Editorial

Liebe Leserinnen und Leser,

Container und Cluster, Kubernetes und Docker, Images und Volumes: Viel Fremdes begegnet einem beim Abtauchen in die Welt des modernen Software-Deployments. Das ist – wie so oft – anstrengend, aber sehr lohnenswert.

Mit diesem Heft geben wir Ihnen nicht nur die Begrifflichkeiten und Konzepte an die Hand, um die populäre Containertechnik zu verstehen, sondern zeigen ganz praktisch, wie Sie damit arbeiten und sich das Leben vereinfachen. Ein Container ist schnell gestartet, aber die wahre Magie der Technik entfaltet sich mit seiner Zerstörung: Denn einerseits bleiben keine Spuren auf dem System zurück und andererseits geht trotzdem nichts von Wert verloren. Eine einmal definierte Container-Umgebung kann man jederzeit wiederherstellen.

Aus dieser Erkenntnis ergeben sich vielfältige Möglichkeiten. Updates und deren Tests verlieren ihren Schrecken. Hakt etwas, kann man jederzeit auf die vorherige Version der Software zurückwechseln. Auf dem Produktivsystem läuft nicht das gleiche Betriebssystem wie auf der Maschine des Entwicklers? Macht nichts, denn ein Container sieht immer gleich aus – ganz egal, wo er läuft. Der Root-Server ächzt unterm Kundenansturm? Kubernetes hilft und verteilt Lasten auf zahlreiche Maschinen oder schiebt die Infrastruktur gleich in eine kommerzielle Cloud.

All das klingt nach mehr Pflegebedarf als herkömmliche Admin-Arbeit, ist aber meist mit nur einer Änderung einer Konfigurationsdatei oder einem Kommandozeilenbefehl getan. Container machen vieles einfacher.

Lassen Sie sich bei der Dockerei nicht entmutigen, wenn etwas nicht klappt, denn: „Es ist noch kein Meister vom Himmel gefallen“, wusste schon der Lateiner.

Jan Mahn, Merlin Schumacher, Peter Siering

Inhalt

EINFÜHRUNG

Die Container-Technik, Docker und Kubernetes verändern den Umgang mit (Server-)Software. Von der neuen Flexibilität profitieren Unternehmensumgebungen genauso wie der eigene Heim- oder Webserver oder die Hausautomation.

HARDWARE

Nicht nur auf x86- und Root-Servern ist Docker zuhause: Wir geben Tipps zum Umgang mit Docker für den Raspi und in der Cloud und zeigen, wie man eigene Multi-Architektur-Images baut.

PRAXISWISSEN

Container sind im Handumdrehen vernetzt. Mit Docker-Compose baut man komplexe Infrastrukturen auf. Dank moderner Docker-GUIs findet Container-Management längst nicht nur auf der Kommandozeile statt.

PROJEKTE

Fremde Images wollen genau beäugt werden. Wir haben eine Auswahl von Container-Images zusammengestellt, die viel Arbeit ersparen, durchdacht sind und gut gepflegt werden. Einige stellen wir im Detail vor.

ENTWICKLER

Die Cloud lässt sich auch nutzen, um den Bau von Images zu automatisieren. Wer seine Projekte lieber lokal verwaltet, kann mit GitLab oder GitHub Actions und Docker eine eigene Softwarefabrik hochziehen. Verpackt man fertige Projekte in ein Docker-Image, verlieren Versionsstände und Paketabhängigkeiten ihren Schrecken.

ZUM HEFT

Alle ScreenCasts zu c't Docker & Co. finden Sie auf ct.de/wgcz

c't Hands on!

ScreenCast
ct.de/w8gh

Warum Container?

Die Container-Technik und Docker verändern den Umgang mit (Server)-Software. Von der neuen Flexibilität profitieren Unternehmensumgebungen genauso wie der eigene Heim- oder Webserver oder die Hausautomation.

