2019

c't wissen

DOCKER

mit Screencasts

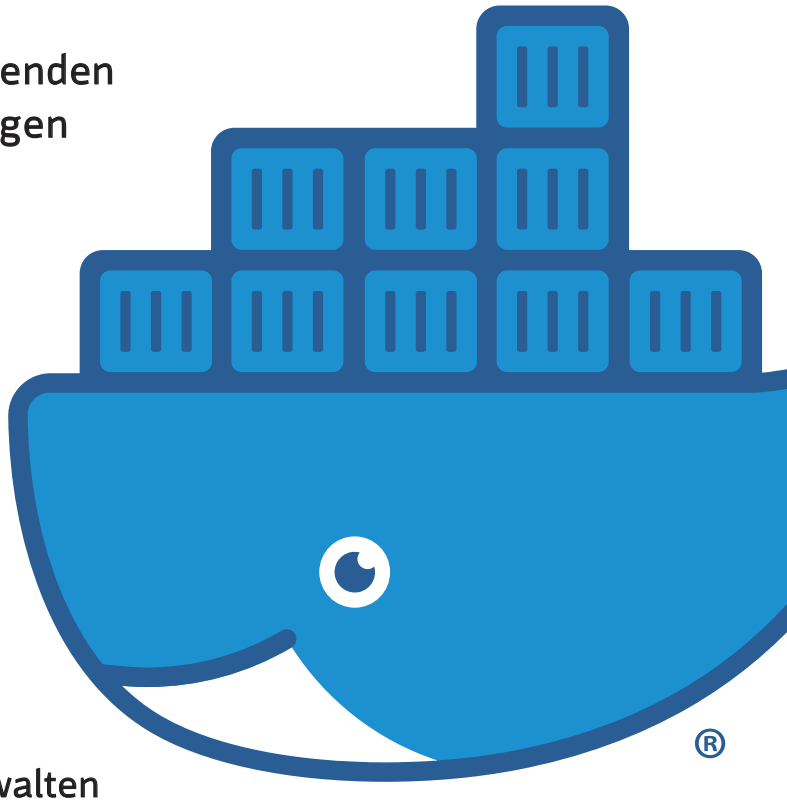
Komplexe Software einfach einrichten

Docker-1×1

Container-Konzept verstehen und anwenden
Dienste einrichten, verwalten und pflegen

Tipps & Workshops

Wie Einsteiger profitieren
Wie Entwickler eigene Images bauen
Wie Admins gute Images erkennen



Nützliche Helfer

Compose: Container verknüpfen
Swarm: Anwendungen skalieren
Portainer & Co: Infrastruktur grafisch verwalten

Container in der Praxis

Smart-Home-Zentrale auf dem Raspi aufsetzen
Die eigene Cloud auf dem NAS betreiben
Software-Projekte: Node.js und PHP wegsperren

www.ctspecial.de

Heise

Inhalt

EINFÜHRUNG

Container-Technik und Docker erfordern einen veränderten Umgang mit (Server)-Software. Erfahren Sie, wie Sie die Container-Plattform auf dem Server oder PC installieren und weshalb es wichtig ist, fremde Images sorgfältig zu prüfen.

PRAXISWISSEN

Container sind im Handumdrehen vernetzt. Mit Docker-Compose arbeiten Container verteilt über mehrere Systeme. Dank moderner Docker-GUIs findet Container-Management längst nicht nur auf der Kommandozeile statt.

HARDWARE

Auch auf dem NAS und Raspberry Pi ist Docker zuhause: So werden Sie mit der eigenen Nextcloud-Instanz auf dem NAS unabhängig von Google & Co. Wir geben Tipps zum Umgang mit Docker für den Raspi und zeigen, wie man eigene Multi-Architektur-Images baut.

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 und Docker eine eigene Softwarefabrik hochziehen. Verpackt man fertige Projekte in ein Docker-Image, verlieren Versionsstände und Paketabhängigkeiten ihren Schrecken.

