KUBERNETES

FEBRUAR 23, 2021

„Kubernetes ist der Steuermann für die Beladung deines Container-Schiffes. Er sagt dir, wo du welchen Container am besten platzierst, um höchste Effizienz gewährleisten zu können. Er ist der Manager für all deine Microservice-Systeme!“

Kubernetes (oder auch k8s) ist eine Open-Source-Plattform, welche von Google Ingenieuren entwickelt und als Spende an die Cloud Native Computing Foundation übergeben wurde. Sie wird zur Orchestrierung von Systemen mit vielen einzelnen Microservices genutzt und automatisiert die Verwaltung containerbasierter Anwendungen. Es ist quasi DAS Tool für ein effizientes Management von Containern.

Als Orchestrierungstool automatisiert Kubernetes die Verwaltung und Steuerung komplexer Container-Systeme. Der aktuelle Status des Systems wird mit Hilfe des Kubernetes Dashboard oder der CLI überwacht. In der Praxis wird das bestehende Monitoring häufiger um Prometheus, Alertmanager und Grafana* erweitert. Außerdem verändert Kubernetes das System so, dass stets der gewünschte Zustand (desired state) erreicht wird.

*Prometheus ist das Überwachungs- und Warnmeldungs-Toolkit. Die Warnmeldungen werden mit Hilfe des Alertmanager verarbeitet und via Grafana visualisiert.

INHALTSVERZEICHNIS

1. ZUSAMMENSPIEL VON KUBERNETES UND (DOCKER-) CONTAINERN

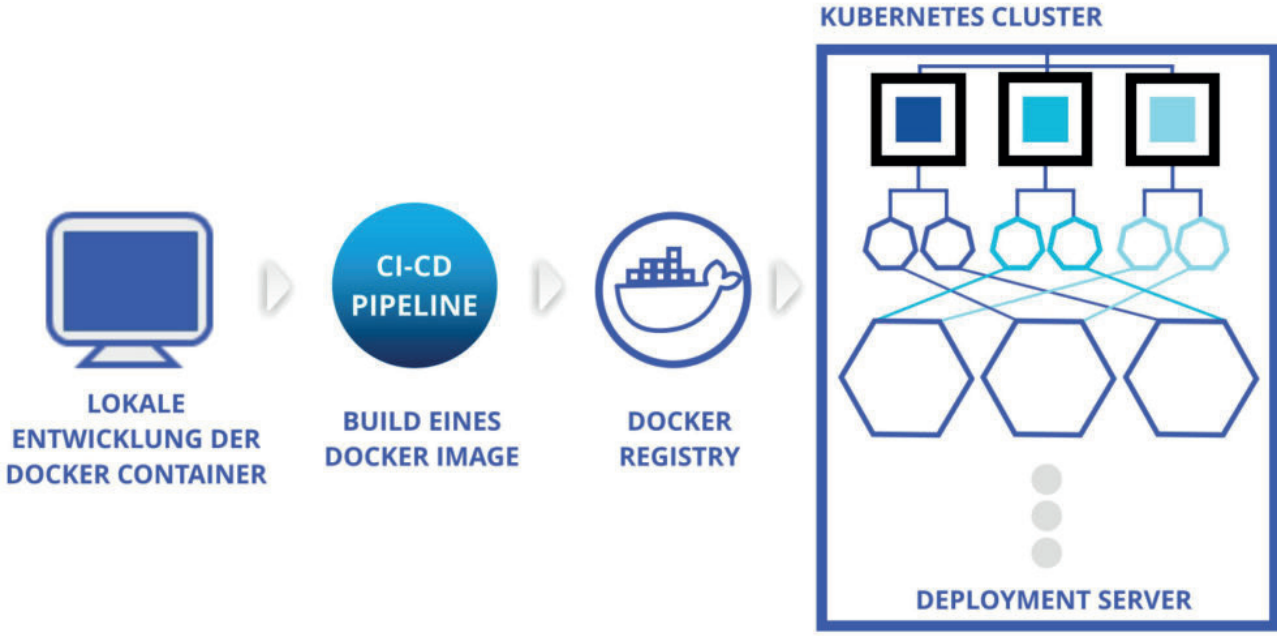

ABBILDUNG: VOM CONTAINER ZUM KUBERNETES CLUSTER

Der CI-CD-Build-Prozess übernimmt das Verpacken in Docker-Container und stellt sie den Servern bereit. Sobald diese Container auf einem der Produktionsserver oder einem der Staging-Server bereitgestellt wurden, sollten diese überwacht werden. Ab diesem Zeitpunkt ist ein Orchestrierungstool – also z.B. Kubernetes – nötig. Kubernetes übernimmt die Überwachung, Skalierung und Verteilung. Kubernetes bietet also die komplette Orchestrierung aller Container, die wir auf unseren Servern bereitgestellt haben. Orchestrierungstools sind somit eine optimale Ergänzung für die Arbeit mit Containisierungstools wie Docker.

