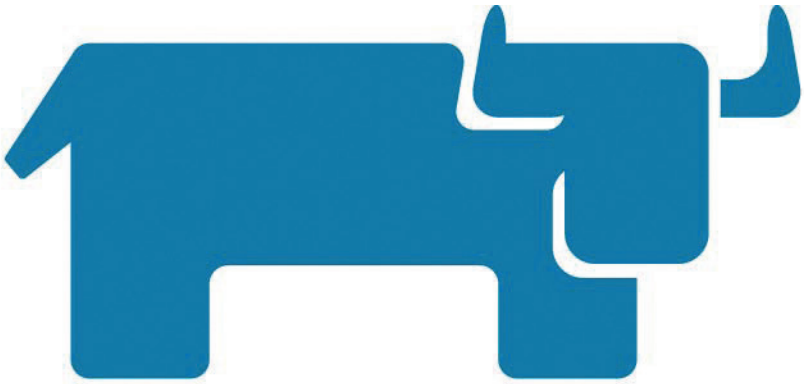

RANCHER

Docker-Application-Stacks mit Rancher

Patrick Busch, Polyas GmbH

Bei Rancher handelt es sich um eine Container-Management-Plattform, die darauf abzielt, das Arbeiten mit Docker-Containern so einfach wie möglich zu gestalten. Im Vergleich zu Kubernetes, Mesos etc. erfindet Rancher das Rad nicht neu – macht aber einiges anders. Dieser Artikel zeigt, wie schnell und einfach eine Rancher-Installation aufgesetzt ist und wie das Deployment eigener Services funktioniert.

Voraussetzung für einen erfolgreichen Einsatz ist lediglich ein laufender Docker-Daemon auf dem Host-System. Für die Absicherung des Systems ist der Nutzer selbst verantwortlich. Als Basis-Installation in den Beispielen dient ein aktuelles Debian Jessie, auf dem der Docker-Daemon installiert wurde.

Grund-Architektur

Rancher setzt einen Master-Slave-Mechanismus ein. Der Master, auch Server genannt, der zuerst eingerichtet werden muss, ist die Steuerzentrale für alle anderen Hosts. Auf dem Server kann das Rancher-UI bedient werden und er stellt das API zur Verfügung, um programmatisch alle Funktionen bedienen zu können. Die Hosts, auf denen die Container später laufen, sind in Environments unterteilt.

Abbildung 1: Startbildschirm nach erfolgreicher Installation

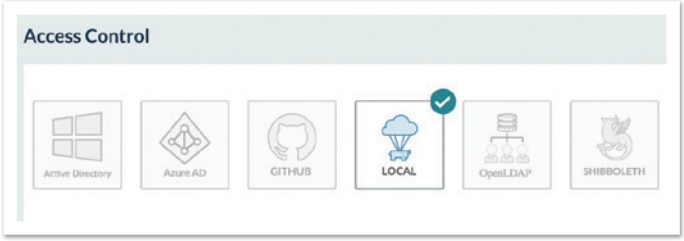

Abbildung 2: Die Möglichkeiten zur Nutzerverwaltung

Ein Environment kann beliebig viele Hosts beinhalten, wobei zwischen den Hosts innerhalb eines Environments ein IPSec-Netzwerk aufgebaut ist. Bestehen mehrere Environments, dann kennen sich diese untereinander nicht. Über das UI ist hier eine feingranulare Rechtevergabe an verschiedene Nutzer möglich.

Erstinstallation

Der Rancher-Server selbst ist ein Docker-Image, das in der einfachsten Variante ohne Konfiguration mit „docker run -d --restart=unless-stopped -p 8080:8080 rancher/server" gestartet werden kann. Ohne weitere Konfiguration wird hier eine im Container laufende Datenbank verwendet. Für einen produktiven Einsatz wird empfohlen, eine extern laufende Datenbank zu nutzen. Der Rancher-Server ist HA-fähig; die Konfigurationsanleitung kann direkt über das Frontend abgerufen werden.

Nach erfolgreichem Start ist Rancher über den Browser erreichbar. Es empfiehlt sich, die Installation nur über Reverse Proxy nach außen verfügbar zu machen. Nach erfolgreicher Einrichtung erscheint die in *Abbildung 1* gezeigte Webseite.

