Thorsten Leemhuis

Warenverkehr

Container mit Docker bauen, umschlagen und betreiben

Docker erleichtert Admins und Entwicklern den Alltag: Es schnürt Server-Anwendungen mit allen benötigten Komponenten zu Paketen, die auf verschiedensten Linux-Distributionen laufen und sich bequem übers Netz verteilen lassen.

WordPress installieren? Mit Docker-Containern gelingt das in einer Minute. Dabei muss man sich nicht mit den Eigenarten des jeweiligen Linux herumschlagen. Und wenn nach dem Einspielen der Sicherheitsupdates mal was schiefläuft, stellt Docker in Sekundenschnelle den alten Stand wieder her. Das alles erzeugt wenig Overhead, denn im Unterschied zu den virtuellen Maschinen von KVM, Xen oder VMware bereitet die Container-Virtualisierung der Hardware kaum Zusatzarbeit.

Durch solche Vorteile hat sich Docker einen Namen gemacht. Um von ihnen zu profitieren, muss man sich allerdings auf neue Arbeitsweisen einlassen. Die offenbaren sich bei den ersten Gehversuchen mit Docker. Greifen Sie für diese zu einer aktuellen Version einer der großen Linux-Distributionen wie CentOS, Fedora, OpenSuse oder Ubuntu. Typischerweise müssen Sie Docker über die Paketverwaltung nachinstallieren und den Docker-Daemon aktivieren. Bei Ubuntu 15.10, das wir für diesen Artikel verwendet haben, erledigt ein `apt-get install docker.io` beides. Dieser und nahezu alle der folgenden Befehle erfordern Root-Rechte, daher müssen Sie die Kommandos als Root oder durch Voranstellen von `sudo` ausführen.

Was Docker attraktiv macht, zeigt die Einfachheit einer WordPress-Installation:

```
mkdir /srv/mysqldata
docker run -d --name mysql-container \
 -v /srv/mysqldata/:/var/lib/mysql/ \
 -e MYSQL_ROOT_PASSWORD=mysqlpwd \
 mysql
docker run -d --name wordpress-container \
 --link mysql-container:mysql \
 -p 8080:80 \
 wordpress
```

Diese Befehle, deren Parameter der Artikel im Verlauf noch erläutert, laden die einige Hundert MByte großen Docker-Images mit MySQL und WordPress herunter. Mit ihnen legt Docker dann zwei verbundene Container an. In einem läuft WordPress, das Sie im Browser unter „localhost:8080“ ansprechen können. Seine Konfiguration und die Nutzdaten legt WordPress in einer MySQL-Datenbank ab; die läuft in dem anderen Container, der seine Datenbankdateien in einem eigens auf dem Host angelegten Verzeichnis speichert.

Diese Befehle ersetzen mehrere Dutzend fehlerträchtige und distributionsspezifische Konfigurationsschritte, die eine manuelle Installation von MySQL oder WordPress normalerweise erfordert. Was Docker dabei alles erledigt, lässt sich mit einem simpleren Container besser erklären:

```
docker pull debian
```

