt in Form eines Slices aus Bytes – um das JSON-Objekt lesbar auszugeben, muss man es mit `string(bytes)` in einen String konvertieren. Das Ergebnis der Umwandlung:

```
{"Name":"Beispiel","ID":1,"Tags":["Test","123"]}
```

```
package cmd

import (
  "fmt"
  "log"
  "net/http"
  "github.com/gorilla/mux"
)

func serve(){
  r := mux.NewRouter()
  r.StrictSlash(true)
  r.HandleFunc("/news", handleAll)
  r.HandleFunc("/news/{id}", handleSingle)
  r.HandleFunc("/api/news",handleAllApi).Methods("GET")
  r.HandleFunc("/api/news/{id}", handleSingleApi).Methods("GET")
  go log.Fatal(http.ListenAndServe(":8080", r))
}

func handleAll(w http.ResponseWriter, r *http.Request) {
   fmt.Fprint(w, "<h1>Alle News</h1>")
}
func handleAllApi(w http.ResponseWriter, r *http.Request) {
  fmt.Fprint(w, "<h1>Alle NEWS als JSON-API</h1>")
}
func handleSingle(w http.ResponseWriter, r *http.Request) {
  vars := mux.Vars(r)
  id := vars["id"]
  fmt.Fprintf(w, "<h1>Nachricht %s</h1>", id)
}
```

Mit dem Routing-Paket Gorilla sind auch große Webanwendungen und APIs kein Problem.

Nicht immer ist es erwünscht, dass die Elemente im JSON-Objekt denselben Namen tragen wie im Go-`struct`. Schon die Großschreibung der Schlüssel ist für JSON-Daten eher unüblich (innerhalb des Go-`struct` aber notwendig, damit die Inhalte zugänglich sind). Wenn der Name im JSON-Objekt abweichen soll, fügt man bei der Definition des `struct` sogenannte Tags an:

```
type Example struct{
  Name string    `json:"example_name"`
```

```
  ID   int      `json:"id,omitempty"`
  Tags []string `json:"tags"`
}
```

Der JSON-Marshaller wertet diese Tags aus und berücksichtigt sie bei der Umwandlung. Die optionale Angabe `omitempty` weist den Marshaller an, das Element in der JSON-Ausgabe wegzulassen, wenn es mit dem Null-Wert belegt ist.

JSON zerlegen

Auch der Rückweg vom String zum `struct` ist einfach, wenn man die genaue Struktur der Daten kennt und sie bereits als `struct` definiert hat. Die oben angelegten Tags funktionieren auch in die andere Richtung mit dem JSON-Unmarshaller:

```
func main() {
 str := `{"example_name": "test", "id":55, "tags":
["1", "2"]}`
 bytes := []byte(str)
 ex := Example{}
 json.Unmarshal(bytes, &ex)
 fmt.Println(ex)
 fmt.Println(ex.ID)
}
```

Kommt das JSON-Objekt wie im Beispiel als String an, müssen Sie es erst in ein Byte-Slice konvertieren. Dann brauchen Sie eine `Example`-Instanz als leere Hülle für das `struct`, im Beispiel die Variable `ex`. Die Funktion `json.Unmarshal()` bekommt die Daten und einen Zeiger auf diese Hülle und füllt letztere mit den Inhalten. Anschließend können Sie auf die Daten des Objekts zugreifen.

Beim Modellieren des `struct` müssen Sie nicht immer das gesamte JSON-Objekt nachbilden, das Sie erwarten. Es reicht, sich auf die Teile zu beschränken, mit denen Sie arbeiten möchten. Nicht definierte Elemente ignoriert der Unmarshaller einfach.

Probleme kann es dennoch geben. Wenn der JSON-String nicht den Erwartungen entspricht oder syntaktische Fehler enthält, enthält `ex` nach dem Umwandeln ein leeres `struct`. Fehlerbehandlung ist daher dringend angeraten:

```
err := json.Unmarshal(bytes, &ex)
if err != nil{
 fmt.Println(err.Error())
 return
}
```

Gleichung mit Unbekannten

Nicht immer kennt man den genauen Aufbau der Datenstruktur des JSON-Objekts. In diesem Fall kann man sich das Go-Konzept des leeren Interface zunutze machen. Statt eines `struct` übergibt man dem Unmarshaller einen Zeiger auf eine leere Hülle unbekannten Typs, das leere Interface heißt `interface{}`:

```
str := {"name":"Beispiel","id":1}
var ex interface{}
json.Unmarshal([]byte(str), &ex)
```

Um anschließend auf Werte zuzugreifen, muss man Annahmen darüber treffen, welche Datentypen im Objekt liegen – ganz ohne Typisierung funktioniert eine stark typisierte Sprache wie Go nicht. Eine solche Type Assertion formuliert man mit der Syntax `objekt.(<Typ>)`. Das dekodierte JSON-Objekt ist auf der obersten Ebene ein Array mit Schlüsseln im String-Format (zum Beispiel `"name"` oder `"id"`). In Go entspricht das einer Map – die Inhalte sind weitere unbekannte Interfaces. Das Objekt `ex` sollte man im ersten Schritt wie folgt auflösen:

```
m := ex.(map[string]interface{})
```

Über die Syntax eines Arrays kommt der Inhalt zum Vorschein. Auch darüber müssen Sie Annahmen treffen. Vermutlich handelt es sich beim Namen (`"name":`) um einen String:

```
name := m["name"].(string)
fmt.Println("Der Name ist " + name)
```

So kann man flexibel auf unterschiedliche Daten reagieren, ohne die Typisierung aufzugeben. Wenn Sie das JSON-Objekt etwa von einem API beziehen und dessen Entwickler es mit den Typen nicht so genau nehmen und die ID mal korrekt als Zahlwert (`"id": 1`) und mal als String verschicken (`"id": "2"`), können Sie darauf per `switch-case`-Konstruktion reagieren. Mit `m["id"].(type)` erfahren Sie den Typ.

