BIESER

Der Kubernetes-Guide für Führungskräfte

1 Management Summary

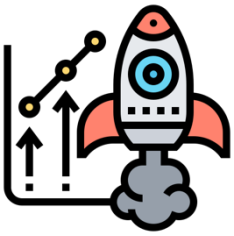
Die **Flexibilität und Schnelligkeit** durch **Containerisierung** und die **Orchestrierung** vieler Container mit **Kubernetes** ist der Treibstoff für eine hohe Geschwindigkeit bei außergewöhnlicher Stabilität in der modernen IT.

Kubernetes hat sich als De-facto-Standard für die Orchestrierung von Container-basierten Anwendungen etabliert. Angesichts der aktuellen technologischen Entwicklungen und Marktanforderungen ist es für IT-Abteilungen von entscheidender Bedeutung, jetzt auf Orchestrierungstools wie Kubernetes zu setzen. Dieser Schritt ermöglicht es Unternehmen, ihre IT-Infrastruktur zu modernisieren, Agilität zu erhöhen und Wettbewerbsvorteile zu sichern.

Marktreife und Stabilität

Kubernetes hat sich seit seiner Einführung durch Google im Jahr 2014 rasant entwickelt und ist nun eine ausgereifte und stabile Plattform. Die weitreichende Unterstützung durch eine starke Open-Source-Community und führende Technologieunternehmen sorgt für kontinuierliche Innovation und Stabilität. Unternehmen können von dieser Reife und dem bewährten Erfolg profitieren, um zuverlässige und skalierbare Anwendungen zu betreiben.

Flexibilität und Skalierbarkeit

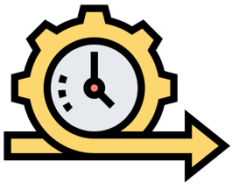


Kubernetes bietet außergewöhnliche Flexibilität und Skalierbarkeit, die für moderne IT-Abteilungen unerlässlich sind. Es ermöglicht das einfache Deployment, Management und Scaling von Anwendungen in unterschiedlichen Umgebungen – ob On-Premises, in der Cloud oder in hybriden Setups. Diese Skalierbarkeit ist entscheidend, um auf veränderte Geschäftsanforderungen schnell reagieren zu können.

Kostenoptimierung

Durch den Einsatz von Kubernetes können IT-Abteilungen ihre Ressourcennutzung optimieren und Kosten senken. Kubernetes ermöglicht eine effiziente Verwaltung von Rechenressourcen durch automatische Skalierung und Ressourcenverteilung. Dies führt zu einer besseren Auslastung der vorhandenen Hardware und einer Reduktion der Betriebskosten.

Unterstützung von DevOps und CI/CD



Kubernetes fördert die Implementierung von DevOps-Praktiken und Continuous Integration/Continuous Deployment (CI/CD) Pipelines. Dies beschleunigt die Software-Entwicklung und -Bereitstellung, erhöht die Qualität der Software und verkürzt die Markteinführungszeit. Unternehmen können dadurch schneller auf Marktveränderungen reagieren und Innovationen vorantreiben.

Verbesserung der Anwendungsportabilität

Durch den Einsatz von Containern und Kubernetes wird die Portabilität von Anwendungen erheblich verbessert. Anwendungen können problemlos zwischen verschiedenen Umgebungen migriert werden, ohne dass umfangreiche Anpassungen erforderlich sind. Dies erleichtert die Nutzung von Multi-Cloud-Strategien und verringert die Abhängigkeit von einzelnen Anbietern.

Sicherheit und Compliance

Kubernetes bietet umfangreiche Sicherheitsfunktionen wie Netzwerksegmentierung, rollenbasierte Zugriffskontrolle (RBAC) und Geheimnisverwaltung. Diese Funktionen helfen IT-Abteilungen, Sicherheits- und Compliance-Anforderungen zu erfüllen und sensible Daten zu schützen.

Fazit

Jetzt ist der richtige Zeitpunkt, auf Kubernetes zu setzen. Die Plattform bietet eine ausgereifte und stabile Lösung zur Orchestrierung von Containern, unterstützt agile und effiziente Entwicklungsprozesse und fördert Kosteneinsparungen. Unternehmen, die Kubernetes implementieren, sind besser gerüstet, um den Herausforderungen der digitalen Transformation zu begegnen und langfristige Wettbewerbsvorteile zu sichern. Es ist an der Zeit, die IT-Infrastruktur zu modernisieren und Kubernetes als strategische Technologie in der IT-Abteilung zu verankern.

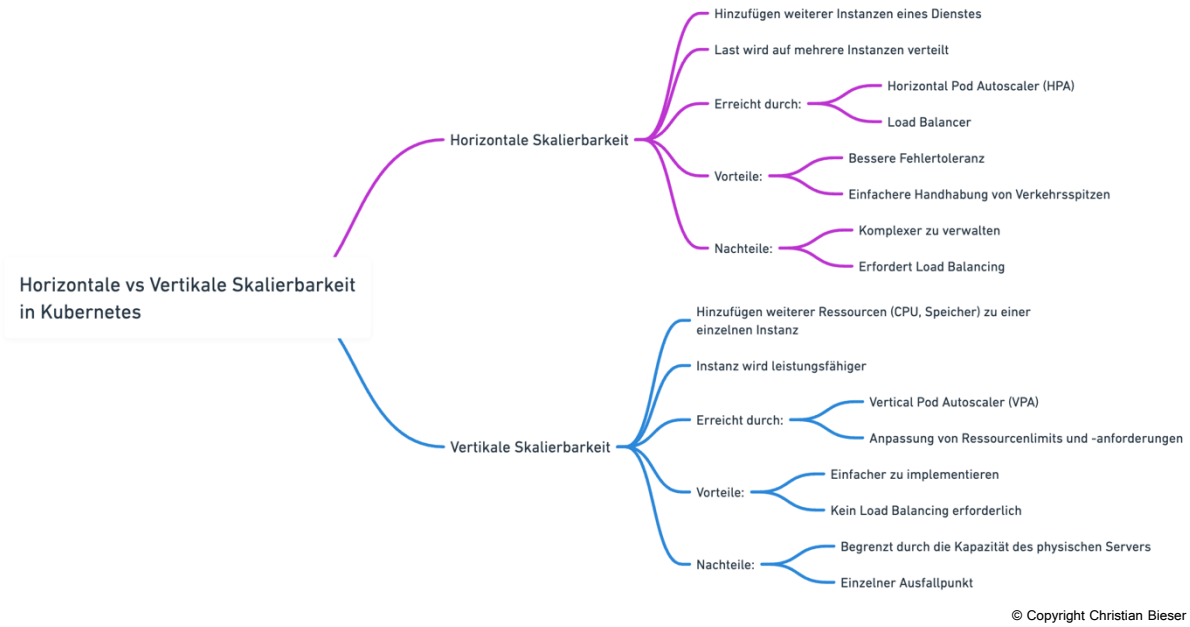
Images in Management Summary: Flaticon.com

