Deploying Rock Solid Applications with Kubernetes

@jelmersnoeck

Jelmer Snoeck

FIND ME

github.com/jelmersnoeck

twitter.com/jelmersnoeck

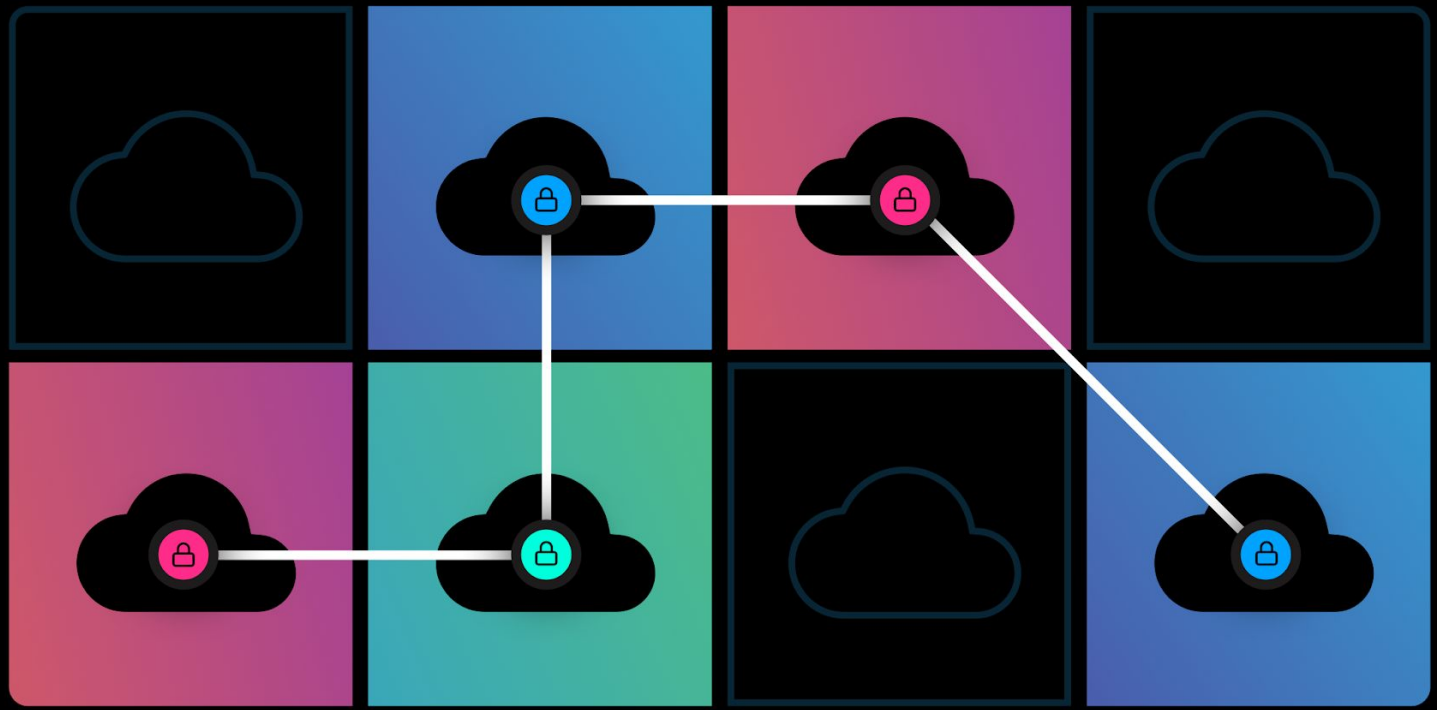
ABOUT ME

- Tech Lead at manifold.co
- <3 Golang