2. EINSATZGEBIET VON KUBERNETES

Angenommen man hat ein großes und komplexes System, das aus vielen einzelnen kleinteiligen Services (Microservices) besteht. Ergänzend hat man die Erwartung, dass noch weitere unzählige Microservices hinzukommen könnten. Man möchte natürlich gewährleisten, dass die einzelnen Anwendungen nicht ausfallen und zudem soll das System modular erweiterbar sein.

Genau in diesem Fall ist die Einführung eines Kubernetes Clusters sinnvoll, da eine hohe Skalierbarkeit und Ausfallsicherheit mit einem solchen Orchestrierungstool sichergestellt werden kann. Natürlich stellt dies keine Garantie dar und Kubernetes ist keine Allzwecklösung. Ergänzend kann man neben einem eigenen Cluster auch eine Cloud-Plattform wie Azure, GCP oder AWS verwenden.

Falls im Unternehmen bereits ein Kubernetes Cluster vorhanden ist, ist die Hürde deutlich geringer Systeme entsprechend zu überführen. Falls jedoch noch kein Cluster vorhanden ist, stellt sich grundsätzlich die Frage, ob für den jeweiligen Anwendungsfall das Aufsetzen eines Clusters in einem sinnvollen Nutzen-Kosten-Verhältnis steht.

BEISPIELSZENARIO

Mein gewünschter Zustand ist, dass die Anwendungen X kontinuierlich läuft. Das wiederum teile ich meinem Orchestrator mit. Somit steht der desired state fest. Mit Hilfe von Kubernetes erschaffe ich ein System, indem ich viele Knoten (Nodes) habe und die Software könnte auf jeglichem Node laufen.

Welche Vorteile habe ich?
Falls ein Node ausfällt, auf dem meine Anwendung X läuft, ist das kein Problem! Durch meinen Orchestrator gibt es nicht nur EINEN Node, worauf die Software läuft. Somit stellt der Ausfall eines Nodes keinerlei Probleme dar, denn meine Anwendung X läuft auf einem anderen Node weiter.

Hinweis: Für ein System mit wenigen Microservices ist das Aufsetzen eines Kubernetes Clusters nicht sinnvoll! Das Beispiel soll lediglich verbildlichen, in wie fern Kubernetes Abhilfe leisten kann.

3. VORTEILE VON KUBERNETES

Zusammengefasst bietet das Open-Source Orchestrierungstool Kubernetes folgende Vorteile:

Rollouts und Rollbacks von Anwendungen werden automatisiert (Rolling Updates)
In Deployments werden Container-Images und die Anzahl der benötigten Replikationen von Kubernetes zusammengefasst. Kubernetes überwacht die Deployments nicht nur selbstständig, sondern sorgt zudem dafür, dass die Anwendungen ohne Unterbrechungen durchlaufen (sollte dies gewünscht sein). Falls jedoch einzelne Anwendungen geupdatet werden, sorgt Kubernetes für einen Rollout. Möchte man diesen Prozess rückgängig machen, ist dies ebenfalls möglich – das ist das sogenannte Rollback.

Computer-Ressourcen werden optimiert
Der Entwickler gibt den Rahmen vor, indem er beispielsweise die Höchstwerte der Prozessorleistung (CPU) oder des Speicherplatzes der jeweiligen Container festlegt, und Kubernetes verteilt die Anwendung. Hierdurch kann eine effizientere Ressourcennutzung gewährleistet werden.

Service Discovery ist integriert
Kubernetes erkennt Netzwerkverbindungen zu anderen Containern und auch ins Internet völlig selbstständig. Mittels DNS ist eine voll funktionsfähige Service Discovery in Kubernetes integriert.

Hohe Skalierbarkeit des Systems
Kubernetes stellt drei Skalierungstools zur Verfügung: Cluster Autoscaler, Vertical Autoscaler und Horizontal Autoscaler. Der Cluster Autoscaler wird zur Skalierung der Nodes genutzt, hingegen skaliert der Vertical Autoscaler die Größe der Pods und der Horizontal Autoscaler die Anzahl der Pods. Die hohe Skalierbarkeit des Systems ist gegeben, wenn die drei Skalierungstools bedarfsentsprechend konfiguriert werden.

