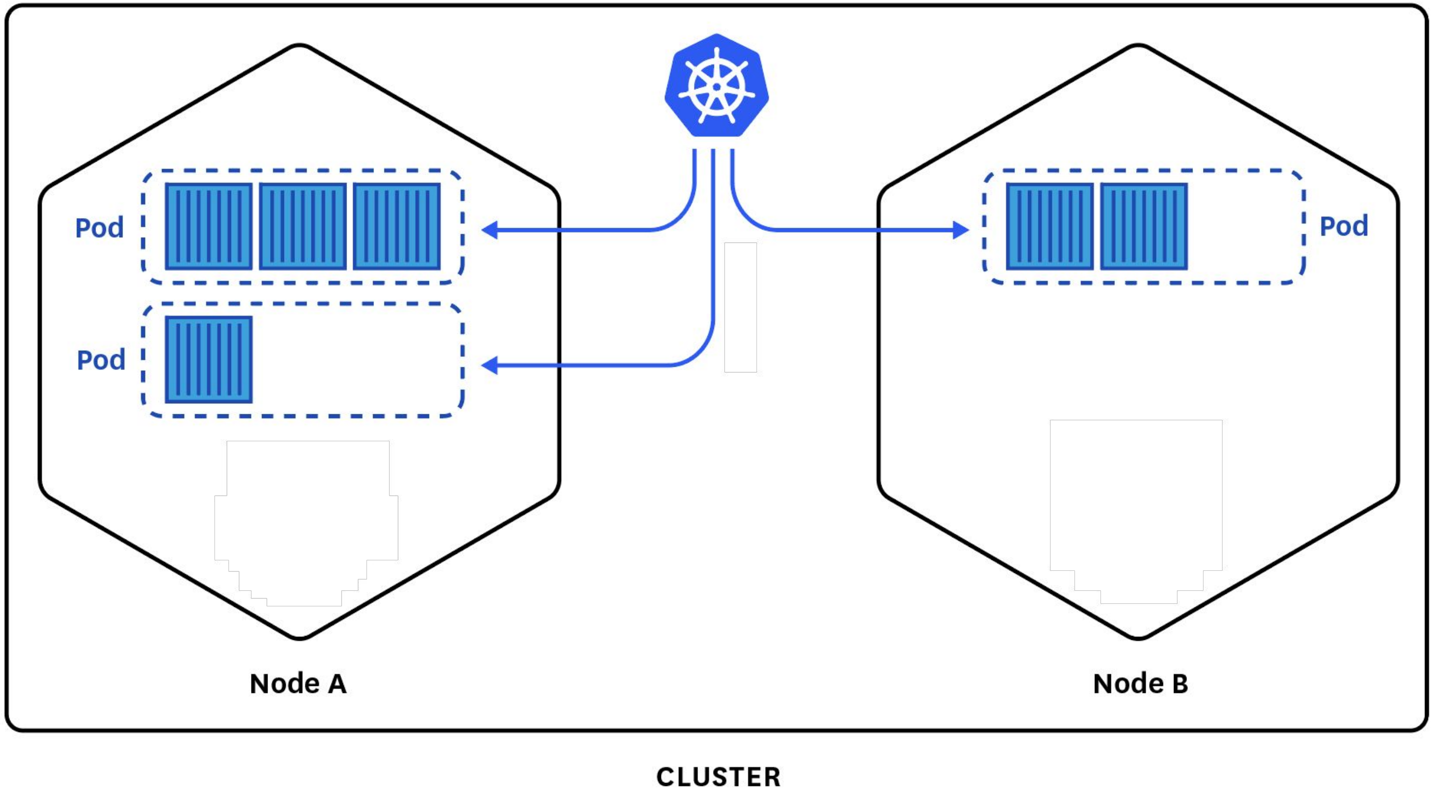
Kubernetes workshop

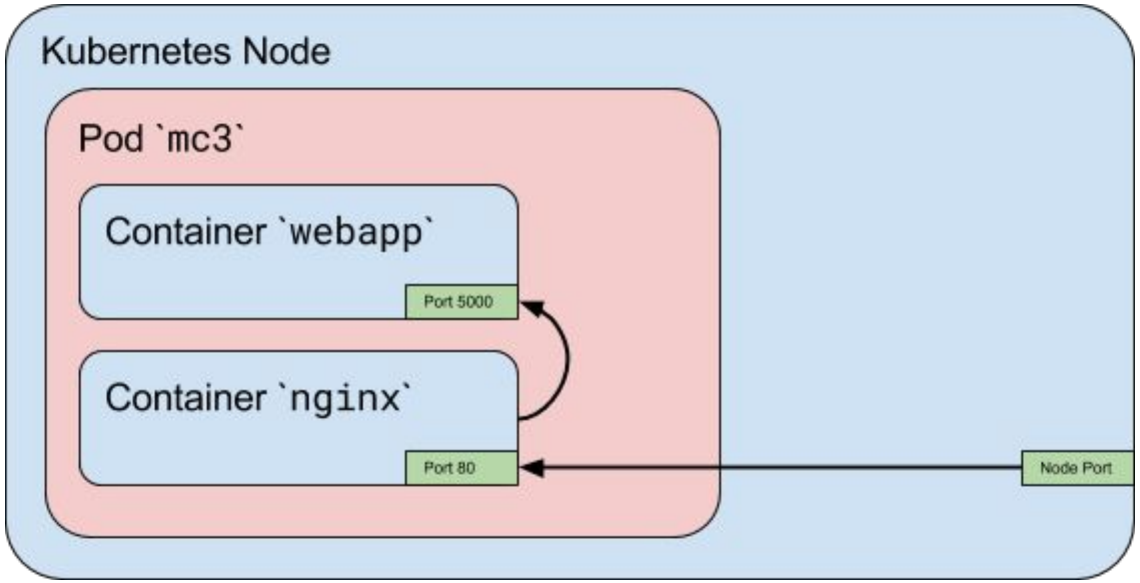
Konsepter

- Node
- Pod
- ReadinessProbe, Livenessprobe
- Deployment
- Service
- Service Discovery
- Ingress
- Configmap