Von Jan Mahn, Merlin Schumacher und Peter Siering

Software-Container machen den Test und den Betrieb von Server-Diensten einfach: Ein Container enthält eine Software mit all ihren Abhängigkeiten. Gestartet wird ein solcher Container aus einem Abbild, dem Container-Image. Er verhält sich auf jedem Gerät exakt gleich – auf der Entwicklermaschine genauso wie auf dem Root-Server oder beim Cloud-Anbieter. Die Software im Container merkt nicht, dass sie im Container steckt – für sie sieht ihre Umgebung wie eine eigene Maschine aus. Löscht man einen Container, hinterlässt er keine Spuren, keine Konfigurationsreste im Betriebssystem. Das sind die Versprechen, mit denen eine Container-Umgebung wie Docker Administratoren und Entwickler überzeugen will.

Damit Container die Arbeit wirklich erleichtern, sollte man die zugrunde liegenden Ideen und Begrifflichkeiten verstanden haben. Dieses Grundwissen liefern dieser und der folgende Artikel. Ein wichtiges Paradigma der Docker-Welt lautet: ein Dienst pro Container. Man wirft also nicht etwa Datenbank und Webserver in dasselbe Image, um eine Web-

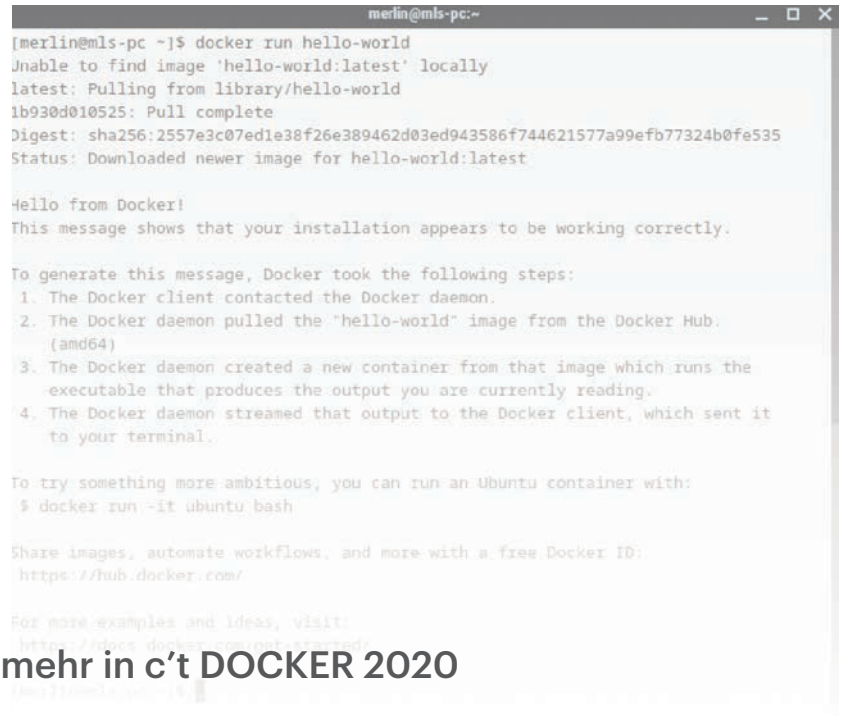
Anwendung zu starten. Vielmehr stecken sowohl Datenbank als auch Webserver in ihren eigenen Images.

Grundbegriffe

Die wichtigste Information vorab: Docker-Container sind keine virtuellen Maschinen. Jeder laufende Container ist ein eigener Prozess innerhalb des Host-Betriebssystems. Hier gibt es keinen Hypervisor und keine virtuelle Hardware, auf der ein weiteres vollständiges Betriebssystem läuft. Die Container und das Host-Betriebssystem greifen auf denselben laufenden Linux-Kernel zurück. Das sieht man deutlich bei einem Blick in die Prozessliste des Host-Betriebssystems: Dort werden die in den Containern laufenden Prozesse genauso aufgelistet wie solche, die nicht in Containern stecken. Anders als eine virtuelle Maschine beansprucht ein Container also keine Ressourcen, um ein vollständiges Gastbetriebssystem laufen zu lassen. Container sind „nur“ durch Namespaces voneinander getrennte Prozesse – interessant wird Docker durch das Drumherum, also die Verwaltung von Images, Netzwerken und Volumes. Aber der Reihe nach.

