Kubernetes: Eine effiziente Automatisierungslösung für Container

Michael Schulze, Opitz Consulting Deutschland

Dass modulare Infrastrukturen herkömmliche Serversysteme, aber auch komplexe virtuelle Umgebungen ablösen, ist allgemein bekannt. Dabei ist ein Trend hin zu verteilter und skalierbarer Software mit integriertem Monitoring zu beobachten. Das schlägt sich konkret nieder in der Nutzung von Microservices und der Kapselung von Anwendungen zur Erfüllung von Mandantenfähigkeit und Security-Aspekten. Wir haben es hier mit einem Paradigmenwechsel zu tun, bei dem Container-Architekturen wie Docker eine zunehmende Rolle spielen und monolithische Systemlandschaften nach und nach ablösen. Denn für eine modulare Verteilung ist es notwendig, Anwendungen zustandslos und versioniert über Container zur Verfügung zu stellen. Ein weiterer wichtiger Punkt ist die Skalierbarkeit, um unterschiedlichen Ressourcenanforderungen gerecht zu werden.

Diese Entwicklung ist in allen Software-Lösungen zu beobachten, und auch im Cloud-Umfeld ist sie längst angekommen. Modulare Lösungen benötigen zudem Ausfallsicherheit, Lastverteilung und eine gezielte Verwaltung der Ressourcen. Vor diesem Hintergrund sind Orchestrierungslösungen für Container-Architekturen entstanden, wie die populäre Orchestrierungsplattform Kubernetes, mit der sich der Artikel im Detail beschäftigen wird.

Grundlagen der Container-Architektur

Im Gegensatz zur Virtualisierung mit Hypervisoren, wo eine VM ein komplettes Betriebssystem für die Anwendungen bereitstellt, teilen sich in der Containerarchitektur mehrere Container den OS-Kernel des darunterliegenden Betriebssystems und nutzen somit dessen Infrastruktur. Dabei werden zur Container-Laufzeit alle notwendigen Prozesse der Anwendung isoliert im OS gestartet und verwaltet. Der Container selbst entspricht nur einer Datei, in der Anwendungscode, die Konfiguration und die Abhängigkeiten (z. B. notwendige Bibliotheken) gebündelt provisioniert werden. Dabei verwaltet eine zum OS kompatible Container-Plattform die bereitgestellten Containerdateien *(siehe Abbildung 1)*. Das spart Platz und ermöglicht die modulare, zustandlose sowie versionierte Bereitstellung und Kapselung von Anwendungen.

Die Container Runtime selbst nutzt zur Laufzeit dann die Inhalte des Containers in Interaktion mit dem darunterliegenden Betriebssystem. Dabei können mithilfe implementierter Netzwerkmechanismen verschiedene Container Runtimes miteinander kommunizieren *[1]*.

Container-Lösungen gibt es in verschiedenen Ausprägungen schon länger, aber erst Docker als Open-Source-Container-Implementierung revolutionierte ab 2013 den Markt und entwickelte sich seither zu einem Quasi-Standard insbesondere in Development-Umgebungen.

Diese Entwicklung machte den Weg frei für eine flexible, serverübergreifende Bereitstellung von modularisierten Anwendungsstrukturen (Microservices) auf verschiedenen Plattformen und Umgebungen.

Insgesamt vereinfacht sich durch den Einsatz von Container-Technologie das Provisionieren, die Änderung und die Erweiterung komplexer Anwendungen erheblich.

Neben der Container-Technologie Docker gibt es weitere Container Engines wie rkt, LXC, LXD, die jedoch bis heute in ihrer Verbreitung nicht die Bedeutung von Docker erreichen konnten *[2]*.

Kubernetes: eine Einführung

Wie bereits erwähnt, nimmt im Entwicklungsbereich die Notwendigkeit, Anwendungen einer großen Anzahl von Benutzern zur Verfügung zu stellen und komplexere Strukturen abzubilden, stetig zu. Diesen erweiterten Anforderungen begegnet man schon seit Längerem mit Container-Orchestrierungswerkzeugen. Durch ihren Einsatz wird es möglich, Cluster von Containern sicher zu betreiben sowie Lastverteilung und Skalierungsoptionen zu realisieren. Lösungen wie Docker-Swarm etablierten sich, wurden allerdings von der schnellen Entwicklung des Container-Orchestrierungstools Kubernetes überholt.