Erster Schritt im Server ist, die Installation gegen unbefugten Zugriff abzusichern. Über den Menüpunkt „Access Control" unterhalb von „Admin" stehen verschiedene Möglichkeiten zur Verfügung *(siehe Abbildung 2)*. Im Beispiel aktivieren wir die lokale Nutzerverwaltung. Dazu ist ein Admin-Account erforderlich. Anschließend sollte das in *Abbildung 3* dargestellte Login-Formular sichtbar sein.

Aufbau eines Environments

Bisher gibt es nur einen Master-Server, aber noch keine Environments, auf denen wir tatsächlich Container starten können. Ein Environment ist eine Gruppe von Hosts, die durch ein IPSec-Netzwerk miteinander verbunden sind. Das Default-Environment, das beim Start des Servers angelegt wird, basiert auf Cattle. Dahinter verbirgt sich der Rancher-eigene Unterbau zur Orchestrierung; er ist vergleichbar mit Kubernetes, Mesos oder Swarm. Diese Alternativen sind in Rancher auch als Unterbau für die Orchestrierung möglich; die Auswahl ist beim Neuanlegen eines Environments zu treffen. *Abbildung 4* zeigt die derzeit möglichen Environment-Templates.

Bei der Auswahl von Mesos oder Kubernetes als Unterbau für ein Environment kümmert sich Rancher um das Deployment der jeweiligen Container zum Management des Clusters. Ebenso kann der Zugriff über die jeweiligen UIs/CLIs erfolgen. Im Artikel bewegen wir uns im bei der Installation angelegten Default-Environment, also auf Cattle.

Ein erster Host kann im Frontend-Menüpunkt „Infrastructure" über den Unterpunkt „Hosts" hinzugefügt werden. Über den entsprechenden Button wird der Prozess gestartet. Vor dem Hinzufügen des ersten Hosts ist die Host-Registration-URL anzugeben *(siehe Abbildung 5)*. Die Host-Registration-URL muss von jedem Host aus erreichbar sein und auf das API des Servers leiten. Per Default ist dies die URL, über die auch das UI erreichbar ist. Nach der einmaligen Eingabe der Host-Registration-URL lässt sich über das Formular der neue Host konfigurieren *(siehe Abbildung 6)*.

Der Screenshot zeigt die Konfiguration für einen selbst aufgesetzten Host. Bei den zur Verfügung stehenden Cloud-Anbietern gestal-