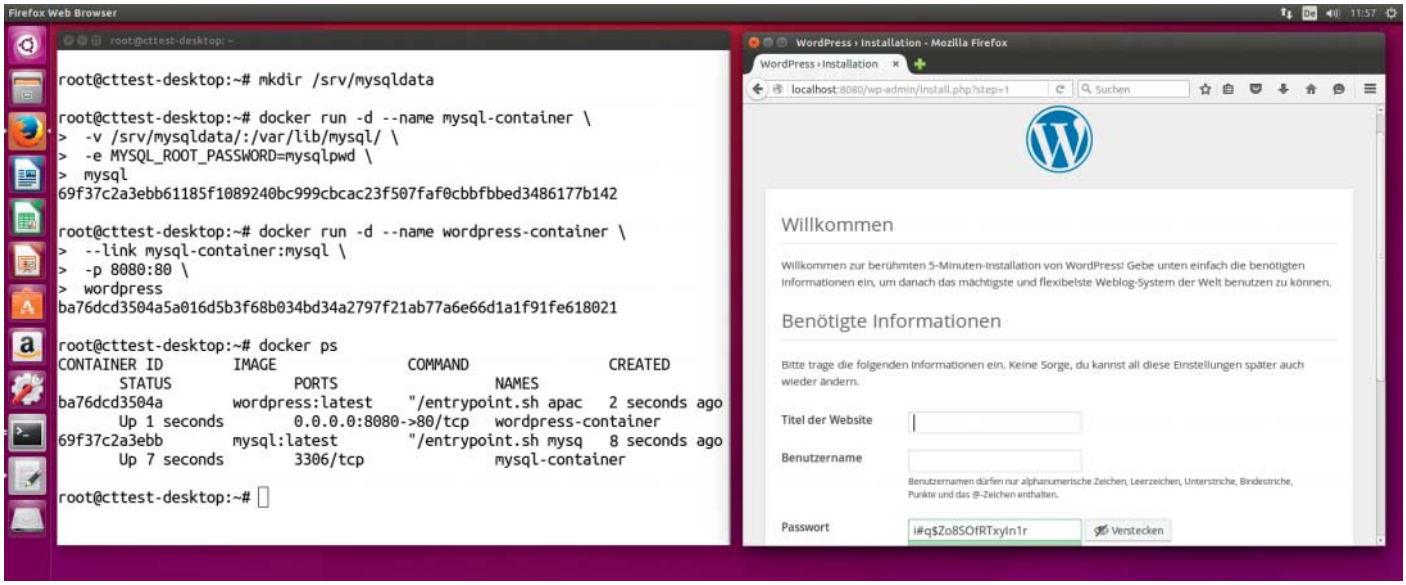
Dieses Kommando ruft ein Image mit einer Minimalinstallation eines aktuellen Debian beim Docker-Hub ab. Es ist eines von vielen bei diesem Webdienst abrufbaren Docker-Images. Legen Sie nun mit dem Debian-Image Ihren ersten Container an:

```
docker create --name=meincontainer -t -i debian
```

Er bekommt die Bezeichnung „meincontainer“. Das in der Container-Konfiguration hinterlegte `-t -i` veranlasst Docker, bei diesem Container ein interaktives Terminal-Device bereitzustellen. Ferner legt Docker ein Verzeichnis für die im Container geschriebenen Daten an.

Setzen Sie den so angelegten Container nun in Gang:

```
docker start -a -i meincontainer
```



Zwei Docker-Kommandos und ein paar Minuten Geduld, schon ist WordPress nahezu einsatzbereit.

Dadurch startet der im Hintergrund arbeitende Docker-Daemon den Container. Durch das `-a -i` verbindet der Docker-Client die aktuelle Konsole mit der Bash des Containers, die im Debian-Image als Standard-Startkommando definiert ist. Der Eingabe-Prompt wechselt dadurch auf eine Bezeichnung wie `root@063d97caacfb:/#`, die den Anfang einer automatisch erzeugten Container-ID enthält. Wenn Sie dort den Befehl `ps -A` absetzen, wird die Liste nur die im Container laufenden Prozesse `bash` und `ps` enthalten. Docker blockiert nämlich den Blick auf diese und andere Ressourcen des Hosts mit Namespace-Techniken des Linux-Kernels.

Deshalb kommen Sie auch an die Dateien des Hosts-Betriebssystems oder Ihr Home-Verzeichnis nicht heran. Ein `cat /etc/os-release` zeigt vielmehr, dass die Bash in der mit dem Image heruntergeladen Debian-Umgebung läuft. Wenn Sie hier Dateien anlegen und verändern, dann speichert Docker die Daten in dem Verzeichnis, das es beim Anlegen des Containers erstellt hat. Dieses wird nämlich mit Dateisystem-Overlay-Techniken als oberste Schicht über das Debian-Image gelegt. Das Image bleibt so unverändert, daher kann Docker es als Basis für andere Container nutzen und so Speicherplatz sparen.

Statt einer physischen Netzwerkschnittstelle ist im Container eine virtuelle verfügbar, die Docker eingerichtet hat. Docker konfiguriert zudem den Host als Router und legt eine passende Route im Container an, damit dort laufende Software den Weg ins Netz findet.