Alle Zahlwerte legt der Unmarshaller in eine Variable vom Typ `float64`. So wird aus allen möglichen Eingabeformaten ein einheitlicher Integer für die ID, mit dem man später zum Beispiel rechnen könnte:

```
var id int
switch i := m["id"].(type) {
  case string:
```

```
    id,_ = strconv.Atoi(i)
  case float64:
    id = int(i)
}
fmt.Println(id)
```

Die Funktion `Atoi()` aus dem Paket `strconv` wandelt Strings in Integer um, sofern das möglich ist. Der Name des Pakets gehört ebenfalls in die Liste der Importe am Anfang der Datei.

Der Umgang mit JSON-Objekten erscheint Ihnen vielleicht etwas sperrig, wenn Sie bisher mit klassischen Websprachen wie PHP oder JavaScript gearbeitet haben, die schwach typisiert sind. Das Wandeln von JSON in Objekte und andersherum ist dort in einer Zeile erledigt – wenn man in Go mit `structs` arbeitet, hat man nicht nur die Sicherheit, dass man es mit dem richtigen Datentyp zu tun hat. Der Code ist später auch leichter zu lesen.

API-Ausgaben

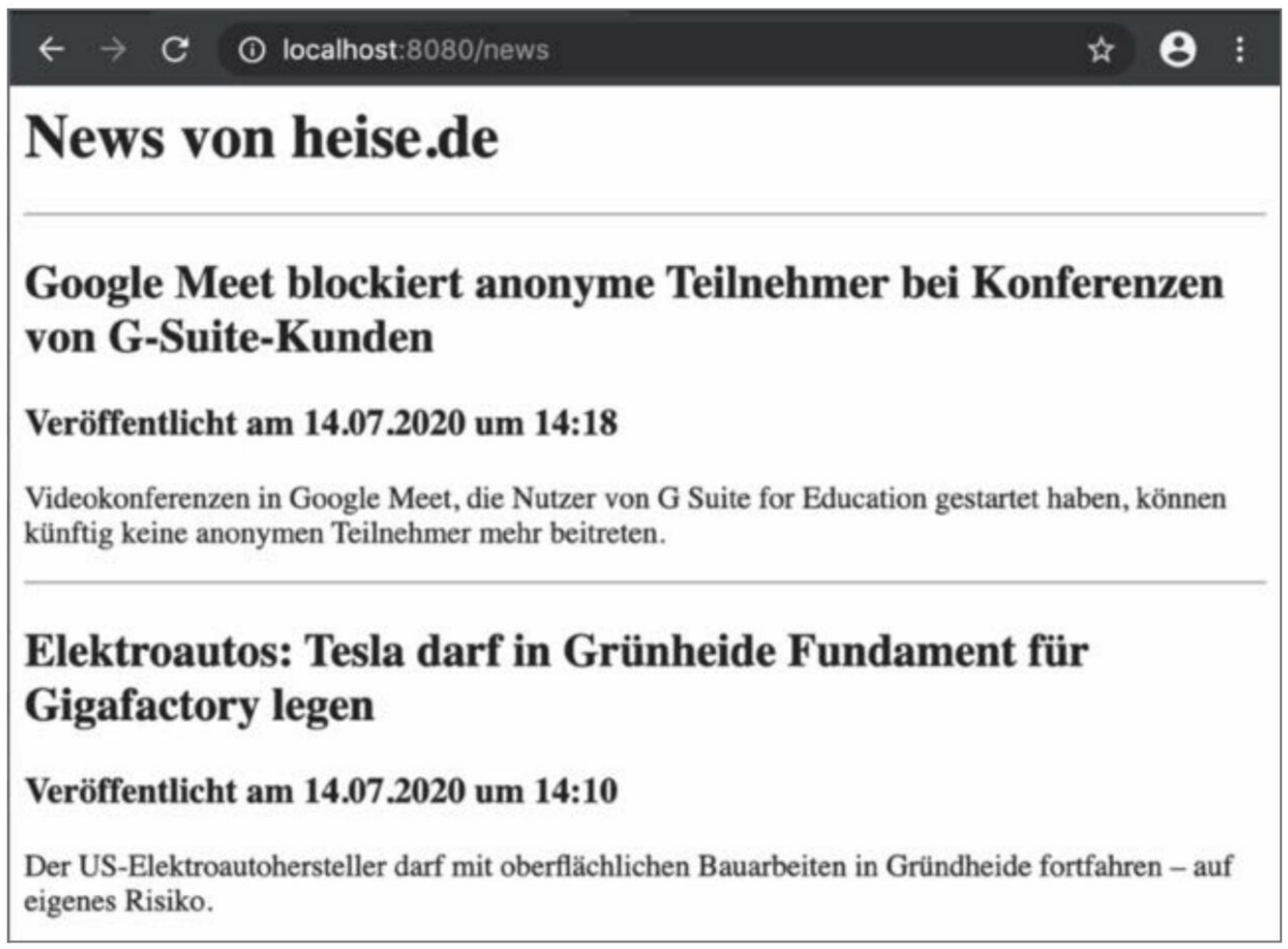
`structs` sind in jedem Fall die sauberste Lösung, wenn man JSON-Daten erzeugen möchte. Bisher sind die Funktionen im Beispielprojekt zum Abrufen von einem oder mehreren Nachrichten aus dem Newsfeed dafür noch nicht ansatzweise bereit. Programmlogik und Darstellung sind zum Beispiel in der Funktion `describe()` unschön miteinander vermischt. Zeit für etwas Refactoring – eine der Arbeiten, bei denen man viel über die Sprache lernt. Wenn Sie die Erkenntnisse selbst sammeln möchten, nehmen Sie sich den Stand aus dem Git-Branch `part-3` vor und versuchen Sie, die abgerufenen Nachrichten selbst in `structs` zu verpacken.

