Engineer

Kubernetes

⁂ et system som holder en flåte av containere i gang

Hva det er

Kubernetes, ofte skrevet K8s, er åpen kildekode som kjører containeriserte applikasjoner på en gruppe maskiner, kalt en klynge, cluster på engelsk, og holder dem i samsvar med en beskrivelse du gir den: hvilket image hver applikasjon kjører, hvor mange kopier som finnes, hvor mye minne og CPU hver kopi får bruke, og hvordan andre programmer når den. Prosjektets egen dokumentasjon kaller det en portabel og utvidbar plattform med åpen kildekode for å forvalte containeriserte arbeidslaster og tjenester. Navnet er gresk for rorsmann eller los, K8s teller de åtte bokstavene mellom K og s, og Google gjorde prosjektet til åpen kildekode i 2014.

En klynge har to halvdeler. Kontrollplanet holder beskrivelsen: API-serveren er det ene HTTP-grensesnittet hver endring går gjennom, etcd lagrer det den får beskjed om, scheduleren tildeler hver ny Pod en egnet node, og controller manager kjører løkkene som handler ut fra beskrivelsen. På hver arbeidsnode sørger kubelet for at Pods som er tildelt den, kjører, kube-proxy vedlikeholder nettverksreglene bak Services, og en containerkjøretid som containerd eller CRI-O starter containerne. Docker Engine sluttet å være en innebygd kjøretid i Kubernetes 1.24, og images bygd med Docker kjører uendret, fordi de er standard OCI-images.

Beskrivelsen er et sett objekter, som regel skrevet i YAML. En Pod er den minste enheten Kubernetes planlegger: én eller flere containere som deler en nettverksadresse og lagring på én maskin. En Deployment sier hvor mange identiske Pods som skal finnes, og bytter dem ut trinn for trinn når malen deres endres. En Service gir et skiftende sett Pods ett stabilt navn og én stabil adresse, og velger medlemmene sine etter etikett. En liten webapplikasjon ser slik ut:

```
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
```

Den veies som regel opp mot tre mindre alternativer. Docker Compose beskriver containere på én maskin, og når den maskinen forsvinner, gjør de det også. En tjenestestyrer som systemd kan starte en krasjet container på nytt og starte den ved oppstart, noe som dekker svært mange behov med langt mindre maskineri, men den har ikke noe begrep om å flytte arbeid til en annen maskin. Plattformer i skyen som Google Cloud Run eller Azure Container Apps kjører containere uten å gi deg et kontrollplan å drive, mot at du får færre innstillinger. Dokumentasjonen sier også tydelig hva Kubernetes ikke er: en tradisjonell, altomfattende plattform som tjeneste. Den bygger ikke applikasjonen din, og den leverer ingen egne databaser eller meldingskøer.

Bruk begrepet i en beslutning

Bruk «Hva er Kubernetes?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Pakket slik at den oppfører seg helt likt på en bærbar PC, på en testserver og i produksjon, og skalerer opp når en travel uke kommer.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Driftet 20 Docker-apper på Google Cloud og AWS — 20 containeriserte applikasjoner produksjonssatt og overvåket

Driftet 20 Docker-apper på Google Cloud og AWS Ta kontakt →

Hvorfor det er viktig

Den bærende ideen er at klyngen hele tiden arbeider mot en skrevet beskrivelse, i stedet for å kjøre en liste med kommandoer én gang. Å bruke filen ovenfor starter ingenting direkte; den registrerer et ønske. Kontrollere er kontrollsløyfer som følger med på klyngen og handler for å flytte den nåværende tilstanden nærmere den ønskede. Dør en maskin, og tre kopier blir to, ser kontrolleren for Deployment-objektet avviket, en ny Pod opprettes, og scheduleren plasserer den på en maskin med plass. Fordi beskrivelsen er et sett filer, kan den ligge i et repo og gå gjennom den samme gjennomgangen som koden den kjører.

Utgivelser følger med plattformen. Standardstrategien for en Deployment er en rullerende oppdatering: nye Pods starter, og gamle stopper etter hvert som de nye blir klare, innenfor grenser som som standard er 25 % ekstra og 25 % manglende til enhver tid. Hver endring av Pod-malen tas vare på som en revisjon, ti som standard, og `kubectl rollout undo` går tilbake til den forrige. Skriptet som en gang tømte en server, byttet en binærfil og satte den inn igjen, blir innebygd oppførsel.

Scheduleren endrer også økonomien i en maskinpark. Den plasserer hver Pod etter den CPU-en og det minnet Pod-en ber om, så mange små tjenester deler et sett maskiner, i stedet for at hver av dem får en server dimensjonert for sin verste dag.

Prisen er reell. En klynge er et distribuert system i seg selv, med versjoner som må oppgraderes etter en plan, et nettverkslag, et lagringslag, sertifikater og tilgangskontroll, og noen må forstå alt dette når det oppfører seg rart. Administrerte tilbud som Google Kubernetes Engine, Amazon EKS og Azure Kubernetes Service driver kontrollplanet for deg, men hvert manifest, og som regel arbeidsnodene, forblir ditt ansvar. For en håndfull tjenester på én eller to maskiner gjør en vanlig tjenestestyrer ofte jobben med færre bevegelige deler.

