Engineer

Kubernetes

System, das eine Flotte von Containern am Laufen hält

Was es ist

Kubernetes, oft K8s geschrieben, ist quelloffene Software, die containerisierte Anwendungen über eine Gruppe von Maschinen, einen Cluster, ausführt und an eine vorgegebene Beschreibung bindet: welches Image jede Anwendung ausführt, wie viele Kopien existieren, wie viel Speicher und CPU jede verwenden darf und wie andere Programme sie erreichen. Die Projektdokumentation nennt es „eine portable, erweiterbare Open-Source-Plattform zur Verwaltung containerisierter Arbeitslasten und Dienste“. Der griechische Name bedeutet Steuermann oder Lotse. K8s behält K und s und ersetzt die acht Buchstaben dazwischen durch ihre Anzahl. Google veröffentlichte das Projekt 2014 als Open Source.

Ein Cluster besteht aus zwei Teilen. Die Steuerungsebene hält die Beschreibung: Der API-Server ist die eine HTTP-Schnittstelle, durch die jede Änderung geht. etcd speichert, was ihm mitgeteilt wird. Der Scheduler weist jeden neuen Pod einem geeigneten Knoten zu, und der Controller Manager betreibt die Schleifen, die nach der Beschreibung handeln. Auf jedem Arbeitsknoten stellt das kubelet sicher, dass die dort zugewiesenen Pods laufen. kube-proxy pflegt die Netzwerkregeln hinter Services, und eine Containerlaufzeit wie containerd oder CRI-O startet die Container. Docker Engine ist seit Kubernetes 1.24 keine eingebaute Laufzeit mehr. Mit Docker gebaute Images laufen unverändert, denn es sind standardisierte OCI-Images.