Basis eines jeden Docker-Containers ist ein Docker-Image. Ein solches Image beinhaltet nur die grundlegenden Programme, Bibliotheken und Daten. Aus einem Image lassen sich beliebig viele Container erzeugen. Es ist also wie ein Musterhaus: Niemand wohnt darin und es hat nur eine grundlegende und unpersönliche Einrichtung. Erst das Wohnhaus, das auf Basis des Musterhauses gebaut wurde, wird bezogen und von seinen Bewohnern (den Prozessen) zum Eigenheim gemacht.

Images müssen Sie nicht immer per Hand selbst bauen. Dann wäre Docker nicht so erfolgreich geworden. Images entstehen aus einem Bauplan, dem Dockerfile. Es enthält Anweisungen, auf welchem anderen Image ein neues Image aufbauen soll, kopiert die Software und die Abhängigkeiten hinein und ergänzt Befehle, zum Beispiel Installationsskripte. Wer ein solches Dockerfile für eine Anwendung geschrieben hat, kann das Image in einer Registry veröffentlichen. Die größte Registry ist der Docker-Hub, betrieben von der Docker Inc., der Firma hinter der Software Docker. Wenn man für eine Aufgabenstellung ein fertiges Container-Image sucht, wird man dort aber schnell erschlagen. Im Docker-Hub finden sich schon für ein und dieselbe



Aus dem Image hello-world entsteht ein Container, der ein paar Zeilen auf

Lesen Sie mehr in c’t DOCKER 2020

ScreenCast
ct.de/waug

Container mit Compose einrichten

Docker-Compose erleichtert den Umgang mit komplexen Container-Umgebungen, die zusammenarbeiten sollen. Entwicklern und Administratoren nimmt das Werkzeug viel Arbeit ab und schafft eine Infrastruktur, die überall gleich funktioniert.

Von Jan Mahn

Hat man ein passendes, gutes und sicheres Container-Image für ein Problem gefunden (siehe S. 42), ist der Container mit Docker schnell gestartet: Mit `docker run <Name des Images>` beginnt der Docker-Daemon, das Abbild aus der Registry zu beschaffen und startet den Container. Mit allerlei Parametern kann man den Container jetzt nach eigenen Wünschen anpassen und zum Beispiel einen Namen vergeben, ein Volume anhängen oder einen Port des Containers auf dem Rechner verfügbar machen. Leider ergibt das schon bei einfachen Aufgaben wie einem Webserver, der statische Inhalte ausliefern soll, einen langen und schlecht lesbaren Befehl. Arbeiten mehrere Contai-

ner zusammen, die aufeinander angewiesen sind, ist eine Sammlung von Zeilen mit `docker run` nicht mehr praktikabel.

Mit Docker-Compose definieren Sie in einer Yaml-Datei, welche Container zusammenarbeiten sollen, wie sie heißen, welche Images sie verwenden und wie sie miteinander kommunizieren dürfen. Mit dem Befehl `docker-compose up` fährt die Komposition dann hoch – auf dem heimischen Entwicklungs-PC genauso wie auf dem Root-Server.