Hohe Ausfallsicherheit der Anwendungen
Die bereits angesprochenen Vorteile zeigen, dass Kubernetes für die fortlaufende Funktionsfähigkeit des gesamten Systems sorgt. Fällt hypothetisch eine Anwendung aus, sorgt Kubernetes dafür, dass die Anwendung in einem anderen Pod weiterläuft.

Im Folgenden wird erläutert, welche Elemente für welche Leistungen zuständig sind.

4. FUNKTIONSWEISE VON KUBERNETES

Kubernetes benötigt eine Beschreibung welche Container und wie viele davon gebraucht werden. Alles andere erfolgt automatisiert, da es die Beschreibung des desired states darstellt.

Das Kubernetes-System besteht aus verschiedenen Elementen, die im Folgenden näher erläutert werden:

Kubernetes Cluster
Das Kubernetes Cluster ist das Gesamtsystem bestehend aus Master und den Nodes.

Master
Der Master steuert und kontrolliert alle Teile innerhalb des Kubernetes-Clusters wie beispielsweise die Nodes und den Kubernetes Ingress. Ergänzend weiß der Master auf welchen Pods welche Services laufen. Falls also der Client eine Anfrage startet, führt der Master diesen direkt zum entsprechenden Zielort.

Ingress
Die Nodes eines Kubernetes-Clusters sind nicht direkt im öffentlichen Internet zugänglich, um Risiken bei der Nutzung von Kubernetes zu minimieren. Ingress sorgt dafür, dass partiell Services, welche öffentlich zugänglich sein sollten, auch zugänglich sind. Sie stellt Netzwerkrouten außerhalb des Clusters für ausgewählte Services zur Verfügung und ist somit eine Lösung für die Handhabung verschiedener Schnittstellen (APIs).

Damit Ingress als Ressource genutzt werden kann, muss ein Ingress-Controller auf dem Kubernetes Cluster installiert werden. Dieser Controller dient als Verbindungsstück zwischen Cluster und den öffentlich zugänglichen Schnittstellen. Keine Sorge: Die meisten Cloud-Anbieter, welche Kubernetes hosten, stellen einen speziellen Ingress-Controller bereit. Diese speziellen Ingress-Controller sind angepasst an den jeweiligen Cloud-Anbieter und bieten somit auch unterschiedliche Funktionalitäten an.

Kurzgefasst: Ingress bildet die Schnittstelle zur Außenwelt und fungiert als Load Balancer sowie Verteiler zwischen den Services.

Nodes
Nodes beinhalten wiederum die Pods.

Pods
Hierbei handelt es sich um die kleinste Kubernetes-Einheit. Diese beinhaltet mindestens einen Container. Ein Pod entspricht in etwa einem Virtuellen Server auf dem Node. In einem Pod werden die jeweiligen Container automatisch gestartet und gestoppt. Der Container beinhaltet wiederum mindestens eine Anwendung.
Das Klassifikationsmodell zeigt, dass die Datenwerte nur einer Klasse von einer begrenzten Anzahl an Klassen zugehören können – in diesem Fall handelt es sich um ein binäres Klassifikationsmodell, welches lediglich zwei Klassen als Output-Werte ausgeben kann.