Im Listing auf dieser Doppelseite sehen Sie eine mögliche Herangehensweise. Das `struct NewsList` enthält ein Slice aus `News`. Abgerufen werden die ersten Nachrichten von der neuen Funktion `newsList()`, die eine `NewsList` produziert.

Diese Funktion kann man in einer eigenen Datei im Paket `cmd` unterbringen. Auf `newsList()` sollen alle Funktionen zugreifen, die News-Listen ausgeben, also auch der Handler für das JSON-API:

```
func handleAllApi(
 w http.ResponseWriter, r *http.Request){
 news := newsList(5)
  js, _ := json.Marshal(news)
  w.Write(js)
}
```

Weil die Daten bereits als `structs` vorliegen, reichen wenige Zeilen aus. Der Marshaller erstellt ein



Der Webserver liest den Atom-Newsfeed ein, formatiert die Inhalte und generiert eine HTML-Ausgabe.

Quelltext zum Projekt
ct.de/wjfw

```
type NewsList struct{
  News         []News    `json:"news,omitempty"`
  BaseUrl      string    `json:"base_url,omitempty"`
}
type News struct{
  Date         string    `json:"date,omitempty"`
  ID           string    `json:"id"`
  Title         string    `json:"title"`
  Description  string    `json:"description"`
}

func newsList(count int) NewsList{
  list := NewsList{ BaseUrl: "http://heise.de/"}
  fp := gofeed.NewParser()
  feed, _ := fp.ParseURL("https://www.heise.de/rss/heise-atom.xml")

  for i := 0; i < count; i++ {
   item := feed.Items[i]
   id := strings.Replace(item.GUID, list.BaseUrl, "", -1)
   n := News{ item.Published, id, item.Title, item.Description}
   list.News = append(list.News, n)

  }
 return list

}
```

Die Daten aus dem Newsfeed landen in Structs. Damit wird der Export in JSON-Objekte leicht.

Byte-Slice und `w.Write()` schickt es an den Client zurück. Um dem Client zu signalisieren, dass er JSON erhält, empfiehlt es sich, vorher den passenden Header zu senden:

```
w.Header().Set("Content-Type","application/json")
```

Web-Alternative

Die Beispiele zeigen, dass Go eine attraktive Alternative für Web- und vor allem für API-Projekte ist. Das Gorilla-Framework enthält nicht nur den Router, sondern auch Lösungen für andere typische Aufgaben von Webanwendungen wie die Arbeit mit Cookies oder Formulareingaben.

Der gesamte Server steckt nach dem Kompilieren in einer Binärdatei, die man für Windows, Linux oder macOS erzeugen kann. Externe Abhängigkeiten gibt es keine – im Vergleich zu einem Webserver mit PHP, Python oder einer Node.JS-Umgebung ist die Einrichtung daher extrem einfach: Programm auf den Server legen und ausführen. Auch Updates gelingen ohne Stress, weil man nur diese eine Datei austauschen muss. Wenn Sie Ihr Go-Programm in einen Container verpacken und in der Cloud ausführen wollen, können Sie einen minimalen Docker-Container bauen: Statt das Dockerfile mit einem großen Base-Image zu beginnen, starten Sie einfach mit `FROM scratch` . Außer Ihrem kompilierten Go-Binary muss der Container nichts weiter enthalten. Weil der Container nicht größer ist, als er unbedingt sein muss, ist Go eine gute Wahl für Ihre eigene Cloud-Anwendung. (jam) c't