ZUM HEFT

ScreenCast
ct.de/w55q

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-

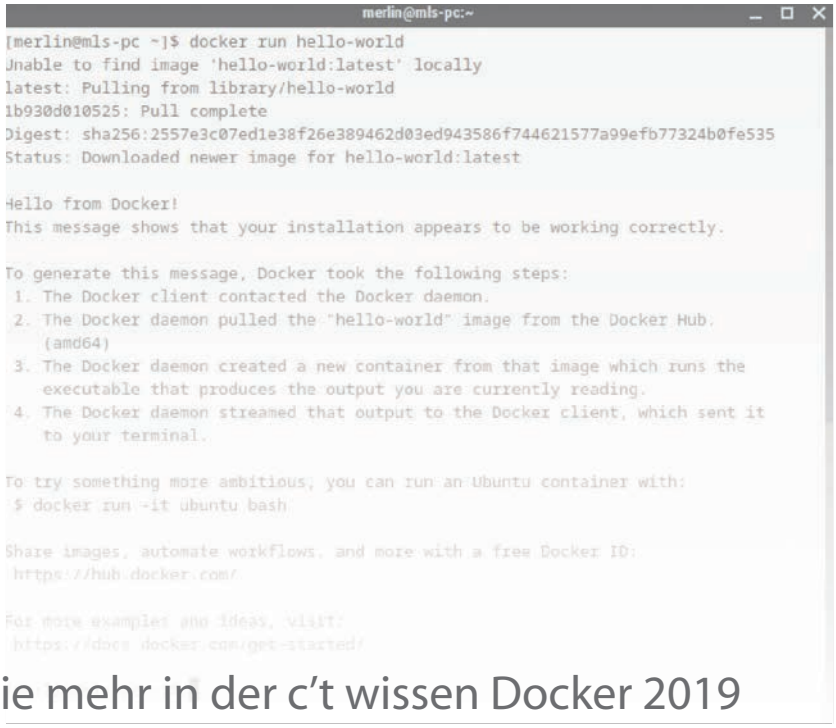
Anwendung zu starten. Vielmehr stecken sowohl Datenbank als auch Webserver in ihren eigenen Images.

Grundbegriffe

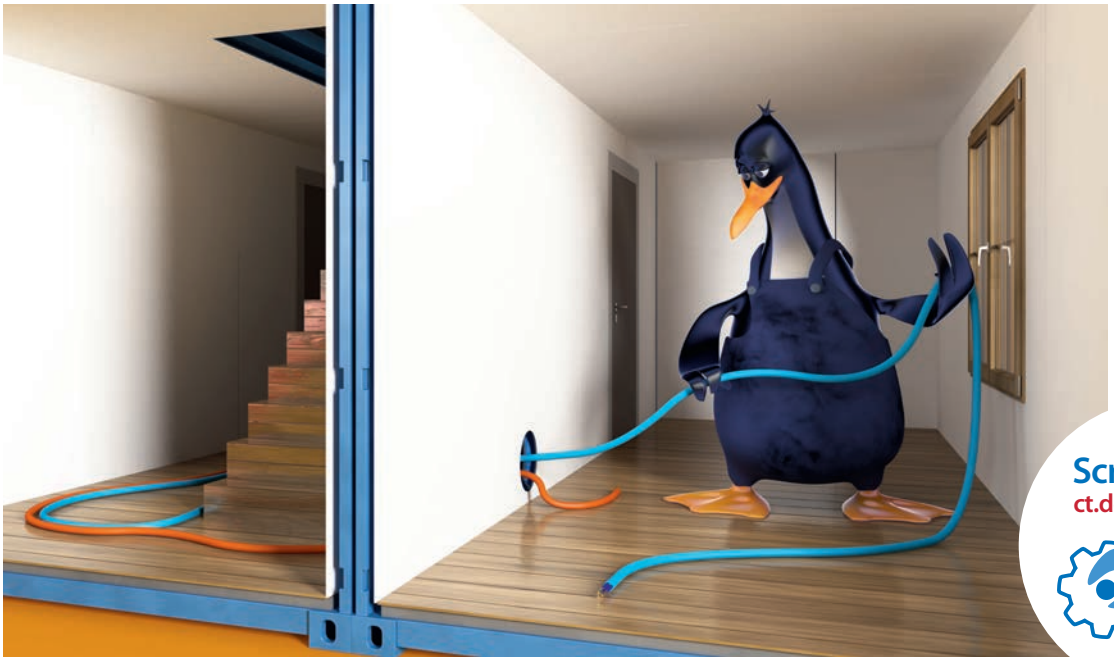
Die wichtigste Information vorab: Docker-Container sind keine virtuellen Maschinen. Jeder laufende Container ist ein eigener Prozess innerhalb des Host-Betriebssystems. Hier gibt es keinen Hypervisor und keine virtuelle Hardware, auf der ein weiteres vollständiges Betriebssystem läuft. Die Container 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 die



ScreenCast
ct.de/w14m

Container selbst vernetzen

Wurden Webserver und Datenbank in bester Docker-Manier in jeweils eigene Container verfrachtet, muss der Webserver neue Wege finden, auf die Datenbank zuzugreifen. Früher wurden solche Container miteinander verlinkt, doch diese Methode ist veraltet und soll bald entfernt werden. Die Zukunft gehört den User Defined Networks mit Embedded DNS-Server inklusive Namensauflösung für Container.