- <3 Kubernetes

manifold


What even are Rock Solid Applications?
@jelmersnoeck

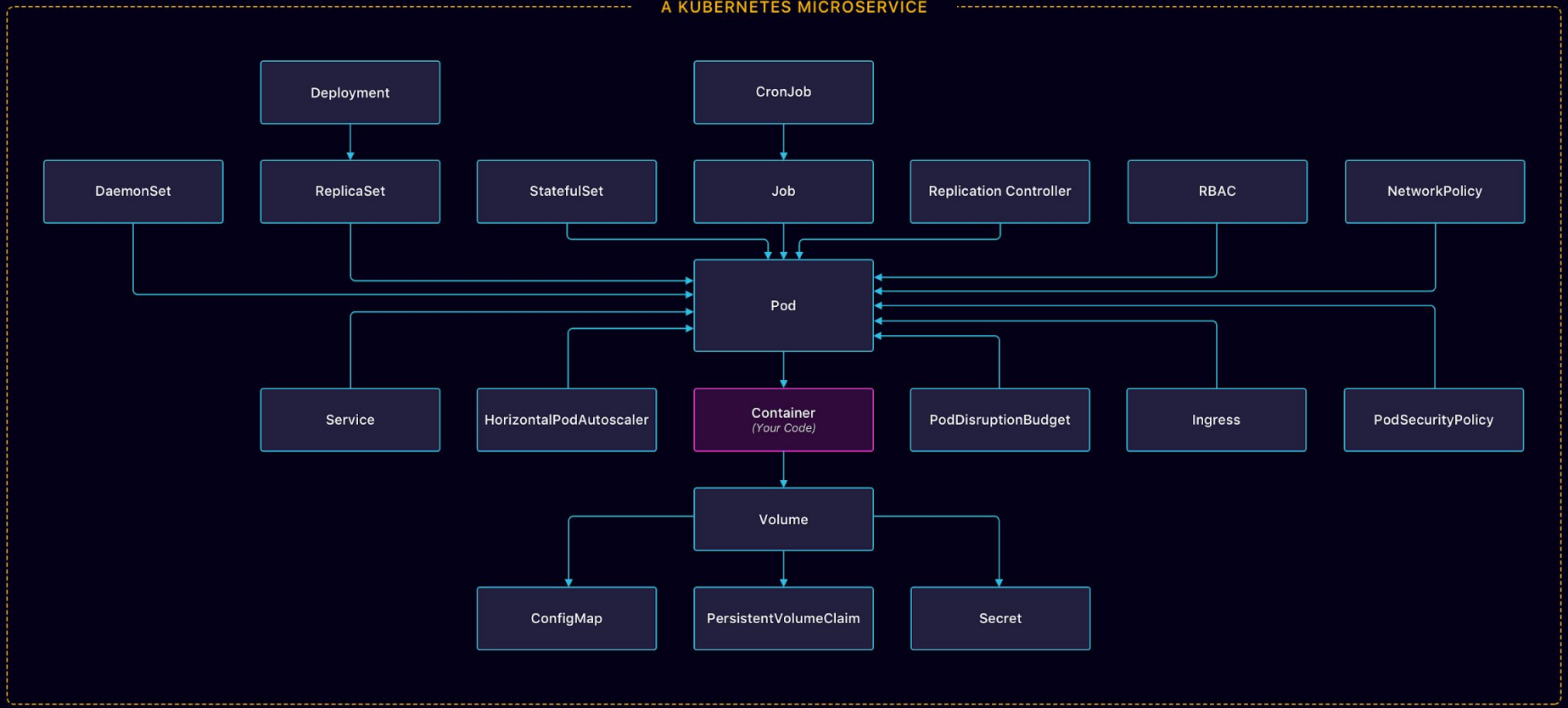
Secure Applications



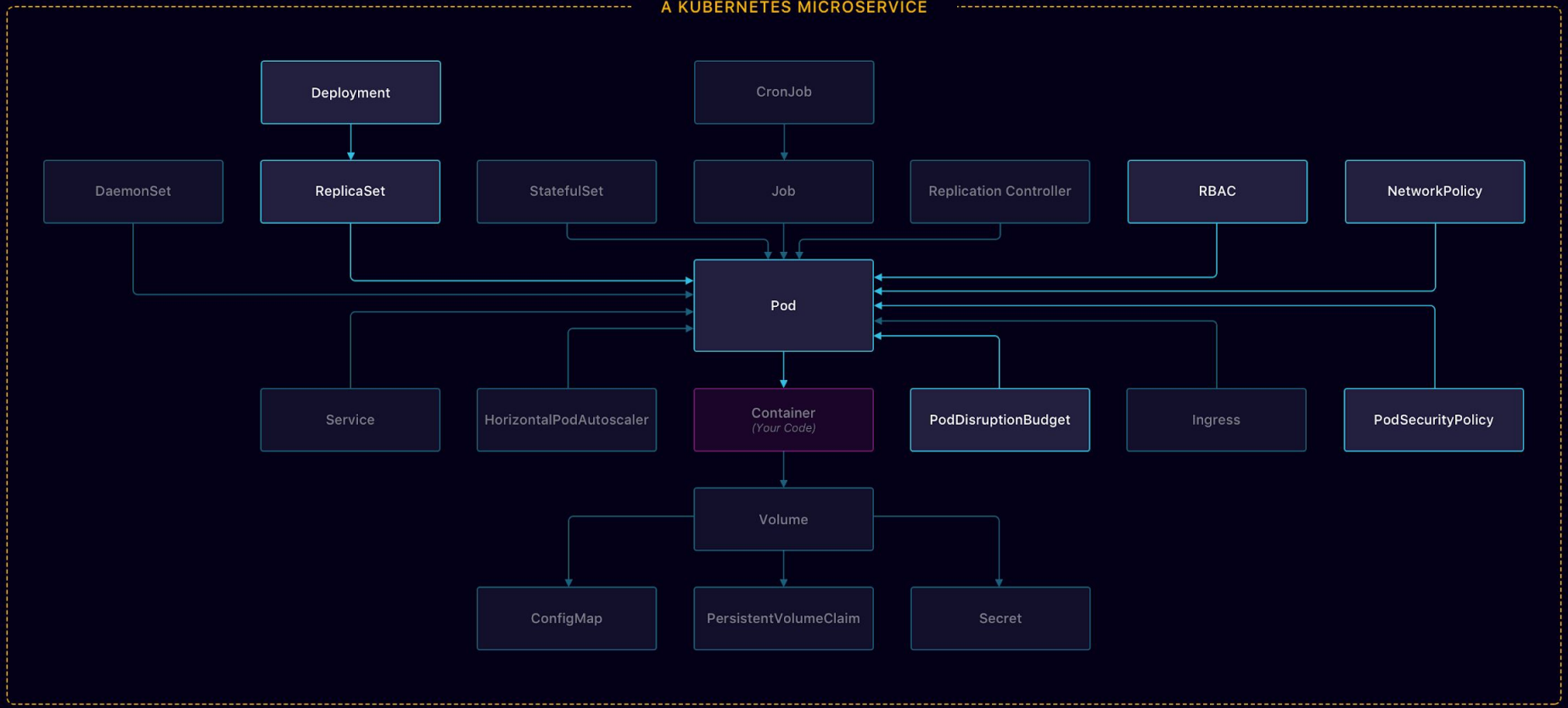

Highly Available Applications

@jelmersnoeck

