OpenStack-Deployment mit Kolla

Einfach aufgeschichtet

David Rabel, Robin van der Linden, Christian Berendt

OpenStack händisch zu installieren ist zwar lehrreich, für den Produktiveinsatz allerdings kaum zu empfehlen. Kolla verspricht ein einfaches, anpassbares und automatisiertes Setup.

OpenStack besteht aus einem immer komplexeren Ökosystem von Projekten: schwierig zu überschauen und noch schwieriger zu verwalten. Gängige Deployment-Frameworks vereinfachen das nur zum Teil. Während eine zum großen Teil automatisierte Installation mittlerweile zum Standard gehört, fehlen oft die Möglichkeiten zum nahtlosen Upgrade der einzelnen Dienste. Auch Anpassbarkeit und Herstellerunabhängigkeit sind nicht immer gegeben. Kolla verpackt einzelne Dienste in Docker-Containern und setzt zur Konfiguration und Orchestrierung auf Ansible oder Kubernetes. Es ist flexibel und ermöglicht relativ einfach ein tief greifendes Anpassen der Umgebung.

Wer einmal versucht hat, eine funktionstüchtige OpenStack-Umgebung von Hand zu bauen, kennt deren Komplexität. Eine interessante Erfahrung, keine Frage – und wer viel Zeit und Frustrationstoleranz mitbringt, lernt auf diese Weise wahrscheinlich am schnellsten etwas über das Innenleben von OpenStack und das Zusammenspiel der einzelnen Komponenten. Für den Produktiveinsatz ist ein manuelles Deployment aber nicht tauglich. Dafür gibt es zu viele Konfigurationsmöglichkeiten, zu viele Stolpersteine, zu viel, was schiefgehen kann, und zu viele Einzelteile, die aufeinander abgestimmt sein wollen. Mittlerweile besteht die Software aus über dreißig Einzelprojekten. Hinzu kommen zahlreiche weitere Projekte, die nicht offiziell Teil von OpenStack sind, aber in dessen Umfeld existieren.

Ein Monster an Komplexität

Es liegt nahe, Installation und Konfiguration der Einzelkomponenten zu automatisieren. Deployment-Frameworks für OpenStack gibt es schon länger. Deren Bandbreite ist groß: vom dünnen Wrapper um bekannte Konfigurationsmanagement-Tools wie Ansible, Puppet oder Chef bis zu hübschen Webinterfaces, von der Installation direkt aus den Sourcecode-Repositories bis zu vorgefertigten Paketen. Die verschiedenen Ansätze haben ihre Stärken und Schwächen. Eine Entscheidung für das richtige Deployment-Framework ist deshalb vor allem eine Frage der Anforderungen.

Zunächst wäre da die Anpassbarkeit. Leider scheint der Wunsch danach dem nach einer hübschen Oberfläche entgegenzulaufen. Wer täglich mit einem so komplexen Gebilde wie OpenStack arbeiten muss, wird vermutlich mehr Wert auf Funktion als auf Aussehen legen. Natürlich ist nicht alles schlecht, was nicht auf der Kommandozeile passiert. Es ist aber wichtig, immer einen Blick hinter den Vorhang zu werfen. Denn allzu oft kommt man in Situationen, die die Entwickler des Deployment-Frameworks nicht im Kopf hatten. Und dann ist es von Vorteil, sich nicht durch viele Abstraktionsebenen zu arbeiten, sondern nah am tatsächlichen Geschehen zu sein und Änderungen an der untersten Ebene durchführen zu können – ohne die Konsistenz zu gefährden und ohne dass diese Änderungen bei der nächsten Gelegenheit wieder überschrieben werden.

Ein weitere Schwachstelle vieler Deployment-Frameworks sind Upgrades. Oft beherrschen Frameworks den Schritt von einer zur nächsten OpenStack-Release nicht oder erwarten dafür einige manuelle Nacharbeit. Neue Releases von OpenStack erscheinen aber halbjährlich und werden bereits nach einem Jahr nicht mehr mit offiziellen Updates versorgt. Aufgrund dieses zügigen Erscheinungszyklus empfiehlt es sich, von vornherein die Möglichkeit nahtloser Upgrades so-

wohl auf Zwischenversionen als auch auf Major Releases einzuplanen.