Warum ist Kubernetes wichtig für die IT-Abteilung der Zukunft?

1.1 Skalierbarkeit und Effizienz

Kubernetes ermöglicht es, Anwendungen in Containern zu orchestrieren, was eine hohe Skalierbarkeit und Effizienz bietet. Containerisierung trennt Anwendungen und ihre Abhängigkeiten vom zugrunde liegenden Betriebssystem, wodurch Anwendungen konsistent und zuverlässig in verschiedenen Umgebungen ausgeführt werden können.

Die Skalierung von Kubernetes-Clustern kann sowohl horizontal als auch vertikal erfolgen. Hier sind die wichtigsten Konzepte und Mechanismen, die dabei eine Rolle spielen:



1.1.1 Horizontale Skalierung

Die horizontale Skalierung bezieht sich auf das Hinzufügen oder Entfernen von Instanzen (Pods) einer Anwendung, um die Arbeitslast zu bewältigen.

Horizontal Pod Autoscaler (HPA)

Der Horizontal Pod Autoscaler (HPA) skaliert die Anzahl der Pods basierend auf der beobachteten CPU-Auslastung (oder anderen benutzerdefinierten Metriken) eines Deployment oder ReplicaSet.

Funktionsweise: HPA überwacht die Ressourcenauslastung der Pods und passt die Anzahl der replizierten Pods automatisch an, um die gewünschte Auslastung zu erreichen.

Konfiguration: HPA wird durch eine Kubernetes-Ressource definiert, die Zielmetriken und Schwellenwerte angibt.

Wie sieht Beispielcode in Kubernetes dabei aus?

yaml Code kopieren

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: example-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: example-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```


Cluster Autoscaler

Der Cluster Autoscaler skaliert die Anzahl der Nodes in einem Cluster basierend auf den Ressourcenanforderungen der Pods.

Funktionsweise: Wenn nicht genügend Ressourcen vorhanden sind, um neue Pods zu planen, fügt der Cluster Autoscaler neue Nodes hinzu. Wenn Nodes unterausgelastet sind, entfernt der Cluster Autoscaler sie.

Anwendungsfall: Dies ist besonders nützlich in Cloud-Umgebungen, in denen die Anzahl der verfügbaren Maschinen dynamisch angepasst werden kann.

1.1.2 Vertikale Skalierung

Die vertikale Skalierung bezieht sich auf das Anpassen der Ressourcen (CPU und Speicher), die einem einzelnen Pod zugewiesen sind.

Vertical Pod Autoscaler (VPA)

Der Vertical Pod Autoscaler passt die CPU- und Speicheranforderungen der Pods an, um sicherzustellen, dass jeder Pod über genügend Ressourcen verfügt, um effizient zu arbeiten.

Funktionsweise: VPA überwacht die Ressourcennutzung und passt die Ressourcengrenzen und -anforderungen der Pods entsprechend an. Im Gegensatz zur horizontalen Skalierung ändert VPA die Größe der bestehenden Pods, anstatt neue Pods hinzuzufügen.

Konfiguration: VPA wird durch eine Kubernetes-Ressource definiert.

1.1.3 Manuelle Skalierung

Kubernetes ermöglicht auch die manuelle Skalierung der Pods und Nodes durch direkte Benutzerinteraktion.

Kubectl-Befehl

Ein Administrator kann die Anzahl der Pods manuell skalieren, indem er den kubectl scale Befehl verwendet.

1.1.4 Skalierung von StatefulSets und DaemonSets

Kubernetes unterstützt auch die Skalierung von speziellen Workloads wie StatefulSets und DaemonSets.

StatefulSets

StatefulSets bieten Mechanismen für den Betrieb von stateful Applikationen. Diese können ebenfalls horizontal skaliert werden, wobei jeder Pod eine eindeutige, stabile Identität und persistente Speicher erhält.

DaemonSets

DaemonSets stellen sicher, dass auf jeder Node im Cluster eine Kopie des Pods ausgeführt wird. Dies ist besonders nützlich für Anwendungen wie Logging und Monitoring.

1.2 Agilität und Schnelligkeit

Durch die Nutzung von Containern können IT-Abteilungen schneller auf Änderungen reagieren. Containerisierte Anwendungen können einfach und schnell aktualisiert, skaliert und ausgerollt werden, was die Entwicklungs- und Bereitstellungszyklen erheblich verkürzt.

1.3 Kostenreduktion

Die effiziente Ressourcennutzung durch Container und die Automatisierung von Deployments und Management-Aufgaben mit Kubernetes reduzieren die Betriebskosten. Ressourcen können nach Bedarf zugewiesen und freigegeben werden, was die Gesamtkosten der IT-Infrastruktur senkt:

Höhere Auslastung der Hardware

- Containerisierung: Mit Kubernetes können mehrere Container auf einer einzigen VM ausgeführt werden, wodurch die Hardwareauslastung optimiert wird. Dies reduziert die Anzahl der benötigten VMs und damit die Infrastrukturkosten.
- Ressourcenallokation: Kubernetes teilt Ressourcen wie CPU und RAM effizient auf, sodass weniger Ressourcen ungenutzt bleiben.