- Secret
- PersistentVolume
- PersistentVolumeClaim
- Namespace
- Label

Node og Pod

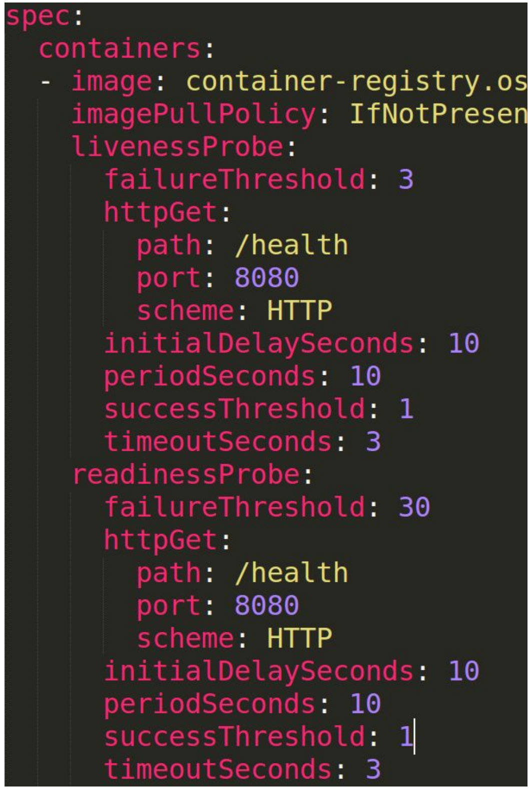


Pod


Kubernetes Node
Pod `mc3`
Container `webapp`
Port 5000
Container `nginx`
Port 80
Node Port