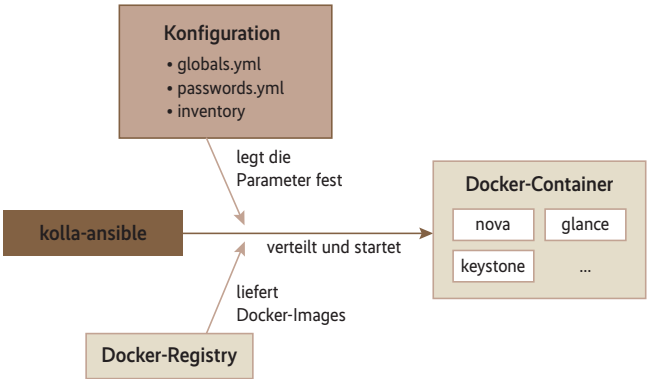
Viele Entwickler sorgen für Sicherheit

Ein häufig erwähntes Argument für freie Software ist das Vermeiden eines Vendor-Lock-in. Dieser Vorteil besteht bei OpenStack unabhängig vom Deployment-Framework. Auch die meisten Tools zum Installieren und Verwalten der eigenen OpenStack-Landschaft stehen unter einer freien Lizenz. Wenn der Hersteller sich entschließt, diese nicht weiterzuentwickeln, ist es immer möglich, die Entwicklung oder zumindest nötige Bugfixes selbst voranzutreiben.

So weit die Theorie. In der Praxis sieht man immer wieder, dass Tools verwaisen, wenn der ursprüngliche Hersteller sie nicht weiterentwickelt. Ein Beispiel aus der jüngeren Vergangenheit ist das Deployment-Framework Fuel [1], das fast ausschließlich von Mirantis betreut wurde. Seitdem Mirantis auf ein anderes Tool setzt, entwickelt es kaum noch jemand weiter. Deshalb empfiehlt es sich, immer einen Blick auf die Struktur der Entwickler-Community zu werfen (siehe „Alle Links“ am Ende des Artikels): Arbeitet primär eine einzelne Firma an der Software oder sind Entwickler aus unterschiedlichen Unternehmen daran beteiligt?

Automatisiert und in Container verpackt

Kolla ist ein Deployment-Framework für OpenStack, das konsequent alle Komponenten in Docker-Containern betreibt – sowohl OpenStack-eigene Services als auch darunterliegende Infrastrukturdienste wie Datenbanken oder den Message-Broker RabbitMQ. Als zugrunde liegendes Konfigurationsmanagement-Tool setzt es auf Ansible. Der Ansatz von Kolla ist, eine funktionierende Grundkonfiguration bereitzustellen und diese mit Overlays zu überschreiben. Das hat den zusätzlichen Vorteil, dass man Kolla einfach mit einem All-in-one-Deployment ausprobieren kann, bei dem alle Dienste auf einer einzelnen Maschine laufen.



Kolla verteilt Docker-Container auf Hosts, um somit einen OpenStack-Cluster aufzusetzen. Die Rollen der einzelnen Server sind in den Konfigurationsdateien festgelegt (Abb. 1).

Passende Docker-Images können direkt aus dem Quelltext gebaut werden. Die Community stellt aber zu jeder Release auch fertige Images zur Verfügung. In jedem Fall ist es sinnvoll, die Images in einer lokalen Docker-Registry vorzuhalten – besonders bei Produktivumgebungen. Das Deployment in Applikations-Containern erlaubt, die einzelnen Dienste getrennt voneinander und vom zugrunde liegenden Hostsystem zu verwalten.

Leichte Hülle um Ansible

Kolla besteht prinzipiell aus zwei Komponenten: den Docker-Images und einem Deployment-Tool (siehe Abbildung 1). Dieser Artikel bezieht sich bei Letzterem auf das schon länger von Kolla eingesetzte Ansible, obwohl in naher Zukunft auch Kubernetes zur Orchestrierung verwendet werden kann.

Das Tool *kolla-ansible* spielt dabei eine zentrale Rolle. Es erlaubt das Vorbereiten *(bootstrap-servers)* und Testen *(prechecks)* der Server sowie das Ausrollen *(deploy)*, Ändern *(reconfigure)* und Aktualisieren *(upgrade)* der OpenStack-Landschaft. Dabei ist *kolla-ansible* tatsächlich nur ein relativ kleines Skript, das einen leichtgewichtigen Wrapper um Ansible darstellt. Kollas Stärke besteht in der Vielzahl vorgefertigter Ansible-Tasks, aufgeteilt auf verschiedene Playbooks und Rollen, die je nach Kommando zum Tragen kommen.

