Intro til Kubernetes

fra begyndelsen

Plan for workshop

Muligheder:

- Intro til Kubernetes
  - Gennemgang af begreber, værktøjer, demo’er…
  - Praktiske øvelser
  - Mini-projekt: oversætte Docker Compose miljø til Kubernetes
- Selvstudium, eksperimenter
- Hybrid af ovenstående

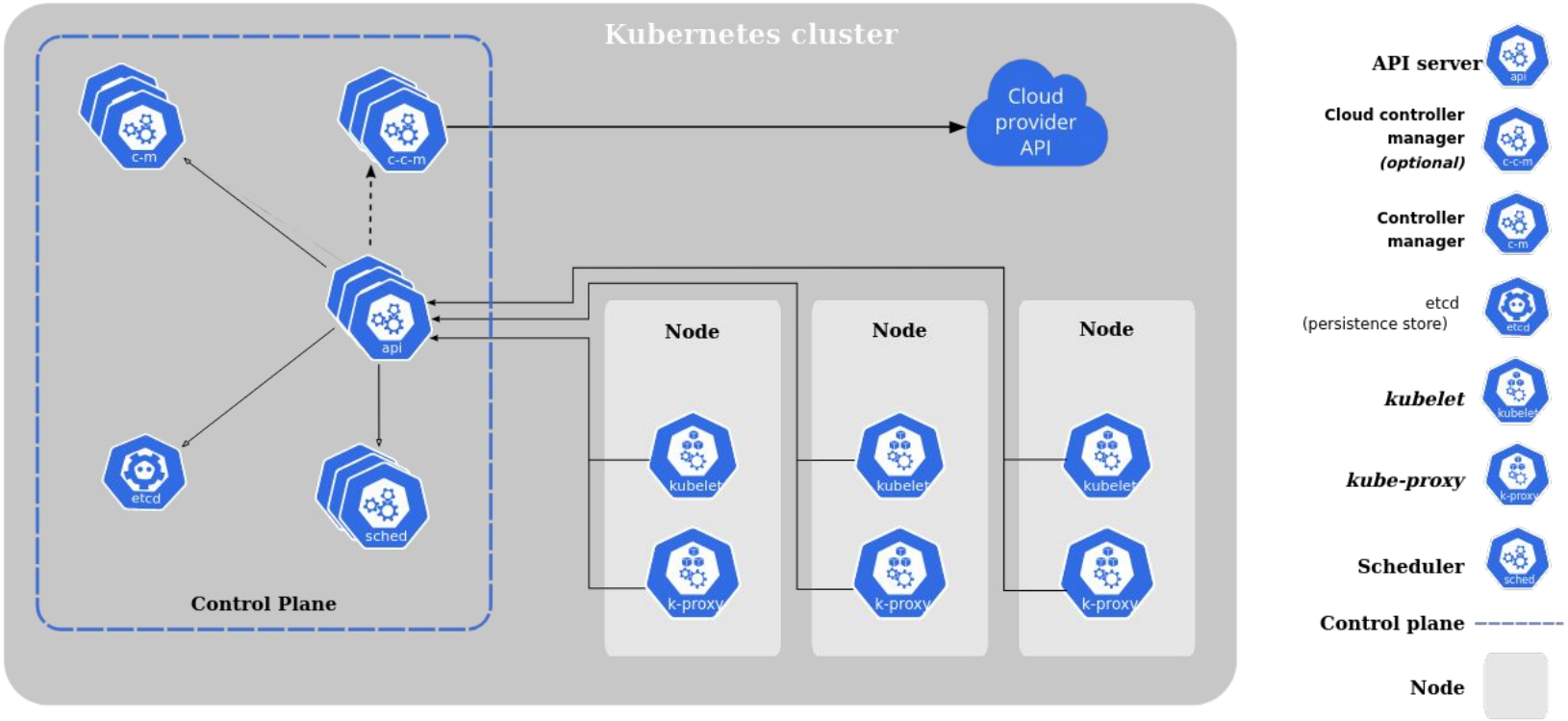
Intro til Kubernetes

- Kort intro til Kubernetes
- Lokale udviklingsmiljøer (Minikube, kind,...)
- Kommandolinjeværktøjer til at interagere med et Kubernetes cluster: `kubectl` og `k9s`
- Basale Kubernetes ressourcer (Pods, Deployments, Services, ConfigMaps, Secrets, Namespaces, Persistent volumes,...)
- Helm og Helm charts (herunder pakkehåndtering i Kubernetes)
- Ingress
- Autoskalering
- Flux
- …

Materiale til workshop

Kan ses her: https://github.com/andreaskring/kubernetes

Kubernetes



https://kubernetes.io/docs/concepts/overview/components/

Kubernetes ressourcetype (Pod)

“*Pods* are the smallest deployable units of computing that you can create and manage in Kubernetes.”