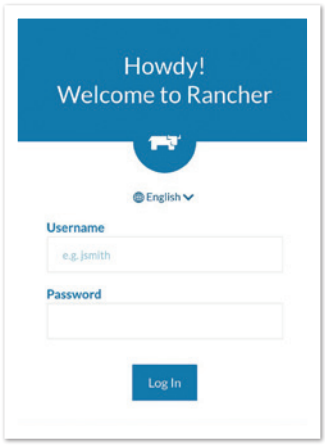

Abbildung 3: Das Login-Formular

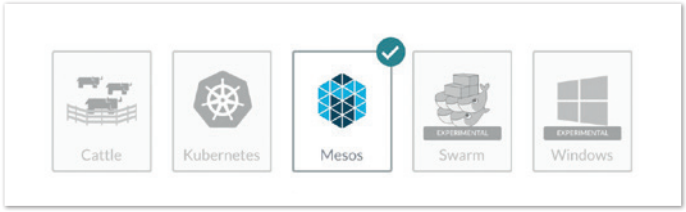

Abbildung 4: Die Environment-Templates

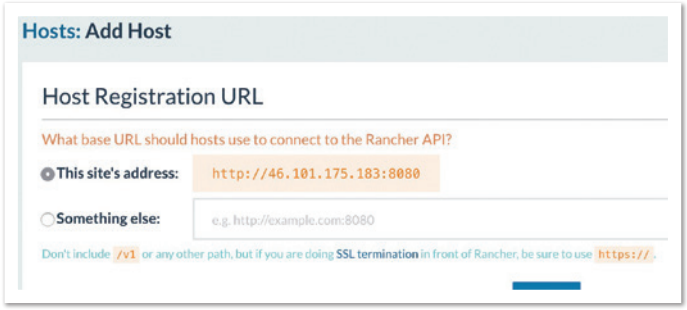

Abbildung 5: Die Eingabe der Host-Registration-URL

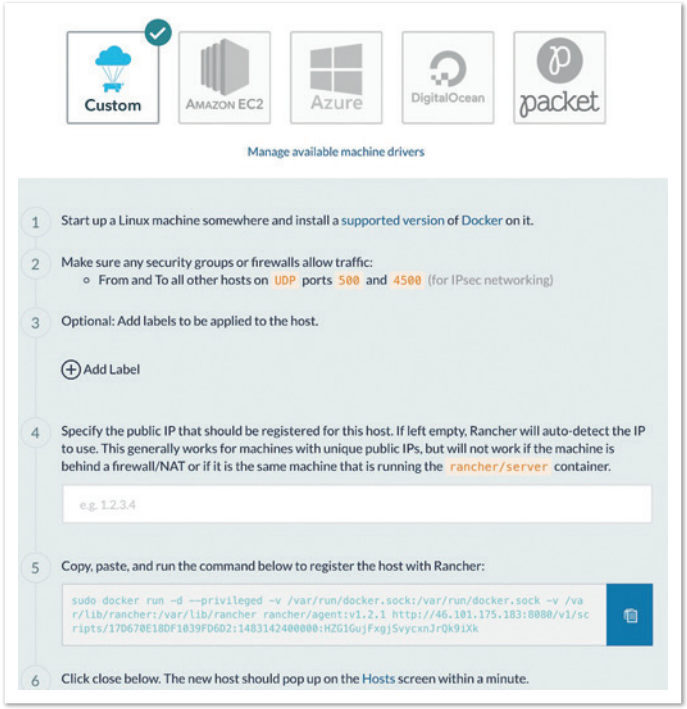

Abbildung 6: Die Konfiguration eines neuen Hosts

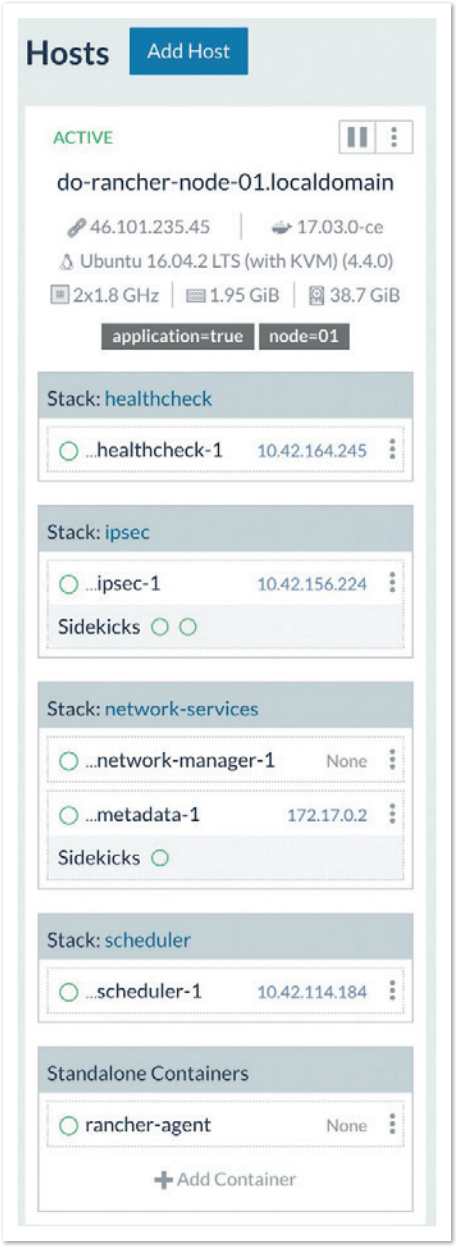

Abbildung 7: Der laufende Host

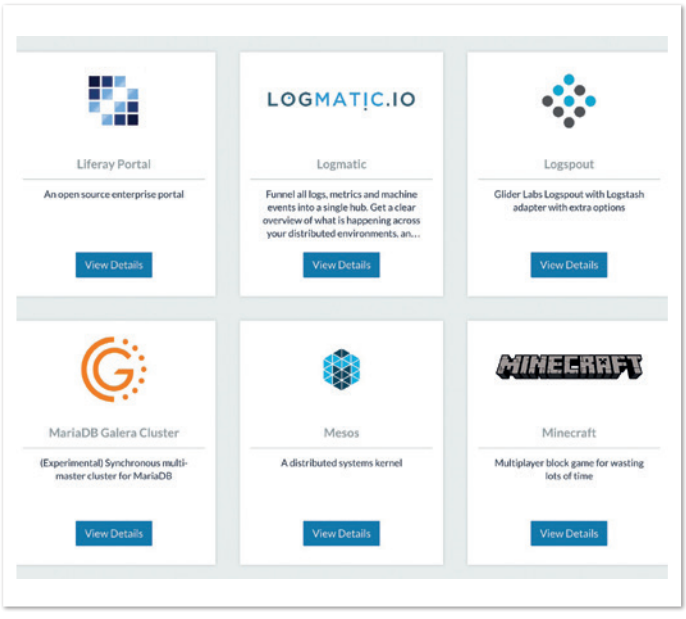

Abbildung 8: Beispiele für Applikationen im Rancher Catalog

tet sich die Konfiguration entsprechend anders. Bei DigitalOcean genügt es beispielsweise, den entsprechenden API-Key anzugeben, um dann die Hosts dort direkt provisionieren zu können.

Dieses Formular hilft beim manuellen Aufsetzen dabei, den Host zu konfigurieren. Voraussetzung ist, wie beim Master-Server, ein Linux-Host mit installiertem Docker-Daemon. Das Hinzufügen eines neuen Hosts ist so einfach wie der Start des Master-Servers; mit einem kleinen Unterschied. Das Formular hilft lediglich, den entsprechenden Docker-Befehl zusammenzustellen, der auf dem neuen Host ausgeführt wird *(siehe Listing 1)*.

Nach dem Ausführen des Befehls sollte der neue Host innerhalb kurzer Zeit im Server erscheinen *(siehe Abbildung 7)*. Das Formular stellt neben zusätzlichen Informationen, unter anderem zu Ports, die Möglichkeit bereit, Hosts mit Labels zu versehen. Über diese Labels lässt sich später das Deployment-Verhalten von Containern bestimmen. Die einfachste Variante lautet, die Hosts durchzunummerieren, wobei die empfohlene Variante lautet, den Hosts fachlich passende Labels zu geben. Es ist beispielsweise denkbar, ein heterogenes Environment mit Hosts, die auf starke I/O-Last ausgelegt sind, und Hosts, die lediglich Datenhaltung betreiben, aufzubauen. Durch entsprechende Labels auf den Hosts kann den Containern dann beim Starten vorgegeben werden, auf welchen Hosts mit dazugehörigen Labels sie starten dürfen. Für das weitere Vorgehen in diesem Artikel wurde ein Environment von drei Hosts aufgebaut.