Von Mirko Dölle

Für jeden Dienst einen eigenen Container, so könnte man die Docker-Doktrin kurz zusammenfassen. Nimmt man sie ernst, zerfällt ein traditioneller LAMP-Server mit Linux, Apache, MySQL und PHP in mindestens einen Webserver- und einen eigenen Datenbank-Container – mancher lagert sogar noch PHP in einen eigenen Container aus. Doch der gewünschte Effekt, dass die Dienste unabhängig voneinander arbeiten, hat auch Nebenwirkungen: Es müssen Kommunikationskanäle konfiguriert werden.

Beim traditionellen monolithischen LAMP-Server ist die Konfiguration trivial, Webserver und MySQL-Datenbank laufen auf demselben Host

und tauschen ihre Daten lokal über das Loopback-Interface oder einen Socket aus. In Docker-Container aufgeteilt handelt es sich nun voneinander getrennte Prozesse mit eigenem Prozessraum und eigener IP-Adresse, die weiterhin miteinander kommunizieren müssen. Als Beispiel dient ein Webserver mit PHPMyAdmin als Datenbank-Frontend und abgetrennter MySQL-Datenbank. Beides gibt es als fertige Container-Images im Docker Hub.

Starten Sie den Webserver-Container mit dem Kommando

```
docker run --name webserver \
-p 80:80 -d phpmyadmin/phpmyadmin
```

so wird zunächst einmalig das PHPMyAdmin-Image vom Docker Hub heruntergeladen. Da es keine Angabe zum Netzwerk gibt, das dieser Container benutzen soll, hängt Docker ihn an die Default Bridge mit dem Netzsegment 172.17.0.0/24 an. Folglich erhält der Webserver eine IP-Adresse aus dem Subnetz der Standard-Bridge `bridge`. Welche das ist, erfahren Sie erst nach dem Start per `docker network inspect bridge`, zum Beispiel 172.17.0.2.

Veraltete Verlinkung

Einfluss auf diese IP-Adressvergabe haben Sie im Netzsegment der Standard-Bridge nicht, Sie können nicht mal einem Container eine manuelle IP-Adresse zuweisen. Das wird zum Problem, wenn Sie den Datenbank-Container starten:

```
docker run --name dbserver \
-e MYSQL_ROOT_PASSWORD=123456 \
-d mysql:8
```

Auch dieser Container landet ohne abweichende Netzwerkkonfiguration in der Standard-Bridge und erhält eine IP-Adresse aus dem Segment 172.17.0.0/24. Auch wenn die IP-Adressen lange unverändert bleiben, spätestens nach einem Neustart des Hosts werden die Karten neu gemischt – weshalb es keine Lösung ist, die von Docker zugewiesene IP-Adresse der Datenbank in PHPMyAdmin auf dem Webserver einzutragen. Abhilfe brachte bisher die Option `--link`:

```
docker run --name webserver \
--link dbserver:db -p 80:80 \
-d phpmyadmin/phpmyadmin
```

des Docker-Containers mit dem Namen `dbserver` an – sodass PHPMyAdmin über den Hostnamen `db` auf die MySQL-Datenbank zugreifen kann. Das setzt allerdings voraus, dass alle (PHP-)Anwendungen des Webservers die Namensauflösung auch nutzen und nirgends die Angabe einer IP-Adresse erfordern. Die Festlegung auf `db` als (internen) Hostnamen für den Datenbank-Container hat übrigens das PHPMyAdmin-Projekt getroffen. Solche Angaben finden Sie meist in der Beschreibung der einzelnen Docker-Images auf Docker Hub.

Doch wie schon eingangs erwähnt ist die Link-Option seit langem „deprecated“, also veraltet, und könnte bei jedem künftigen Release entfernt werden. Als Ersatz wurden die User Defined Networks geschaffen.

Netz nach Maß

Der große Vorteil der User Defined Networks liegt in der Separierung, ganz nach Docker-Mentalität: Die verschiedenen Netze sind voneinander getrennt, Zugriff haben stets nur die Container, die mit dem jeweiligen Netz verknüpft sind. Das dürfen auch durchaus mehrere sein, etwa bei einem Container, der Datenbank-Backups von Containern verschiedener Kunden einsammelt. Der Container besitzt dann mehrere virtuelle Netzwerk-Devices.