Welche Tasks dabei auf welche Nodes angewendet werden, legt wie bei Ansible üblich ein Inventory-File fest. Das von Kolla mitgelieferte Inventory enthält eine Vielzahl aufeinander aufbauender Definitionen, sodass meist nur noch die Hostnamen der Server und ihre gewünschte Rolle in der OpenStack-Landschaft eingetragen werden müssen.

Die weitere Konfiguration geschieht vor allem in der Datei *globals.yml* im Verzeichnis */etc/kolla*, die *kolla-ansible* unverändert an Ansible weiterreicht. Die dort gesetzten Variablen wiederum werden in den von Kolla mitgelieferten Ansible-Playbooks ausgewertet. Sie definieren, welche Tasks wo angewendet werden.

Ein Deployment mit Kolla verläuft immer ähnlich: Es gibt einen Seed-Node, von dem aus man alle anderen Nodes einrichtet. Diesen grundlegend vorzubereiten, ist genau wie die Installation des Betriebssystems auf allen Nodes als Vorarbeit zu leisten. Sobald *kolla-ansible* mit all seinen Abhängigkeiten installiert ist, läuft der Rest hierüber: die Installation weiterer benötigter Pakete, das Vorkonfigurieren der anderen Nodes, das Downloaden der Docker-Images und letztendlich das Deployment.

Kolla hat also keine hübsche Oberfläche, sondern überzeugt durch Schlichtheit und einfache Benutzbarkeit. Sowohl das Erstellen von Images mit Docker als auch das zum eigentlichen Deployment eingesetzte Ansible ist leicht zu verwenden und gut strukturiert. Das ermöglicht nicht nur eine große Anpassbarkeit, sondern erleichtert auch das Troubleshooting erheblich. Schlägt zum Beispiel das Deployment fehl, sieht man sofort, in wel-

iX-TRACT

- OpenStack ist eine komplexe Sammlung von Einzelkomponenten, deren händische Installation viel Zeit und Sorgfalt erfordert.
- Werkzeuge, die diesen Vorgang automatisieren, lassen sich häufig nicht ausreichend anpassen. Vielversprechende Helfer sind wieder vom Markt verschwunden, weil niemand sie weiterentwickelt.
- Kolla kombiniert die Fähigkeiten von Ansible und Docker, um automatisiert einen OpenStack-Cluster aufzusetzen. Dabei gibt es viel Spielraum für Individualisierung.

chem Ansible-Task es ein Problem gab, und kann ihn mit detaillierterer Ausgabe wiederholen.

Schlicht, praktisch, überschaubar

Durch die konsequente Verwendung von Docker-Containern können einzelne Dienste oder die gesamte Umgebung leicht auf einen neueren (oder älteren) Versionsstand gebracht werden, und das nahezu ohne Unterbrechung.

Ein Vendor-Lock-in hat man bei Kolla kaum zu befürchten. Insgesamt hat keine einzelne Firma mehr als fünfzehn Prozent der Commits zum Sourcecode beigetragen. Seit der letzten Release sieht es ganz ähnlich aus.

Im Folgenden soll ein All-in-one-Deployment von OpenStack mit Kolla erläutert werden. Grundlage hierfür ist ein Server oder eine virtuelle Maschine mit mindestens 2 CPU-Kernen, 8 GByte RAM, 20 GByte Storage und 2 Netzwerkschnittstellen. Empfohlen sind 4 Kerne, 10 GByte RAM und 100 GByte Disk Space. Als Betriebssystem kommt CentOS 7 zum Einsatz. Eines der Netzwerk-Interfaces sollte eine statische oder per DHCP vergebene IP-Adresse aufweisen. Das andere wird während der Installation konfiguriert.