Slik ser det ut i praksis

Selskapets side om teknologier sier regelen rett ut: Docker og Kubernetes for de arbeidslastene som fortjener dem, og vanlige systemd-enheter for dem som ikke gjør det. Kubernetes velges per arbeidslast, ut fra størrelsen og formen på jobben, og ikke som et standardvalg.

Arbeidslaster som fortjente det, dukker opp i ett oppdrag. Tjue Docker-applikasjoner ble produksjonssatt og vedlikeholdt i Google Cloud og AWS, blant dem R-baserte analyseapper, med konsekvent konfigurasjon og overvåking, slik at ingen applikasjon ble et spesialtilfelle. Feilsøkingen gikk gjennom Podman, og feilsøkingen av orkestreringen gjennom Kubernetes. Siden rapporterer at problemer ble diagnostisert raskere, og at produksjonssettingene forble stabile og reproduserbare.

Den andre halvdelen av regelen vises på selskapets egen plattform. Et containerlag bygd på Podman og Quadlet kjører hver container som en tjeneste under operativsystemets egen tjenestestyrer, beskrevet i den samme deklarative formen som alt annet på verten. Containerne bruker vertens nettverk og publiserer ingen porter, så brannmurreglene som herdingsrollen installerer, beholder kontrollen. En utgivelse er ett fast skript som en uprivilegert bruker kjører, der Kubernetes ville tatt imot et manifest gjennom API-et, og skriptet bygger imaget og starter tjenesten på nytt. På én vert gir det hver container en skrevet beskrivelse og den tjenestestyreren verten allerede stoler på, uten noen ekstra prosessbehandler. Det som mangler, er plassering på tvers av maskiner og erstatning av en død node, og det er der Kubernetes begynner å betale for vekten sin.

Hvor det går galt

Det meste av trøbbelet med Kubernetes kommer fra avstanden mellom det et manifest ber om, og det klyngen melder tilbake.

En utrulling blir stående halvferdig mens leveransepipelinen melder suksess. Når nye Pods aldri blir klare, på grunn av en feil image-tagg, et krasj ved oppstart, en readiness-probe som feiler eller en kvote som mangler, slutter Deployment-objektet å gjøre fremskritt. Etter fristen for fremdrift, 600 sekunder som standard, registrerer den årsaken `ProgressDeadlineExceeded`, og dokumentasjonen sier at Kubernetes ikke gjør noe mer: de gamle Pods som fortsatt kjører,

fortsetter å betjene trafikken, og ingenting rulles tilbake av seg selv. `kubectl rollout status` avslutter med en annen kode enn null når fristen er passert, så et steg for produksjonssetting kan vente på den og angre utgivelsen:

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

En container starter på nytt med `OOMKilled`, eller blir treg på en node med ledig CPU. Systemkjernen håndhever grensene, og de to ressursene oppfører seg forskjellig. En container som går over minnegrensen sin, kan bli drept og startet på nytt; en som når CPU-grensen sin, blir strupet, så den går tregere uten å dø, og det ser ut som et problem i applikasjonen. Scheduleren plasserer Pods etter det de ber om, ikke etter det de bruker, så lave forespørsler stapper en node full, og høye lar maskiner stå ledige. En grense som gis uten en forespørsel, blir forespørselen. Sett minnet ut fra målt bruk, og behandle CPU-grenser som en bevisst avgjørelse.

En applikasjon som starter sakte, blir drept før den er ferdig med å starte. En liveness-probe som feiler, får kubelet til å drepe containeren og starte den på nytt, så en applikasjon som trenger ett minutt på å laste, og som sjekkes etter ti sekunder, starter på nytt i det uendelige. En startup-probe holder de andre sjekkene tilbake til applikasjonen er oppe. En readiness-probe tar en Pod ut av sin Service uten å starte den på nytt, noe som passer for en Pod som er opptatt eller varmer opp en cache. En liveness-probe som kaller databasen, starter hver replika på nytt samtidig når databasen er treg, og gjør en treg avhengighet om til nedetid.

En Pod startet for hånd forsvinner, og en rettelse gjort for hånd blir opphevet. En Pod som er opprettet alene, blir på noden sin til den avsluttes, slettes eller kastes ut, eller til noden svikter, og ingen kontroller oppretter den på nytt; dokumentasjonen anbefaler en Deployment eller en Job selv for én enkelt kopi. En endring gjort med `kubectl edit` finnes bare i klyngen, og neste gang noen bruker filene fra repoet, kommer den gamle verdien tilbake. En nødrettelse hører derfor hjemme i filene så snart nødsituasjonen er over.

Les mer

- Kubernetes overview — prosjektets egen beskrivelse av hva Kubernetes er, hva det tilbyr, og hva det utelater.
- Pods — den minste enheten Kubernetes planlegger, livssyklusen dens og hvorfor Pods sjelden opprettes for hånd.
- Deployments — rullerende oppdateringer, tilbakerulling, revisjonshistorikk og fristen for fremdrift.
- Service — stabile navn og adresser foran et skiftende sett Pods.
- Resource Management for Pods and Containers — forespørsler, grenser og hvordan kjernen håndhever hver av dem.
- Configure Liveness, Readiness and Startup Probes — de tre probene og hvordan du beskytter en container som starter sakte.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er Kubernetes?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/kubernetes/