- Gruppe af en eller flere (sammenhængende) containere
- Co-located og co-scheduled
- Containerne deler storage og netværksressourcer
- Init-containere: kører og terminerer før hovedcontaineren starter

https://kubernetes.io/docs/concepts/workloads/pods/

Lokale udviklingsmiljøer

- Minikube (https://kubernetes.io/docs/tutorials/hello-minikube/)
- Kind (https://kind.sigs.k8s.io/)

Minikube og kubectl

Minikube

```
$ minikube start
$ minikube start --cpus=2 --memory=4g [--kubernetes-version=v1.26.3]
```

kubectl

```
$ kubectl get pod[s]
$ kubectl run my-nginx --image=nginx
$ kubectl delete pod my-nginx
$ kubectl create cronjob my-job --image=... --schedule="*/1 * * * *"
$ kubectl apply -f my-pod.yaml
$ …
```

Heads-up: kubectl og context!

```
$ kubectl config get-contexts
CURRENT   NAME                                          CLUSTER                                       AUTHINFO                                                NAMESPACE
          do-fra1-k8s-1-22-8-do-0-fra1-1650268880803    do-fra1-k8s-1-22-8-do-0-fra1-1650268880803    do-fra1-k8s-1-22-8-do-0-fra1-1650268880803-admin
          kind-crypto                                   kind-crypto                                   kind-crypto
          kind-gcoord                                   kind-gcoord                                   kind-gcoord
          kind-os2mo                                    kind-os2mo                                    kind-os2mo
*         minikube                                      minikube                                      minikube                                                default
          os2mo-ci-magenta-az                           os2mo-ci-magenta-az                           clusterUser_moraci_os2mo-ci-magenta-az
          os2mo-dev-magenta-az                          os2mo-dev-magenta-az                          clusterUser_moradev_os2mo-dev-magenta-az
          os2mo-dev-silkeborg-az                        os2mo-dev-silkeborg-az                        clusterUser_os2mo-rg_os2mo-dev-silkeborg-az
          os2mo-prod-magenta-az                         os2mo-prod-magenta-az                         clusterUser_morademo_os2mo-prod-magenta-az
          os2mo-prod-silkeborg-az                       os2mo-prod-silkeborg-az                       clusterUser_os2mo_prod_rg_os2mo-prod-silkeborg-az
          os2mo-saas-magenta-az                         os2mo-saas-magenta-az                         clusterUser_morasaas_os2mo-saas-magenta-az
          os2mo-test-magenta-az                         os2mo-test-magenta-az                         clusterUser_moratest_os2mo-test-magenta-az
          os2mo-test-silkeborg-az                       os2mo-test-silkeborg-az                       clusterUser_os2mo_test_rg_os2mo-test-silkeborg-az
          test-cluster                                  test-cluster                                  clusterUser_morasaas_test-cluster
```

k9s

- Din bedste Kubernetes-ven!
- Text GUI wrapper til kubectl
- Hentes fra https://k9scli.io/

Opsætning af miljø

---

1. Installér og start Minikube (https://minikube.sigs.k8s.io/docs/start/)
2. Installér kubectl (https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/)
3. Kør følgende:
   ```
   $ kubectl get pod
   $ kubectl run my-nginx --image=nginx
   $ kubectl get pod
   ```
4. Installér k9s (https://k9scli.io/)
   Nemmest at hente binary fra https://github.com/derailed/k9s/releases