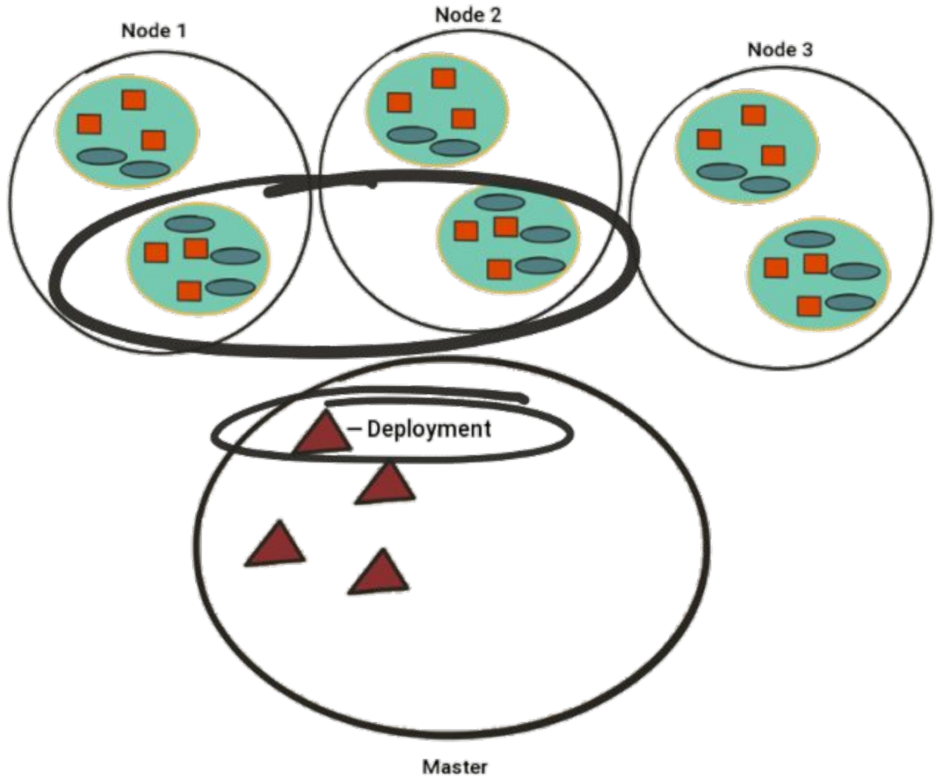
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

Helsesjekk



```
spec:
  containers:
  - image: container-registry.os
    imagePullPolicy: IfNotPresen
    livenessProbe:
      failureThreshold: 3
      httpGet:
        path: /health
        port: 8080
        scheme: HTTP
      initialDelaySeconds: 10
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 3
    readinessProbe:
      failureThreshold: 30
      httpGet:
        path: /health
        port: 8080
        scheme: HTTP
      initialDelaySeconds: 10
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 3
```

Deployment



Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Service

Pods

- Kræsjer
- Skifter IP

Løsning: Services

- Har fast IP
- Ruter trafikk videre til pods

Service

```yaml
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 9376
```

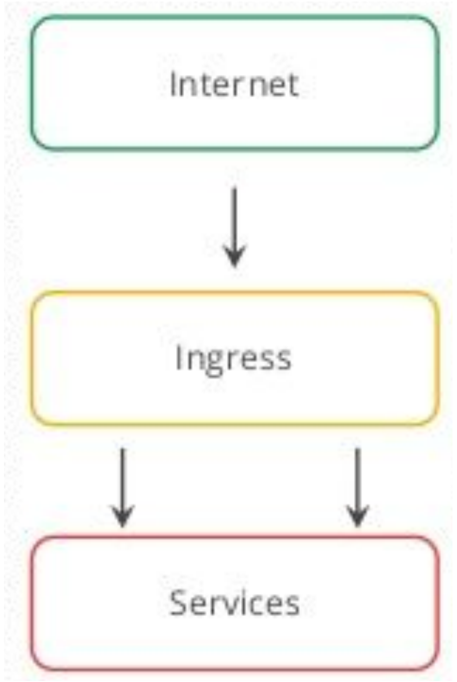
Service discovery

For å nå en service fra en pod/container:

Eks 1: http://servicename

Eks 2: $ ping servicename

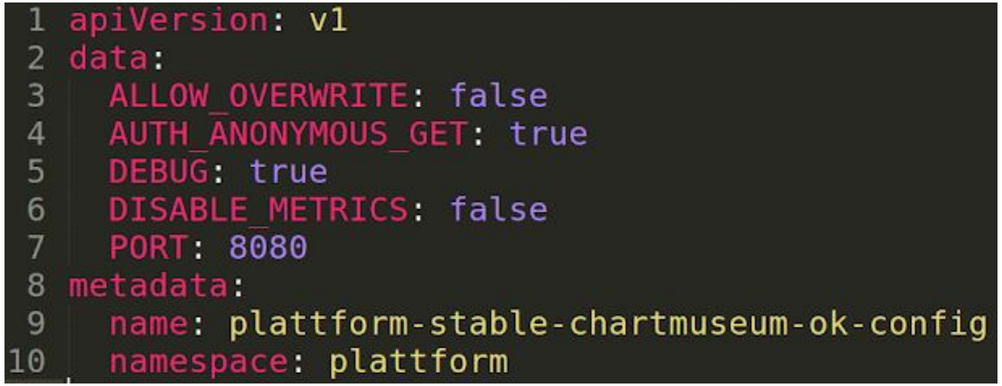
Ingress



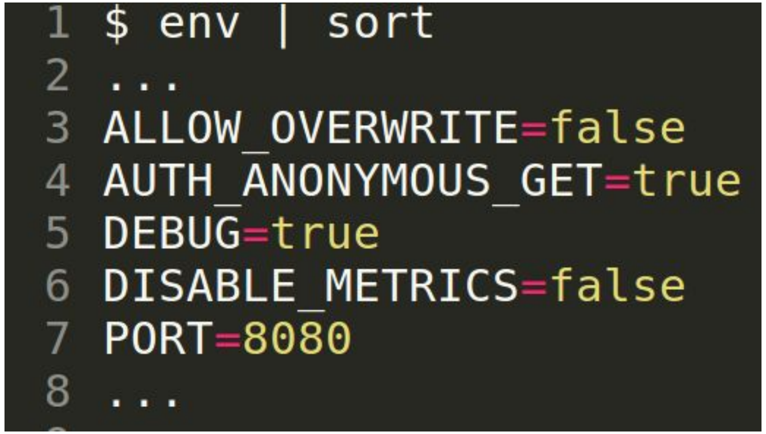
Ingress

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /helloworld
        backend:
          serviceName: my-service
          servicePort: 80
```

ConfigMap



```yaml
apiVersion: v1
data:
  ALLOW_OVERWRITE: false
  AUTH_ANONYMOUS_GET: true
  DEBUG: true
  DISABLE_METRICS: false
  PORT: 8080
metadata:
  name: plattform-stable-chartmuseum-ok-config
  namespace: plattform
```



```
$ env | sort
...
ALLOW_OVERWRITE=false
AUTH_ANONYMOUS_GET=true
DEBUG=true
DISABLE_METRICS=false
PORT=8080
...
```

Secret

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: YWRtaW4=
  password: MWYyZDFlMmU2N2Rm
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: mysecret
```

```
$ cat /etc/foo/mysecret/username

admin
```