Klarmachen

Bevor es losgehen kann, benötigen Sie das Programm Docker-Compose. Windows- und macOS-Nutzer müssen nur Docker für ihr Betriebssystem installieren und bekommen Docker-Compose direkt mitgeliefert. Unter Linux müssen Sie es eigens installieren – und die Paketquellen sind nicht immer eine geeignete Anlaufstelle. So bietet Debian im Februar 2019 noch Docker-Compose 1.8.0 aus dem Sommer 2016 an, obwohl Version 1.23.2 aktuell ist. Weil sich in der Docker-Welt in zwei Jahren viel verändert hat, lohnt die Installation der neuen Version. Sie sollten daher auf den Installationsweg setzen, den Docker vorschlägt. Folgender Befehl, den Sie auch über ct.de/waug in der Docker-Dokumentation finden, lädt automatisch die richtige Version für Ihr Betriebssystem herunter:

```
sudo curl -L "https://github.com/␣
↳docker/compose/releases/download/␣
↳1.23.2/docker-compose-$(uname -s)␣
↳-$(uname -m)" -o /usr/local/bin/␣
↳docker-compose
```

Passen Sie die Versionsnummer an, wenn Sie diesen Befehl einige Monate nach Erscheinen des Hefts ausführen. Jetzt muss das Programm ausführbar gemacht werden:

```
sudo chmod +x /usr/local/bin/␣
↳docker-compose
```

Der Befehl `docker-compose --version` sollte jetzt eine Versionsnummer zurückgeben. Auf dem Raspberry Pi, der sich für Docker-Compose-Projekte zwar grundsätzlich eignet, funktioniert dieses Vorgehen aber nicht. Am einfachsten kommen Sie hier mit dem Python-Paketmanager pip zum Ziel, der bei Raspbian [illegible]

Schreibhilfe

Die Docker-Compose-Datei wird im Yaml-Format verfasst, das jeder Texteditor erzeugen kann – mit einer IDE können Sie sich das Leben aber deutlich erleichtern.

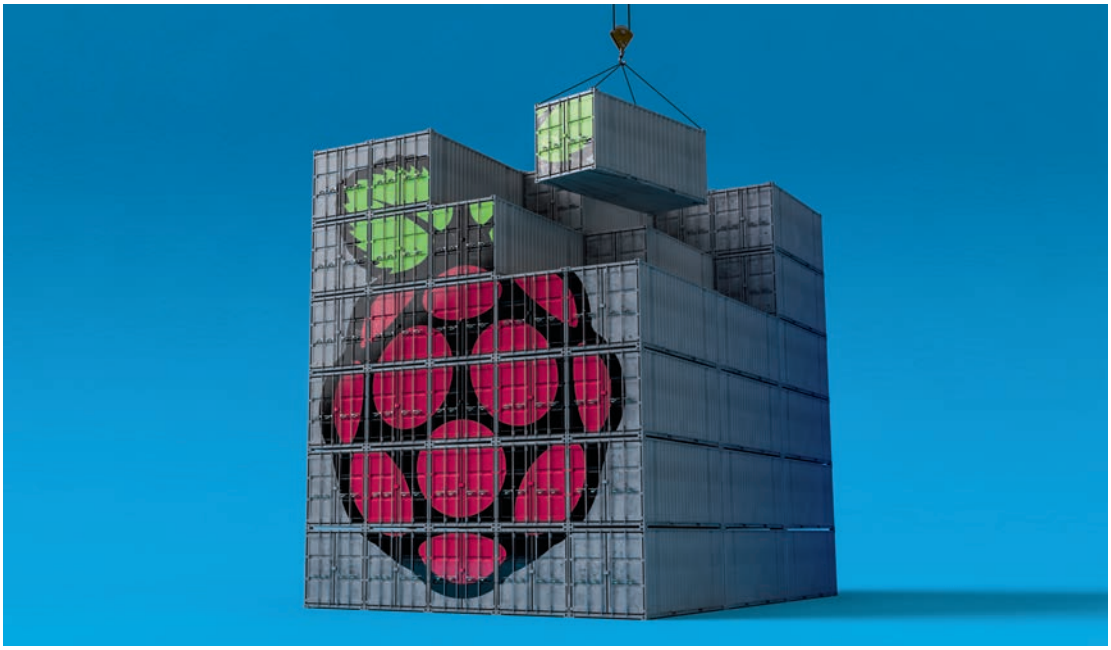
In Yaml wird die Struktur der Daten über Einrückungen festgelegt, eine Unterebene wird durch zwei Leerzeichen am Zeilenanfang eingeleitet. Hat Ihr bevorzugter Editor eine Möglichkeit, Leerzeichen sichtbar zu machen, sollten Sie diese Funktion nutzen. Empfehlenswert ist die Open-Source-IDE Visual Studio Code (siehe ct.de/waug). Mit Strg+Umschalt+P öffnen Sie die Befehlseingabe. Tippen Sie dort den Begriff „Rendern“ ein und wählen Sie „Ansicht: Rendern von Leerzeichen umschalten“. Leerzeichen sind jetzt nicht mehr unsichtbar. Wer viel mit Docker-Compose und Visual Studio Code arbeitet, kann sich zusätzlich die Erweiterung „Docker Compose“ installieren. Sie fügt eine kleine GUI für Docker-Compose in der Oberfläche der IDE hinzu, über die man einzelne Container inspizieren und verwalten kann.