Der auffälligste Unterschied ist, dass Sie vor dem Start des ersten Containers das Netzwerk einmalig anlegen müssen:

```
docker network create webnet
```

Geben Sie wie hier nur den Namen des neuen Netzes an, legt Docker automatisch ein neues Class-B-Subnetz als Bridge an, das Sie genauso wie die Standard-Bridge namens `bridge` nutzen können: Container in diesem Netz haben Internetzugriff und sind sogar von außen erreichbar, sofern Sie die Ports freigeben. Darüber hinaus hat dieses User Defined Network `webnet` aber auch einen selbstlernenden DNS-Server, der die Namen der Container auflöst. Damit der Datenbank-Server im `webnet` gestartet wird, müssen Sie beim Aufruf von `docker run` lediglich den Parameter `--network webnet` ergänzen:

```
docker run --network webnet \
-e MYSQL_ROOT_PASSWORD=123456 \
--name dbserver -d mysql:8
```

NAS für Container

Docker ist als „Server-Lego“ zu Recht beliebt: Man kann neue Dienste mit wenig Aufwand ressourcensparsam erproben und bei Gefallen einfach laufen lassen. Wir haben uns vier NAS-Modelle mit Docker-Option genauer angesehen.

Von Ernst Ahlers

Netzwerkspeicher oder NAS (Network Attached Storage) sind seit Langem als gemeinsames Datenlager für Arbeitsgruppen, Wohngemeinschaften oder Familien gebräuchlich. Über die Zeit haben sie immer mehr Funktionen dazugelernt: Privatleute finden etwa Medienserver nützlich, die Filme aus der gemeinschaftlichen Sammlung ausliefern, während Firmen eher Pakete für CRM (Customer Relationship Management) oder ERP (Enterprise Resource Planning) schätzen.

Der Knackpunkt bei allen nachinstallierbaren Erweiterungen: Man ist auf das Repository (Software-Lager) des NAS-Herstellers angewiesen, nicht nur als Quelle, sondern auch für die fortlaufende Pflege. Bei den großen Anbietern ist auch die Auswahl groß, aber man findet immer wieder Pakete, die einmal erstellt, aber dann nicht mehr aufgefrischt wurden.

Das ist mit Docker anders: Hier kommen die Erweiterungen (Container-Images) aus einem Lager, mit dem die NAS-Hersteller nichts zu tun haben. Natürlich veralten auch zahlreiche Docker-Images, aber zumindest gibt es eine große Community statt viele kleinere. So wird Docker immer interessanter, wenn man ein NAS nicht nur als Datenlager, sondern auch als kleinen Server verwenden will.

Wir haben drei ungefähr gleich teure NAS mit x86-Prozessor und zwei Plattenbuchten beschafft, für die es bereits eine Docker-Implementierung gibt (QNAP, Synology) oder die eine andere Art der Virtualisierung bieten (KVM/Qemu bei Thecus). Außer Konkurrenz läuft das rund 50 Euro günstigere TerraMaster F2-220 mit, das sich mit etwas zusätzlicher Hardware unter dem Betriebssystem Openmediavault 4 als NAS und Mikroserver nutzen lässt [1, 2]. OMV4 bietet mehr Optionen und nachrüstbare Funktionen als die Herstellersoftware, unter anderem eben auch Docker.

Die Geräte mussten sich mit zwei 4-TByte-Platten ST4000VN008 im RAID-1-Verbund beweisen. Von unserer bisherigen Testbestückung mit 3-TByte-Modellen sind wir abgerückt, weil jetzt die etwas größeren Platten den Sweet Spot beim Preis pro Gigabyte haben. Mit älteren NAS-Tests sind die Ergebnisse bei Durchsatz, Geräuschentwicklung und Leistungsaufnahme deswegen nicht vergleichbar.

NAS-Performance vermessen

Wir testeten die NAS-Performance mit unserem c't-NAS-Bench (Download unter ct.de/wymr). Sie dürfen ihn gern für eigene Messungen verwenden, wenn Sie bei Veröffentlichungen unseren Link angeben.

Der c't-NAS-Bench kopiert mit den Windows-internen Funktionen verschieden viele, verschieden große Dateien zwischen einer hinreichend großen RAM-Disk und dem Prüfling hin und her. Dabei berücksichtigt er Verzögerungen im Betriebssystem, misst den Durchsatz also so, wie er sich in der täglichen Praxis einstellt. Würde man diesen Lag herausrechnen, entstünden viel zu optimistische Werte.