Rancher Catalog

Der Rancher Catalog ist ein von der Community gepflegter Katalog aus fertig deploybaren Applikationen und die einfachste Möglichkeit, um schnell zu Ergebnissen zu kommen. Eine Applikation heißt Rancher-intern „Stack", wobei ein Stack aus verschiedenen Containern besteht. Die Definition eines Stacks wird als „docker-compose.yml" vorgegeben. Zusätzlich besitzt Rancher ein eigenes „rancher-compose.yml"-Format. Letzteres enthält lediglich zusätzliche Informationen, die beispielsweise die Skalierung der im „docker-compose.yml" definierten Container oder deren Health-Management betreffen. Der Rancher Catalog selbst ist eine auf GitHub gepflegte Ansammlung solcher Service-Definitionen verschiedenster Ausprägungen.

Abbildung 8 zeigt, wie beispielsweise ein MariaDB-Galera-Cluster vollständig und auf Knopfdruck über den Catalog gestartet werden kann. Beim Auswählen der Option erscheinen hierzu applikationsspezifische Konfigurationsmöglichkeiten. Als Beispielapplikation starten wir nun „Rocket.Chat". Dazu suchen wir den entsprechenden Service im Rancher Catalog, wählen eine Version aus und starten ihn. *Abbildung 9* zeigt den darauf startenden Vorgang: Das Deployment und Starten der einzelnen Container. Da in der Konfiguration keine Regeln für das Deployment nach Host-Labels hinterlegt sind, werden die Hosts zufällig ausgewählt. *Abbildung 10* zeigt die vollständig laufende Applikation. Durch einen Klick auf den blau hinterlegten Port können wir direkt zur laufenden Applikation springen; allerdings nur auf diesem einen Host und vorausgesetzt, der Port ist nicht durch die Firewall geblockt.