Ladeliste

Für das erste Compose-Projekt legen Sie einen neuen Ordner an und erstellen hier die Datei docker-compose.yml. Der Ordnername landet später in den Containernamen, sollte also nicht zu lang sein.

Eine Compose-Datei beginnt immer mit der Angabe `version:`, die die Version der Compose-Definition angibt. Welche Version Sie verwenden können, hängt von der eingesetzten Docker-Version (nicht etwa der Docker-Compose-Version) zusammen. Wenn Sie jetzt einsteigen und die aktuelle Docker-Version 18.09 nutzen, können Sie direkt `version: 3.7` fordern.

Als Beispiel soll ein WordPress-Blog dienen. Damit das läuft, braucht es einen Container mit WordPress und einem Webserver – in diesem Fall Apache. Die Container werden im Abschnitt `services:` definiert. Der erste Service bekommt den Namen „wp“ und das Image „wordpress:4-php7.2-apache“. Zum Ablegen der Daten braucht WordPress eine MySQL- oder MariaDB-Datenbank. Sie bekommt den Namen „db“ und soll aus dem Image „mysql:5.7“ entstehen. Die Compose-Datei beginnt mit folgenden Zeilen:



Docker auf dem Raspberry Pi

Die Container-Software Docker unterstützt nicht nur x86-Systeme, sondern auch den Raspi. Inzwischen gibt es eine stattliche Anzahl von Containern für den kleinen Rechner, mit denen man ihn zum Familien-Wiki oder zur Smart-Home-Zentrale machen kann. Wir geben Tipps zum Umgang und zeigen darüber hinaus, wie man eigene Multi-Architektur-Images für PC und Raspi baut.

Von Merlin Schumacher

Entwickler und Unternehmen schätzen Software-Container schon seit Langem. Mit ihnen kann man im Handumdrehen Wordpress, MediaWiki, Node-Red und andere Dienste einrichten und konfigurieren. War Docker anfangs nur für x86-Prozessoren verfügbar, läuft es nun auf zahlreichen Architekturen, darunter auch ARM-CPUs des Einplatinenrechners Raspberry Pi. So kann man Dienste, die man noch argwöhnisch beäugt, problemlos und gut weggesperrt als Container auf den Raspi befördern. Oder man lässt unterschiedliche Versionen von Laufzeitumgebungen wie NodeJS nebeneinander laufen, ohne dass sie sich ins Gehege kommen.

