NINE · CLOUD NAVIGATORS

Mit Kubernetes starten

Eine praktische Checkliste für Engineering-Teams

Kubernetes ist die Industriestandard-Plattform für den Betrieb containerisierter Workloads in der Produktion. Aber der Weg dahin erfordert gezielte Vorbereitung. Diese Checkliste führt durch die wichtigsten Fragen, die jedes Engineering-Team vor und während dem Deployment auf Kubernetes beantworten sollte. Die Punkte bauen aufeinander auf: Arbeite sie der Reihe nach durch.

1 Hast du deinen Container-Build und -Push automatisiert?

Kubernetes bezieht Container-Images aus einer Registry, daher ist eine zuverlässige, automatisierte Pipeline unerlässlich. Falls du deinen Build- und Push-Prozess noch nicht automatisiert hast, fange hier an: Commit ins Git-Repository, Build auslösen, ein OCI-kompatibles Image erzeugen und in eine Registry pushen (privat oder public).

Arbeite dich von dort rückwärts vor: automatisierte Tests, Code-Review-Gates und alle weiteren Workflow-Schritte, die vorhanden sein müssen. Eine solide CI-Pipeline ist das Fundament für alles Weitere.

---

2 Scannst du deine Container-Images auf Sicherheitslücken?

Ein kleines Image ist ein sichereres Image, aber Grösse allein reicht nicht. Container-Images können bekannte Sicherheitslücken (CVEs) in Basis-Layern oder Abhängigkeiten enthalten. Integriere Image-Scanning in deine CI-Pipeline mit Tools wie **Trivy** oder **Grype**. Viele Container-Registries, darunter auch unsere private Registry, bieten integriertes Scanning.

Lasse die Pipeline bei kritischen oder hochgefährlichen Findings fehlschlagen und aktualisiere Basis-Images regelmässig.

---

3 Wie schlank ist dein Container?

Dein Container-Image wird von verschiedenen Systemen in deiner Pipeline bezogen. Um Startzeit und Bandbreitenkosten zu optimieren, entferne alles Unnötige: Nutze Multi-Stage-Builds, verwende minimale Basis-Images (z.B. Distroless oder Alpine) und vermeide Build-Tools im Produktions-Image.

Schlankere Images bedeuten auch weniger Pakete mit potenziellen Sicherheitslücken: Sicherheit und Effizienz gehen Hand in Hand.

---

4 Hast du deine Kubernetes-Konfiguration geschrieben?

Falls du deine Kubernetes-Manifests noch nicht geschrieben hast, prüfe die verfügbaren Templating-Optionen. **Helm Charts** sind der De-facto-Standard für die Paketierung von Kubernetes-Anwendungen. Lerne Helm, auch wenn du mit einfachen YAML-Manifests startest, denn die meiste Software von Drittanbietern wird als Helm Chart ausgeliefert.

Kubernetes verwendet ein deklaratives Konfigurationsmodell. Tools wie **Kustomize** (in kubectl integriert), **cdk8s** oder **Pulumi** bieten verschiedene Ansätze für das Management von Konfigurationen auf unterschiedlichen Komplexitätsstufen. Behandle deine Kubernetes-Konfiguration als erstklassigen Teil deiner Anwendung: Versioniere, reviewe und teste sie.

---

5 Kennst du das Performance-Profil deiner Anwendung?

Um das Beste aus Kubernetes herauszuholen, musst du verstehen, wie deine Anwendung unter Last performt: mindestens das CPU- und RAM-Profil. Damit kannst du Resource Requests und Limits korrekt konfigurieren, die Kubernetes für Scheduling und Self-Healing verwendet.

Ohne korrekte Ressourcenkonfiguration könnte deine Anwendung nicht neu starten, wenn es ein Problem gibt, oder in den `CrashLoopBackOff`-Status geraten. Es ist sehr unwahrscheinlich, dass du das beim ersten Versuch richtig triffst: Monitore die tatsächliche Auslastung und passe regelmässig an. Kubernetes v1.27+ unterstützt vertikales In-Place-Scaling ohne Pod-Neustart (derzeit Beta).