Kubernetes verwendet Docker als Standard Container Engine. Mit Kubernetes können jedoch auch weitere Container Engines wie rkt verwendet werden. Die enthaltenen Komponenten bieten eine effiziente Verwaltung des automatisierten Betriebs, des Deployments, Skalierungsoptionen sowie Monitoring und Wartung von Container-basierten Anwendungen. Durch die Clusterfunktionalität und die

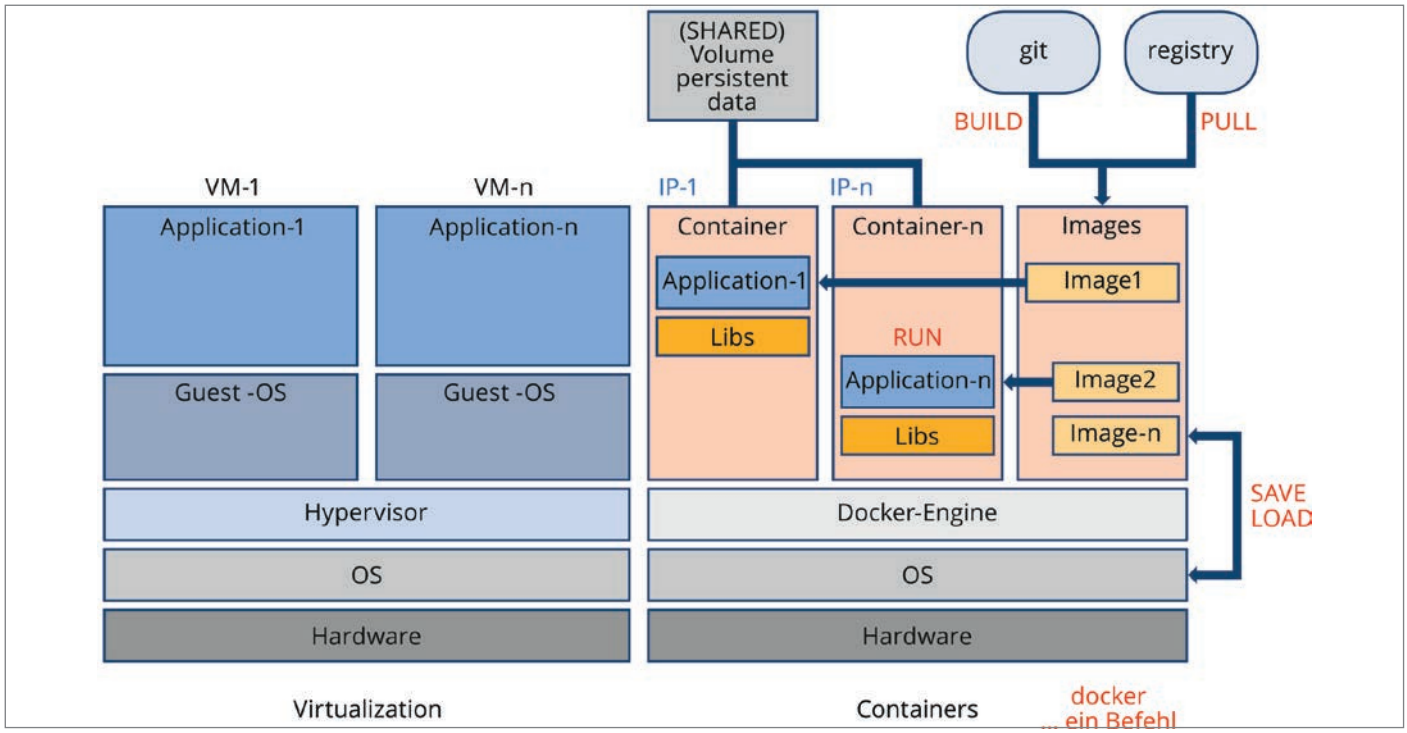

Abbildung 1: Aufbau einer Container-Architektur (Quelle: Michael Schulze)

Möglichkeit des Multi-Node-Betriebs können Container innerhalb des Clusters als eine Einheit verwaltet werden [3] [4].