Automatisierung

- Deployment und Management: Kubernetes automatisiert viele Verwaltungsaufgaben wie Rollouts, Rollbacks und das Management von Containern, was den Bedarf an manuellen Eingriffen und damit die Personalkosten reduziert.
- Self-Healing: Kubernetes erkennt und ersetzt automatisch fehlerhafte Container, was die Notwendigkeit für manuelle Fehlerbehebung verringert.

Wartung und Verwaltung

- Einheitliche Plattform: Kubernetes bietet eine einheitliche Plattform für die Verwaltung von Anwendungen, was die Komplexität reduziert und die Effizienz des IT-Teams erhöht.
- Open Source: Als Open-Source-Software entfallen Lizenzkosten, die bei kommerziellen Orchestrierungslösungen anfallen könnten.

1.4 Verbesserte Fehlertoleranz und Verfügbarkeit

- **Automatische Neustarts:** Kubernetes startet fehlerhafte Pods automatisch neu, was die Ausfallzeiten reduziert und die Verfügbarkeit erhöht.
- **Load Balancing:** Kubernetes verteilt den Datenverkehr automatisch auf gesunde Pods, was die Leistung optimiert und Ausfallzeiten minimiert.
- Kubernetes bietet Mechanismen für **Backup und Recovery**, die Datenverlust verhindern und die Notwendigkeit teurer Notfallwiederherstellungslösungen verringern.

2 Welchen Reifegrad muss meine IT-Abteilung vor der Einführung von Kubernetes haben?

2.1 Technisches Know-how

Vor der Einführung von Kubernetes sollte die IT-Abteilung ein solides Verständnis der Grundlagen der Containerisierung und der Funktionsweise von Kubernetes haben. Kenntnisse in Docker, Container-Orchestrierung und Cloud-Infrastruktur sind unerlässlich.

2.1.1 Grundlegendes technisches Know-how

Containerisierung

- Docker: Verstehen der Grundlagen von Docker, Erstellen von Dockerfiles, Arbeiten mit Docker-Images und -Containern.

Kubernetes-Grundlagen

- Architektur: Verständnis der Kubernetes-Architektur, einschließlich Master- und Worker-Nodes.
- Kubernetes-Objekte: Kenntnisse über grundlegende Objekte wie Pods, Deployments, Services, ConfigMaps und Secrets.
- Kubectl: Beherrschung der Befehlszeilenschnittstelle (CLI) von Kubernetes, um Cluster zu verwalten.

Netzwerk und Sicherheit

- Netzwerkkonzepte: Verständnis von Service Discovery, Load Balancing und Networking in Kubernetes.
- Sicherheit: Kenntnisse über Netzwerkisolierung, Zugriffskontrollen (RBAC) und Pod-Sicherheit.

Speicherlösungen

- Persistente Volumes: Verstehen der verschiedenen Speicheroptionen in Kubernetes und der Verwaltung von persistenten Daten.

CI/CD und DevOps

- Continuous Integration/Continuous Deployment: Vertrautheit mit CI/CD-Pipelines und deren Integration mit Kubernetes.
- Automatisierung: Kenntnisse in der Automatisierung von Deployments und Skalierung.

Cloud- und Infrastrukturkenntnisse

- Cloud-Provider: Erfahrung mit Cloud-Plattformen wie AWS, GCP oder Azure, insbesondere mit Managed Kubernetes-Diensten (EKS, GKE, AKS).
- Infrastruktur als Code (IaC): Nutzung von Tools wie Terraform oder Ansible zur Automatisierung der Infrastrukturverwaltung.

2.1.2 Rollen in einem Kubernetes-Umfeld

Im Umfeld von Kubernetes sind **neue Rollen** in der IT-Abteilung notwendig:

- **Site Reliability Engineers**
- **Kubernetes Administratoren**
- **Kubernetes Developers**
- **(DevOps Engineers)**
- **(Security Engineers)**

Diese gilt es sinnvoll zu beschreiben und zu besetzen.

Kubernetes Administrator*in

Verantwortlich für die Installation, Konfiguration und Verwaltung von Kubernetes-Clustern. Überwacht die Cluster-Performance und führt Upgrades und Patches durch.

Deine Aufgaben	Dein Profil
- Verwaltung und Optimierung von Kubernetes-Clustern in verschiedenen Umgebungen (On-Premises, Cloud, Hybrid) - Implementierung und Wartung von Continuous Integration/Continuous Deployment (CI/CD) Pipelines - Monitoring und Performance-Tuning von Kubernetes-Anwendungen - Sicherstellung der Hochverfügbarkeit und Skalierbarkeit der Systeme - Verwaltung von Netzwerken und Sicherheitsrichtlinien innerhalb der Kubernetes-Umgebung - Fehlerbehebung und Support für Entwicklungs- und Betriebsteams	- Abgeschlossenes Studium im Bereich Informatik oder eine vergleichbare Qualifikation - Fundierte Kenntnisse in Docker und Container-Orchestrierung - Erfahrung mit CI/CD-Tools wie Jenkins, GitLab CI, oder Argo CD - Kenntnisse in Cloud-Plattformen wie AWS, Azure oder Google Cloud - Starke Problemlösungsfähigkeiten und eine proaktive Arbeitsweise - Gute Kenntnisse in Skriptsprachen wie Bash & Python

Kubernetes Developer*in

Entwickelt und deployed containerisierte Anwendungen auf Kubernetes. Arbeitet an der Optimierung und Skalierung von Anwendungen innerhalb des Clusters.