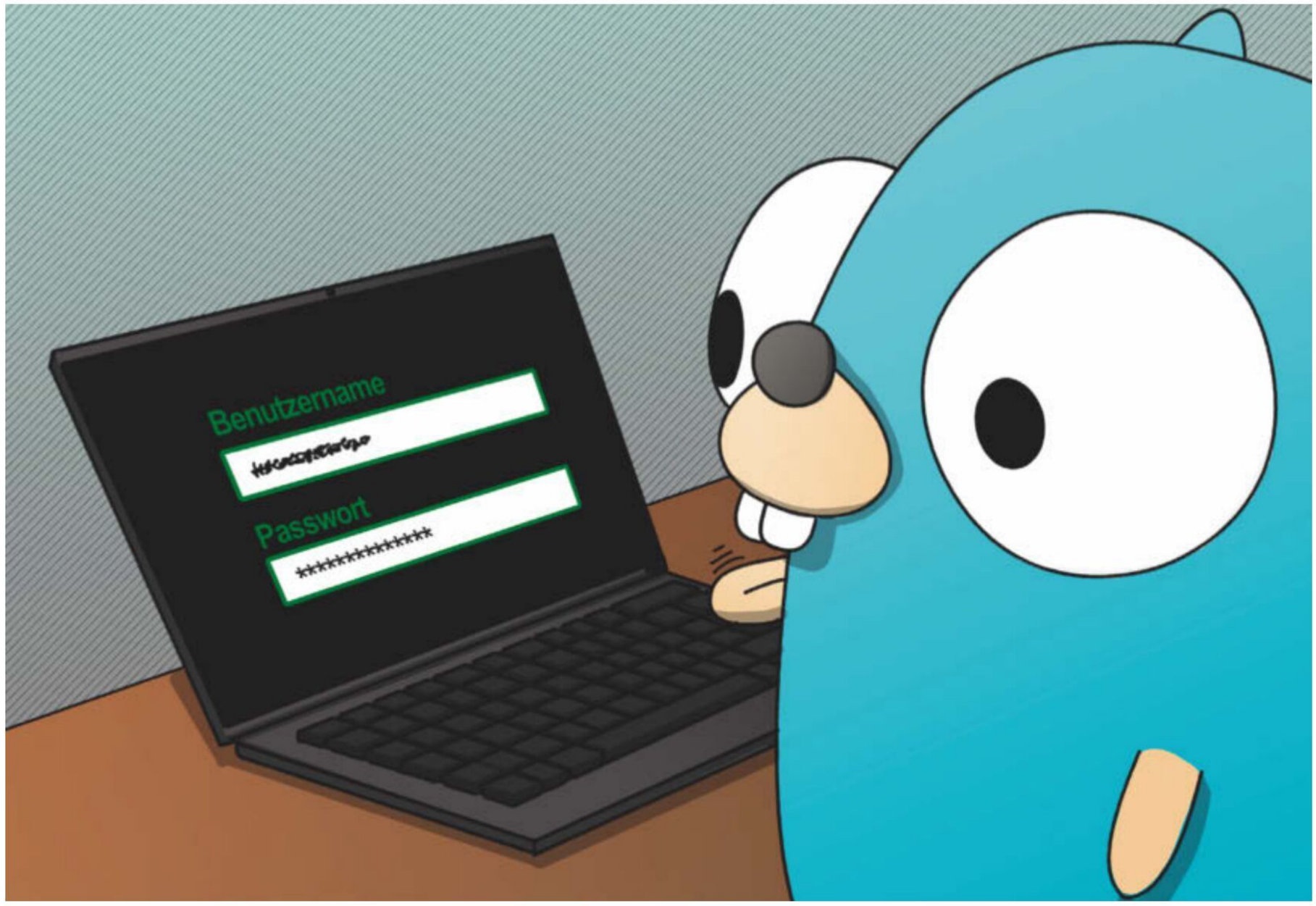


Bild: Thorsten Hübner

Go-Einstieg: Zugangsdaten

Viele Programme erfragen Anmeldedaten vom Benutzer, um sich an einem Server anzumelden. Oft landet das Kennwort dann im Klartext in einer Datei auf der Festplatte – dabei ist es kein Hexenwerk, den sicheren Kennwortspeicher des Betriebssystems zu nutzen. Mit Go gelingt das plattformübergreifend.

Von **Jan Mahn**

Die „schönsten“ Probleme beim Programmieren sind solche, die man vorm Programmieren noch gar nicht hatte. Eigentlich hatte man sich mit einem ganz anderen Ziel an den Computer gesetzt und verbringt dann die meiste Zeit mit einer ganz neuen Baustelle. Für manche Probleme gibt es nämlich ästhetische Lösungen, deren Schönheit darin besteht, dass der Nutzer von ihnen nichts mitbekommt. Die Gewissheit, eine wirklich schöne Lösung gefunden und umgesetzt zu haben, muss als

Lohn für die Arbeit dann ausreichen. Wenn man die Welt dabei noch ein Stückchen sicherer gemacht hat, ist das befriedigend genug.

Das sichere Speichern von Zugangsdaten in einer Anwendung ist ein gutes Beispiel, das eine schöne Lösung verlangt. Auf die Existenz dieses Problems stieß ich natürlich nebenbei: Ein kleines Kommandozeilenwerkzeug sollte her, mit dem Kollegen und ich per HTTPS auf ein internes API im Verlag zugreifen können. Eigentlich keine große Sache: Meine neue Lieblingssprache Go bringt alles mit, um eine Kommandozeilenanwendung zu bauen und für alle gängigen Betriebssysteme zu kompilieren. Unter Windows, Linux und macOS sollte das Programm laufen, damit möglichst viele Kollegen in den Genuss kommen können. Die ersten Schritte waren auch schnell erledigt, die Hülle einer Kommandozeilenanwendung mit Go und dem CLI-Framework Cobra finden Sie ab Seite 142. Das Cross-Compiling erledigt Go voller Eleganz und schnell hatte ich auf dem Mac drei Binärdateien für die drei Betriebssysteme. Auch das Abrufen von Informationen aus einem HTTP-API ist mit den Bordmitteln von Go keine große Herausforderung.