Schnelles All-in-one-Deployment zum Testen

Des Weiteren sollte die CPU Hardwarevirtualisierung beherrschen. Ist diese aktiv, ist in der Datei */proc/cpuinfo* eines der Flags *vmx* oder *svm* gesetzt. Falls nicht, lässt sich das im BIOS oder UEFI aktivieren. Im Falle einer virtuellen Maschine muss Nested Virtualization auf dem Hostsystem aktiviert sein und die CPU-Konfiguration des Hosts korrekt an das Gastsystem weitergereicht werden. Außerdem muss je nach CPU entweder das Kernelmodul *kvm_intel* oder *kvm_amd* geladen sein. Im ersten Schritt bringt man mit *yum* das System auf den aktuellen Stand und fügt das Repository *epel* hinzu, das die für die Installation benötigten Pakete enthält:

```
yum update && yum upgrade
yum install -y epel-release
```

Im weiteren Verlauf benötigt Kolla noch einige Pakete aus den Repositories. Den OpenStack-Client, das Kommandozeilentool von OpenStack, sowie *kolla-ansible* installiert man direkt aus dem Python Package Index:

```
yum install -y python-pip && pip install -U pip
yum install -y python-devel libffi-devel \
        gcc openssl-devel libselinux-python \
                                        ansible wget
pip install python-openstackclient kolla-ansible
```

Kolla liest beim Deployment die zwei Konfigurationsdateien *passwords.yml* und *globals.yml* ein. Entsprechende Template-Dateien sind im Paket *kolla-ansible* bereits enthalten und müssen nur noch nach */etc/kolla/* kopiert und angepasst werden:

```
cp -r /usr/share/kolla-ansible/etc_examples/
                                        kolla /etc/kolla/
```

Darüber hinaus benötigt *kolla-ansible* ein Ansible Inventory, in dem die Nodes und ihre Rolle in der OpenStack-Landschaft aufgelistet sind. Dafür kopiert man das mitgelieferte Inventory für das All-in-one-Deployment in das momentane Verzeichnis, um später nicht den gesamten Pfad angeben zu müssen:

```
cp /usr/share/kolla-ansible/ansible/
                                inventory/all-in-one .
```

Anpassen der Konfiguration geschieht in der Datei */etc/kolla/globals.yml*, in der die folgenden Parameter anzupassen sind:

```
network_interface: "ens33"
neutron_external_interface: "ens36"
kolla_internal_vip_address: "192.168.178.150"
enable_haproxy: "no"
openstack_release: "4.0.0"
```

Die tatsächlichen Namen der Netzwerk-Interfaces verrät *ip a*. Hier ist *ens33* die mit IP versehene Schnittstelle, *ens36* besitzt keine Konfiguration. *kolla_internal_vip_address* enthält die Adresse des ersten Interface. Im All-in-one-Szenario ist HA-Proxy ausgeschaltet. Als Release kommt OpenStack 4.0.0 Ocata zum Einsatz.

Ansible erfüllt seine Aufgabe

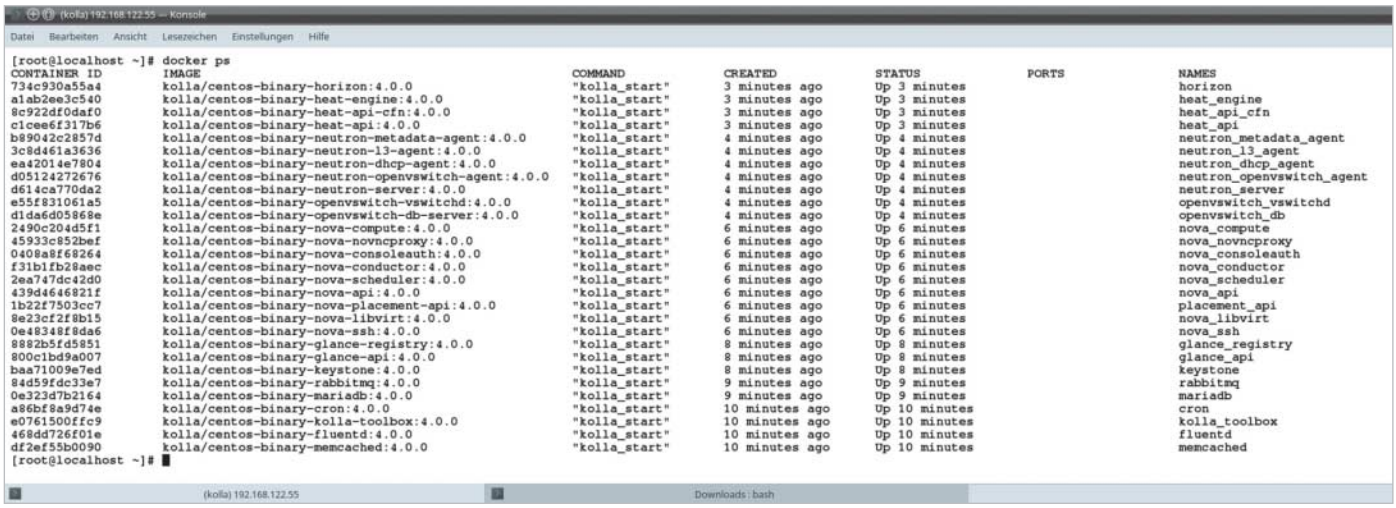
Alle OpenStack-Services arbeiten mit verschiedenen Datenbanktabellen, Usern und Kennwörtern. Die Passwörter liest Kolla beim Deployment aus der Datei *passwords.yml*. Ein Aufruf von *kolla-gen pwd* füllt die Datei mit zufällig generierten Passwörtern.