Deine Aufgaben

- Entwicklung und Implementierung von Kubernetes-basierten Anwendungen und Services
- Design und Erstellung von Microservices-Architekturen
- Entwicklung von Helm Charts zur Paketierung und Bereitstellung von Anwendungen
- Integration und Automatisierung von CI/CD-Pipelines für Kubernetes-Umgebungen
- Zusammenarbeit mit DevOps- und Infrastrukturteams zur Optimierung der Kubernetes-Deployment-Prozesse
- Monitoring und Optimierung der Leistung von Kubernetes-Anwendungen
- Erstellung und Pflege von technischen Dokumentationen

Dein Profil

- Abgeschlossenes Studium im Bereich Informatik oder eine vergleichbare Qualifikation
- Fundierte Kenntnisse in Programmiersprachen wie Go, Python oder Java
- Erfahrung mit Container-Technologien wie Docker und Container-Orchestrierung
- Kenntnisse in Cloud-Plattformen wie AWS, Azure oder Google Cloud
- Starke Problemlösungsfähigkeiten und eine proaktive Arbeitsweise
- Erfahrung mit CI/CD-Tools wie Jenkins, GitLab CI, oder Argo CD

Site Reliability Engineer*in (SRE)

Überwacht die Zuverlässigkeit und Verfügbarkeit von Anwendungen, die auf Kubernetes laufen. Implementiert Monitoring- und Logging-Lösungen.

Deine Aufgaben

- Entwicklung und Implementierung von Lösungen zur Verbesserung der Zuverlässigkeit, Verfügbarkeit und Performance unserer Systeme
- Aufbau und Pflege von Überwachungs- und Alarmierungssystemen zur proaktiven Fehlererkennung
- Automatisierung von Betriebsaufgaben und Verbesserung der CI/CD-Pipelines
- Zusammenarbeit mit Entwicklungsteams zur Optimierung von Systemarchitekturen und -prozessen
- Durchführung von Fehleranalysen und Root-Cause-Analysen zur Problembehebung
- Erstellung und Pflege von Betriebsdokumentationen und Notfallplänen

Dein Profil

- Abgeschlossenes Studium im Bereich Informatik oder eine vergleichbare Qualifikation
- Fundierte Kenntnisse in der Verwaltung und Skalierung von Cloud-Infrastrukturen (AWS, Azure, Google Cloud)
- Erfahrung in der Automatisierung mit Tools wie Ansible, Terraform oder Chef
- Kenntnisse in Programmiersprachen wie Python, Go oder Bash
- Erfahrung mit Container-Technologien (Docker, Kubernetes) und Microservices-Architekturen
- Starke Problemlösungsfähigkeiten und eine proaktive Arbeitsweise

DevOps Engineer*in

Implementiert CI/CD-Pipelines und automatisiert Deployments. Gewährleistet die Integration von Kubernetes in den gesamten Softwareentwicklungs-Lifecycle.

Deine Aufgaben

- Entwicklung und Implementierung von CI/CD-Pipelines zur Automatisierung von Build-, Test- und Deployment-Prozessen
- Verwaltung und Optimierung von Cloud-Infrastrukturen (AWS, Azure, Google Cloud)
- Automatisierung von Infrastrukturaufgaben mit Tools wie Ansible, Terraform oder Chef
- Monitoring und Logging der Systemleistung sowie Implementierung von Alarmierungssystemen
- Zusammenarbeit mit Entwicklungsteams zur Verbesserung der Softwareentwicklung und -bereitstellung
- Erstellung und Pflege von technischen Dokumentationen und Best Practices

Dein Profil

- Abgeschlossenes Studium im Bereich Informatik oder eine vergleichbare Qualifikation
- Fundierte Kenntnisse in der Verwaltung und Skalierung von Cloud-Infrastrukturen (AWS, Azure, Google Cloud)
- Erfahrung in der Automatisierung mit Tools wie Ansible, Terraform oder Chef
- Kenntnisse in Programmiersprachen wie Python, Bash
- Erfahrung mit Container-Technologien (Docker, Kubernetes) und Microservices-Architekturen
- Starke Problemlösungsfähigkeiten und eine proaktive Arbeitsweise

Security Engineer*in

Sorgt für die Sicherheit von Kubernetes-Clustern durch Implementierung von Sicherheitsrichtlinien und -praktiken. Überwacht und reagiert auf Sicherheitsvorfälle.

Deine Aufgaben

- Entwicklung und Implementierung von Sicherheitsrichtlinien und -verfahren zum Schutz unserer IT-Infrastruktur und Daten
- Durchführung von Sicherheitsbewertungen und Penetrationstests zur Identifizierung von Schwachstellen
- Monitoring und Analyse von Sicherheitsvorfällen sowie Entwicklung von Gegenmaßnahmen
- Verwaltung und Implementierung von Sicherheitslösungen wie Firewalls, IDS/IPS, SIEM und Anti-Malware
- Zusammenarbeit mit DevOps- und Entwicklungsteams zur Integration von Sicherheitsmaßnahmen in den Softwareentwicklungsprozess
-

Dein Profil

- Abgeschlossenes Studium im Bereich Informatik, IT-Sicherheit oder eine vergleichbare Qualifikation
- Fundierte Kenntnisse in Netzwerksicherheit, Anwendungssicherheit und Kryptographie
- Erfahrung mit Sicherheitslösungen und -technologien wie Firewalls, IDS/IPS, SIEM und Anti-Malware-Systemen
- Kenntnisse in Programmiersprachen wie Python, Java oder C++
- Erfahrung mit Cloud-Sicherheitslösungen in Plattformen wie AWS, Azure oder Google Cloud
- Starke Problemlösungsfähigkeiten und eine proaktive Arbeitsweise

2.1.3 Zertifikate und Weiterbildungen

Certified Kubernetes Administrator (CKA)

- Zielgruppe: Kubernetes-Administratoren.
- Inhalt: Installation, Konfiguration, Verwaltung und Fehlerbehebung von Kubernetes-Clustern.
- Anbieter: The Linux Foundation.