Disclaimer: simplified YAML for demo purposes

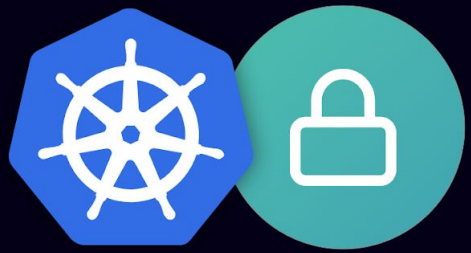


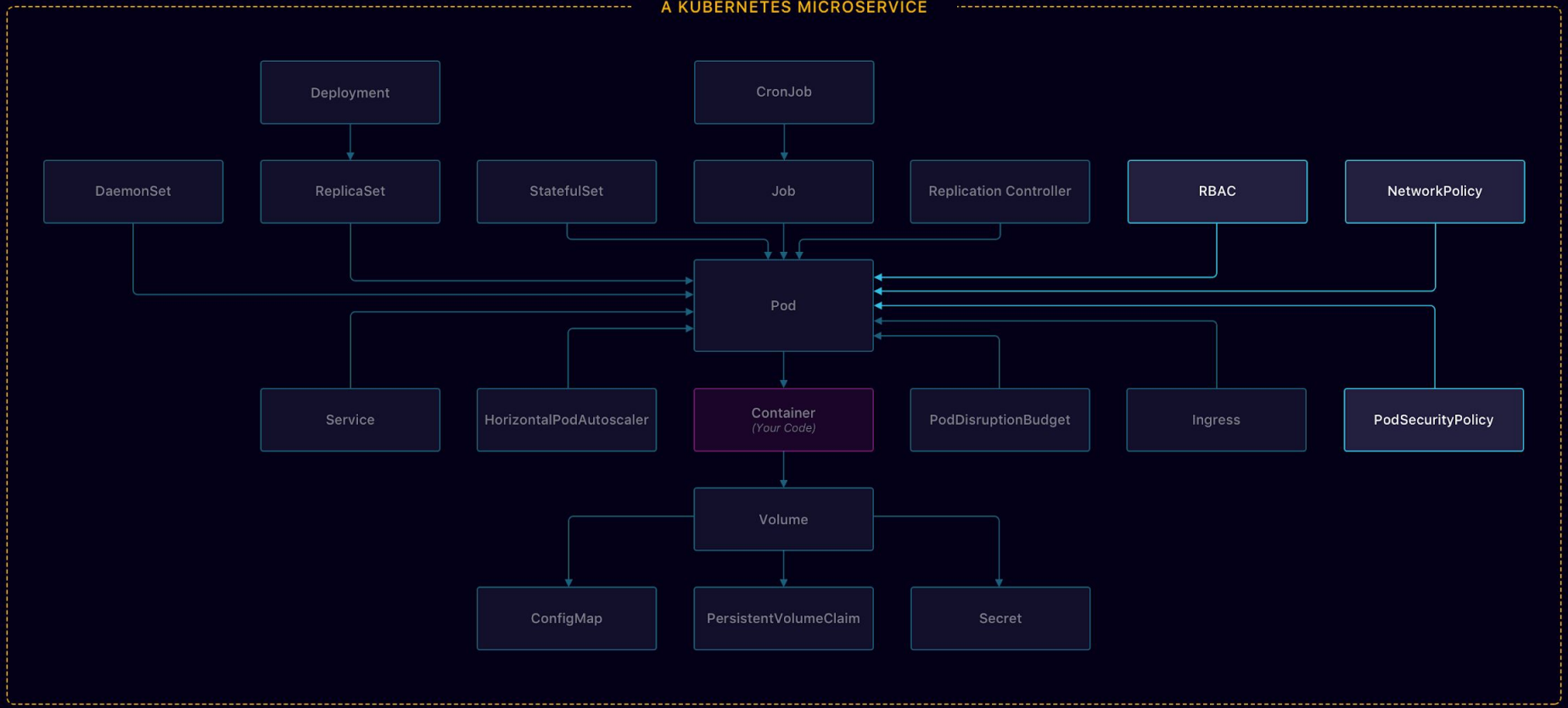
@jelmersnoeck



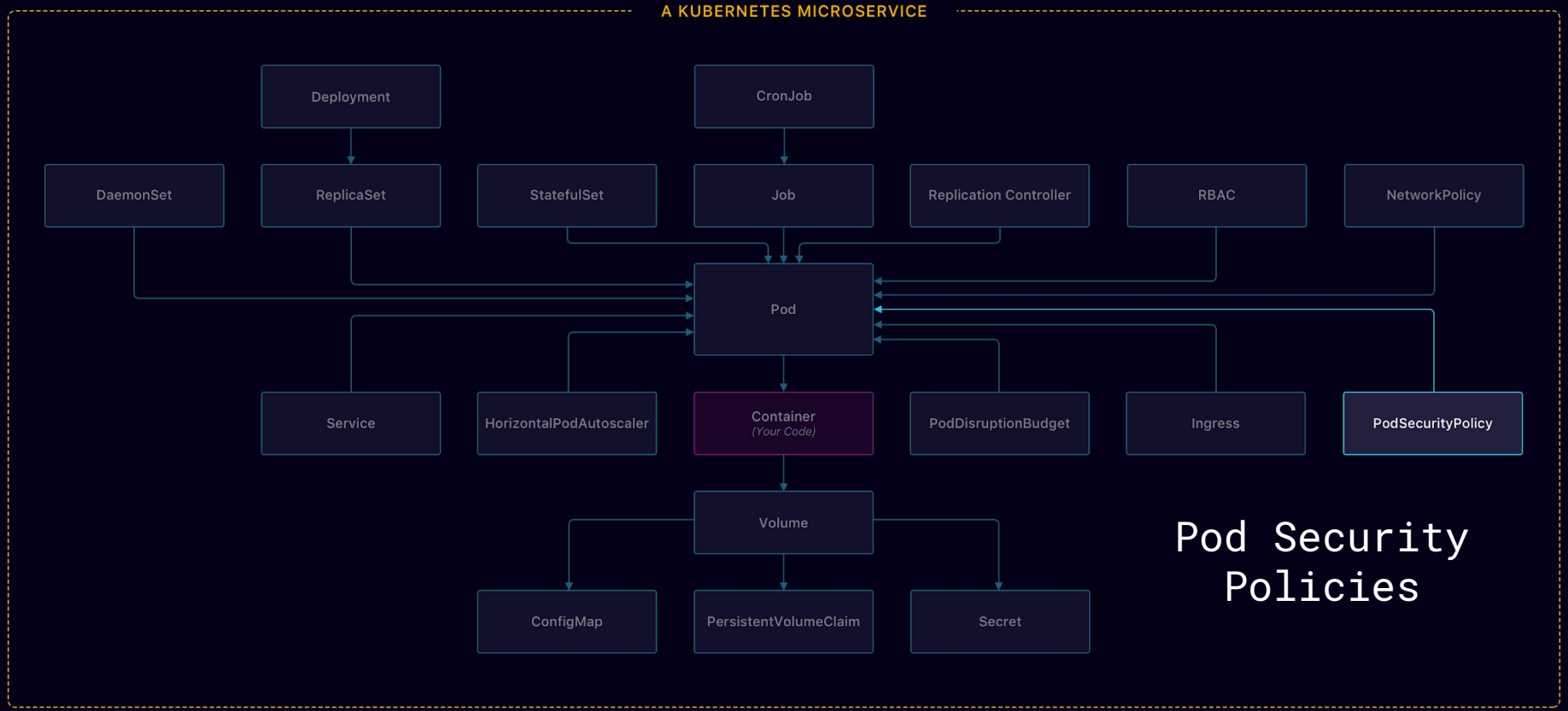
@jelmersnoeck

Security





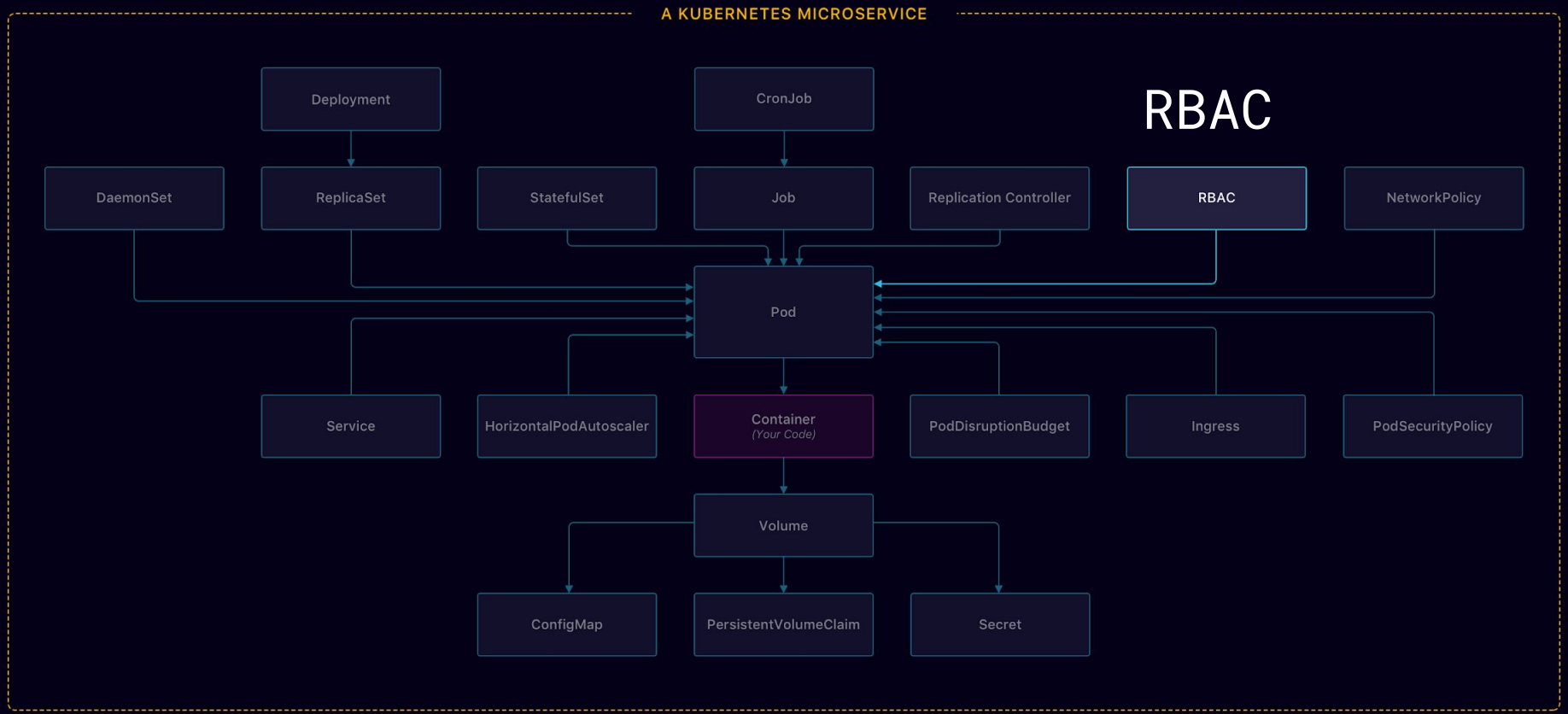