Die Geschichte von Kubernetes

Kubernetes wurde ursprünglich von Google entwickelt und intern eingesetzt (Projektname: Borg). Später stellte Google das Projekt in abgewandelter Form der Open-Source-Gemeinde zur Verfügung. Die Cloud Native Computing Foundation (CNCF), bei der Google Mitglied war, spielte für diese Entscheidung eine große Rolle. In dieser Stiftung sind mittlerweile so gut wie alle wichtigen Cloud Player vertreten [5]. Deshalb findet man Kubernetes-Lösungen heute auch in allen führenden Cloud-Plattformen wieder. Beispiele für den produktiven Einsatz von Kubernetes finden sich bei der New York Times, Philips, SAP und SaaS-Anbietern wie Box und GitHub sowie im Produkt Pokemon-Go [6].

Kubernetes On-Premises

Die Core-Kubernetes-Pakete sind für die gängigsten Distributionen verfügbar. So lassen sich On-Premises komplexe Kubernetes-Cluster-Umgebungen (multinode) installieren. Neben der Core-Installation gibt es weitere auf Kubernetes basierende Management-Plattformen. OpenShift ist beispielsweise eine kommerzielle Lösung der Firma Red Hat, die (auch im DevOps-Kontext) um zusätzliche Komponenten erweitert wurde und als ganzheitliche Lösung zur Verfügung steht [7]. Ein Beispiel aus diesem Umfeld ist OpenDevStack, eine Open-Source-Plattform zur schnellen Bereitstellung von Microservices im DevOps-Kontext [8]. Für einen ersten Einstieg eignen sich One-Node-Kubernetes-Lösungen wie „minikube" oder „minishift", die eine auf bestimmte Ressourcen begrenzte, vollständige Kubernetes-Installation ermöglichen.