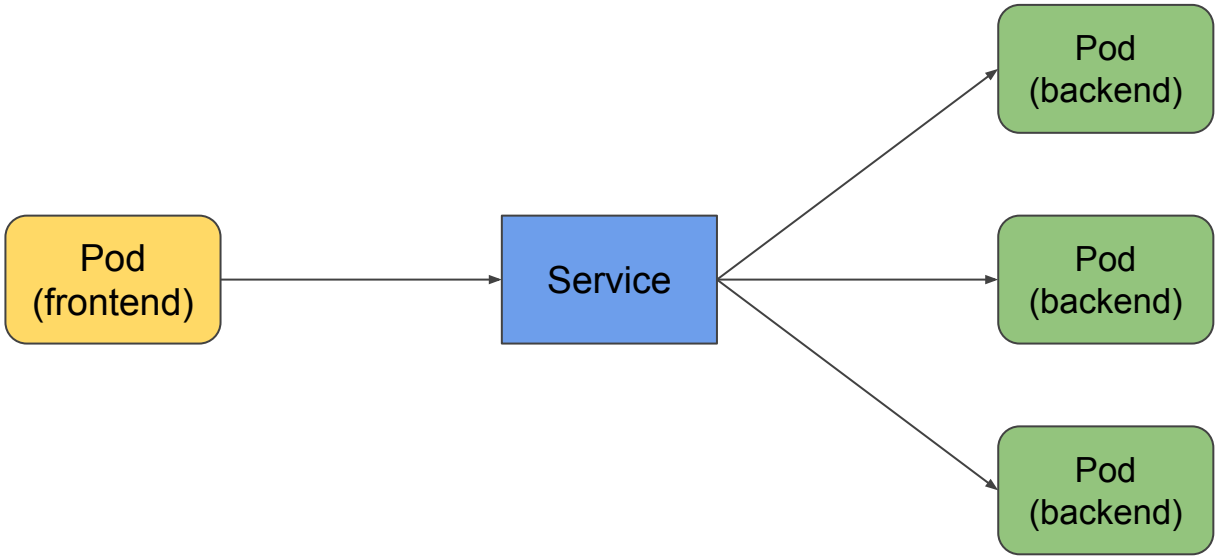
Kubernetes ressourcetype (Deployment)

“A *Deployment* provides declarative updates for Pods and ReplicaSets.”

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

https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Kubernetes ressourcetype (Service)



Kubernetes ressourcetype (Service)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8000
      targetPort: 80
```

Service types:

- ClusterIP
- NodePort
- LoadBalancer
- ExternalName


```
$ kubectl expose deployment nginx-deployment --port=80 --type NodePort
```

Kubernetes ressourcetyper (PV og PVC)

PV = Persistent Volume

PVC = Persistent Volume Claim

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  [storageClassName: xyz]
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Access modes:

- ReadWriteOnce
- ReadOnlyMany
- ReadWriteMany
- ReadWriteOncePod

Konfigurering af pods

- Miljøvariable
- ConfigMaps
- Secrets

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.14.2
      env:
        - name: SOME_ENV
          value: "some string"
      ports:
        - containerPort: 80
```

Kubernetes ressourcetype (ConfigMap)

“A ConfigMap is an API object used to store non-confidential data in key-value pairs. Pods can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a volume”

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"

  game.properties: |
     enemy.types=aliens,monsters
     player.maximum-lives=5
  user-interface.properties: |
     color.good=purple
     color.bad=yellow
     allow.textmode=true
```

https://kubernetes.io/docs/concepts/configuration/configmap/

Kubernetes ressourcetype (Secret)

“A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key”

```
$ echo -n 'admin' | base64
$ echo -n '1f2d1e2e67df' | base64
```

```yaml
---
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
type: Opaque
data:
  username: YWRtaW4=
  password: MWYyZDFlMmU2N2Rm
stringData:
  another-password: something-sensitive
```

https://kubernetes.io/docs/concepts/configuration/secret/

Kubernetes ressourcetype (Namespace)

“In Kubernetes, *namespaces* provides a mechanism for isolating groups of resources within a single cluster”