Certified Kubernetes Application Developer (CKAD)

- Zielgruppe: Kubernetes-Entwickler.
- Inhalt: Entwicklung, Aufbau und Bereitstellung von Anwendungen auf Kubernetes.
- Anbieter: The Linux Foundation.

Certified Kubernetes Security Specialist (CKS)

- Zielgruppe: Kubernetes-Sicherheitsexperten.
- Inhalt: Sicherheitspraktiken, Netzwerkpolicies, und Sicherheitswerkzeuge in Kubernetes.
- Anbieter: The Linux Foundation.

Kubernetes for Developers (Kubernetes auf Coursera)

- Zielgruppe: Entwickler und Anfänger in Kubernetes.
- Inhalt: Grundlegende Konzepte, Arbeiten mit Pods, Deployments und Services.
- Anbieter: Google Cloud auf Coursera.

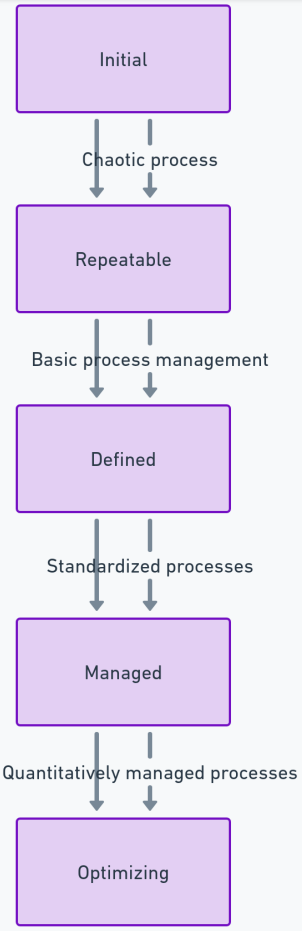
Weitere Ressourcen

- Udemy und Pluralsight: Verschiedene Kurse, die von grundlegenden bis zu fortgeschrittenen Themen reichen.
- Kubernetes Up and Running (Buch): Eine umfassende Einführung in Kubernetes.

2.2 Prozessreife

Es ist wichtig, dass die IT-Abteilung bereits agile Methoden und DevOps-Praktiken etabliert hat. Continuous Integration/Continuous Deployment (CI/CD) Pipelines sollten implementiert sein, um die Automatisierung und Effizienz zu maximieren. Die entsprechenden ITIL-Prozesse mit Fokus auf Service-Design werden auf der linken beschrieben.

ITIL-Service-Design-Prozesse mit Kubernetes



Die genannten ITIL-Prozesse sollten für Kubernetes angepasst werden und mindestens den Reifegrad „Defined“, im Optimalfall „Managed“ haben.

Service Level, Capacity und Availability und Service Continuity Management

Definition und Verwaltung von Service-Level-Agreements (SLAs) für Kubernetes-basierte Anwendungen, Sicherstellen, dass die Kubernetes-Infrastruktur ausreichend Kapazitäten bietet, um aktuelle und zukünftige Anforderungen zu erfüllen. Sicherstellung der Verfügbarkeit von Kubernetes-Clustern und den darauf laufenden Anwendungen. Implementierung von Hochverfügbarkeit, Redundanz und Failover-Mechanismen.Planung und Implementierung von Disaster-Recovery-Strategien für Kubernetes-Cluster, um sicherzustellen, dass Dienste im Falle eines Ausfalls schnell wiederhergestellt werden können.

Information Security Management

Implementierung von Sicherheitsrichtlinien und -maßnahmen für Kubernetes-Cluster, einschließlich Zugangskontrollen, Netzwerksicherheit und Datenverschlüsselung.

Service Transition: Change Management

Relevanz für Kubernetes: Verwaltung und Kontrolle von Änderungen an der Kubernetes-Infrastruktur und den darauf laufenden Anwendungen, um Risiken zu minimieren und Ausfallzeiten zu verhindern.

Release and Deployment Management

Planung, Terminierung und Kontrolle der Bereitstellung neuer Container-Images und Konfigurationen in Kubernetes-Umgebungen. Automatisierung der CI/CD-Pipelines.

Configuration Management

Relevanz für Kubernetes: Verwalten von Konfigurationsdaten und -dokumentationen für Kubernetes-Cluster und -Anwendungen, um eine konsistente und dokumentierte Umgebung zu gewährleisten.

2.3 Infrastruktur und Sicherheit

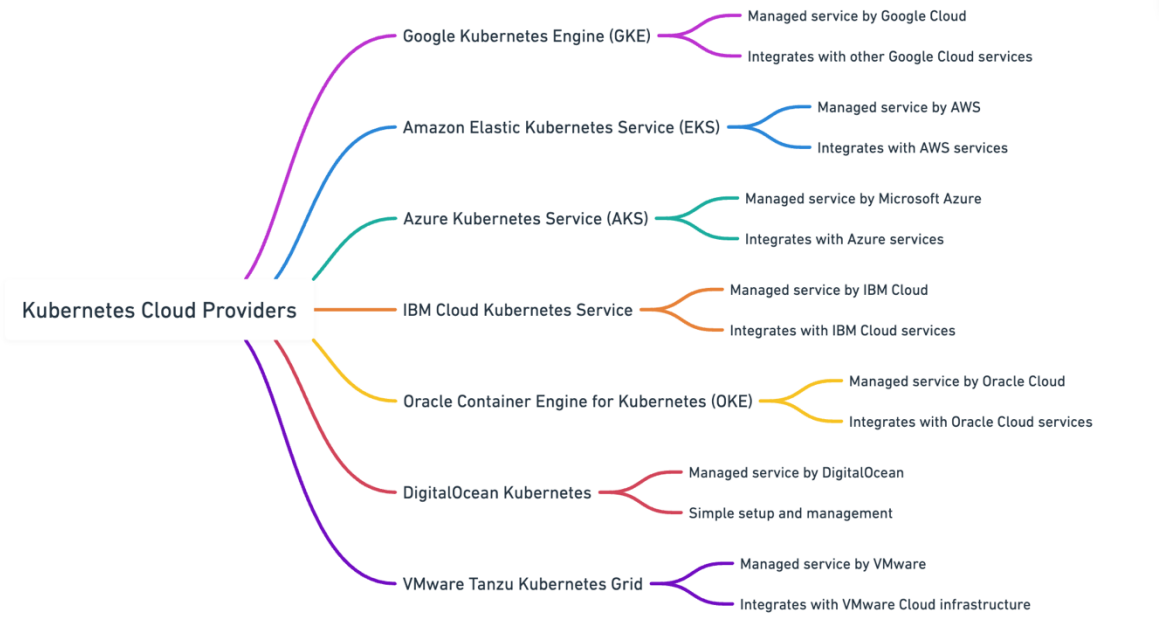
Eine robuste und flexible IT-Infrastruktur ist notwendig, um Kubernetes erfolgreich zu implementieren. Zudem sollten Sicherheitsaspekte wie Netzwerkisolierung, Zugriffskontrollen und regelmäßige Sicherheitsüberprüfungen in den Prozessen verankert sein.

