kubernetes

internkurs – del 1

innhold

- bakgrunn
- teori
- oppgaver!

bakgrunn

bakgrunn

Hvordan skalerer man tusenvis av tjenester til milliarder av brukere?

bakgrunn

Hvordan lar man flere utvikler-team rulle ut tjenester med minimal friksjon?

bakgrunn

Hvordan kan man legge til rette for portabel, skalerbar og uavhengig infrastruktur?

bakgrunn

Behov for…

- deklarativ definisjon av applikasjons- og miljøkonfigurasjon
- felles standarder/API-er på tvers av språk/rammeverk/leverandører

.. og noe som er åpent og gratis!

hva er kubernetes?

“**Kubernetes**, also known as **k8s**, is an open source system for automating deployment, scaling, and management of containerized applications.”

kubernetes i dag

verdens nest-største open-source prosjekt (etter Linux)

50% av selskaper på Fortune 100 bruker Kubernetes

92% markedsandel av verktøy for orkestrering av containere

mer eller mindre alle (utviklings)platformer i verden er basert på kubernetes

“Cloud Native landscape”; du kan bygge din egen sky (hvis du vil)


teori

clustere, noder, pods og containere

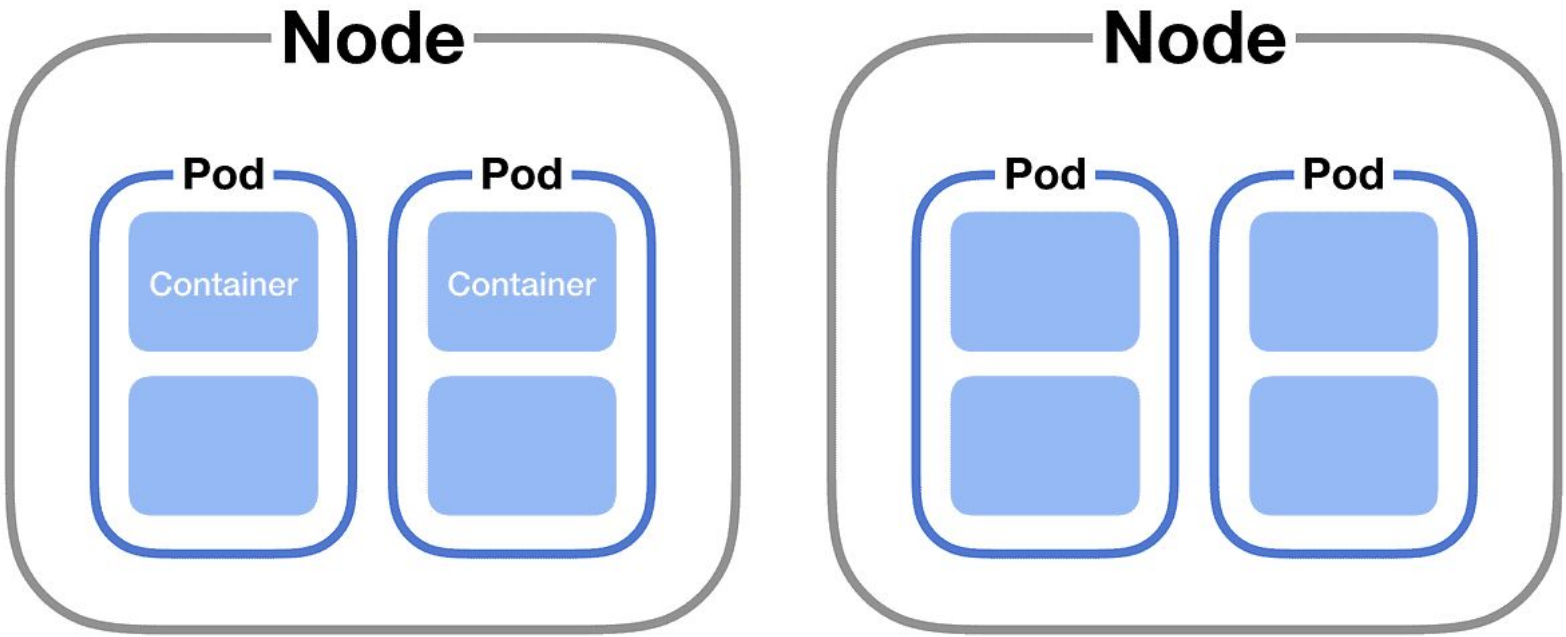
Et **Kubernetes-cluster** består av flere **noder**