```
apiVersion: v1
kind: Namespace
metadata:
  name: some-namespace
```

https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

Øvelse

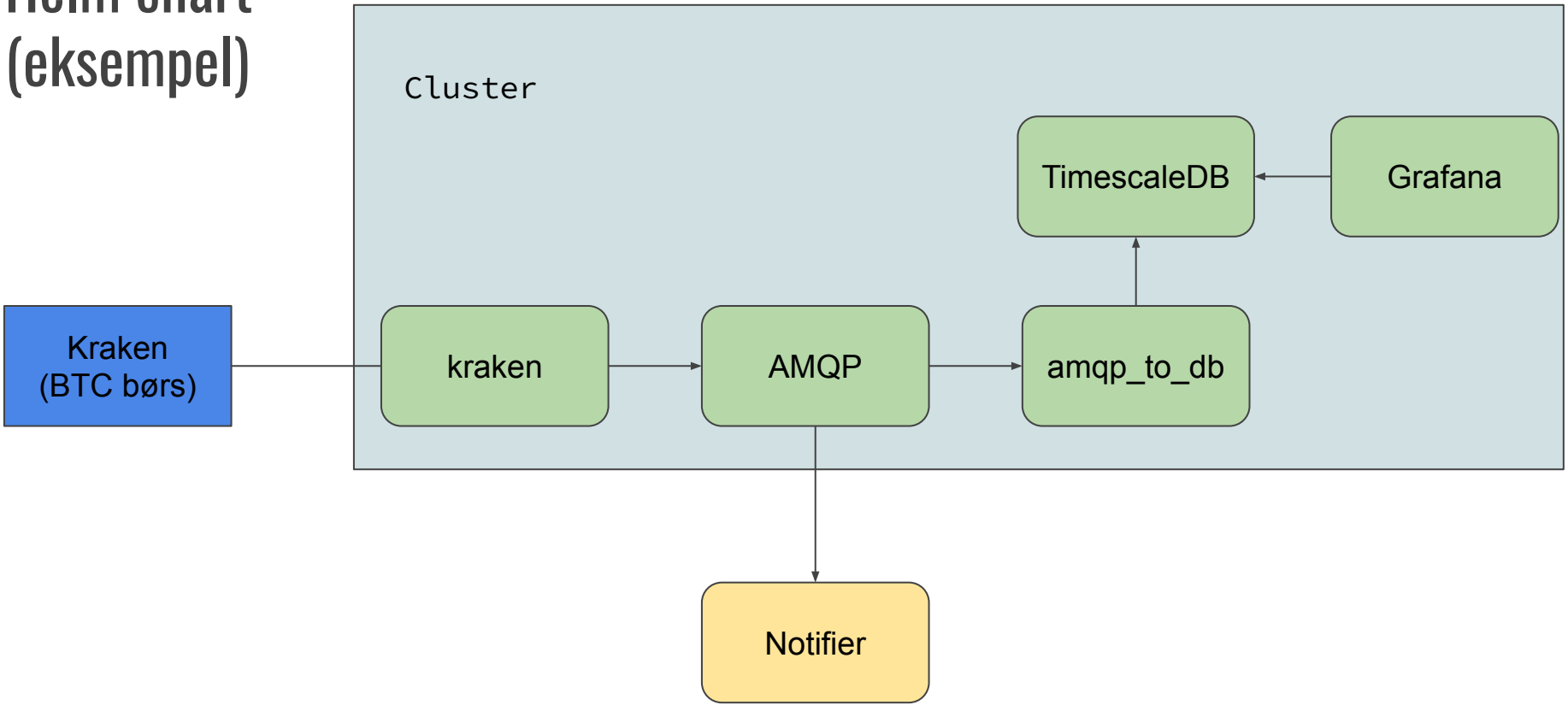
Start en Nginx-applikation i Minikube på flg. måde:

1. Nginx-pod'ene skal administreres af en Deployment
2. Der skal køre to Nginx-pods parallelt
3. En Service skal eksponere de to pods via en NodePort
4. Pod'ene skal konfigureres med to (vilkårlige) miljøvariable fra et ConfigMap og en miljøvariable via en Secret
5. Verificér, at du kan curl'e applikationen (fra hostmaskinen) og at de tre ENVs er sat korrekt

Helm charts

- Templating-mekanisme til Kubernetes ressourcer
- Pakkehåndteringsværktøj til Kubernetes
  - Pakke = chart
  - Repository: sted, hvor charts kan samles og deles
  - Release: instans af chart, som kører i et cluster
- Helm kan styre life cycle af releases (installation, upgrades, rollbacks,...)
- Et Helm chart kan afhænge af andre Helm charts
- Brug applikationseksperternes Helm charts i stedet for at skrive dem selv!

Helm chart (eksempel)



Helm charts