2.4 Kubernetes in der Cloud oder im eigenen Rechenzentrum?

Der Hauptunterschied zwischen Kubernetes in der Cloud und Kubernetes On-Premises liegt in der Umgebung, in der die Kubernetes-Cluster betrieben werden, und den damit verbundenen Herausforderungen und Vorteilen.

2.4.1 Kubernetes in der Cloud

Kubernetes in der Cloud wird von einem Cloud-Provider wie AWS, Google Cloud oder Azure bereitgestellt und verwaltet. Diese Anbieter bieten nahezu unbegrenzte Skalierbarkeit, da Ressourcen bei Bedarf automatisch hinzugefügt oder entfernt werden können. Verwaltete Kubernetes-Dienste in der Cloud übernehmen zudem die automatischen Updates und Patches, wodurch die Wartung vereinfacht wird. Darüber hinaus bieten Cloud-Provider hohe Verfügbarkeit und Redundanz durch Multi-Region- und Multi-AZ-Deployments sowie integrierte Sicherheitsfunktionen und -dienste wie Identitäts- und Zugriffsmanagement, Firewalls und DDoS-Schutz. Ein weiteres wichtiges Merkmal ist das Pay-as-you-go-Kostenmodell, bei dem nur für tatsächlich genutzte Ressourcen gezahlt wird, was besonders bei variablen Workloads kosteneffizient sein kann.



2.4.2 Kubernetes on Premises (im eigenen Rechenzentrum)

Im Gegensatz dazu wird Kubernetes On-Premises auf eigener Hardware und Infrastruktur betrieben, was volle Kontrolle und Anpassungsmöglichkeiten bietet, aber auch mehr Verantwortung für Wartung und Management mit sich bringt. Die Skalierung erfordert hier den Kauf und die Konfiguration zusätzlicher Hardware, was teurer und aufwändiger sein kann. Updates und Patches müssen manuell durchgeführt werden, was zusätzlichen Verwaltungsaufwand bedeutet. Die Sicherstellung hoher Verfügbarkeit und Redundanz ist ebenfalls komplexer und erfordert sorgfältige Planung und zusätzliche Ressourcen. Obwohl die Anfangsinvestitionen in Hardware und Infrastruktur hoch sein können, können die laufenden Kosten bei kontinuierlicher Nutzung niedriger sein als bei Cloud-basierten Lösungen. Bei der Sicherheit sind eigene Maßnahmen erforderlich, was mehr Kontrolle, aber auch mehr Aufwand bedeutet, insbesondere in Bezug auf die Verwaltung der Netzwerkinfrastruktur.

Einkaufsliste für das „eigene“ Kubernetes

1. Architekturdefinition und Planung

Master- und Worker-Nodes: Definieren Sie die Anzahl und die Spezifikationen der Master- und Worker-Nodes.

Netzwerkarchitektur: Planen Sie das Netzwerkdesign, einschließlich IP-Adressierung, Subnetze und VLANs.

2. Hardware und Infrastruktur

Server-Hardware

Master-Nodes: Mindestens 3 Master-Nodes für Hochverfügbarkeit. Jeder Master sollte über ausreichende CPU, RAM und Speicher verfügen.

Worker-Nodes: Anzahl der Worker-Nodes basierend auf dem Ressourcenbedarf der Anwendungen. Auch hier sind ausreichend CPU, RAM und Speicher erforderlich.

Netzwerk

Switches und Router: Leistungsfähige Netzwerkhardware für die interne Kommunikation zwischen den Nodes und externe Verbindungen.

Load Balancer: Für die Verteilung des Traffics und die Gewährleistung der Hochverfügbarkeit der Master-Nodes.

Speicher

Persistente Speichersysteme: SAN, NAS oder verteilte Speicherlösungen wie Ceph oder GlusterFS für persistente Volumes.

3. Software und Tools

Betriebssystem

Linux-Distribution: Eine stabile und unterstützte Linux-Distribution wie Ubuntu, CentOS oder Red Hat Enterprise Linux.

Kubernetes-Software

Kubernetes Distribution: Eine Kubernetes-Distribution wie kubeadm, OpenShift oder Rancher.

Container-Runtime: Docker, containerd oder CRI-O.

Netzwerk-Plugins

CNI-Plugins: Calico, Flannel, Weave, Cilium oder andere für Netzwerk-Overlay und Pod-Kommunikation.

Monitoring und Logging

Monitoring: Prometheus, Grafana für die Überwachung der Cluster-Ressourcen.

Logging: EFK-Stack (Elasticsearch, Fluentd, Kibana) oder Loki für Log-Management.

Jetzt kann die Installation beginnen...