@jelmersnoeck

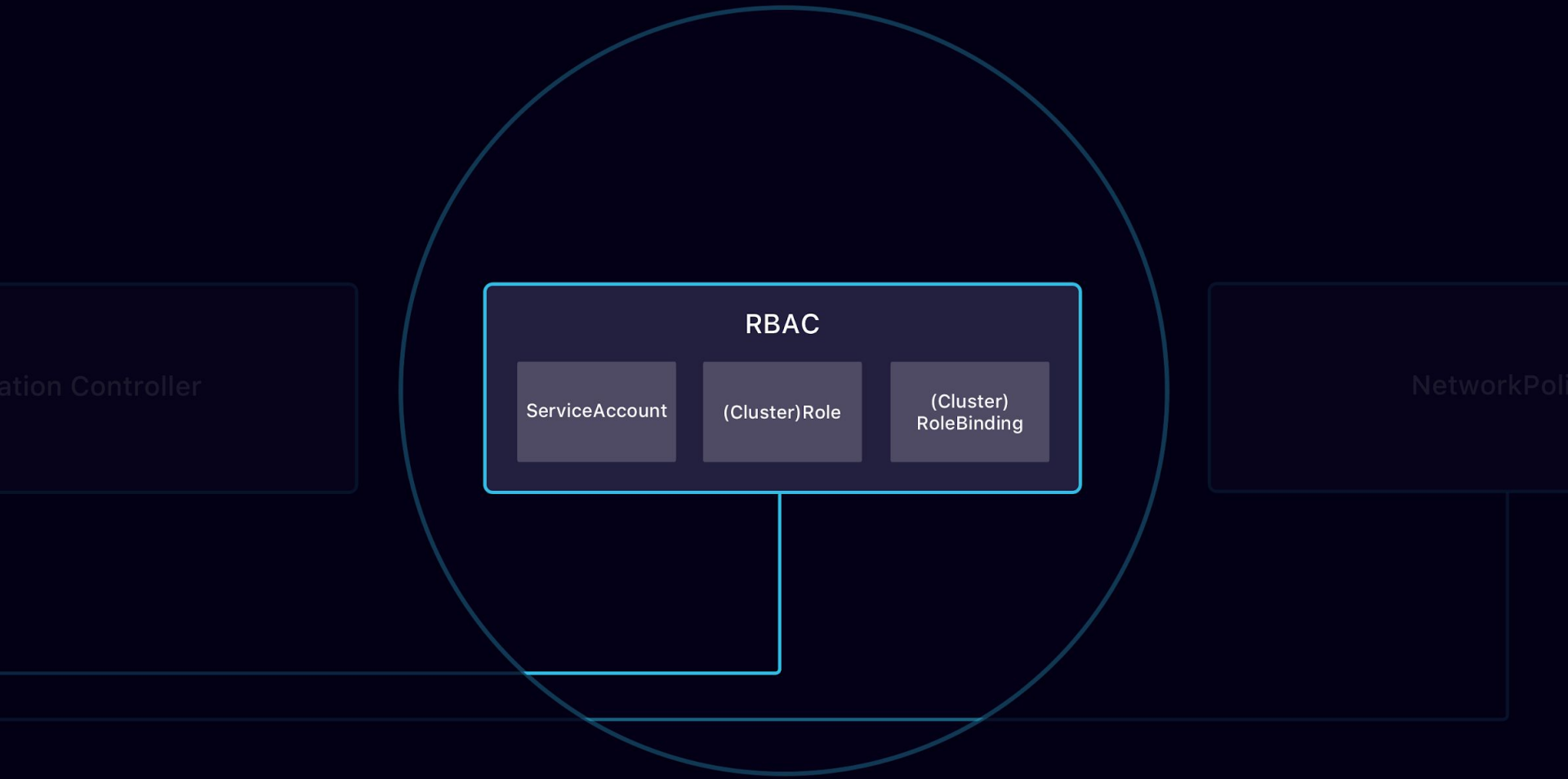


@jelmersnoeck

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: privileged
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
```

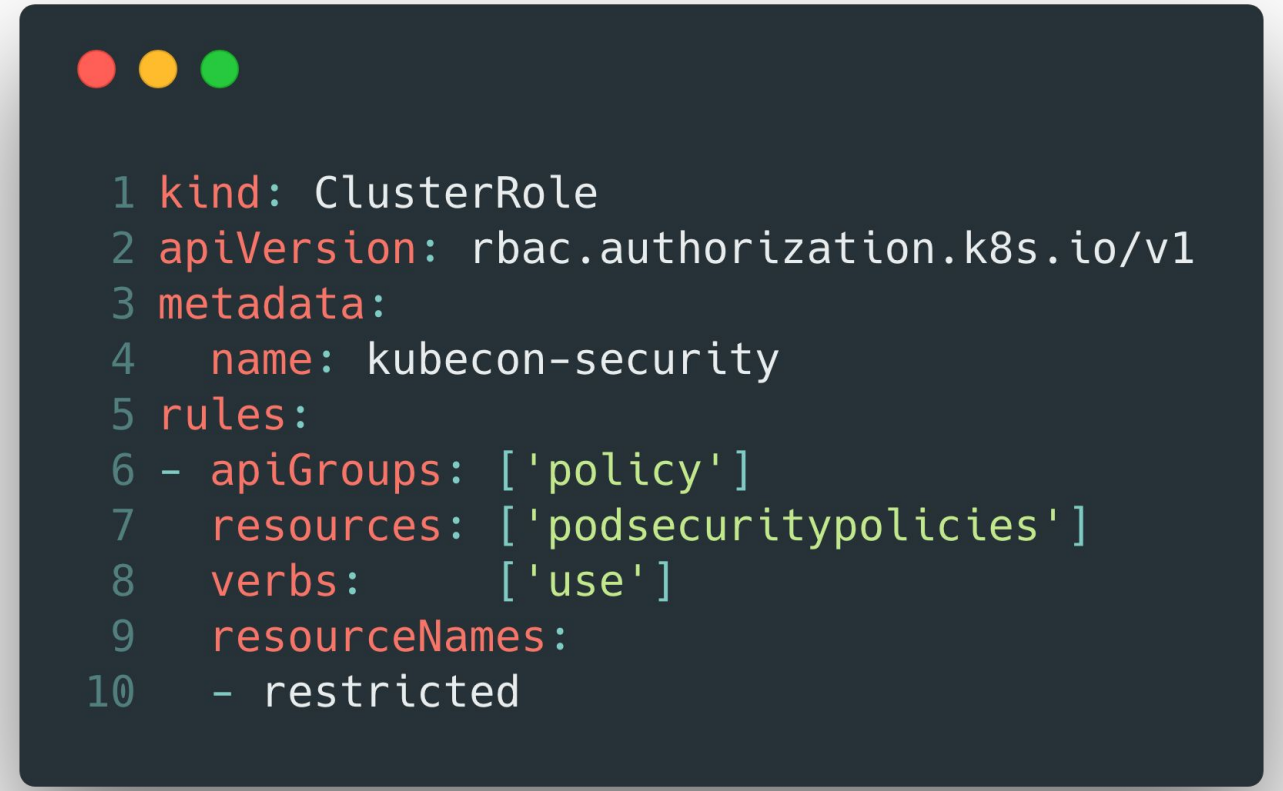