PersistentVolume

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

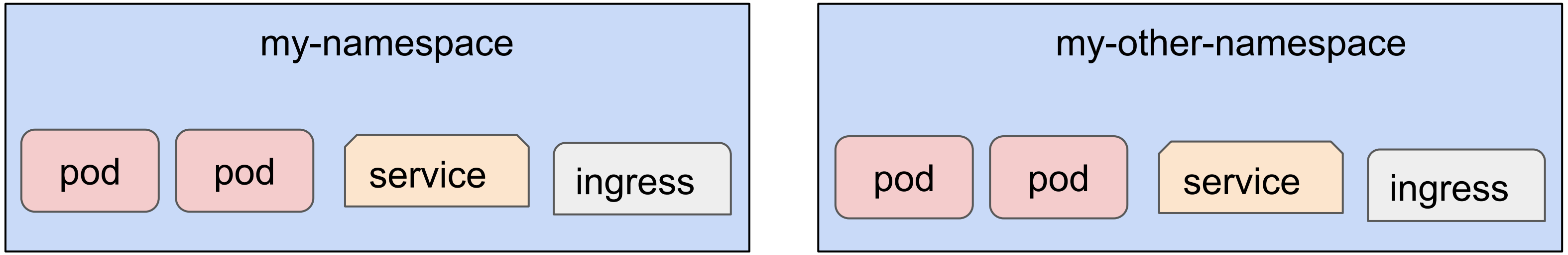
PersistenceVolumeClaim

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  selector:
    matchLabels:
      release: "stable"
```

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: dockerfile/nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: myvolume
  volumes:
    - name: myvolume
      persistentVolumeClaim:
        claimName: myclaim
```

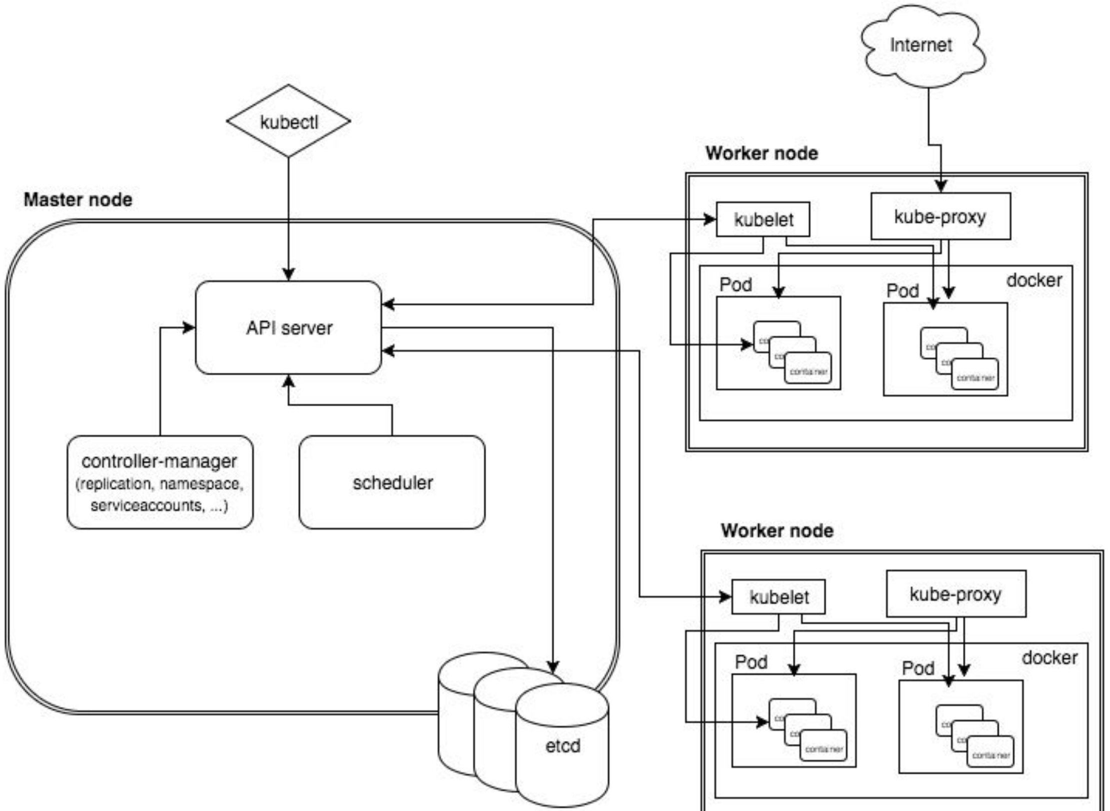
Namespace

En gruppe av ressurser (pod, service, etc)



Label

Kubernetes Architecture



- “Master”
  - Apiserver
  - controller-manager
  - scheduler
  - etcd
- Kubelet
  - Ensures that the pods have running containers
- Workers/Master
  - all other resources uses the apiserver to interact with each other