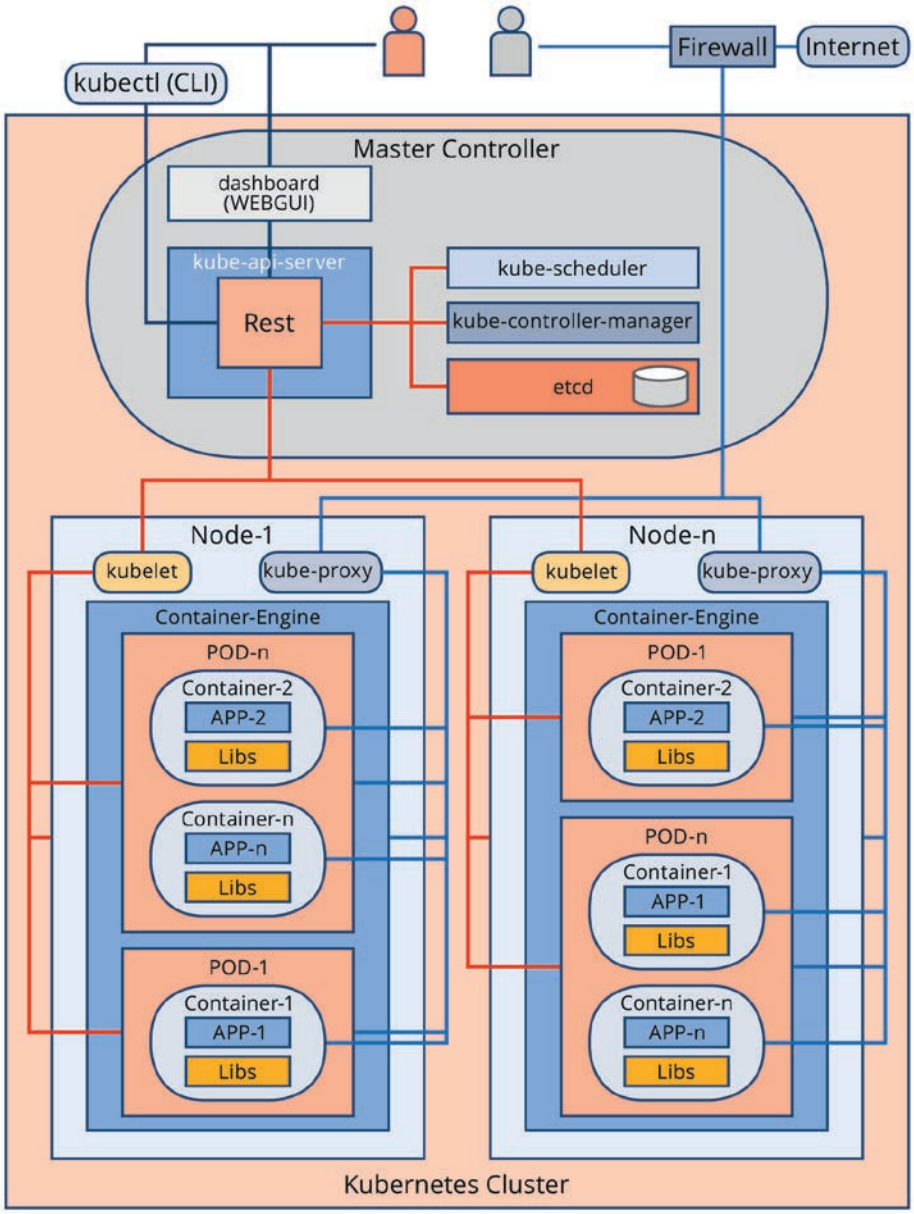

Abbildung 2: Kubernetes-Cluster (Quelle: Michael Schulze)

Kubernetes-Lösungen in der Cloud

Dass Kubernetes im Zeichen der Zeit steht, sieht man auch daran, dass aktuell führende Cloud-Anbieter unterschiedliche Lösungen in verschiedenen Ausprägungen für die Service-Bereitstellung von Kubernetes anbieten. Diese Cloud-Betreiber gehören der CNCF an, der Foundation, der das Kubernetes-Projekt ursprünglich von Google zur Verfügung gestellt wurde. Unter anderem die folgenden Kubernetes-Cloud-Lösungen stehen hier zur Verfügung [9]:

- Google: „GKE" (Google Kubernetes Engine)
  https://cloud.google.com/kubernetes-engine
- Amazon AWS: „EKS" (Elastic Kubernetes Service)
  https://aws.amazon.com/de/eks
- Microsoft Azure: „AKS" (Azure Kubernetes Service)
  https://azure.microsoft.com/de-de/services/kubernetes-service
- Oracle Cloud: „OKE" (Oracle Kubernetes Engine)
  https://cloud.oracle.com/containers/kubernetes-engine
- IBM Cloud Kubernetes Service
  https://www.ibm.com/de-de/cloud/container-service

Kubernetes – Grundlagen der Architektur

Kubernetes besteht aus verschiedenen Core-Komponenten, die sich in Master- und Node-Komponenten sowie in verschiedene Add-ons aufteilen lassen [10].

Diese werden im Folgenden näher erläutert.

Master Core Components

Der Master-Server besteht aus verschiedenen Komponenten:

- kube-controller-manager: Über den Controller Manager wird der Cluster gesteuert. Er kommuniziert mit den Node(s) und POD(s) und stellt die Ausfallsicherheit sowie die Skalierungseinstellungen (Replica) im Cluster sicher.
- kube-api-server: Über den im Master-Server integrierten API-Server einer weiteren Komponente wird eine REST-Schnittstelle zur Verfügung gestellt. Über diese Schnittstelle läuft die interne und externe Kommunikation des Kubernetes-Clusters. Beispiele wären hier die Status-Ermittlung von Ressourcen (GET) oder die Erstellung neuer Ressourcen (POST).
- etcd: Der API-Server kommuniziert mit dem etcd, dem Backend-Datenspeicher von Kubernetes. Diese Komponente besteht aus einem verteilten Open-Source-Schlüsselwertspeicher, in dem Kubernetes alle REST-Schnittstellen-Objekte und seine Zustände sichert. Der Schlüsselwertspeicher stellt das Speichern und das Replizieren von Daten sicher und bildet die Kernfunktion von Kubernetes.
- kube-scheduler: Der Scheduler entscheidet je nach Auslastungskriterien der ermittelten Statusinformationen, auf welchem Server-Knoten Ressourcen zur Verfügung stehen, und verteilt dann entsprechend *[11]*.

Node Core Components

Ein Node ist die Server-Komponente, die Ressourcen wie CPU, RAM, Netzwerk-Infrastruktur etc. für die dort provisionierten Container Runtimes zur Verfügung stellt. Nodes sind über ein Netzwerk miteinander verbunden und stellen so die Cluster-Kommunikation mit dem Master-Server sicher.