```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restricted
spec:
  privileged: false
  allowPrivilegeEscalation: false
  volumes:
  - 'secret'
  hostNetwork: false
  runAsUser:
    rule: 'MustRunAsNonRoot'
```


A KUBERNETES MICROSERVICE
Deployment
CronJob
RBAC
DaemonSet
ReplicaSet
StatefulSet
Job
Replication Controller
RBAC
NetworkPolicy
Pod
Service
HorizontalPodAutoscaler
Container
(Your Code)
PodDisruptionBudget
Ingress
PodSecurityPolicy
Volume
ConfigMap
PersistentVolumeClaim
Secret

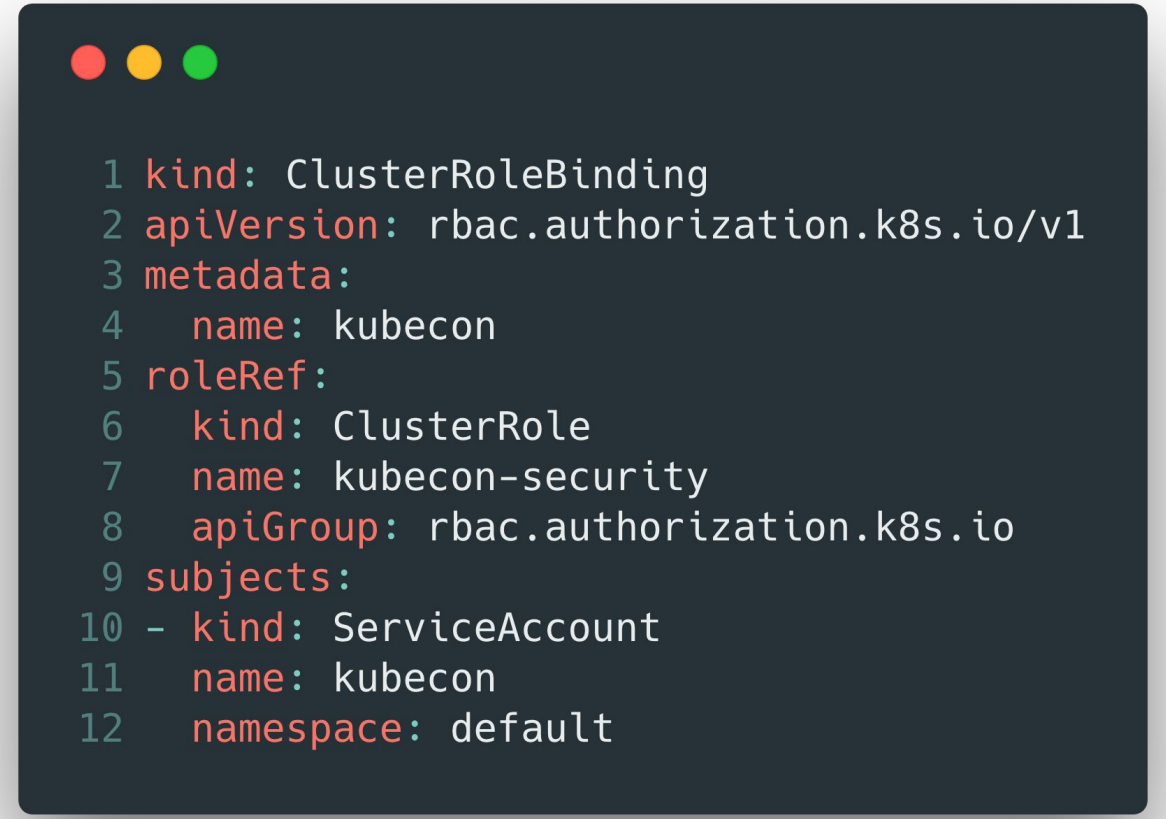

ation Controller
RBAC
ServiceAccount
(Cluster)Role
(Cluster)
RoleBinding
NetworkPol

manifold

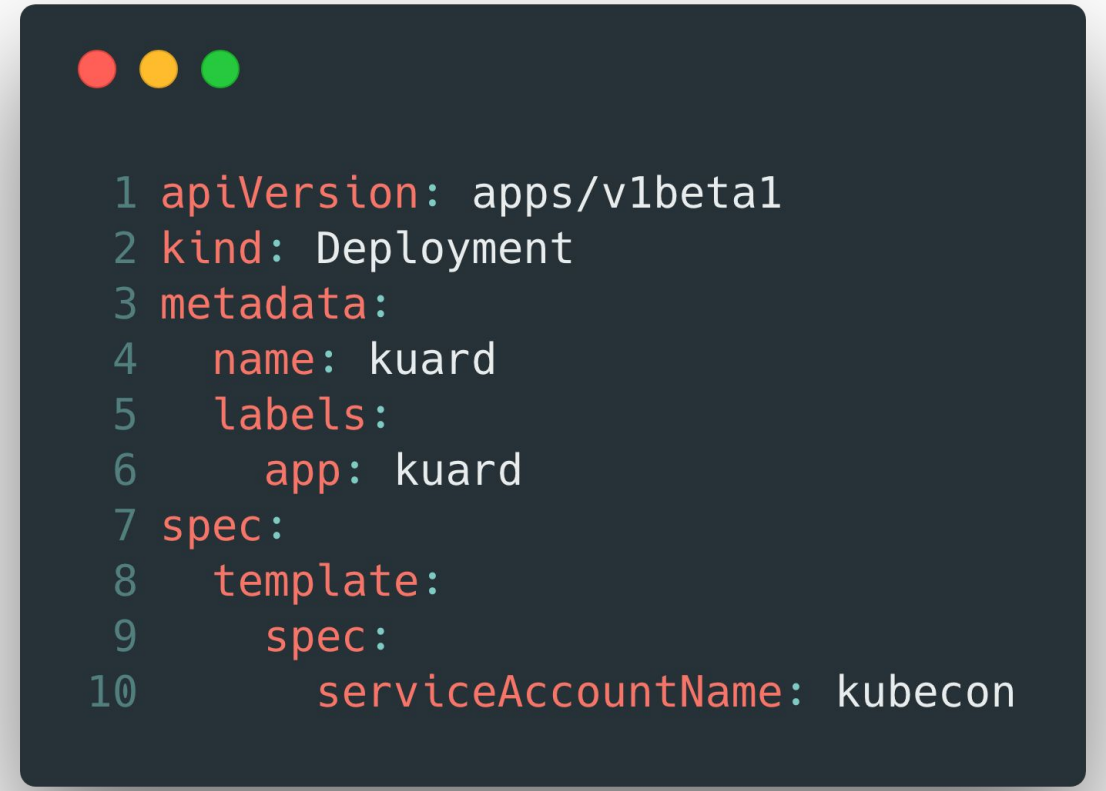
@jelmersnoeck