---

6 Hast du die Skalierbarkeit deiner Anwendung bedacht?

Einer der mächtigsten Aspekte von Kubernetes ist die horizontale Skalierung. Sobald du das Performance-Profil deiner Container kennst, konfiguriere einen **HorizontalPodAutoscaler (HPA)**, um deine Anwendung basierend auf dem aktuellen Traffic oder benutzerdefinierten Metriken zu skalieren. Für Cluster mit variablen Workloads sorgt **Node-Autoscaling** dafür, dass auch die zugrundeliegende Infrastruktur automatisch skaliert.

Traffic-Spitzen müssen kein Problem mehr sein: Lass das Cluster für dich arbeiten.

---

7 Hast du bedacht, wie deine Anwendung auf Neustarts reagiert?

Kubernetes wird deine Container regelmässig starten, stoppen und neu starten: Das ist ein Feature, kein Bug. Stelle sicher, dass deine Anwendung Neustarts sauber übersteht: Vermeide Datenbank-Schema-Migrationen bei jedem Start, implementiere korrekte Liveness- und Readiness-Probes und sorge für schnelle Startzeiten.

Falls deine Anwendung Sidecars verwendet (z.B. Service-Mesh-Proxies, Log-Shipper): **Native Sidecar-Container** haben mit Kubernetes v1.32 General Availability erreicht und verfügen nun über korrektes Lifecycle-Management.

---

8 Hast du die Verfügbarkeit deiner Anwendung bedacht?

Physische und virtuelle Maschinen fallen aus. Aber du kannst deine Workloads so konfigurieren, dass Single Points of Failure vermieden werden. Mehrere Replikas, verteilt auf verschiedene Nodes (**Pod Topology Spread Constraints** oder Pod Anti-Affinity), erhöhen die Verfügbarkeit erheblich. Für geschäftskritische Workloads empfiehlt sich der Betrieb über mehrere Availability Zones hinweg.

---

9 Hast du bedacht, wie deine Anwendung auf nicht verfügbare Services reagiert?

Nichtverfügbarkeit ist immer eine Möglichkeit, unabhängig von Infrastruktur, Anbieter oder Architektur. Entwirf deine Anwendung resilient: Implementiere Retry-Logik mit exponentiellem Backoff, Circuit Breaker und graceful Degradation. Sicherzustellen, dass deine Anwendung auch bei einem Ausfall einer Abhängigkeit erreichbar bleibt und keine Daten verliert, sollte ein zentrales Designziel sein.

---

10 Hast du GitOps für deine Deployments eingerichtet?

Ein ausgereifter Kubernetes-Deployment-Workflow geht über CI/CD-Pipelines hinaus, die Änderungen direkt in Cluster pushen. **GitOps**-Tools wie **ArgoCD** oder Flux behandeln Git als Single Source of Truth für den Cluster-Zustand. Änderungen erfolgen via Pull Requests, werden automatisch mit dem Live-Cluster abgeglichen, und jede Abweichung vom gewünschten Zustand wird erkannt und korrigiert.

Das ergibt eine vollständig auditierbare, versionskontrollierte Deployment-Historie und vereinfacht Rollbacks auf jeden früheren Zustand. GitOps ist für die Produktion kein Luxus, sondern Best Practice.

---

11 Hast du den Betrieb der Services bedacht, von denen deine Anwendung abhängt?

Es gibt viel zu bedenken, wenn man eine Anwendung in Kubernetes betreibt. Unabhängig davon, ob du neu bei Kubernetes bist oder bereits Erfahrung hast: Es kann sehr sinnvoll sein, mit einem erfahrenen Managed-Services-Anbieter zusammenzuarbeiten, der zuverlässige Services auf garantiertem Qualitätsniveau bereitstellt. Wenn du direkten Kontakt mit zertifizierten Engineers, verwaltete Observability, GitOps-Setup, Architekturberatung oder automatisierte Cluster-Upgrades benötigst, bieten wir mit CKA-zertifizierten Engineers und langjähriger Erfahrung im Kubernetes-Produktivbetrieb erheblichen Mehrwert.

Nine Internet Solutions AG · Badenerstrasse 47 · 8004 Zürich · Schweiz
info@nine.ch · +41 44 637 40 00 · nine.ch