Kleine Dateien (1000 × 256 KByte) können die Geräte beim Schreiben in ihrem mehr oder weniger üppigen RAM weitgehend puffern. Dann ist der Durchsatz typischerweise deutlich höher als beim Lesen, wenn die Platten die Daten durch zeitfressende Kopfbewegungen einsammeln müssen. Dieser Effekt fällt schon bei mittelgroßen Dateien (100 × 2 MByte, beispielsweise MP3s) deutlich milder aus. Der Durchsatz klettert nochmals, wenn die NAS bei richtig großen Dateien (10 × 400 MByte, Images oder Backups) aufdrehen können und ihre Gigabit-Ethernet-Anschlüsse nahezu ausreizen.

Schaffte ein NAS beim Schreiben *und* Lesen größer Dateien mehr als 100 MByte/s, dann resultiert ein „Gut“ beim Durchsatz. Hätten beide Werte unter 50 MByte/s gelegen, was im Test nicht vorkam, dann hätte es ein „Schlecht“ gegeben.

Den gleichen Maßstab haben wir beim Zugriff auf verschlüsselte Ordner beziehungsweise Volumes angelegt. Denn alle Geräte unterstützen Verschlüsselung, die man aktiviert, wenn beim Klau einer Platte oder gleich des ganzen NAS nicht auch die Daten abwandern sollen.

Wer das Gerät diebstahlsicher unterbringen kann, der braucht sich um die Verschlüsselung und ihren bei den NAS-Prozessoren ohne AES-NI (TS-251+, F2-220) erkennbaren Performance-Nachteil nicht zu scheren. Dabei helfen abschließbare Schubladen und ein Gehäusedurchbruch nach dem Kensington-

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/wyf3.

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 22 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 den Docker-Daemon und

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
Aufwand	gering

Links zu den Containern:

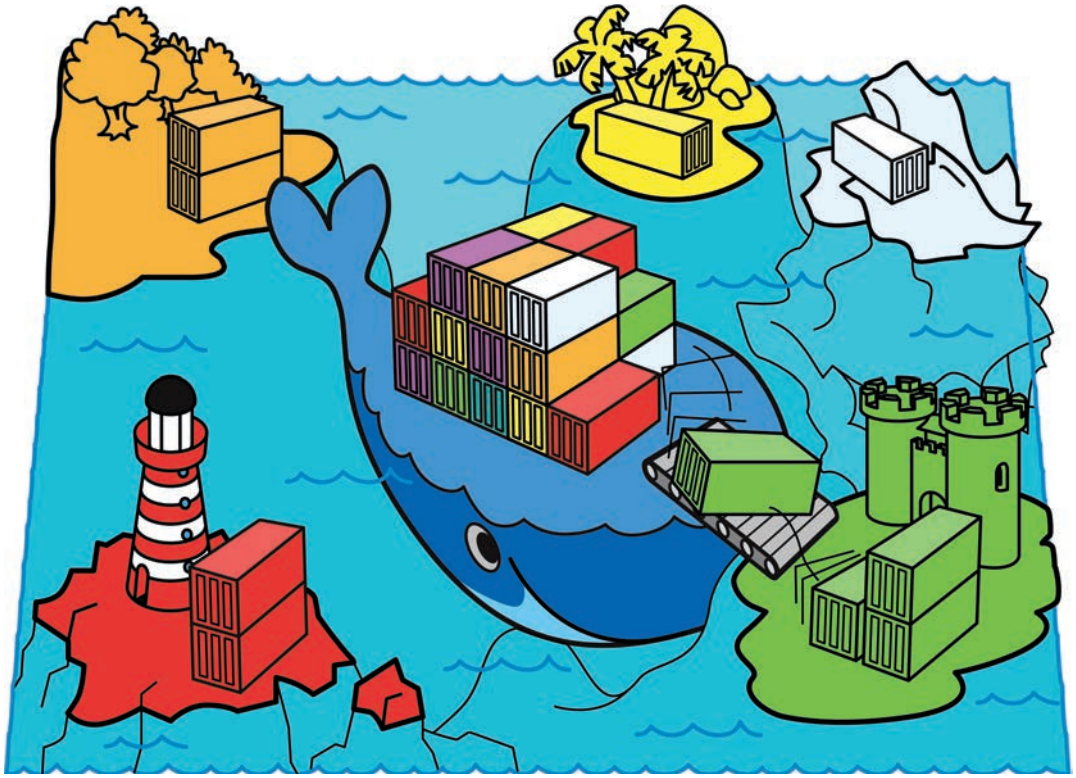


Bild: Rudolf A. Blaha, Illustrator

Container-Images in der Cloud bauen

Wer viele Docker-Images verwaltet, muss diese nach Veränderungen und Aktualisierungen nicht jedes Mal von Hand neu erzeugen. Docker Hub kann Images ohne Zutun aus bestehenden GitHub-Repositories bauen, wenn sich darin etwas ändert.