- kube-proxy: Für die Netzwerkkommunikation innerhalb des Clusters ist die Komponente kube-proxy zuständig. Sie enthält außerdem Routing-Funktionen, die Netzwerkkommunikation von und nach außen ermöglichen.
- kubelet: Auf jedem Node arbeitet ein Agent (kubelet) als Hauptprozess, der den Node und die dort laufenden Container-Runtime-Umgebungen hinsichtlich ihrer Ressourcen und Zustände überwacht und Indikatoren für den Master Controller zur Beurteilung des aktuellen Cluster-Status liefert. kubelet kommuniziert dabei mit dem kube-api-server auf dem Master-Server. Fällt eine Teilkomponente, etwa ein Container, aus, wird er nach den definierten Richtlinien auf einer anderen Ressource (Node/POD) wieder zur Verfügung gestellt.

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
      matchLabels:
          app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
      metadata:
          labels:
              app: nginx
      spec:
          containers:
          - name: nginx
              image: nginx:1.7.9
              ports:
              - containerPort: 80
```

Listing 1

POD

Ein POD (Deutsch: Hülse) ist die kleinste Einheit in einem Kubernetes-Cluster und kann als logische Host-Einheit verschiedene Anwendungscontainer enthalten. Diese Abstraktionsschicht ist auch notwendig, da mit Kubernetes verschiedene Container-Technologien, wie Docker und rkt, verwaltet werden können. Ein POD kann einen oder mehrere Container enthalten. Jeder POD besitzt eine eigene IP-Adresse, die enthaltenen Container nutzen diese gemeinsam *(siehe Abbildung 2)*.

Administrationstools CLI und GUI

kubectl ist die zentrale Commandline-Interface-(CLI)-Steuerung, um im Kubernetes- Cluster enthaltene Komponenten, zum Beispiel PODs, zu administrieren. Ebenso sind hier Möglichkeiten der Skalierung enthalten. kubectl-Befehle kommunizieren direkt mit der REST-Schnittstelle des Master-Servers (kube-api-server). Über diese Schnittstelle werden die Befehle dann direkt auf den Nodes ausgeführt.

Dashboard

Das in Kubernetes enthaltene Dashboard ist die Web-GUI-Schnittstelle, die einen großen Teil der Kubernetes-Funktionalität visualisiert. Auch hier können analog zu den verfügbaren CLI-Tools Parameter angepasst und neu definiert werden. Das Dashboard bietet als Steuerzentrale von Kubernetes umfangreiche Möglichkeiten sowie Monitoring-Funktionalitäten und ermöglicht so einen Überblick über die Kubernetes-Umgebung. Dabei nutzt es die REST-Schnittstelle auf dem Master Controller *[12]*.

YAML-Provisionierung am Beispiel von nginx

YAML („yet another markup language“) ist eine vereinfachte Beschreibungssprache und wird für die Definition von Deployment-Beschreibungen in Kubernetes verwendet. YAML bietet im Kubernetes-Kontext den Vorteil einer textbasierten,

versionierbaren Definition von Deployments und ihrer Parameter. Im YAML-File sind alle Informationen enthalten, die erforderlich sind, um den Container in der Kubernetes-Umgebung zu provisionieren und dort auszuführen. Da Kubernetes darauf basiert, komplexe Container-Applikationsumgebungen mit definierten Voreinstellungen in Kubernetes zu deployen, zu parametrisieren und zu ändern, vereinfacht das den Deployment-Prozess erheblich.

Das folgende Beispiel beschreibt die Bereitstellung einer Web-Tier-Umgebung auf einem nginx-Webserver und verdeutlicht die Funktionsweise der Mechanismen der Deployment-Bereitstellung in Kubernetes. Insbesondere das Zusammenspiel mit kubectl wird dadurch transparenter.

In der dargestellten Deployment-Beschreibung *(siehe Listing 1)* bestimmt das Image nginx:1.7.9, welches Container-Image aus dem Repository angewendet werden soll. Dieses wird dann in Kubernetes deployt und ausgeführt. Dabei sollen zwei Instanzen entstehen (Skalierungsoption) und der Webservice soll später auf Port 80 erreichbar sein.

Deployment-Vorgang

Die vorbereitete Deployment-Beschreibung durchläuft in der Anwendung für unser Beispiel die folgenden Schritte:

- Einlesen der Deployment-YAML-Beschreibung aus der Online-Ressource.
- Container-Image ermitteln (Repository).
- Container Runtime auf Node(s) und Pod(s) bereitstellen, entsprechend der definierten Skalierung, und starten.
- Anwendung steht nun zur Verfügung.

Die Umsetzung der genannten Schritte und die Kontrolle erfolgen durch das CLI-Interface kubectl *(siehe Listing 2)*.

Wie man sieht, wurde der Webserver, wie zuvor definiert, in zwei Instanzen bereitgestellt.