Es wird allerdings nicht alles abgeschirmt: Ein `uname -r` zeigt die Version des vom Host verwendeten Kernels, schließlich führt dieser die Bash zusammen mit der drumherum geschaffenen Abschottung aus, die den Container ausmacht. Daher sind die im Container laufenden Prozesse in der Prozessliste des Host normal zu sehen, haben dort allerdings eine andere Prozess-ID.

Docker-Container haben eine eigene Betriebssystemumgebung, wo die im Container laufenden Prozesse abgeschottet vom Rest des Systems arbeiten.

Wenn Sie die Bash durch Eingabe von `exit` schließen, endet damit auch der Container. Sie können ihn aber jederzeit erneut starten, denn die Container-Definition samt Datenverzeichnis hält Docker weiterhin vor. Der Befehl

```
docker ps
```

zeigt alle gerade laufenden Container. Dort tauchen der MySQL- und WordPress-Container auf, sofern Sie diese mit den ersten Beispielkommandos des Artikels eingerichtet haben. Sie laufen im Hintergrund und lassen sich mit dem Stop-Kommando beenden:

```
docker stop mysql-container wordpress-container
```

Der Befehl `docker ps -a` zeigt auch angelegte Container an, die gerade nicht laufen. Sollte ein solcher ausgedient haben, können Sie ihn löschen:

```
docker rm meincontainer
```

Dabei gehen alle Daten verloren, die im Container gespeichert wurden. Viel Speicherplatz wird dabei allerdings nicht frei: Die Images, mit denen Sie den jeweiligen Container angelegt haben, hält Docker zur erneuten Verwendung weiter vor. Alle vorgehaltenen Images listen Sie über `docker images` auf. Jedes davon können Sie mit `docker rmi` entfernen, sofern keine Container-Konfiguration es mehr referenziert.

Im Detail

Zum besseren Verständnis hat es sich dieser Gang durch die grundlegenden Docker-Funktionen ein wenig umständlicher gemacht, als es nötig gewesen wäre. Den simplen Debian-Bash-Container erhalten Sie nämlich auch mit nur einem Kommando:

```
docker run --name meincontainer -t -i debian
```

Das ist komfortabler, verschleiert aber den Unterschied zwischen einem Docker-Image, dem darüber angelegten Docker-Container und dem Start dieses Docker-Containers.

Durch das Beispiel mit der Shell kann man sich im Container umsehen und ein Gefühl für die Container-Virtualisierung mit Docker

```
root@cttest-desktop: ~
root@cttest-desktop:~# grep PRETTY_NAME /etc/os-release
PRETTY_NAME="Ubuntu 15.10"
root@cttest-desktop:~# ps -auxw | wc -l
245
root@cttest-desktop:~# docker exec -t -i wordpress-container bash
root@ba76dcd3504a:/var/www/html# grep PRETTY_NAME /etc/os-release
PRETTY_NAME="Debian GNU/Linux 8 (jessie)"
root@ba76dcd3504a:/var/www/html# ps -auxw
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.3 313680 31552 ? Ss 10:56 0:00 apache2 -DFOREGROUND
www-data 55 0.0 0.4 393184 39476 ? S 10:56 0:00 apache2 -DFOREGROUND
www-data 65 0.0 0.1 313812 9712 ? S 10:56 0:00 apache2 -DFOREGROUND
www-data 66 0.0 0.1 313780 9600 ? S 10:56 0:00 apache2 -DFOREGROUND
www-data 67 0.0 0.1 313780 9564 ? S 10:56 0:00 apache2 -DFOREGROUND
root 107 0.0 0.0 20232 3240 ? Ss 11:51 0:00 bash
root 114 0.0 0.0 17500 2080 ? R+ 11:51 0:00 ps -auxw
root@ba76dcd3504a:/var/www/html#
```

bekommen. Für solche „Betriebssystem-Container“ und den interaktiven Einsatz von Kommandozeilenprogrammen wie der Bash ist Docker aber gar nicht gedacht. Vielmehr ist es auf den Betrieb von „Anwendungs-Containern“ (Application Container) ausgelegt, wo eine einzelne Anwendung als Container läuft, die normalerweise als Hintergrunddienst ausgeführt würde. Neben MySQL und WordPress sind das etwa auf Node.js aufbauende Webanwendungen, Suchmaschinen wie Elasticsearch oder von Firmen für eigene Zwecke programmierte Software.