Von Merlin Schumacher

Docker-Images zu pflegen, ist oft mühselig: Man muss regelmäßig Änderungen einpflegen, Pakete aktualisieren, ein neues Image bauen und dann hochladen. Wenn sich das Basis-Image ändert, muss man erneut ran. Der Autobuild-Dienst des Docker Hub nimmt einem einen Großteil dieser Arbeit ab und erzeugt aus einem GitHub-Repository vollautomatisch fertige Images.

Spielt man eine Änderung im GitHub-Repository ein, legt der Dienst los und aktualisiert das Image.

Schlägt der Bau fehl, wird man darüber informiert und kann im Log analysieren, woran es hakt. Für den Bau von Images stellt der Hub eine Container-Umgebung bereit, die das Notwendigste zum Bau von Images mitbringt. Was darin an Software fehlt, kann man jederzeit nachinstallieren oder nachladen. Die Dockerfiles für das Image unterscheiden sich nicht von denen beim lokalen Bau.

Fundament

Erzeugen Sie zunächst ein GitHub-Repository, in dem nur ein einfaches `Dockerfile` liegt. Dockers Autobuild-Dienst kann sowohl auf GitHub- als auch Bitbucket-Repositories zurückgreifen. Dieser Artikel beschränkt sich jedoch auf die Nutzung mit GitHub. Exemplarisch erzeugen wir ein Image, das mit Hilfe des Programms Screenfetch Informationen über das System ausgibt:

```
FROM ubuntu
RUN apt-get update
RUN apt-get -y upgrade
RUN apt-get install -y screenfetch
ENTRYPOINT ["/usr/bin/screenfetch"]
```

Das Dockerfile kam bereits im Artikel ab S. 88 als Beispiel für den Betrieb von Docker auf dem Raspi zum Einsatz. Ein für diesen Artikel angepasstes Repository, das Sie für Autobuild-Versuche einfach forken, und weiterführende Links finden Sie auf ct.de/w869.

Rufen Sie den Docker Hub auf hub.docker.com auf. Sollten Sie noch keinen Account haben, registrieren Sie sich über den Knopf „Sign Up“ rechts oben. Ansonsten melden Sie sich über „Sign In“ an. Ist Ihr GitHub-Account noch nicht mit dem Hub verknüpft, klicken Sie rechts oben auf Ihren Benutzernamen und dann auf „Account Settings“. Scrollen Sie runter bis zum Abschnitt „Linked Accounts“ und klicken Sie hinter der Zeile „GitHub“ auf „Connect“. Bestätigen Sie GitHubs folgende Rückfrage, ob der Zugriff durch den Docker Hub in Ordnung ist. Nun sind Docker Hub und GitHub miteinander verbunden und können Daten austauschen.

Bauvorhaben

Klicken Sie im Docker Hub oben auf „Repositories“ und anschließend auf den großen Knopf „Create Repository“, um ein neues Docker-Image-Repository zu erzeugen. Legen Sie einen Namen und auf Wunsch auch eine Beschreibung fest. Wenn Sie den Code oder Ihre Versuche mit dem Docker Hub nicht veröffentlichen wollen, klicken Sie auf „Private“, ansonsten belassen Sie es auf „Public“. Für das weitere Vorgehen ist die Einstellung unerheblich.

Der wichtigste Teil der Einstellungen sind die „Build Settings“, denn sie regeln die Details des Bauvorgangs. Klicken Sie zunächst auf das GitHub-Symbol, anschließend erscheint eine neue Zeile, in der Sie das für den Bau als Quelle vorgesehene GitHub-Repository auswählen. Mit einem Klick auf „Click here to customize the build settings“, öffnet sich eine Liste, in der Sie Regeln für den Bau der Images festlegen. Lassen Sie die Einstellungen erst

Für den Bau von Images kann man im Docker Hub komplexe