```
kubectl apply -f https://k8s.io/examples/application/deployment.yaml
kubectl get pods

Output:
deployment.apps "nginx-deployment" created

NAME                                READY     STATUS     RESTARTS  AGE
nginx-deployment-76bf4969df-279r4   1/1       Running        0      1h
nginx-deployment-76bf4969df-d796n   1/1       Running        0      1h
```

Listing 2

Zusammenfassung und Ausblick

Im Artikel wurde das Kubernetes Framework mit seinen Komponenten und Funktionalitäten vorgestellt; wir haben am Beispiel von nginx gesehen, wie einfach es ist, eine kleine Applikation im Cluster zur Verfügung zu stellen und Skalierungsoptionen anzuwenden. Mithilfe von YAML-Files als versionierbarer Grundlage für Deployment-Definitionen in Kubernetes können auch komplexe Container-Applikationsstrukturen gut verwaltet werden.

Die gestiegenen Anforderungen des Markts mit der Notwendigkeit einer schnellen, ausfallsicheren und flexibel skalierbaren Bereitstellung von Anwendungen über Container-Lösungen und deren Verwaltung hat dazu geführt, dass sich Kubernetes als produktionsreife Orchestrierungslösung bereits etabliert hat.

Kubernetes bietet Cluster- sowie Skalierungsfunktionalitäten und stellt eine effiziente Managementlösung für Container-Landschaften dar. Laut einer Erhebung der Crisp-Research AG planen 57% der deutschen Unternehmen im Jahr 2019 den Einsatz von Kubernetes *[13]*. Für Kubernetes stehen On-Premises-Installationsoptionen sowie diverse Cloud-Varianten aller namhaften Anbieter zur Verfügung.

Damit wird Kubernetes in Zukunft sicher noch an Bedeutung gewinnen und sich langfristig weiter als Standard für Container-Cluster-Orchestrierung etablieren.

Orchestrierungswerkzeuge für Container-Lösungen einzusetzen, wäre der nächste logische Schritt, um Container-Umgebungen verwaltbar zu machen. Kubernetes spielt hier eine besondere Rolle. Als Open-Source-Projekt der CNCF steht das Projekt zudem unter dem Einfluss namhafter Cloud-Anbieter. Das garantiert eine permanente Weiterentwicklung.

Quellen

[1] *https://www.expertenderit.de/blog/7-linuxbasierte-container-plattformen-im-%C3%BCberblick*
[2] *https://www.docker.com/resources/what-container*
[3] *https://kubernetes.io/docs/reference/kubectl/overview*
[4] *https://kubernetes.io/docs/concepts*
[5] *https://www.cncf.io/blog/2017/12/06/cloud-native-technologies-scaling-production-applications*
[6] *https://blog.risingstack.com/the-history-of-kubernetes*
[7] *https://kubernetes.io/docs/concepts/overview/components*
[8] *https://www.redhat.com/de/technologies/cloud-computing/openshift*
[9] *https://kubernetes.io/docs/concepts/cluster-administration/cloud-providers*
[10] *https://coreos.com/etcd*
[11] OpenDevStack stellt Werkzeuge zur schnellen Provisionierung von Anwendungen bereit und wird bei Opitz Consulting bereits intern sowie auch bei einigen Kunden erfolgreich eingesetzt. *https://www.opitz-consulting.com/portfolio/software-development/devops-services/opendevstack.html*
[12] *https://github.com/kubernetes/dashboard*
[13] *https://www.crisp-research.com/kubernetes-und-cloud-native-trends-2019-das-jahr-de*

Michael Schulze
michael.schulze@opitz-consulting.com