

Engineer

Kubernetes

⁂ et system der holder en flåde af containere kørende



Hvad det er

Kubernetes, ofte forkortet K8s, er open source-software, der kører containeriserede applikationer på en gruppe maskiner, en klynge, og hele tiden holder dem på den beskrivelse, man har givet den: hvilket image hver applikation kører, hvor mange kopier der skal findes, hvor meget hukommelse og CPU hver kopi må bruge, og hvordan andre programmer når frem til den. Projektets egen dokumentation kalder det en portabel og udvidelig platform til at styre containeriserede arbejdsbyrder og tjenester. Navnet er græsk for rorsmand eller pilot, tallet i K8s tæller de otte bogstaver mellem K og s, og Google frigav projektet som open source i 2014.

En klynge har to halvdele. Kontrolplanet rummer beskrivelsen: API-serveren er den ene HTTP-grænseflade, som enhver ændring går igennem, etcd gemmer det, den får at vide, scheduleren tildeler hver ny pod en passende node, og controller-manageren kører de løkker, der handler på beskrivelsen. På hver arbejdsnode sørger kubelet for, at de pods, der er tildelt den, kører, kube-proxy vedligeholder netværksreglerne bag Services, og en container-runtime som containerd eller CRI-O starter selve containerne. Docker Engine holdt op med at være en indbygget runtime i Kubernetes 1.24, men images bygget med Docker kører uændret, fordi de er almindelige OCI-images.