Doch dann musste ich mich am API-Server anmelden, ganz klassisch per Benutzername und Kennwort. Irgendwie widerstrebte es mir, auch zu Testzwecken, mein Kennwort mal kurz in den Code einzubauen. Allzu schnell landen solche Experimente versehentlich in der Versionsverwaltung und dann schlimmstenfalls bei GitHub. Ein vermeidbarer Fehler. Eine ästhetische Login-Routine musste her, bevor ich den Rest der eigentlichen Anwendung bauen konnte.

Was andere machen

Das Rad neu zu erfinden, war in der Geschichte der Softwareentwicklung nur selten eine gute Idee. Also sah ich mir an, was andere getan haben. Das Docker-CLI zum Beispiel ist ausreichend weit verbreitet, um als Anhaltspunkt zu dienen. Ich dachte mir: „Was die gebaut haben, kann nicht ganz falsch sein.“ Die Anmeldung an einer Container-Registry funktioniert dort mit der Syntax:

```
docker login <Adresse des Servers>
```

Danach wird man nach Benutzername und Kennwort gefragt. Hat die Anmeldung am Server funktioniert, muss man diesen Schritt auf der Maschine auch nicht mehr erledigen, die Daten werden also irgendwo gespeichert – doch wo eigentlich genau?

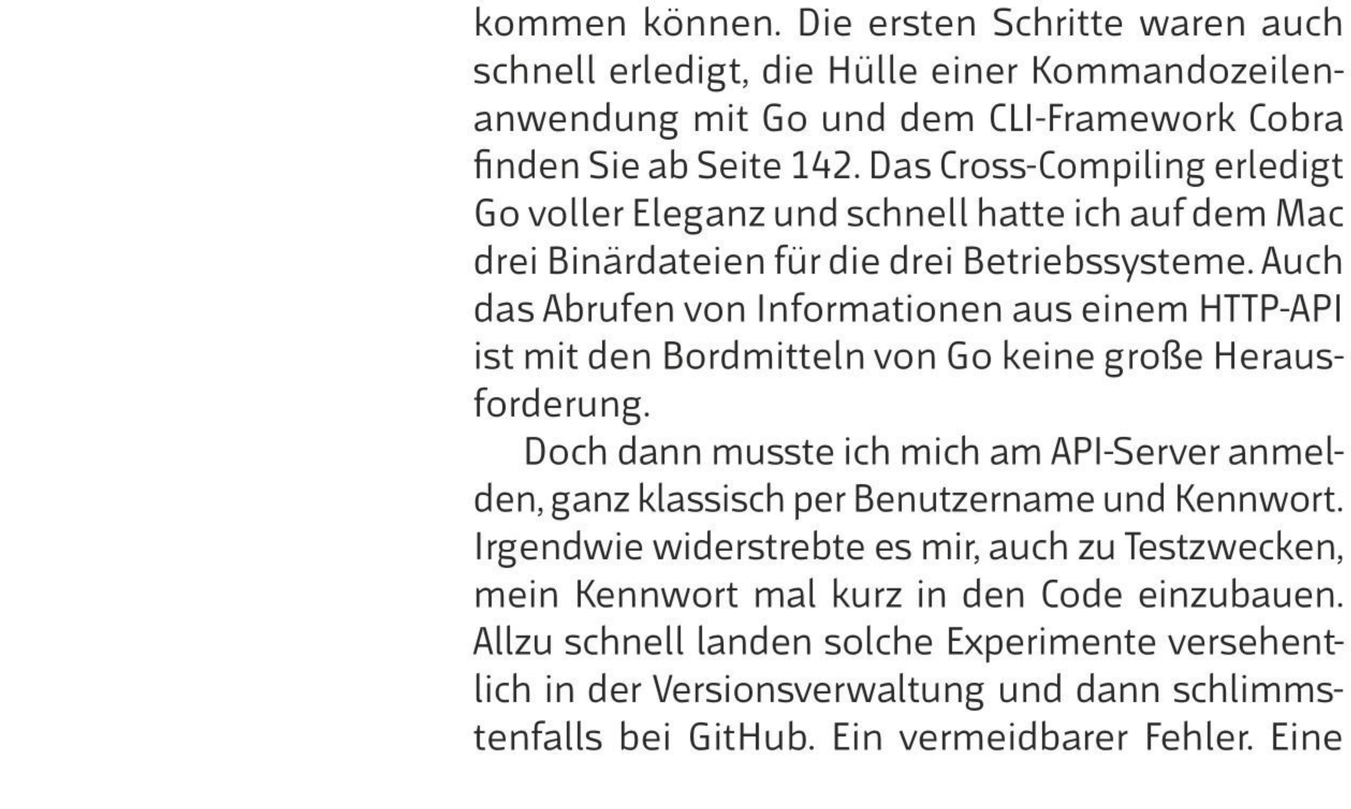
Um das herauszufinden, schaute ich mir die drei gängigsten Betriebssysteme einmal an, zunächst Linux mit Gnome Desktop und einen Ubuntu-Server ohne GUI. Fündig wurde ich jeweils im Ordner .docker im Home-Verzeichnis. Dort legt Docker die Datei config.json an:

```
"auths": {
 "docker.pkg.github.com": {
 "auth":<Name & Passwort in Base64>
 }
}
```

Für jede Registry, an der ich mich anmeldete, landete das Kennwort in dieser JSON-Datei, konvertiert als Base64-String. Base64 ist natürlich keine Verschlüsselung, sondern nur ein anderer Zeichensatz und leicht rückkonvertierbar. Nach dem sicherheitstechnisch letzten Schrei sah mir das noch nicht aus. Wer an die Datei kommt, hat das Kennwort. Und nicht jeder Nutzer hat eine Festplattenverschlüsselung aktiviert.