Wegwerf-Prinzip

Ein Faktor, der Docker zu seiner Popularität verholfen hat: Wenn man sich ganz auf die Docker-Philosophie einlässt, kann man in Sekundenschnelle auf den alten Stand zurückwechseln, falls es nach einem Software-Update Probleme gibt.

Damit solch ein Rollback gelingt, muss ein Container unberührt bleiben, wenn man die darin enthaltene Software oder die dort genutzte Betriebssystemumgebung aktualisieren will. Stattdessen baut man ein neues Image oder lädt es herunter. Mit ihm erzeugt und testet man dann einen neuen Container. Anschließend stoppt man den alten und startet den neuen; falls der dann zickt, lässt sich der letzte Schritt schnell revidieren. Wenn sich der neue Container hingegen bewährt hat, kann man den bisherigen Container und das alte Image wegwerfen.

Damit dieser Ansatz funktioniert, muss der Container „stateless“ sein – darf also keine Zustandsdaten enthalten. Das lässt sich bewerkstelligen, indem Sie diese extern ablegen. Dazu können Sie ein Verzeichnis vom Host in den Container hineinreichen, wie es das Beispiel am Artikelanfang für den MySQL-Container gezeigt hat. Durch den Parameter `-v /srv/mysqldata/:/var/lib/mysql/` hängt Docker das Verzeichnis `/srv/mysqldata/` im Container unter `/var/lib/mysql/` ein – an der Stelle also, wo ein MySQL standardmäßig seine Datenbank-Dateien ablegt, die so auf dem Host statt im Container landen.

Der MySQL-Container lässt sich dadurch später durch einen neueren ersetzen, dem man dieses Verzeichnis mitgibt. Solange der neuere Container dabei die Daten nicht in inkompatibler Weise ändert, kann man leicht wieder zum alten Container zurück; solch inkompatible Änderungen an Daten gibt es bei MySQL und vielen anderen Programmen normalerweise nur bei großen Versionssprüngen.

Ein explizites Hineinreichen eines Datenverzeichnisses ist beim MySQL-Image übrigens nicht zwingend erforderlich: In dessen Metadaten gibt es eine Anweisung, durch die Docker selbst ein Verzeichnis für die Datenbank auf dem Host anlegt und passend einbindet, wenn man das nicht manuell macht. Die Datenbankdateien liegen dann aber in den Untiefen unterhalb von `/var/lib/docker/`, wo man sie schnell aus den Augen verliert und beim Anlegen von Nachfolge-Containern oder Backups leicht vergisst.

Die beim Docker-Hub abrufbaren Images sind typischerweise einsatzbereit vorkonfiguriert. Sie eignen sich trotzdem für die verschiedensten Einsatzzwecke, weil sie leicht anpassbar sind. Im einfachsten Fall geht das über Image-spezifische Variablen, die man per `-e` übergibt. Das Beispiel am Artikelanfang hat auf diese Weise ein Datenbank-Passwort im MySQL-Container spezifiziert.

Die Macher eines Image legen die Variablen fest und erläutern ihren Einsatz normalerweise in der Dokumentation, die das Webinterface des Docker-Hub zeigt. In der finden sich häufig auch viele andere Tipps zum Einsatz der Images.

Duplo-Steine

In den Images beim Docker-Hub steckt typischerweise nur die Anwendung, die in Containern laufen soll, zusammen mit den von ihr benötigten Betriebssystem-Komponenten. Docker kann diese Images verbinden, um Größeres zu schaffen.