Alle weiteren Vorbereitungen kann Kolla automatisch übernehmen. Bootstrap präpariert die Server so weit, dass anschließend die Docker-Container darauf verteilt werden können:

```
kolla-ansible -i all-in-one bootstrap-servers
```

Weiterhin kann das Tool einige automatisierte Checks durchführen:

```
kolla-ansible -i all-in-one prechecks
```



In einem All-in-one-Deployment landen alle für OpenStack relevanten Container auf einem Host. Die Ausgabe von *docker ps* listet sie auf. Ohne Automatisierung müssten die Dienste händisch installiert und gepflegt werden (Abb. 2).

Die benötigten Docker-Images werden von Docker Hub heruntergeladen, sofern keine andere Docker-Registry angegeben wurde. Die Images dort baut die Kolla-Community:

```
kolla-ansible -i all-in-one pull
```

Schlussendlich kann der Roll-out ausgeführt werden. Nachdem dieser durchlaufen ist, stehen die OpenStack-Services zur Benutzung bereit. Folgender Befehl startet den Prozess:

```
kolla-ansible -i all-in-one deploy
```

Je nach Hardware dauert dieser Vorgang 20 bis 60 Minuten. Danach sollte eine Zusammenfassung von Ansible zu sehen sein – optimalerweise ohne Fehler:

```
PLAY RECAP *************************
localhost : ok=251 changed=124 ↵
                    unreachable=0 failed=0
```

Zugriff über Konsole und Dashboard

War die Installation bis hier erfolgreich, erreicht man unter der IP, die zuvor in der Konfigurationsdatei bei *kolla_internal_vip_address* eingetragen wurde, das OpenStack-Dashboard Horizon. Nutzername zum Login ist *admin*, das Passwort findet sich in der erwähnten Datei */etc/kolla/passwords.yml* unter *keystone_admin_password*. Auf der Kommandozeile gibt der Befehl *docker ps* einen Überblick über die laufenden Container.

Im Folgenden werden kurz die Schritte erklärt, die möglichst schnell zu einer laufenden virtuellen Maschine in der OpenStack-Umgebung führen. Dieser Abschnitt ist nicht Kolla-spezifisch und entspricht nur teilweise der empfohlenen Vorgehensweise. Um von der Kommandozeile mit dem OpenStack-Client arbeiten zu können, sollte man ein paar Umgebungsvariablen setzen. Dies geschieht am besten mit einer vordefinierten OpenRC-Datei, die man aus dem Horizon-Dashboard erhält, indem man rechts oben auf „admin“ und anschließend auf „OpenStack RC File v3“ klickt.

Umgebung wie in jeder OpenStack-Installation

Wieder in der Kommandozeile setzt folgender Befehl die Umgebungsvariablen für den Kommandozeilenclient:

```
source <pfad-zur-datei>/admin-openrc.sh
```

Das Erstellen einer virtuellen Maschine erfordert ein virtuelles Netzwerk (entspricht einem physischen Netzwerk) und ein Subnetz (entspricht einem IP-Netzwerk).

```
openstack network create --share \
 --provider-physical-network physnet1 \
 --provider-network-type flat testing_net
openstack subnet create --network testing_net \
 --subnet-range 192.168.122.128/25 testing_ ↵
                                        subnet
```

Als Image für die virtuelle Maschine empfiehlt sich CirrOS, ein abgespecktes Linux-Image, das für solche Testzwecke gedacht ist:

```
wget https://download.cirros-↵
cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img

openstack image create cirros \
 --file cirros-0.3.5-x86_64-disk.img
```