Ich wechselte den Computer: Auch auf dem Mac legt Docker die Datei config.json an, dort allerdings ohne das Kennwort. Stattdessen fand ich dort den Eintrag `"credsStore": "desktop"`. Docker Desktop wählt hier eine schönere Alternative zur Klartextdatei im Dateisystem: die betriebssystemeigene Ablage für Geheimnisse.

Unter macOS heißt sie Schlüsselbundverwaltung und verwaltet neben Kennwörtern von Programmen



Unter macOS landen die Kennwörter in der Schlüsselbundverwaltung.

auch Zertifikate, die Kennwörter des Apple-Browsers Safari und WLAN-Schlüssel. Die grafische Oberfläche finden Sie über die Spotlight-Suche mit dem Suchbegriff „Schlüsselbundverwaltung“. Dort fand ich auch meine Docker-Zugangsdaten, gesichert mit meinem Benutzerkennwort. Zum Verschlüsseln nutzt macOS Crypto-Hardware wie den Apple T2, der in neuen Geräten verbaut ist. Mac-Anwendungen gebrauchen diesen Tresor sehr zahlreich, die Liste ist gut gefüllt.

Unter Windows gibt es ebenfalls eine ins Betriebssystem integrierte Anmeldeinformationsverwaltung, die aber eher ein Nischendasein fristet. Die grafische Oberfläche findet man mit dem Suchbegriff „Anmeldeinformationsverwaltung“ in der Suchleiste oder über die Systemsteuerung. Auch Windows kann ein TPM (Trusted Platform Module), also die Crypto-Hardware des Rechners, zum Verschlüsseln benutzen. Auf meinen Testgeräten war die Liste der Kennwörter, die andere Anwendungen hier abgelegt hatten, aber sehr überschaubar. Auch weit verbreitete Software wie die Spotify-App oder Thunderbird machen offensichtlich keinen Gebrauch von diesem Speicher.

Zurück zur Linux-Maschine mit dem Gnome-Desktop: Dort gibt es „GnomeKeyring“, eine Programmsammlung, die die Spezifikation des „DBus Secret Service API“ umsetzt. In vielen Distributionen ist die grafische Oberfläche Seahorse für diese Schlüsselverwaltung bereits installiert. Wenn nicht: Das Paket heißt `seahorse`. Die Schlüsselsammlung bewahrt GnomeKeyring verschlüsselt auf, wenn sie nicht benutzt wird. Ein Blick in das Seahorse-GUI zeigt, ähnlich wie auch unter Windows: Viele Programme machen noch keinen Gebrauch von dieser Funktion.

Solange der Nutzer, dem die Geheimnisse gehören, nicht angemeldet ist, werden sie verschlüsselt gespeichert, sofern vorhanden kommt die Crypto-Hardware zum Einsatz. Beim Anmelden am Betriebssystem werden sie mit dem Nutzerkennwort entschlüsselt.

Mein Entschluss stand fest: Diese drei Ablageorte für Kennwörter sollte mein kleines Werkzeug benutzen – für andere Betriebssysteme (zum Beispiel Linux mit KDE oder Server-Linux ohne grafische Oberfläche) kann man später immer noch eine Fallback-Lösung mit einer JSON-Datei einbauen.

```
func login(server string) error {
 reader := bufio.NewReader(os.Stdin)
 fmt.Print("Enter Username: ")
 user, _ := reader.ReadString('\n')
 user = strings.Replace(user, "\n", "", -1)
 if len(user) < 2 {
 fmt.Println("no username entered")
 return nil
 }

 service := "secret-api"
 pass := ""
 pass, err := keyring.Get(service, user)
 if err != nil {
 fmt.Print("Enter Password: ")
 bytePassword, err := terminal.ReadPassword(int(syscall.Stdin))
 if err != nil {
 fmt.Println("no password entered.")
 }
 pass = string(bytePassword)

 err = keyring.Set(service, user, pass)
 if err != nil {
 fmt.Println("could not save to keychain.")
 return nil
 }
 }

 fmt.Println("we have a password")
 fmt.Println(pass)
 // Hier könnte die eigentliche HTTP-Operation folgen
 return nil
}
```

Mit go-keyring speichern Sie Kennwörter im Kennwortspeicher des Betriebssystems.

Eine Bibliothek für alle

Die drei Schnittstellen für die Kennwortspeicher von Windows, macOS und Gnome Desktop wollte ich ungern zu Fuß bedienen und war froh, dass man beim Versandhändler Zalando offenbar nicht nur Schuhe in Postpakete verpackt, sondern auch Code in Open-Source-Pakete. Für das Verwalten von Kennwörtern mit Go haben sie das Paket `go-keyring` entwickelt und auf GitHub veröffentlicht (Link via ct.de/w6gc).