En **node** er en virtuell maskin der flere **pods** (og andre ressurser) kan bo

En **pod** er en vanlig **container** (men kan også ha flere containere)

En **container** er en enkelt Linux-prosess (altså programmet ditt)

Cluster



interagering med kubernetes

Kubernetes har en egen API-server

Hvert “objekt” i Kubernetes er en **ressurs**

Alle ressurser har en **full spesifikasjon** som API-et forstår

En ressurs representerer en form for **“desired state”**:

- en **Pod** beskriver en container
- en **Secret** beskriver en hemmelighet
- en **ConfigMap** bestemmer konfigurasjon
- etc…

ressurser

Alle “objekter” i k8s er “ressurser”

De representerer en desired state

Alle har sin egen spec

K8s følger med de fleste man trenger

Kan lage egne ressurser (CR) og speccer for disse (CRD)

Lar deg representere alt i k8s som k8s-ressurser

Hemmeligheter, lagring, databaser, infrastruktur, sårbarheter osv… the sky is the limit

eksempel på spec (Pod)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: min-kule-app
spec:
  containers:
    - image: nginx:latest
      name: nginx
```

interagering med kubernetes (imperativt)

kubectl kan brukes for å nå API-et fra kommandolinjen:

```
kubectl run nginx --namespace min-kule-app --image nginx
```

interagering med kubernetes (deklarativt)

```
# pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: min-kule-app
spec:
  containers:
    - image: nginx:latest
      name: nginx
```

```
kubectl apply -f pod.yaml
```

namespaces

namespaces er en måte å isolere ressurser, og styre tilgang

vanlig å ha ett namespace **per team/system/applikasjon**

“alt som hører sammen” burde være i samme namespace

tilgang gis også gjerne på namespace-nivå

namespace må alltid settes, hvis ikke bruker vi namespacet **“default”**

```
kubectl -n min-kule-app get pods
```

Pod

representerer en kjørende container

den enkleste byggestenen for kjørende tjenester

ReplicaSet

representerer *n* antall **Pods**

bygger på spec'en til **Pod** (bokstavelig talt)

brukes aldri alene

e.g., et **ReplicaSet** med `replicas: 2` vil kreve at 2 **Pods** alltid kjører

ReplicaSet

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
  namespace: my-app
  …
spec:
  replicas: 3
  …
    spec:
      …
```