Die Wahl zwischen Kubernetes in der Cloud und On-Premises hängt letztlich von den spezifischen Anforderungen, dem Budget und den technischen Ressourcen eines Unternehmens ab. Cloud-Lösungen bieten Bequemlichkeit, Skalierbarkeit und oft geringere Anfangsinvestitionen, während On-Premises-Lösungen mehr Kontrolle, Anpassungsmöglichkeiten und potenzielle Kosteneinsparungen auf lange Sicht bieten können.

3 Wie sieht eine schrittweise Einführung von Kubernetes aus?

3.1 Schritt 1: Planung und Bewertung

Zunächst sollten die aktuellen Bedürfnisse und Ziele der IT-Abteilung bewertet werden. Eine Roadmap für die Einführung von Kubernetes sollte erstellt werden, die auch die erforderlichen Schulungen und Infrastruktur-Updates umfasst.

Ein Beispielhafter Projektstrukturplan könnte folgendermaßen aussehen:

Projektphase	Aufgabe	Beschreibung	Dauer	Verantwortlicher
Planung	Bedarfsanalyse	**Anforderungen der Anwendungen ermitteln**	2 Wochen	Projektmanager
	Anwendungsanforderungen	**Ressourcenbedarf, Hochverfügbarkeit und Redundanz bestimmen**	1 Woche	Systemarchitekt
	Skalierbarkeitsanforderungen	**Skalierungsanforderungen für Workloads festlegen**	1 Woche	Systemarchitekt
	Cluster-Architektur	**Design der Cluster-Architektur**	3 Wochen	Systemarchitekt
	Master- und Worker-Nodes	**Anzahl und Spezifikationen der Nodes festlegen**	1 Woche	Systemarchitekt
	Netzwerkarchitektur	**Netzwerkdesign planen (IP-Adressierung, Subnetze, VLANs)**	2 Wochen	Netzwerkadministrator
Infrastrukturaufbau	Hardwarebereitstellung	**Server-Hardware bereitstellen**	4 Wochen	IT-Infrastruktur-Team
	Master-Nodes	**Hardware für Master-Nodes beschaffen und konfigurieren**	2 Wochen	IT-Infrastruktur-Team
	Worker-Nodes	**Hardware für Worker-Nodes beschaffen und konfigurieren**	2 Wochen	IT-Infrastruktur-Team
	Netzwerkhardware	**Switches, Router und Load Balancer einrichten**	4 Wochen	Netzwerkadministrator
	Speichersysteme	**Persistente Speichersysteme (SAN, NAS) einrichten**	3 Wochen	Speicheradministrator
Softwareinstallation	Betriebssysteminstallation	**Linux-Distribution auf allen Nodes installieren**	2 Wochen	IT-Infrastruktur-Team
	Kubernetes-Tools installieren	**Installation von kubeadm, kubectl, kubelet**	1 Woche	DevOps-Team
	CNI-Plugin installieren	**Installation und Konfiguration des gewählten Netzwerk-Plugins**	1 Woche	DevOps-Team
Cluster-Bootstrapping	Master-Node-Setup	**Initialisieren des ersten Master-Nodes**	1 Woche	DevOps-Team
	Worker-Nodes hinzufügen	**Hinzufügen der Worker-Nodes zum Cluster**	1 Woche	DevOps-Team
Konfiguration	Netzwerk-Setup	**Konfiguration des CNI-Plugins für Netzwerkkommunikation**	1 Woche	Netzwerkadministrator
Verwaltung und Betrieb	Verwaltungstools einrichten	**Installation und Konfiguration von Verwaltungstools (Dashboard, kubectl)**	2 Wochen	DevOps-Team
	Sicherheitsmaßnahmen implementieren	**RBAC, Secrets Management, Netzwerk-Security**	3 Wochen	Sicherheitsbeauftragter
Monitoring und Logging	Monitoring-Tools einrichten	**Installation und Konfiguration von Prometheus, Grafana**	2 Wochen	DevOps-Team
	Logging-Tools einrichten	**Installation und Konfiguration des EFK-Stacks oder Loki**	2 Wochen	DevOps-Team
Backup und Recovery	Backup-Strategien entwickeln	**Regelmäßige Backups der ETCD-Datenbank und persistenter Volumes planen**	2 Wochen	DevOps-Team
	Disaster-Recovery-Plan entwickeln	**Implementierung eines Disaster-Recovery-Plans**	2 Wochen	DevOps-Team
Wartung und Updates	Regelmäßige Updates und Patches	**Planmäßige Updates der Kubernetes-Software und Sicherheits-Patches**	Laufend	DevOps-Team
	Performance-Monitoring und Kapazitätsplanung	**Kontinuierliche Überwachung und Ressourcenoptimierung**	Laufend	DevOps-Team

3.2 Schritt 2: Auswahl der Anwendungen für das Pilotprojekt

Welche Applikationen eignen sich für einen Piloten?

Ein Kubernetes-Pilotprojekt lässt sich am einfachsten für Anwendungs-Technologien aufsetzen, die bereits containerisiert sind und von Natur aus skalierbar und cloud-native sind. Hier sind einige Technologien und Anwendungstypen, die sich besonders gut für ein Kubernetes-Pilotprojekt eignen:

1. Microservices

Microservices-Architekturen bestehen aus kleineren, unabhängig deploybaren Diensten, die sich hervorragend für Kubernetes eignen. Jede Microservice kann in einem eigenen Container laufen und Kubernetes kann die Skalierung und Verwaltung übernehmen.

Beispiele: Spring Boot, Node.js, Go-Microservices

2. Web-Anwendungen

Containerisierte Web-Anwendungen lassen sich leicht auf Kubernetes deployen und skalieren. Sie profitieren von der automatischen Lastverteilung und Selbstheilung.

Beispiele: NGINX-basierte Anwendungen, Apache HTTP Server, Node.js, Ruby on Rails

5. Big Data und Analytics

Big Data-Anwendungen können auf Kubernetes in verteilten Umgebungen laufen, um große Datenmengen zu verarbeiten und zu analysieren.