Load Balancing und SSL

Rancher bietet fertige Services zum Load Balancing an, die auch SSL terminieren können. Sie heißen im UI „Load Balancer". Unter der Haube steckt ein HAProxy-Server. Dementsprechend können über die

Defaults hinausgehende Einstellungen über „HAProxy.cfg"-Dateien vorgenommen werden. Um damit SSL terminieren zu können, müssen zuerst die gewünschten SSL-Zertifikate eingepflegt werden.

Über die Menüpunkte „Infrastructure" und „Certificates" lassen sich einzelne Zertifikate hinzufügen. Für jedes Zertifikat können hier ein Name und eine Beschreibung hinterlegt sein. Außerdem sind mindestens ein Zertifikat und ein Private Key anzugeben. Der Private Key ist nach dem Abschicken des Formulars nicht mehr einsehbar. Um Änderungen an der Zertifikatseinstellung vorzunehmen, muss er daher jedes Mal neu angegeben werden.

Um das Zertifikat zu verwenden, muss ein öffentlich erreichbarer Load Balancer definiert sein. Hierzu sollten ein neuer Stack angelegt und die Konfigurationsoptionen leer gelassen werden. Innerhalb dieses Stacks wird dann über den blauen Dropdown-Button ein Load Balancer angelegt.

Abbildung 11 zeigt, wie ein Load Balancer per Default auf allen Hosts eines Environments ausgerollt wird. Damit die Lastverteilung sauber funktioniert, sollten die Requests von außen auch gleichverteilt auf allen Hosts ankommen. Erreicht werden kann dies beispielsweise durch ein DNS-Round-Robin-Verfahren, bei dem eine Domain auf mehrere IP-Adressen zeigt. Als Best Practice hat es sich bewährt, für jedes Environment solche Domains einzurichten und die Domains, über die die Services erreicht werden sollen, als CNAME auf dieser Environment-spezifischen Domain einzurichten. Das hält den Pflegeaufwand beim Hinzufügen neuer Hosts in Grenzen, da nur eine Domain bearbeitet werden muss.

In der Konfiguration des Load Balancer sollte der Access auf „public" gesetzt sein. Als Protokoll ist HTTPS auf jeden Fall vorzuziehen; dafür muss aber bereits ein entsprechendes SSL-Zertifikat eingerichtet worden sein. Dieses kann im unteren Abschnitt des Formulars konfiguriert werden. „Request Host" sollte die Domain sein, über die der Service erreichbar sein soll. Die Domain sollte darüber hinaus auch mit dem SSL-Zertifikat übereinstimmen. Der Port wird entsprechend zum Protokoll angegeben. Als Target ist der Service definiert, an den die Requests weitergeleitet werden sollen, und als Port gilt der, auf den der Service lauscht. In unserem Fall sind das „rocketchat" und „3000". Dabei ist es unerheblich, ob der Port, auf den weitergeleitet wird, auf eine öffentliche Adresse hört, da wir hier in einem komplett internen IPSec-Netzwerk sind. „Rocket.Chat" ist von hier an unter der eingegebenen Domain und dem konfigurierten Protokoll erreichbar.

Für den Fall, dass wir SSL konfiguriert haben und auch nur darüber erreichbar sein wollen, bietet es sich an, einen Service einzurichten, der alle Requests von Port 80 auf Port 443 weiterleitet *(siehe Listing 2)*. In *Abbildung 12* ist dafür eine beispielhafte Konfiguration dargestellt. Es handelt sich hierbei um einen NginX-Proxy, der über die untenstehende Konfiguration alle einkommenden Requests auf Port 443 umleitet.

Wichtig in der Konfiguration ist, dass auf jedem Host eine Instanz davon läuft, wenn DNS-Round-Robin für das Environment eingerichtet ist. Das vorkonfigurierte Image heißt „pbusch/https-proxy:1.0". Hier wird über die Port-Map der öffentliche Port 80 auf den Container-Port 80 weitergeleitet. Dafür muss Port 80 auf allen Maschinen offen sein. Nach dem Start des Service ist „Rocket.Chat"

```
docker run -d --privileged -v
/var/run/docker.sock:/var/run/docker.sock -v
/var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.1
https://your-host-registration-
url/v1/scripts/4C321E7DD75BAC37052A:1483142400000:OIqF
s2R66XKj36McyFdr66io9Fo
```