Deployment

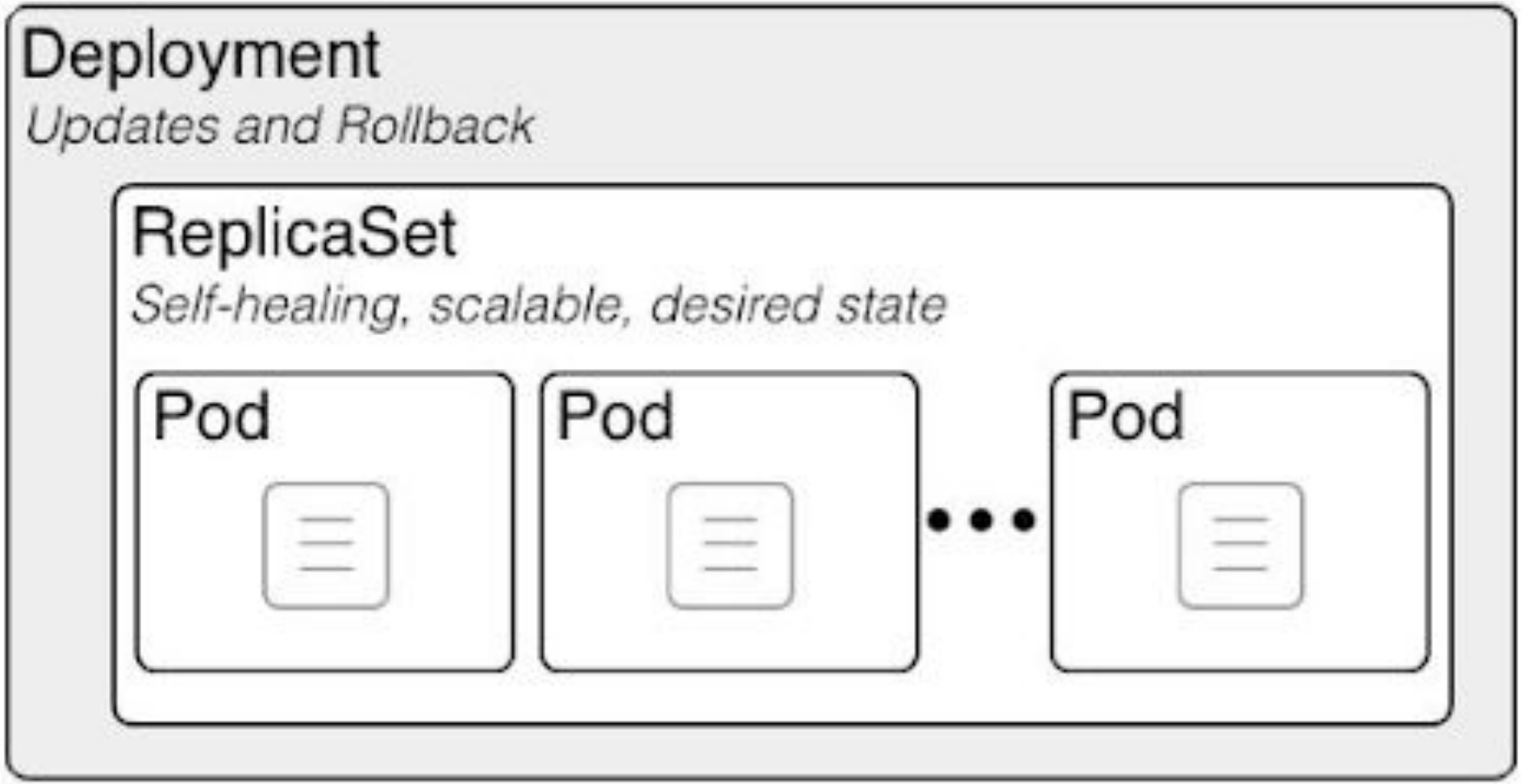
representerer utrulling av en tjeneste

bygger på spec'en til **ReplicaSet** (bokstavelig talt)

lar deg rulle ut/tilbake **ReplicaSets**

Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: my-app
  …
spec:
  replicas: 3
  …
    spec:
      …
```


Deployment
Updates and Rollback
ReplicaSet
Self-healing, scalable, desired state
Pod
Pod
Pod

ConfigMap

representerer *konfigurasjon*, kan brukes av f.eks. **Pods**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: frontend-config
  namespace: my-app
data:
  ENVIRONMENT: dev
  NEXT_PUBLIC_API_URL: https://my-app.example.com/api/v1
```

Secret

representerer *hemmelige verdier*, kan brukes av f.eks. **Pods**

```
apiVersion: v1
kind: Secret
metadata:
  name: very-secret-password
  namespace: my-app
data:
  PASSWORD: azhzLXdvb2hvb28=
  MY_OTHER_PASSWORD: aGVpc2Fubgo=
```

Service

representerer eksponering av en **Pod** til omverdenen

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: my-app
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - port: 80
      targetPort: 9376
```

NetworkPolicy

representerer begrensninger på nettverkskall fra/til **Pods**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: database
  namespace: five31
spec:
  policyTypes:
  - Ingress
  - Egress
  …
```

NetworkPolicy

representerer begrensninger på nettverkskall fra/til **Pods**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: database
  namespace: five31
spec:
  …
  podSelector:
    matchLabels:
      app: database
  …
```

NetworkPolicy

representerer begrensninger på nettverkskall fra/til **Pods**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: database
  namespace: five31
spec:
  …
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: backend
  …
```

spørsmål?

oppgaver

baksetercx/kubernetes-internkurs