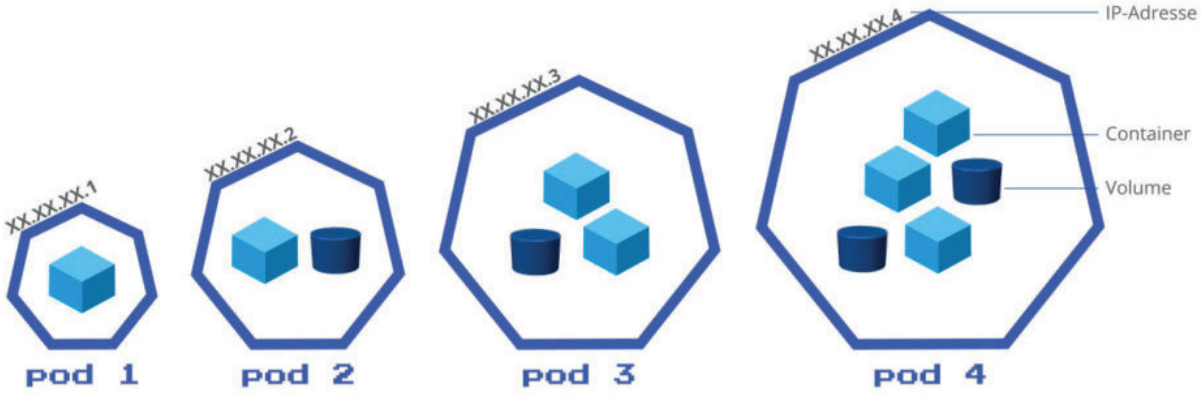

ABBILDUNG: KUBERNETES PODS

ReplicaSets

ReplicaSets sind Sets von Kopien (Replicas) eines Pods, welche die gleichen Container und Services oder nach einem Update neue Versionen der jeweiligen Services beinhalten. Dem Set wird mitgeteilt, wie viele Replicas benötigt werden, um einen Service bereitstellen zu können. Falls also ein Pod ausfällt, wird Kubernetes ein Replica davon erstellen, um den Ausfall aufzufangen und die Services können -wie gehabt- bereitgestellt werden

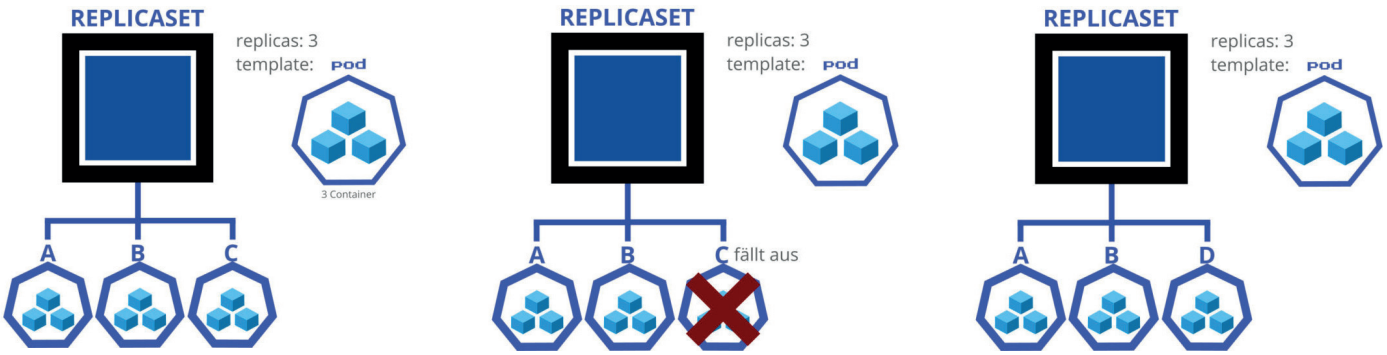

ABBILDUNG: AUSFALL EINES PODS

kubectl/Kubernetes-Weboberfläche

Es kann sowohl die Kubernetes-Weboberfläche, aber auch das Kommandozeilen-Tool kubectl zur Kommunikation mit Kubernetes genutzt werden.

5. KUBERNETES VS DOCKER SWARM UND ANDERE ORCHESTRIERUNGSTOOLS

Der große Unterschied zwischen Kubernetes und anderen Orchestrierungstools ist die Virtualisierungsumgebung. Beispielsweise setzt Docker Swarm die Containerumgebung Docker voraus – das ist bei Kubernetes nicht der Fall. Mit Kubernetes können die unterschiedlichsten Container-Virtualisierungssysteme kontrolliert und gesteuert werden. Diese können sogar innerhalb des Systems völlig unterschiedlich sein (z.B. zwei unterschiedliche Virtualisierungssysteme) und Kubernetes stellt weiterhin alle Funktionen zur Verfügung.

6. FAZIT ZU KUBERNETES

Microservice-Architekturen erfordern nicht nur Containisierungstools, sondern eine effiziente Verwaltung dieser. Die Einführung eines Kubernetes Clusters ist empfehlenswert, wenn mehrere Microservices mit hoher Skalierbarkeit verwaltet werden müssen. Ist ebendies gefordert und eine langfristige Erweiterung des Systems in Planung, eignet sich Kubernetes bestens, um das Monitoring, die Skalierung und das Management von Microservice-Systeme zu erleichtern.

QUELLEN:

1. **Startseite Kubernetes –** https://kubernetes.io/ – abgerufen am 19.02.2021

2. **What is Kubernetes?** – abgerufen am 19.02.2021

3. **Startseite Grafana** – abgerufen am 19.02.2021

4. **Prometheus Alert Manager** – abgerufen am 19.02.2021