Hallo Welt! Der erste Container ist schnell gestartet und gibt ein paar Basisinformationen zur Arbeitsweise von Docker aus.

```
pi@raspberrypi:~ $ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
61e750ce94d2: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499788
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm32v7)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Sollten Sie bereits mit Docker vertraut sein und sich nur für die Spezifika von Docker auf ARM-Architekturen interessieren, können Sie die nächsten Abschnitte überspringen und bei „Image-Suche“ weiterlesen.

Um den gesamten Artikel inklusive Erstellung von Containern, die sowohl für Raspi und PC geeignet sind, nachzuvollziehen, brauchen Sie einen Raspberry Pi 2 oder neuer und einen PC mit einer funktionierenden Docker-Installation. Falls Sie noch keine Docker-Installation haben und Debian, Ubuntu, Fedora oder CentOS einsetzen, können Sie die im Artikel für den Raspberry Pi empfohlenen Installationsschritte auch auf dem PC durchführen. Für macOS und Windows finden Sie auf der Docker-Homepage Anleitungen zur Installation. Den Link dazu und alle weiteren zum Artikel finden Sie über ct.de/wq9w.

Container-Crashkurs

Auch wenn es vielleicht auf den ersten Blick so erscheint: Container sind keine virtuellen Maschinen, sondern laufen als Prozesse wie ganz normale Software auf dem System, werden aber durch spezielle Kernel-Techniken von anderen Prozessen getrennt. Dadurch hat man praktisch keine Performance-Einbußen, aber dennoch voneinander sauber getrennte Dienste. In den Containern läuft typischerweise nur ein einziger Prozess. Wird dieser beendet, verschwindet damit auch der Container.

Container sind sogenannte Images

Docker-Distros für den Raspi

Wie beim Raspi üblich, haben sich schnell spezialisierte Distributionen für Docker entwickelt. Um den Raspi bemühen sich: Hypriot, resinOS und Rancher OS. Hypriot basiert auf Raspbian und bringt die notwendigen Docker-Werkzeuge gleich mit, Updates und Optimierungen gibt es aber relativ selten. Die weitere Konkurrenz hat einige Nachteile. So muss man resinOS mit einem speziellen Konfigurationswerkzeug anpassen. Die Zukunft von Rancher OS ist ungewiss, denn seit Version 2.0 setzt das darauf aufsetzende Container-Verwaltungstool Rancher auf die Container-Orchestrierung Kubernetes, und Rancher OS wurde noch nicht entsprechend angepasst.

Gut gepflegte Docker-Container

Gute Container-Images zu finden ist aufwendig, denn das Angebot ist fast unendlich und oft lösen sie nur Teilprobleme. Wir haben eine Auswahl von Images zusammengestellt, die einem viel Arbeit ersparen, durchdacht sind und gut gepflegt werden.

Von Merlin Schumacher

Die sprichwörtliche Nadel im Heuhaufen ist nichts gegen das Finden eines guten Container-Images im Docker Hub. Meist erhält man hunderte Treffer, von denen keiner taugt: zu alt, voller Lücken, riesengroß. Oder man findet etwas, das auf den ersten Blick perfekt erscheint, aber genau das, was man wirklich braucht, nicht kann. Wie gut die Container ihren Job wirklich machen, zeigen sie leider oft erst im Praxiseinsatz. Wir haben bewährte Docker-Container aus unserem Alltag gesammelt

und stellen sie hier vor. Die Links zu den Images im Docker Hub finden Sie unter ct.de/wnky.

Unsere Auswahl ist zwar ganz konkret, aber auch exemplarisch zu sehen. So stehen Nextcloud oder WordPress als Beispiel für komplexere Container-Images, Tvheadend für den Umgang mit Hardware oder Google Pagespeed als Beispiel für das Kompilieren innerhalb von Images. Die Images dienen auch als Inspiration und Beispiele für eigene Dockerfiles und Container.

Eine konkrete Empfehlung ist Portainer als Web-Oberfläche, die einem das Hantieren auf der Kommandozeile erspart und einen schönen Überblick über den Zustand der eigenen Docker-Installation(en) bietet. Ebenso nützlich ist Watchtower, das die eigenen Container auf dem neuesten Stand hält.