Beskrivelsen består af objekter, som regel skrevet i YAML. En pod er den mindste enhed, Kubernetes placerer: en eller flere containere, der deler netværksadresse og lager på én maskine. Et Deployment angiver, hvor mange ens pods der skal findes, og udskifter dem trinvis, når deres skabelon ændres. En Service giver en skiftende flok pods ét fast navn og én fast adresse og vælger sine medlemmer ud fra labels. En lille webapplikation ser sådan ud:

```
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 holdes typisk op mod tre mindre løsninger. Docker Compose beskriver containere på én maskine, og forsvinder maskinen, forsvinder de med den. En tjenestestyring som systemd kan genstarte en container, der går ned, og starte den, når maskinen booter; det dækker rigtig mange behov med langt mindre maskineri, men den kan ikke flytte arbejde til en anden maskine. Hostede platforme som Google Cloud Run eller Azure Container Apps kører containere uden at overlade et kontrolplan til dig, til gengæld med færre indstillinger. Dokumentationen siger også ligeud, at Kubernetes er noget andet end en traditionel alt-i-en-PaaS: det bygger ikke din applikation og leverer hverken databaser eller beskedkøer selv.

Brug begrebet i en beslutning

Brug »Hvad er Kubernetes?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Pakket, så den opfører sig identisk på en bærbar, en testserver og i produktion, og skalerer op, når der kommer en travl uge.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har drevet 20 Docker-apps på Google Cloud og AWS — 20 containeriserede applikationer udrullet og overvåget

Har drevet 20 Docker-apps på Google Cloud og AWS Kontakt os →

Hvorfor det betyder noget

Grundtanken er, at klyngen hele tiden arbejder hen imod en skreven beskrivelse i stedet for at køre en række kommandoer én gang. Når filen ovenfor anvendes, starter den ikke noget direkte; den registrerer et ønske. Controllere er kontrolløkker, der holder øje med klyngen og handler for at bringe den aktuelle tilstand tættere på den ønskede. Dør en maskine, så tre kopier bliver til to, ser Deploymentets controller forskellen, en ny pod oprettes, og scheduleren placerer den på en maskine med plads. Og fordi beskrivelsen er filer, kan den ligge i et repository og gennemgås på samme måde som den kode, den kører.

Udrulning følger med platformen. Et Deployments standardstrategi er en rullende opdatering: nye pods starter, og gamle stopper i takt med, at de nye bliver klar, inden for grænser, der som standard er 25 % ekstra og 25 % manglende ad gangen. Hver ændring af pod-skabelonen gemmes som en revision, ti som standard, og `kubectl rollout undo` går tilbage til den forrige. Det script, der engang tømte en server, skiftede en binær fil ud og satte serveren i drift igen, bliver til indbygget adfærd.

Scheduleren ændrer også regnestykket for en flåde af maskiner. Den placerer hver pod efter den CPU og hukommelse, poden anmoder om, så mange små tjenester kan dele et sæt maskiner i stedet for at få hver sin server dimensioneret til den værste dag.

Prisen er reel. En klynge er selv et distribueret system med versioner, der skal opgraderes efter en plan, et netværkslag, et lagerlag, certifikater og adgangskontrol, og nogen skal forstå det hele, når det driller. Administrerede tilbud som Google Kubernetes Engine, Amazon EKS og Azure Kubernetes Service driver kontrolplanet for dig, men hvert manifest og som regel også arbejdsnoderne forbliver dit ansvar. Til en håndfuld tjenester på en eller to maskiner klarer en almindelig tjenestestyring ofte opgaven med færre bevægelige dele.

Sådan ser det ud i praksis

Virksomhedens side om teknologier trækker en klar grænse: Docker og Kubernetes bruges til de arbejdsbyrder, der har brug for dem, og almindelige systemd-units til resten. Kubernetes vælges efter opgavens størrelse og form, ikke som en vane.

Arbejdsbyrder af den første slags ses i én opgave. Tyve Docker-applikationer kørte i drift på Google Cloud og AWS, heriblandt analyseapps skrevet i R, med ensartet konfiguration og overvågning, så hver applikation fulgte samme mønster. Fejlsøgning foregik gennem Podman, og fejlsøgning af orkestreringen gennem Kubernetes. Siden melder om hurtigere diagnose af problemer og om udrulninger, der holdt sig stabile og gav samme resultat hver gang.

Den anden halvdel af reglen ses på virksomhedens egen platform. Et containerplan bygget på Podman og Quadlet kører hver container som en tjeneste under styresystemets egen supervisor, beskrevet i samme deklarative form som resten af værten. Containerne bruger værtens netværk og publicerer ingen porte, så de firewallregler, hærdningsrollen installerer, bevarer kontrollen. En uprivilegeret bruger udruller ved at køre ét fast script, der bygger imaget og genstarter tjenesten. På én vært giver det hver container en skreven beskrivelse og den supervisor, værten allerede stoler på, uden en ekstra processtyring ved siden af. Det, opsætningen lader ligge, er placering på tværs af maskiner og erstatning af en død node, og det er netop dér, Kubernetes begynder at være sin vægt værd.

Hvor det går galt

De fleste problemer med Kubernetes opstår i afstanden mellem det, et manifest beder om, og det, klyngen melder tilbage.

En udrulning går i stå halvvejs, mens udrulningspipelinen melder succes. Når nye pods aldrig bliver klar, på grund af et forkert image-tag, et nedbrud ved start, en readiness-probe, der fejler, eller en kvote, der slipper op, holder Deploymentet op med at gøre fremskridt. Efter fremdriftsfristen, som standard 600 sekunder, registrerer det årsagen `ProgressDeadlineExceeded`, og ifølge dokumentationen gør Kubernetes ikke mere end det: de gamle pods, der stadig kører, betjener fortsat trafikken, og intet rulles tilbage af sig selv. `kubectl rollout status` afslutter med en exitkode forskellig fra nul, når fristen er overskredet, så et deploy-trin kan vente på den og rulle udgivelsen tilbage:

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

En container genstarter med `OOMKilled` eller bliver langsom på en node med ledig CPU. Grænserne håndhæves af kernen, og de to ressourcer opfører sig forskelligt. En container over sin hukommelsesgrænse kan blive dræbt og genstartet, når kernen mærker pres på hukommelsen; en container ved sin CPU-grænse bliver droslet og kører langsommere uden at dø, hvilket ligner en fejl i applikationen. Scheduleren placerer pods efter det, de anmoder om, og ikke efter det, de bruger, så lave requests overfylder en node, og høje lader maskiner stå ubrugte hen. En grænse uden request bliver selv til requesten. Sæt hukommelse ud fra målt forbrug, og gør CPU-grænser til en bevidst beslutning.

En applikation, der starter langsomt, bliver dræbt, før den er kommet op. Når en liveness-probe fejler, dræber kubelet containeren og starter den igen, så en applikation, der skal bruge et minut på at starte, men tjekkes efter ti sekunder, genstarter i det uendelige. En startup-probe holder de andre tjek tilbage, indtil applikationen kører. En readiness-probe tager en pod ud af sin Service uden at genstarte den, hvilket passer til en pod, der har travlt eller varmer en cache op. En liveness-probe, der kalder databasen, genstarter alle replikaer på én gang, når databasen er langsom, og gør dermed en langsom afhængighed til et nedbrud.

En pod, der er startet i hånden, forsvinder, og en rettelse lavet i hånden bliver rullet tilbage. En pod oprettet på egen hånd bliver på sin node, til den afslutter, slettes, smides ud, eller noden svigter, og ingen controller genskaber den; dokumentationen anbefaler et Deployment eller et Job, selv til én enkelt kopi. En ændring lavet med `kubectl edit` findes kun i klyngen, og næste apply fra repositoriet sætter den gamle værdi tilbage, så en nødrettelse hører hjemme i filerne, så snart nøden er overstået.

Læs mere

- Kubernetes overview — projektets egen beskrivelse af, hvad Kubernetes er, hvad det tilbyder, og hvad det lader ligge.
- Pods — den mindste enhed, Kubernetes placerer, dens livsforløb, og hvorfor pods sjældent oprettes i hånden.
- Deployments — rullende opdateringer, tilbagerulning, revisionshistorik og fremdriftsfristen.
- Service — faste navne og adresser foran en skiftende flok pods.

- Resource Management for Pods and Containers — requests, grænser, og hvordan kernen håndhæver hver af dem.
- Configure Liveness, Readiness and Startup Probes — de tre prober, og hvordan en container, der starter langsomt, beskyttes.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er Kubernetes?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af **D.**

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/kubernetes/