Die Bibliothek besteht im Kern aus zwei Funktionen, die im Hintergrund das Betriebssystem erkennen und dann über eine geeignete Schnittstelle mit dem Kennwortspeicher interagieren. Ein Kennwort ist mit nur drei Zeilen im internen Kennwortspeicher abgelegt:

```
err := keyring.Set(name, user, pass)
if err != nil {
 log.Fatal(err)
}
```

Als ersten Parameter erwartet `keyring.Set()` den Namen der Anwendung, unter dem das Kennwort gespeichert werden soll. Diesen Namen kann der Entwickler vergeben, zum Beispiel `secret-api`. Wenn Sie diese paar Zeilen in ein minimales Go-Projekt eingebaut und ausgeführt haben, sollten Sie im GUI Ihres Betriebssystems einen Eintrag mit dem Namen finden.

Mit der Go-typischen Fehlerbehandlung, also einer Variablen `err` für den Fehler, kann man zum Beispiel erkennen, dass das Betriebssystem keinen passenden Kennwortspeicher mitbringt, und hier eine Alternative einbauen. Ist `err` nicht `nil`, hat etwas nicht funktioniert.

Auch das Auslesen eines Eintrags ist schnell erledigt, man braucht nur den Namen des Programms und den des Benutzers:

```
secret, err := keyring.Get(name, user)
if err != nil {
 log.Fatal(err)
}
log.Println(secret)
```

Der Workflow

Mit diesen Schnipseln ist eine Funktion schnell gebaut, die den Nutzer nach dem Kennwort fragt und



es im Speicher ablegt. Den ganzen Code inklusive der CLI-Hülle finden Sie über ct.de/w6gc bei GitHub. Auf Seite 168 finden Sie eine Funktion, die als Inspiration für eigene Abläufe dienen kann.

Ein paar Teilprobleme muss man zusätzlich zum Speichern noch lösen. Um den Benutzernamen vom Nutzer zu erfragen, braucht man einen Lesepuffer, den `bufio.NewReader()` erzeugt. Die Methode `ReadString('\n')` liest dann alle Eingaben bis zum Drücken der Enter-Taste. Nachdem man den Zeilenumbruch abgeschnitten hat, bleibt der Benutzername übrig.

Das Kennwort sollte man nicht auf demselben Weg erfragen, schließlich soll es beim Tippen nicht lesbar auf der Kommandozeile landen. Schöner funktioniert das mit `ReadPassword()` aus dem Paket `"golang.org/x/crypto/ssh/terminal"`.

Feinschliff

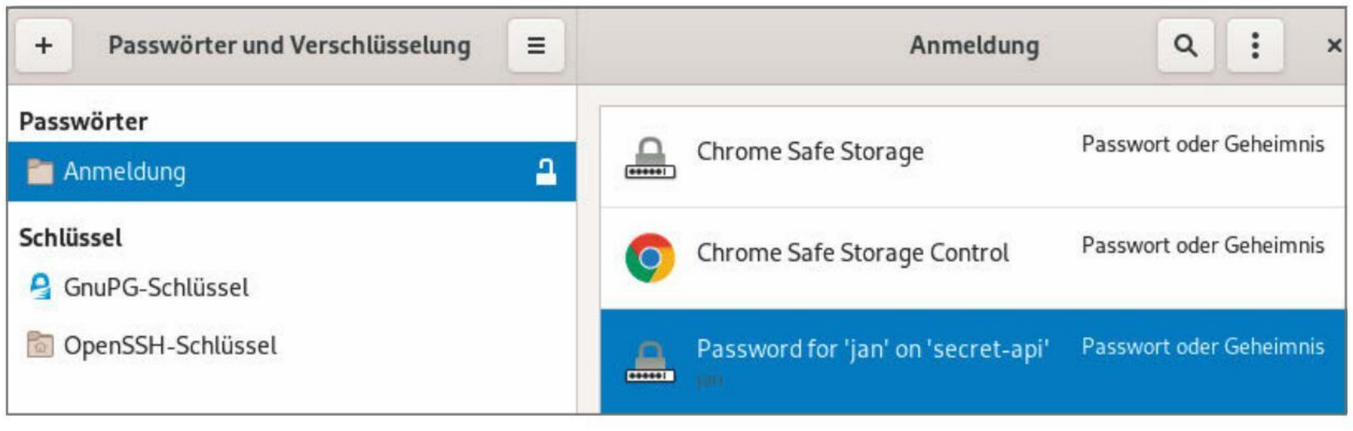
Neugierig kompilierte ich das Programm für alle Ziel-Betriebssysteme. Eine kleine Skriptdatei erleichterte diese Arbeit ungemein:

```
export GOARCH=amd64
export GOOS=darwin
go build -o ./build/mac/secret-api
export GOOS=linux
go build -o ./build/linux/secret-api
export GOOS=windows
go build -o ./build/win/secret-api.exe
```

Die Kompilate beförderte ich auf meine Testmaschinen und startete sie auf der Kommandozeile mit `secret-api login example.org`. Beim ersten Durchgang wurde ich nach dem Benutzernamen und auch nach dem Kennwort gefragt, beim zweiten Durchgang mit demselben Benutzernamen kam das Programm ohne diese Frage aus und bezog die Information aus dem Kennwortspeicher. Für jeden neuen Benutzernamen entsteht ein weiterer Eintrag. Möchte man dem Nutzer erlauben, sich bei verschiedenen Servern anzumelden, kann man die Serveradresse zum Beispiel an den Nutzernamen anhängen:

```
user = user + "@" + server
```

Auf allen Testumgebungen funktionierte das Verfahren wie geplant, in den GUIs fand ich meine gespeicherten Kennwörter jeweils wieder. Die Entwickler der Bibliothek weisen noch darauf hin, dass unter Linux eine Sammlung mit dem Namen `login` existieren muss und man diese möglicherweise zuerst über Seahorse anlegen muss. Auf allen gängigen Linux-Distributionen ist diese aber von Haus aus angelegt und einen Dialog zum Anlegen neuer Sammlungen fand ich auch nicht. Der Hinweis ist offenbar überholt.