Die Beschreibung besteht aus Objekten, gewöhnlich als YAML geschrieben. Ein Pod ist die kleinste von Kubernetes eingeplante Einheit: ein oder mehrere Container, die sich auf einer Maschine Netzwerkadresse und Speicher teilen. Ein Deployment sagt, wie viele identische Pods existieren sollen, und ersetzt sie schrittweise, wenn sich ihre Vorlage ändert. Ein Service gibt einer wechselnden Gruppe von Pods einen stabilen Namen und eine Adresse und wählt die Mitglieder nach Labels aus. Eine kleine Webanwendung sieht so aus:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: web
          image: registry.example.com/web:1.4.2
          resources:
            requests:
              cpu: 250m
              memory: 256Mi
            limits:
              memory: 256Mi
          readinessProbe:
            httpGet:
              path: /healthz
              port: 8080
```

Kubernetes wird gewöhnlich gegen drei kleinere Möglichkeiten abgewogen. Docker Compose beschreibt Container auf einer Maschine. Fällt diese weg, sind auch die Container weg. Ein Dienstmanager wie systemd kann einen abgestürzten Container neu starten und ihn beim Systemstart ausführen. Das deckt viele Anforderungen mit erheblich weniger Technik ab, kennt aber keine Verlagerung von Arbeit auf eine andere Maschine. Gehostete Plattformen wie Google Cloud Run oder Azure Container Apps betreiben Container, ohne Ihnen eine Steuerungsebene zum Betreiben zu überlassen, bieten dafür aber weniger Einstellungen. Die Dokumentation sagt auch deutlich, was Kubernetes nicht ist: eine herkömmliche, alles umfassende Platform as a Service. Es baut Ihre Anwendung nicht und liefert keine eigenen Datenbanken oder Nachrichtenwarteschlangen mit.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Kubernetes?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. So verpackt, dass sie auf dem Laptop, dem Testserver und im Produktivbetrieb gleich funktioniert und bei höherer Auslastung mitwächst.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 20 Docker-Apps auf Google Cloud und AWS betrieben — 20 bereitgestellte und überwachte Containeranwendungen

20 Docker-Apps auf Google Cloud und AWS betrieben Kontakt aufnehmen →

Warum es wichtig ist

Der Kerngedanke ist, dass der Cluster fortlaufend auf eine schriftliche Beschreibung hinarbeitet, statt einmal eine Befehlsliste auszuführen. Das Anwenden der obigen Datei startet direkt nichts, sondern hält einen Wunsch fest. Controller sind Regelschleifen, die den Cluster beobachten und seinen aktuellen Zustand dem gewünschten annähern. Fällt eine Maschine aus und werden aus drei Kopien zwei, sieht ein Controller die Lücke. Ein neuer Pod wird erstellt, und der Scheduler setzt ihn auf eine Maschine mit Platz. Weil die Beschreibung aus Dateien besteht, kann sie im Repository liegen und dieselbe Prüfung durchlaufen wie der ausgeführte Code.

Kubernetes bringt Abläufe für Releases mit. Die Standardstrategie ist eine fortlaufende Aktualisierung: Neue Pods starten, und alte stoppen, sobald die neuen bereit sind. Dabei gelten Grenzen, die standardmäßig jederzeit 25 % zusätzliche und 25 % fehlende Pods erlauben. Jede Änderung der Podvorlage wird als Revision aufbewahrt, standardmäßig zehn. `kubectl rollout undo` kehrt zur vorherigen zurück. Was früher ein Skript erledigte, das einen Server aus dem Verkehr nahm, eine Binärdatei austauschte und ihn wieder zuschaltete, ist nun eingebautes Verhalten.

Der Scheduler ändert außerdem die Wirtschaftlichkeit einer Serverflotte. Er platziert jeden Pod nach seinen CPU- und Speicheranforderungen. Viele kleine Dienste teilen sich dadurch Maschinen, statt jeweils einen Server für ihre höchste zu erwartende Last zu erhalten.

Das hat seinen Preis. Ein Cluster ist selbst ein verteiltes System, mit planmäßig zu aktualisierenden Versionen, einer Netzwerk- und Speicherschicht, Zertifikaten und Zugriffskontrolle. All das muss jemand verstehen, wenn es sich falsch verhält. Verwaltete Angebote wie Google Kubernetes Engine, Amazon EKS und

Azure Kubernetes Service betreiben die Steuerungsebene für Sie. Jedes Manifest und gewöhnlich auch die Arbeitsknoten bleiben jedoch Ihre Aufgabe. Für wenige Dienste auf einer oder zwei Maschinen reicht häufig ein schlichter Dienstmanager mit weniger beweglichen Teilen.

Wie es in der Praxis aussieht

Die Technologiestack-Seite des Unternehmens sagt es deutlich: Docker und Kubernetes für die Arbeitslasten, die ihren Einsatz rechtfertigen, gewöhnliche systemd-Units für die übrigen. Kubernetes wird je Arbeitslast nach Größe und Art der Aufgabe gewählt, nicht standardmäßig.

Arbeitslasten, die den Einsatz rechtfertigten, gab es in einem Auftrag. Zwanzig Docker-Anwendungen wurden über Google Cloud und AWS hinweg bereitgestellt und betreut, darunter Analyseanwendungen auf R-Basis. Einheitliche Konfiguration und Überwachung verhinderten, dass eine Anwendung zum Sonderfall wurde. Die Fehlersuche lief über Podman, die Fehlersuche in der Orchestrierung über Kubernetes. Die Seite berichtet von schnellerer Diagnose sowie stabilen und reproduzierbaren Bereitstellungen.

Die andere Hälfte der Regel zeigt die eigene Plattform des Unternehmens. Eine Containerebene auf Podman und Quadlet führt jeden Container als Dienst des betriebssystemeigenen Supervisors aus, in derselben deklarativen Form beschrieben wie alles andere auf dem Host. Die Container verwenden das Hostnetzwerk und richten keine Portfreigaben auf Containerebene ein. Daher bleiben die von der Härtungsrolle installierten Firewallregeln maßgeblich. Wo Kubernetes ein neues Image über seine API ausrollen würde, führt ein Benutzerkonto für die Bereitstellung ohne Rootrechte ein festes Skript aus, das das Image baut und den Dienst neu startet. Auf einem Host erhält so jeder Container eine schriftliche Beschreibung und den Supervisor, dem der Host bereits vertraut, ohne zweiten Prozessmanager. Ausgelassen werden die Verteilung über Maschinen und der Ersatz eines ausgefallenen Knotens. Genau dort beginnt sich der zusätzliche Betriebsaufwand von Kubernetes zu lohnen.

Wo es schiefgeht

Die meisten Kubernetes-Probleme entstehen in der Lücke zwischen den Anforderungen eines Manifests und der Rückmeldung des Clusters.

Eine Bereitstellung bleibt halb fertig, während der Bereitstellungsjob Erfolg meldet. Werden neue Pods nie bereit, etwa wegen eines falschen Imagetags, eines Absturzes beim Start, einer fehlgeschlagenen Readiness-Probe oder eines fehlenden Kontingents, macht das Deployment keinen Fortschritt mehr. Nach Ablauf der Fortschrittsfrist, standardmäßig 600 Sekunden, vermerkt es den Grund `ProgressDeadlineExceeded` . Laut Dokumentation unternimmt Kubernetes dann nichts weiter: Noch laufende alte Pods bedienen weiter Anfragen, und nichts wird selbstständig zurückgesetzt. `kubectl rollout status` endet nach Fristablauf mit einem von null verschiedenen Exitcode. Ein Bereitstellungsschritt kann daher darauf warten und die Veröffentlichung zurücknehmen:

```
kubectl apply -f web.yaml
kubectl rollout status deployment/web --timeout=10m \
  || kubectl rollout undo deployment/web
```

Ein Container startet mit `OOMKilled` neu oder wird auf einem Knoten mit freier CPU langsam. Der Kernel setzt Grenzen durch. Dabei verhalten sich die beiden Ressourcen unterschiedlich. Ein Container über seiner Speichergrenze kann beendet und neu gestartet werden. Einer an seiner CPU-Grenze wird gedrosselt, läuft also langsamer, ohne zu sterben. Das sieht wie ein Anwendungsproblem aus. Der Scheduler platziert Pods nach ihren Anforderungen, nicht nach ihrer tatsächlichen Nutzung. Niedrige Anforderungen überfüllen daher einen Knoten, hohe lassen Maschinen ungenutzt. Eine Grenze ohne Anforderung wird selbst zur Anforderung. Legen Sie Speicher anhand gemessener Nutzung fest und entscheiden Sie bewusst über CPU-Grenzen.

Eine langsam startende Anwendung wird vor Ende ihres Starts beendet. Eine fehlschlagende Liveness-Probe lässt das kubelet den Container beenden und neu starten. Braucht eine Anwendung eine Minute zum Laden, wird aber nach zehn Sekunden geprüft, startet sie endlos neu. Eine Startup-Probe hält die übrigen Prüfungen zurück, bis die Anwendung läuft. Eine Readiness-Probe entfernt einen Pod aus seinem Service, ohne ihn neu zu starten. Das passt für einen beschäftigten Pod oder einen, der einen Cache aufwärmt. Eine Liveness-Probe mit Datenbankaufruf startet bei langsamer Datenbank alle Replikate zugleich neu und macht aus einer langsamen Abhängigkeit einen Ausfall.

Ein von Hand gestarteter Pod verschwindet, und eine manuelle Korrektur wird rückgängig gemacht. Ein eigenständig erstellter Pod bleibt auf seinem Knoten, bis er endet, gelöscht oder verdrängt wird oder der Knoten ausfällt. Kein Controller erstellt ihn neu. Die Dokumentation empfiehlt selbst für eine einzelne

Kopie ein Deployment oder einen Job. Eine mit `kubectl edit` vorgenommene Änderung lebt nur im Cluster. Das nächste Anwenden der Dateien aus dem Repository setzt den alten Wert zurück. Eine Notfallkorrektur gehört daher in die Dateien, sobald der Notfall vorbei ist.

Weiterführende Lektüre

- Kubernetes im Überblick — Die eigene Darstellung des Projekts: Was Kubernetes ist, was es bietet und was es auslässt.
- Pods — Die kleinste von Kubernetes eingeplante Einheit, ihr Lebenszyklus und warum Pods selten von Hand erstellt werden.
- Deployments — Fortlaufende Aktualisierungen, Rollbacks, Revisionsverlauf und Fortschrittsfrist.
- Service — Stabile Namen und Adressen vor einer wechselnden Gruppe von Pods.
- Ressourcenverwaltung für Pods und Container — Anforderungen, Grenzen und ihre Durchsetzung durch den Kernel.
- Liveness-, Readiness- und Startup-Probes konfigurieren — Die drei Prüfungen und der Schutz eines langsam startenden Containers.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Kubernetes?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/kubernetes/