Einen solchen „Link“ zwischen Containern zeigt der eingangs angelegte WordPress-Container, bei dem das `--link mysql-container:mysql` eine Verbindung zum MySQL-Container herstellt. Dadurch baut Docker eine Netzwerkverbindung zwischen den Containern auf und erlaubt dem Startskript des WordPress-Containers, die IP-Adresse des MySQL-Containers abzufragen. Das von den Machern des Wordpress-Image erstellte Startskript kann über den Link auch die Variable abfragen, mit der Sie das Zugriffspasswort beim MySQL-Container gesetzt haben. Die so gewonnenen Informationen trägt das Skript in die Konfiguration von WordPress ein, das so alle Informationen zusammen hat, um MySQL zur Datenablage zu verwenden. Docker-Links und gute Startskripte ersetzen so viele umständliche und fehlerträchtige Konfigurationsschritte.

Die Docker-Macher haben angekündigt, die Link-Funktion mittelfristig zu entfernen. Sie wird durch die „Docker Networks“ ersetzt, die Docker seit einigen Monaten beherrscht; das WordPress-Image nutzt diese mächtigere Verbindungsfunktion aber noch nicht.

Container bekommen eine private IP-Adresse, daher sind darin laufende Anwendungen wie WordPress nicht von außen erreichbar. Damit das gelingt, muss Port-Forwarding her. Der eingangs erstellte WordPress-Container hat das bereits genutzt: Durch den Docker-Run-Parameter `-p 8080:80` werden am Port 8080 des Hosts eingehende Anfragen an den Port 80 des Containers weitergeleitet. Beim WordPress-Container lauscht dort ein Apache-Webserver samt PHP-Modulen, mit denen das WordPress-Image das CMS ausführt. Dass WordPress dabei in einem Container läuft, ist von außen nicht ersichtlich.

Von den drei eingangs erwähnten Befehlen zur WordPress-Installation bleibt nur noch das `-d` unerklärt. Es veranlasst Docker, den Container im Hintergrund zu starten. Der Parameter ist somit das Gegenstück zum Vordergrund-Betrieb, den das `-a` im Debian-Bash-Beispiel verwendet hat.

Der folgende Befehl liefert die Ausgaben, die die Container-Anwendung auf ihre Konsole schreibt:

```
docker logs wordpress-container
```

Sie können in einem laufenden Container auch eine interaktive Shell starten, um Fehlern nachzuspüren:

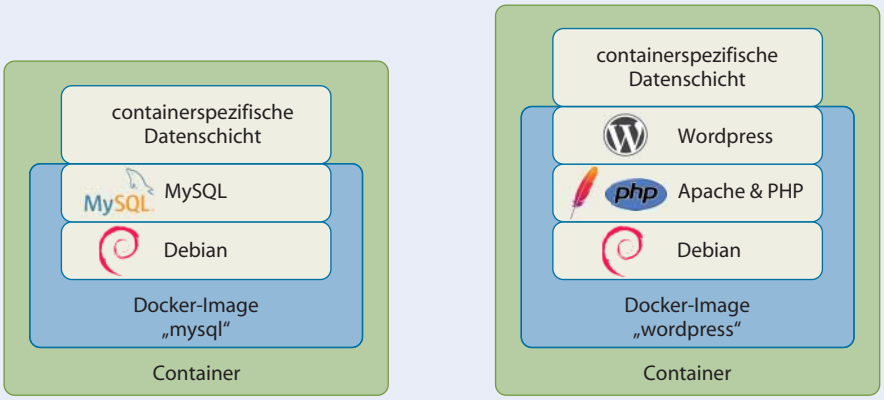
```
docker exec -t -i wordpress-container bash
```

Bilderwerkstatt

Mit den genannten Befehlen und den vielen Images beim Docker-Hub lassen sich schon zahlreiche Aufgaben realisieren. Für komplexere Einsatzzwecke und den Container-Betrieb eigener Anwendungen muss man sich