```
$ helm install <NAME> -f value-overrides.yaml chart/
$ helm upgrade --install <name> -f value-overrides.yaml chart/

$ helm template <NAME> chart/
$ helm repo list
$ helm rollback <RELEASE> <REVISION>
$ helm history <NAME>
$ …
```

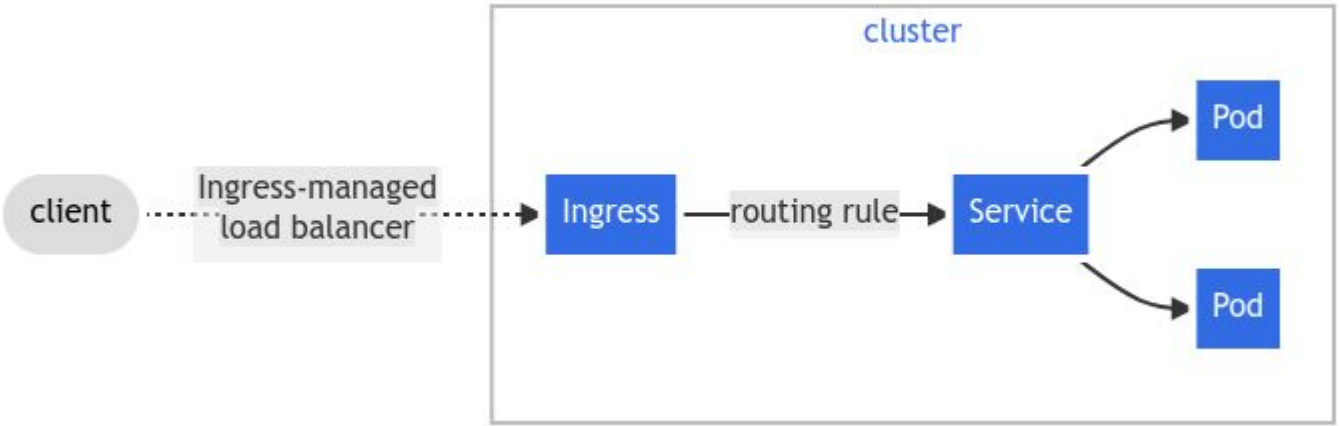
https://helm.sh/

Opsamling fra i går

- Help
  - $ kubectl --help
  - $ minikube --help
  - $ helm --help

- Muligt at bruge logik i Helm charts via template language

Kubernetes ressourcetype (Ingress)



Styring af CPU og memory forbrug for Pods

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
    - name: app
      image: images.my-company.example/app:v4
      resources:
        requests:
          memory: "64Mi"
          cpu: "250m"
        limits:
          memory: "128Mi"
          cpu: "500m"
```

Probes

- Liveness probes
- Readiness probes
- Startup probes

```
---
…
kind: Pod
spec:
  containers:
    …
    livenessProbe:
      httpGet:
        path: /health/live
        port: 5000
      periodSeconds: 60
      timeoutSeconds: 15
      failureThreshold: 3
    readinessProbe:
      httpGet:
        path: /health/ready
        port: 5000
      periodSeconds: 60
      timeoutSeconds: 15
      failureThreshold: 3
    startupProbe:
     httpGet:
        path: /health/live
        port: 5000
      failureThreshold: 150
      periodSeconds: 2
```

Mini-projekt

- Oversæt et Docker Compose miljø til et Helm chart, som kan udrulles i et Kubernetes cluster
- Vælg selv en passende docker-compose.yml fil
  - Meget gerne fra et Magenta-projekt, som du allerede arbejder på
  - Alternativt forslag: brug et af Docker Compose eksempler fra https://docs.docker.com/compose/samples-for-compose/
- Man er velkommen til at arbejde i grupper

Mini-projekt (fremgangsmåde)

---

1. helm create my-awesome-project
2. Tag een komponent ad gangen og lav for denne følgende (test for hvert skridt):
    a. En Deployment (hard-code ENVs mv. i første omgang)
    b. Introducér eventuelle ConfigMaps og Secrets
    c. Template værdier ud i values.yaml
    d. Lav en Service for Deployment’en