Unter Gnome bekommt man mit Seahorse eine Oberfläche, um die gespeicherten Geheimnisse zu verwalten.

Die weiteren Schritte hängen stark vom konkreten Anwendungsfall ab. Bevor man ein neues Kennwort in den Tresor legt, kann man zum Beispiel prüfen, ob es auch beim Server funktioniert. Wie man das bewerkstelligt, kommt auf den Server an. In meinem Fall musste ich Benutzernamen und Kennwort per POST-Request als Formulardaten verschicken und bekam im Erfolgsfall ein Cookie zurück, im Fehlerfall einen HTTP-Fehlercode. Arbeitet man mit einem HTTP-API, bekommt man meist ein Token in einem JSON-Objekt ausgehändigt, das man an zukünftige Anfragen anhängen muss. Je nach Bedarf kann man `go-keyring` dann gleich noch einmal benutzen und das Cookie oder den Token-Inhalt in einem weiteren Eintrag speichern. Alle Funktionen, die später einen Request ausführen sollen, würden es immer zunächst mit diesem Token probieren. Ist dieser nicht mehr gültig, rufen sie eine Login-Funktion auf, die mit den Zugangsdaten aus dem Tresor ein neues Token beschafft und speichert.

Gutes Gefühl

Mein eigentliches Problem, das Auslesen unseres internen API, hatte ich damit noch nicht gelöst, die Anmeldeprozedur mit den Kennwortspeichern gefiel mir aber – vor allem, weil das Ergebnis wirklich plattformübergreifend funktioniert. Die Binärdateien haben keine zusätzlichen Abhängigkeiten und nutzen die Fähigkeiten des jeweiligen Betriebssystems aus. Und lehrreich war der Exkurs allemal. (jam) c't

Code des Beispiels

ct.de/w6gc

Ihr Erste-Hilfe-Set:

Das Notfall-System für den Ernstfall

Heft + PDF mit 29 % Rabatt



Mit der neuen Version **c't Desinfec't 2021/22** sind Sie für den Ernstfall bestens gerüstet:

- ► Vier der neuesten Viren-Scanner
- ► Notfallsystem, das Ihre Daten retten kann
- ► Notfallarbeitsplatz
- ► Verbesserte Hardware-Kompabilität

Heft für 14,90 € • PDF für 12,99 € • Bundle Heft + PDF 19,90 € • Desinfec't-Stick 19,90 €

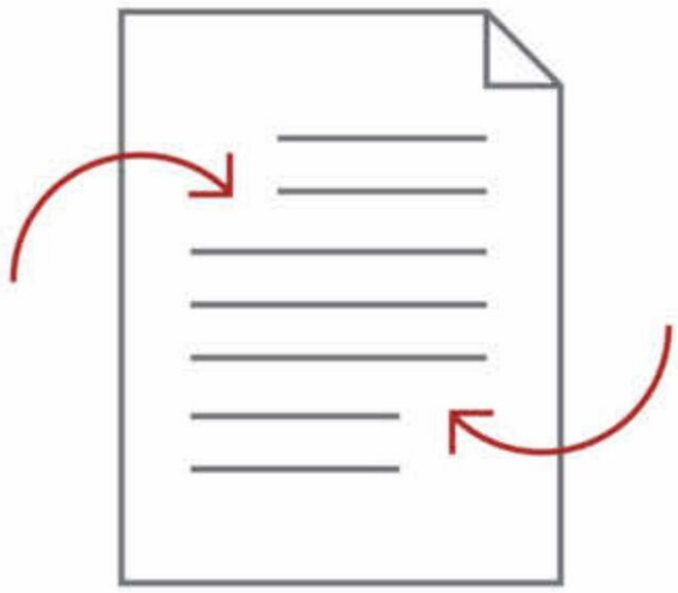
shop.heise.de/desinfect21

Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 €.
Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise Shop

Jetzt neu mit
Videokonferenz
Funktion!

NETFILES



netfiles

Sichere Cloud-Lösungen für Datenaustausch und Collaboration

Einfach

Der netfiles Datenraum ist besonders einfach zu bedienen, bietet umfangreiche Funktionalität und steht Ihnen sofort, ohne Installation von Software oder Plugins zur Verfügung. Ein Webbrowser genügt.

Sicher

Im netfiles Datenraum sind Ihre Daten sowohl bei der Speicherung als auch Übertragung durch 256-bit Verschlüsselung sicher und Compliance-gerecht geschützt.

Bewährt

netfiles gibt es seit mehr als 20 Jahren. Profitieren auch Sie von unserer langjährigen Erfahrung und dem zuverlässigen Betrieb. Wir sind ein deutsches Unternehmen und hosten ausschließlich in Deutschland.

www.netfiles.com

Testen Sie jetzt netfiles 14 Tage kostenlos
oder vereinbaren Sie einen Termin für eine Online-Präsentation.

netfiles GmbH · Marktler Str. 2b · 84489 Burghausen · +49 8677 915 96-12 · vertrieb@netfiles.de