Die Zuteilung von Ressourcen zu virtuellen Maschinen geschieht in OpenStack über Flavors. Zum Testen eignet sich eine Instanz mit einer virtuellen CPU und 256 MByte RAM:

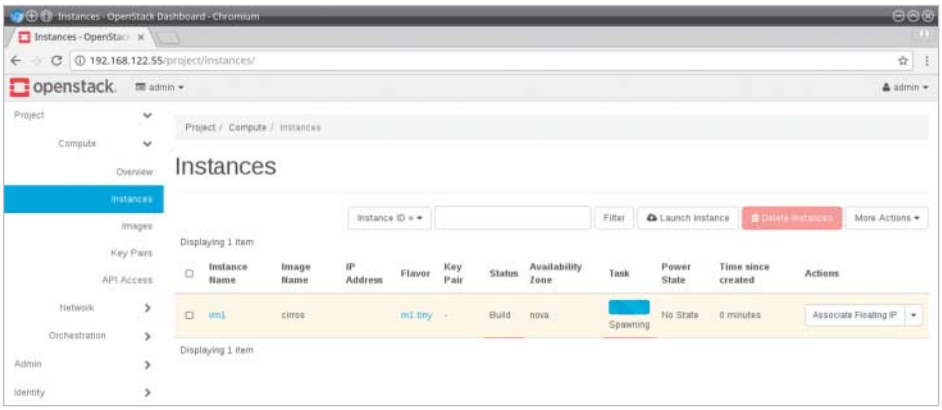
```
openstack flavor create --public \
 --vcpus 1 --ram 256 m1.tiny
```

Nach diesen Vorarbeiten lässt sich die VM einfach starten:

```
openstack server create --flavor m1.tiny \
 --nic net-id=testing_net --image cirros vm1
```

Über Horizon kann man unter „Project –> Compute –> Instances –> vm1 –> Console“ via VNC auf die Konsole der virtuellen Maschine zugreifen, sobald diese eingerichtet ist.

Ein auf mehrere Server verteiltes OpenStack-Deployment mit Kolla ist kaum komplizierter. Zum einen muss man dazu das Inventory-File anpassen. Hier werden die einzelnen Nodes gruppiert und es wird definiert, welcher Service wo installiert wird. Wie für das All-in-one-Deployment gibt es auch für ein Multi-Node-Deployment ein vorgefertigtes Beispiel unter */usr/share/kolla-ansible/ansible/inventory/*. Des Weiteren sind Einstellungen bezüglich Hochverfügbarkeit und Netzwerk-Setup in *globals.yml* einzustellen und gegebenenfalls sinnvoll anzupassen.



Das Dashboard zeigt alle Einstellungen von OpenStack. Neu gestartete Instanzen erscheinen sofort mit ihrem Status (Abb. 3).

Ein Blick in die Zukunft

Kolla ist ein ausgereiftes Tool zum automatischen Setup eines OpenStack-Clusters und trotzdem noch nicht am Ende der Entwicklung angekommen. *kolla-ansible* ist weiterhin eines der aktivsten Projekte der OpenStack-Community, doch gibt es seit einem guten Jahr auch *kolla-kubernetes*, das auf das Container-Orchestrierungssystem Kubernetes statt auf Ansible setzt. Diese Kombination ist zurzeit noch nicht produktionsreif, hat aber erste Hürden bei der Entwicklung genommen. Auf dem letzten Project Team Gathering konnten die Entwickler Fortschritte in wichtigen Designfragen machen. Kolla bleibt damit seiner Linie treu, sich am Puls der Zeit zu befinden und neueste Entwicklungen aus verschiedenen Bereichen zu integrieren, um ein einfach zu bedienendes und gleichzeitig mächtiges Framework zum Deployment von OpenStack zur Verfügung zu stellen. (jab)

David Rabel

ist bei der B1 Systems als Consultant & Trainer im Linux- und OpenStack-Umfeld tätig.

Robin van der Linden

ist Consultant bei der B1 Systems mit Schwerpunkt OpenStack.

Christian Berendt

betreut als Cloud Solution Architect bei der B1 Systems das Thema OpenStack.

Literatur

[1] Martin Gerhard Loschwitz; Infrastrukturmanagement; Vereinfachter Hindernisparcours; Einstieg in OpenStack finden; *iX* 16/2016, S. 42

Alle Links: www.ix.de/ix1711130