Beispiele: Apache Spark, Apache Hadoop, Kafka

6. Machine Learning/AI

Machine Learning-Workloads profitieren von der Skalierbarkeit und Flexibilität von Kubernetes, besonders bei der Verwaltung von Trainings- und Inferenz-Jobs.

Beispiele: TensorFlow, PyTorch, Jupyter Notebooks, Kubeflow

Zusammenfassung

Für ein erfolgreiches Kubernetes-Pilotprojekt eignen sich am besten Anwendungen, die bereits containerisiert sind und in einer Microservices-Architektur oder als cloud-native Anwendungen entwickelt wurden. Web-Anwendungen, CI/CD-Pipelines, Big Data und Analytics, Machine Learning, und API-Gateways sind gute Startpunkte. Diese Technologien profitieren besonders von den Skalierungs-, Orchestrierungs- und Automatisierungsfähigkeiten von Kubernetes.

3.3 Schritt 3: Schulung und Weiterentwicklung

Die kontinuierliche Schulung des IT-Teams ist entscheidend. Dies kann durch interne Workshops, Online-Kurse oder externe Schulungen geschehen. Die gewonnenen Erkenntnisse aus dem Pilotprojekt sollten genutzt werden, um Prozesse und Best Practices zu entwickeln.

3.4 Schritt 4: Skalierung und Optimierung

Nach erfolgreichen Tests und Schulungen kann die Einführung von Kubernetes auf weitere Projekte und Anwendungen ausgeweitet werden. Dabei sollten kontinuierlich Optimierungen vorgenommen werden, um die Effizienz zu maximieren und Fehler zu minimieren.

4 Gibt es Möglichkeiten mit Kubernetes zu experimentieren, ohne ein großes Projekt daraus zu machen?

4.1 Minikube und Kubernetes-Dienste

Minikube ist eine lokale Kubernetes-Umgebung, die sich ideal für erste Experimente eignet. IT-Teams können damit Kubernetes auf einem lokalen Rechner testen, ohne umfangreiche Infrastrukturanforderungen.

4.2 Cloud-basierte Kubernetes-Services

Cloud-Anbieter wie Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS) und Azure Kubernetes Service (AKS) bieten Managed-Kubernetes-Dienste an. Diese ermöglichen es, Kubernetes ohne großen administrativen Aufwand zu testen und erste Anwendungen zu deployen.

4.3 Docker Desktop Kubernetes

Docker Desktop bietet eine einfache Möglichkeit, Kubernetes direkt auf einem lokalen Rechner zu starten. Diese Umgebung eignet sich hervorragend für Entwickler, um erste Schritte mit Kubernetes zu gehen und containerisierte Anwendungen zu testen.

5 Was kann noch verbessert werden, wenn Kubernetes bereits in der IT eingesetzt wird?

5.1 Automatisierung und Monitoring

Die Implementierung von fortgeschrittenen Automatisierungstools und Monitoring-Lösungen kann die Verwaltung und Optimierung von Kubernetes-Clustern erheblich verbessern. Tools wie **Prometheus** und **Grafana** bieten umfassende Überwachungs- und Alarmierungsfunktionen.

5.2 Sicherheitsoptimierungen

Sicherheitsaspekte sollten kontinuierlich überprüft und verbessert werden. Dies umfasst regelmäßige Sicherheitsupdates, Netzwerksegmentierung, und die Implementierung von Best Practices für die Container-Sicherheit.

5.3 Optimierung der Ressourcen

Eine kontinuierliche Überprüfung und Optimierung der Ressourcennutzung kann Kosten senken und die Effizienz steigern. Tools wie Kubernetes Horizontal Pod Autoscaler und Cluster Autoscaler können dabei helfen, die Ressourcennutzung dynamisch anzupassen.

6 Ausblick auf die Entwicklung von Kubernetes und Containerisierung in den nächsten Jahren

6.1 Integration von KI und ML

Kubernetes wird zunehmend in den Bereichen Künstliche Intelligenz (KI) und Maschinelles Lernen (ML) eingesetzt werden. Die Automatisierung und Skalierbarkeit von Kubernetes bieten ideale Voraussetzungen für ML-Workloads.

6.2 Serverless Computing

Die Kombination von Kubernetes mit serverlosen Technologien wird weiter zunehmen. Lösungen wie Knative ermöglichen es, serverlose Anwendungen auf Kubernetes zu betreiben, was die Entwicklung und Bereitstellung weiter vereinfacht.

6.3 Multi-Cloud-Strategien

Kubernetes wird eine zentrale Rolle in Multi-Cloud-Strategien spielen, da es eine einheitliche Orchestrierungsschicht über verschiedene Cloud-Anbieter hinweg bietet. Dies erhöht die Flexibilität und Unabhängigkeit von einzelnen Anbietern.

6.4 Weiterentwicklung der Ökosysteme

Das Kubernetes-Ökosystem wird weiter wachsen, mit neuen Tools und Erweiterungen, die die Verwaltung und den Betrieb noch einfacher und effizienter machen. Projekte wie **Istio** für Service **Meshes** und Helm für Paketmanagement werden weiter an Bedeutung gewinnen.

7 Fazit

Kubernetes und Containerisierung sind wegweisende Technologien für die IT-Abteilungen der Zukunft. Sie bieten immense Vorteile in Bezug auf Skalierbarkeit, Agilität und Effizienz. Eine erfolgreiche Einführung erfordert jedoch eine sorgfältige Planung, Schulung und kontinuierliche Optimierung. Mit den richtigen Ansätzen und Tools können IT-Abteilungen nicht nur ihre aktuellen Herausforderungen meistern, sondern auch zukunftssicher aufgestellt sein.

Bei Fragen stehe ich gerne zur Verfügung:

Christian Bieser

Bieser IT Consulting

71634 Ludwigsburg

christian@bieser-it-consulting.de