Listing 1

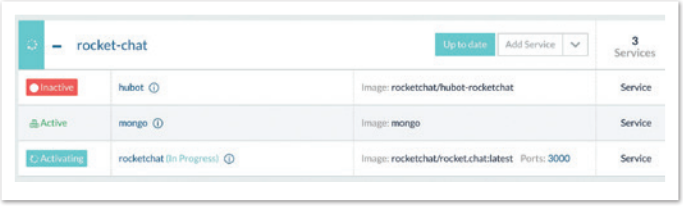

Abbildung 9: Startvorgang eines Applikations-Stacks

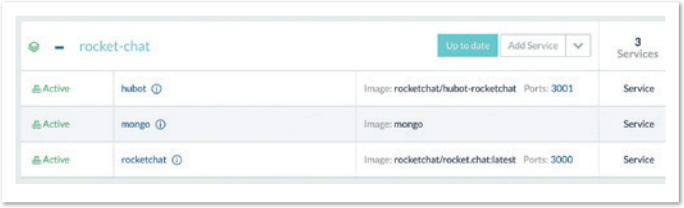

Abbildung 10: „Rocket.Chat"-Stack

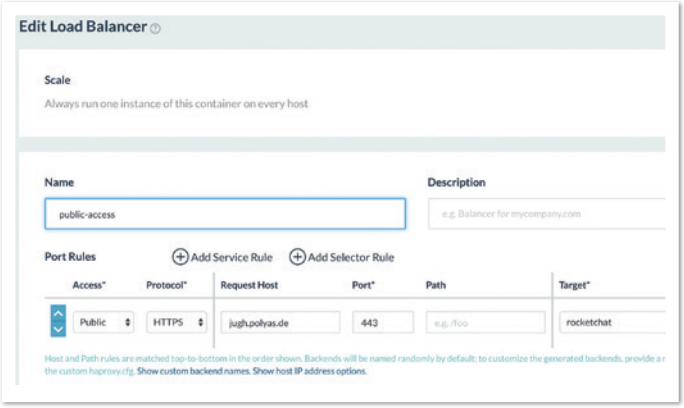

Abbildung 11: Load Balancer mit SSL-Terminierung

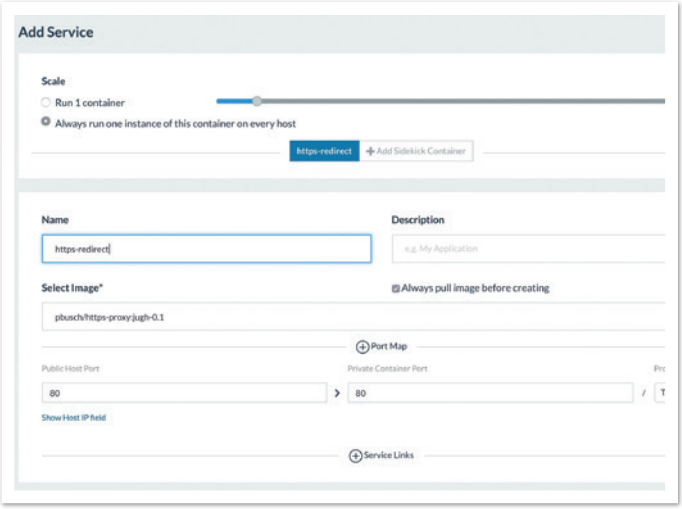

Abbildung 12: Konfiguration für einen Service zur Weiterleitung von HTTP auf HTTPS

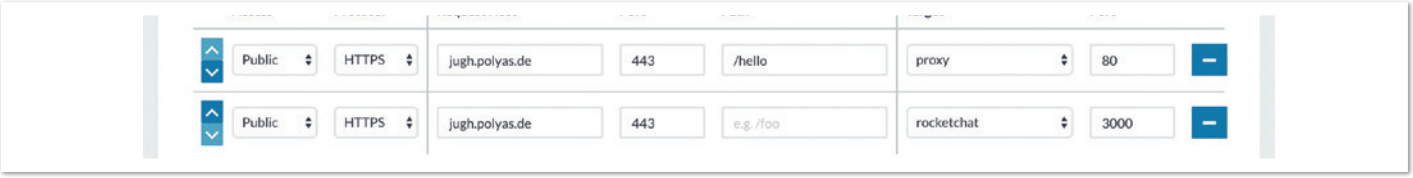

Abbildung 13: Angepasste Konfiguration des Load Balancer mit zusätzlichem Service

nun unter derselben Domain über HTTPS erreichbar und bei dem Versuch, auf HTTP zu verbinden, wird auf HTTPS umgeleitet.