Wachsames Auge

Die Empfehlungen haben wir selbstverständlich getestet und auf Sicherheitslücken geprüft, dennoch ist nicht jedes Image in sechs Monaten noch so sicher wie heute. Man sollte regelmäßig kontrollieren, ob die Maintainer und Entwickler ihre Docker-Images noch pflegen. Ist das nicht mehr der Fall, muss man sich nach einer Alternative umschauen oder selbst Hand anlegen. Das ist ja – Docker sei Dank – oft kein Problem.

Überhaupt ist es sinnvoll, sich eher auf Gruppen wie das Team von Linuxserver.io oder die offiziellen Docker-Pakete zu verlassen, denn im Zweifel schauen dort mehr Augen hin. Ein einzelner Entwickler, der seine Dockerfiles in der Freizeit pflegt und vielleicht irgendwann nicht mehr benötigt, kann Vergleichbares nicht leisten. Wer den ins Auge gefassten Images misstraut, sollte ihnen so zu Leibe rücken, wie es der Artikel ab Seite 42 empfiehlt.

Die Einschätzung des Risikopotenzials der Images ist hauptsächlich von deren Privilegien abhängig. Ein Zugriff auf einige wenige Geräte wie bei dem Tvheadend-Image ist keine unmittelbare Gefahr. Aber Lücken in den zugehörigen Gerätetreibern ermöglichen einem Angreifer aus dem Container heraus Zugriff auf das Hostsystem. Images, die das Management von Docker erleichtern, wie Watchtower oder Portainer, müssen praktisch immer auf den Docker-Socket schreiben können. Über diesen Socket kann ein Container das volle Root-

portainer.io

Docker-GUI

Grafische Oberflächen für Docker sind schon viele gekommen und gegangen – Portainer ist geblieben. Die üblicherweise selbst als Container eingerichtete Anwendung lässt im Browser eine umfassende Verwaltung sämtlicher Container-Bauteile zu: Images, Volumes, Netzwerke, Registries und dazu noch einige Spezialitäten, die den Betrieb eines Docker-Swarms erlauben, also eines Zusammenschlusses mehrerer Serversysteme zu einem Verbund. Mehrere Container bilden dann ähnlich wie beim Einsatz von `docker-compose` eine größere Anwendung. Für diese Betriebsart, die in Portainer nur im Swarm-Betrieb sichtbar ist, lassen sich Vorlagen definieren. Aber auch ohne derlei Möglichkeiten, die man eher im Dunstkreis von Kubernetes und ähnlichen Monstern vermuten würde, ist Portainer durchweg nützlich.

Portainer erlaubt das Anlegen von Benutzern und somit die Vergabe unterschiedlicher Rechte für einzelne Ressourcen, sodass ein Systemverwalter Aufgaben delegieren kann. Der Vollständigkeit halber seien auch die Standardaufgaben aufgezählt, die Portainer erledigt: Log-Dateien anzeigen, die Container-Konsole im Browser zugänglich machen, Konfigurationsdetails bearbeiten sowie statistische Daten sammeln und zeigen. Zur TLS-Absicherung bietet sich für Portainer-Nutzer ein separater Proxy wie Træfik an; geübten Admins genügt vielleicht auch ein SSH-Port-Forwarding.

portainer/portainer	
Offizielles Image	ja, vom Entwickler
Plattformen	amd64, armhf, arm64 u. v. a.
Risikopotenzial	mittel

Links zu den Containern:

Bild: Thorsten Hübner

Erste Schritte 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 20 Jobs parallel laufen lassen – davon höchstens fünf auf macOS. Letzteres ist wohl den höheren Hosting-Kosten von macOS geschuldet, da Apple keinerlei Serverhardware mehr liefert. Laufen die Jobs länger als sechs Stunden, bricht GitHub sie ab. Wer zahlt, erhält größere Kontingente. Eine Übersicht finden Sie in der Tabelle auf Seite 165.

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/w9ky. Wenn in einem Runner Software fehlt, können Sie fehlende Abhängigkeiten nachinstallieren.

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.

```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!“.

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/w9ky).

GitHub-Tarife und Action-Jobs

Tarif	maximale Jobs	davon macOS
Free	20	5
	40	5