Betriebssystem-Umgebung von Containern

Docker-Container erhalten die auszuführende Software und alle von ihr benötigten Bibliotheken aus Docker-Images. Diese setzen sich aus vielen Schichten zusammen, die sich wiederverwenden lassen – MySQL- und Wordpress-Images bauen beispielsweise auf demselben Debian-Image auf.



```
root@cttest-desktop: /home/cttest/imgbau

root@cttest-desktop:/home/cttest/imgbau# cat Dockerfile
FROM debian
MAINTAINER Mein Name <foo@example.com>
RUN apt-get update &&  apt-get install -y nginx
ADD index.html /var/www/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]

root@cttest-desktop:/home/cttest/imgbau# docker build -t meinimage . &> log

root@cttest-desktop:/home/cttest/imgbau# docker run -d -p 8081:80 meinimage
f89269e98f32cef72031a070851131158b5acbbfd3126ceda85395286976ffd1

root@cttest-desktop:/home/cttest/imgbau# curl localhost:8081
<html><body><p>
Nginx in meinem ersten Docker-Image!
</p></body></html>

root@cttest-desktop:/home/cttest/imgbau#
```

Über Steueranweisungen in einem „Dockerfile“ kann Docker automatisch Container-Images erzeugen.

selbst passende Docker-Images bauen. Ein Image mit einem eigenen Nginx-Webserver können Sie beispielsweise mit den folgenden Schritten anlegen:

Erstellen Sie irgendwo ein Verzeichnis (beispielsweise ~/imagebau/) und legen darin eine index.html ab, die Ihr Container später ausliefern soll. Befüllen Sie die Datei mit etwas, um sie später wieder zu erkennen:

```
<html><body><p>
Nginx in meinem ersten Docker-Image!
</p></body></html>
```

Legen Sie im Verzeichnis ferner die Datei Dockerfile an, um darin die Steueranweisungen zur Image-Erzeugung einzutragen:

```
FROM debian
MAINTAINER Mein Name <foo@example.com>
RUN apt-get update && apt-get install -y nginx
ADD index.html /var/www/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Durch das FROM entsteht Ihr Image auf Basis des Debian-Image vom Docker-Hub. Die optionale Maintainer-Angabe nennt einen Ansprechpartner. Die Befehle in der RUN-Anweisung installieren Nginx aus den Debian-Repositories nach. Anschließend kopiert der Docker-Client die zuvor erzeugte Index-Datei an die Stelle des Image, wo Nginx die standardmäßig angezeigte Webseite sucht. Das EXPOSE verrät Docker die Netzwerk-Ports, auf der die Container-Anwendung lauscht. Die letzte Steueranweisung nennt das Kommando, das Docker standardmäßig beim Container-Start ausführt.

Der folgende Befehl baut mit den Dateien ein Image, das den Namen „meinimage“ erhält:

```
docker build -t meinimage ~/imagebau/
```

Dabei sehen Sie unter anderem die Ausgaben der spezifizierten Apt-get-Aufrufe. Anschließend können Sie mit dem Image dann Container anlegen und starten:

```
docker run -d -p 8081:80 meinimage
```

Jetzt finden Sie im Browser unter localhost:8081 die HTML-Index-Datei, die Sie zuvor angelegt hatten.

Auf die beschriebene Weise lassen sich auch andere Container erweitern – das WordPress-Image etwa, falls Sie für WordPress-Plug-ins eine der vielen PHP-Extensions brauchen, die aus Platzgründen außen vor geblieben sind. Prinzipiell könnte man solche Anpassungen auch ohne den Image-Bau vornehmen: Man legt mit einem Basis-Image einen Container an, führt die gewünschten Änderungen manuell aus und leitet daraus mit docker commit ein neues Image ab. Diese Abkürzung ist aber vorwiegend für schnelle, einmalige Anpassungen interessant: Man müsste das Ganze schließlich später jedes Mal wiederholen, wenn man auf ein neueres Basis-Image wechselt, um dort eingeflossene Sicherheitsupdates zu bekommen.

Neben den genannten Dockerfile-Steueranweisungen erläutert die Docker-Dokumentation noch zahlreiche weitere. Darunter sind auch Anweisungen zum Festlegen eines Skripts, um das Container-Verhalten über Variablen beeinflussen zu können, wie es bei MySQL- und WordPress-Images der Fall ist. Impressionen zu den Möglichkeiten liefern die Steuerdateien, mit denen die beim Docker-Hub abrufbaren Images erzeugt werden. Diese finden Sie zumeist über die dort abrufbaren Image-Beschreibungen.

Die MySQL- und WordPress-Images entstehen in Kooperation mit den jeweiligen Projekten und gelten als „offiziell“, weil sie keinen Schrägstrich im Namen enthalten. Beim Docker-Hub kann sich nämlich jedermann registrieren und anschließend beliebig viele Images über docker push hochladen. Anschließend sind diese unter Bezeichnungen wie username/imagename weltweit abrufbar.

Das Hochladen geht sehr schnell, sofern das eigene Image auf einem aufbaut, das es beim Docker-Hub bereits gibt. Docker braucht dann nämlich nur die Unterschiede hochzuladen. Diese sind nur eine weitere Schicht für ein Image, das in der Regel ohnehin schon aus vielen Schichten besteht. Manche dieser Schichten stecken in mehreren Images. Das Debian-Minimal-Image, das am Artikelanfang für erste Beispiele genutzt wurde, ist zugleich eine der Schichten im MySQL- und im WordPress-Image. Dadurch brauchte Docker die Debian-Schicht nur einmal herunterzuladen. Die Schichten sind aber voneinander abhängig – Sie können die Schichten, in denen das MySQL steckt, daher nicht mit einem Ubuntu-Image oder einer neueren Version des Debian-Image kombinieren.

Docker-Hub-User mit kostenlosem Account können eines ihrer Images als nichtöffentlich kennzeichnen. Wer mehr private Images möchte, muss Geld auf den Tisch legen. Das kann man mit einer eigenen Docker-Registry vermeiden, mit der sich ein Image-Austauschdienst in der Art des Docker-Hubs im eigenen Netz aufsetzen lässt. Ein Image, mit dem das recht schnell gelingt, gibt es beim Docker-Hub.

Sie können sich Ihre Images auch vom Docker-Hub bauen lassen. Dazu müssen Sie das Dockerfile und die darin referenzierten Dateien in ein Git-Repository einpflegen, das Sie bei GitHub oder Bitbucket hochladen und im Docker-Hub konfigurieren. Letzterer baut dann automatisch neue Images, sobald sich irgendwas im Git-Repository ändert.

Egal wie man das Image jetzt gebaut hat: Sobald es beim Hub verfügbar ist, lassen sich damit auf jedem Linux-Rechner leicht Container anlegen.

Anpassung

In der Praxis ist der Umgang mit Docker oft doch komplizierter, als es auf den ersten Blick scheint. Das zeigt sich schon beim WordPress-Image vom Docker-Hub, denn das darin enthaltene WordPress aktualisiert sich selbst. Daher kann man nach einem automatischen Update, das Probleme bereitet, eben nicht ohne Weiteres auf den alten Stand zurückgehen. Diesen Weg kann man sich offen halten, indem man regelmäßig per docker commit einen neuen Image-Snapshot vom Container ableitet, um damit notfalls einen neuen Container an den Start bringen zu können.

Auf die Update-Funktion von WordPress alleine dürfen Sie sich auch keineswegs verlassen, schließlich muss man auch die andere Software aktuell halten, die das WordPress-Image mitgebracht hat – also die Debian-Basisumgebung samt Apache und PHP-Modulen. Hier immer wieder neue Images zu erstellen kann schnell in viel Arbeit ausarten. Beim Umstieg auf Docker sollte man sich daher gleich Gedanken über Skripte oder Software-Lösungen machen, die den Image-Bau automatisieren.

Dasselbe gilt für das Anlegen und den Betrieb der Container selbst. Zur richtigen Hochform läuft Docker erst auf, wenn man es mit Orchestrierungstools kombiniert, die der Artikel auf Seite 116 näher erläutert. Der Docker-Ansatz bietet aber auch für die Software-Installationen in kleinerem Umfang einige Vorteile. Womöglich wird daher der Griff zu Docker irgendwann ganz normal, wenn man WordPress auf dem eigenen Webserver einrichten will. (thl@ct.de) c't