Eigene Applikationen

Mit dem nun angeeigneten Vorwissen und den durchgeführten Vorbereitungen ist das Deployment einer eigenen Applikation relativ einfach. Als Beispiel nehmen wir eine simple, vorkonfigurierte Applikation, bestehend aus einem Webserver, der statische Assets ausliefert, und einem vorgeschalteten Proxy. Dafür legen wir zuerst einen leeren Stack an. Als erster Service wird das Image „pbusch/webpage-proxy:1.0" verwendet. Es sollte unter dem Namen „proxy" laufen.

Als zweiter Service kommt das Image „pbusch/webpage-assets:1.0" zum Einsatz. Dieses sollte unter dem Namen „assets" laufen. Über die entsprechenden Namen können die Services eine Namensauflösung über das Rancher-interne DNS machen. Generell ist jeder Service innerhalb eines Environments unter der Domain „servicename.stackname.rancher.internal" erreichbar. Da „rancher.internal" als Suchdomain vorkonfiguriert ist, reicht „servicename.stackname" als Adresse. Innerhalb eines Stacks genügt sogar nur der Name des Service.

Bei Einhaltung eines Namensschemas ist die Verlinkung der Container innerhalb eines Stacks nur für die Namensauflösung nicht nötig. Natürlich kann man auch im Beispiel beliebige Namen verwenden, dann müsste allerdings der Service mit den Assets unter dem Namen „assets" zum Proxy-Service verlinkt werden.

Da hier gar keine öffentlichen Ports definiert wurden und diese Applikation wenn möglich unter demselben Zertifikat laufen soll, muss nun der Load Balancer angepasst werden. Dafür muss in der Konfiguration des Load Balancer eine neue Service-Regel hinzugefügt werden. Diese sollte mit demselben Protokoll, Host und Port wie die bestehende Regel angelegt sein. „Target" sollte der gerade angelegte Proxy-Service auf Port 80 sein. Zusätzlich wird jetzt ein „Request Path" angegeben: „/hello". Die neue Regel sollte dann über der bestehenden Regel stehen, da der Request Path auf den neuen Service beschränkt ist. Die Regeln des Load Balancer werden von oben nach unten abgearbeitet und es greift für jeden Request die erste Regel, bei der alle Bedingungen erfüllt sind. *Abbildung 13* zeigt die Konfiguration. Unter der konfigurierten Domain und dem Pfad „/hello" sollte nun die Seite wie in *Abbildung 14* erscheinen. Somit ist unsere erste eigene Applikation lauffähig.

```
server {
  listen 80 default_server;
  listen [::]:80 default_server;
  server_name _;
  add_header "Strict-Transport-Security" "max-
age=31536000";
  return 301 https://$host$request_uri;
}
```

Listing 2

Abbildung 14: Erster eigener Service

Fazit

Der typische Application Stack mit Frontend, Backend und Datenbank ist mit Rancher einfach zu managen, solange Frontend- und Backend-Services „stateless" gehalten sind. Die Möglichkeiten zur Skalierung ergeben sich hierbei fast von selbst. Rancher ist schnell aufgesetzt, bietet die Möglichkeit, schnell zu Ergebnissen zu kommen, und bringt dennoch die Robustheit für einen Produktivbetrieb mit, solange die einzelnen Host-Systeme entsprechend gegen Angriffe geschützt sind.

Durch die Möglichkeit, verschiedene Environments betreiben zu können, und über die Erstellung und Einbindung eines eigenen Rancher Catalog ist die Einbindung in die eigene Release-Pipeline sogar empfehlenswert. Allerdings sollte dabei ein Augenmerk darauf gelegt werden, dass die eigene Software über private Docker-Registries angebunden ist.

Patrick Busch
p.busch@polyas.de

Patrick Busch ist Head of SaaS Development bei der Polyas GmbH. Aus Karlsruhe stammend, verbrachte er die ersten Jahre seines Berufslebens als Entwickler bei einem dort ansässigen Hosting-Unternehmen. Seit dem Umzug nach Kassel arbeitet er mit der Polyas GmbH, erst als Entwickler, nun in einer Führungsposition, daran, Online-Wahlen auch in Deutschland erfolgreich auf den Weg zu bringen.