```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: kubecon-security
rules:
- apiGroups: ['policy']
  resources: ['podsecuritypolicies']
  verbs:     ['use']
  resourceNames:
  - restricted
```



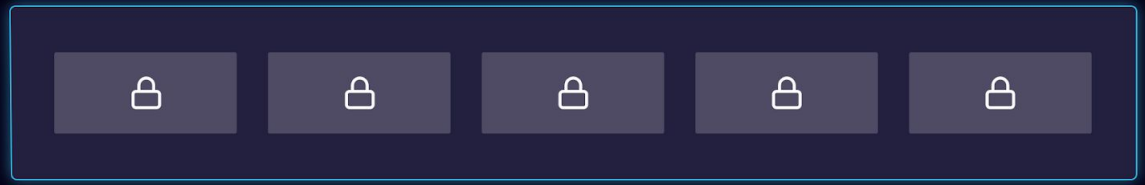
```yaml
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: kubecon
roleRef:
  kind: ClusterRole
  name: kubecon-security
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name: kubecon
  namespace: default
```



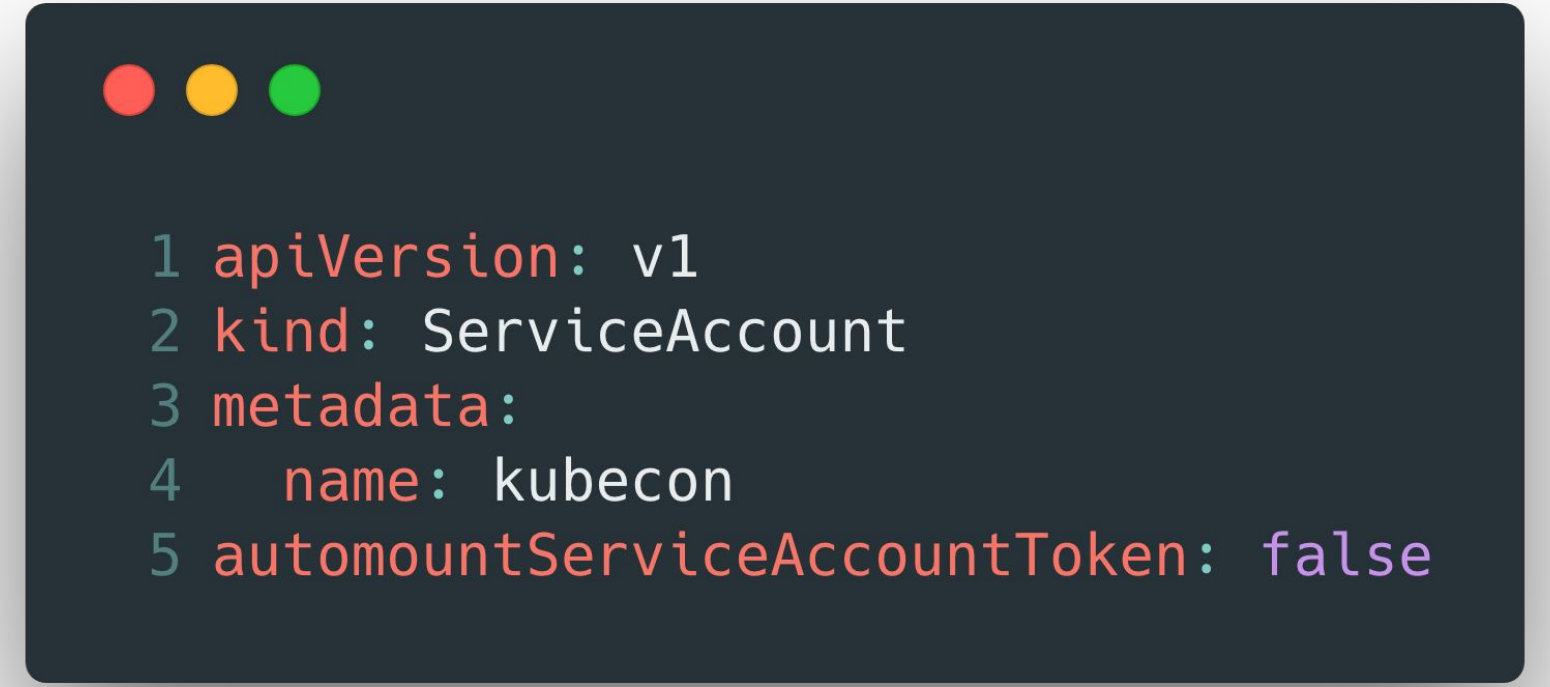
```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: kuard
  labels:
    app: kuard
spec:
  template:
    spec:
      serviceAccountName: kubecon
```

manifold

@jelmersnoeck



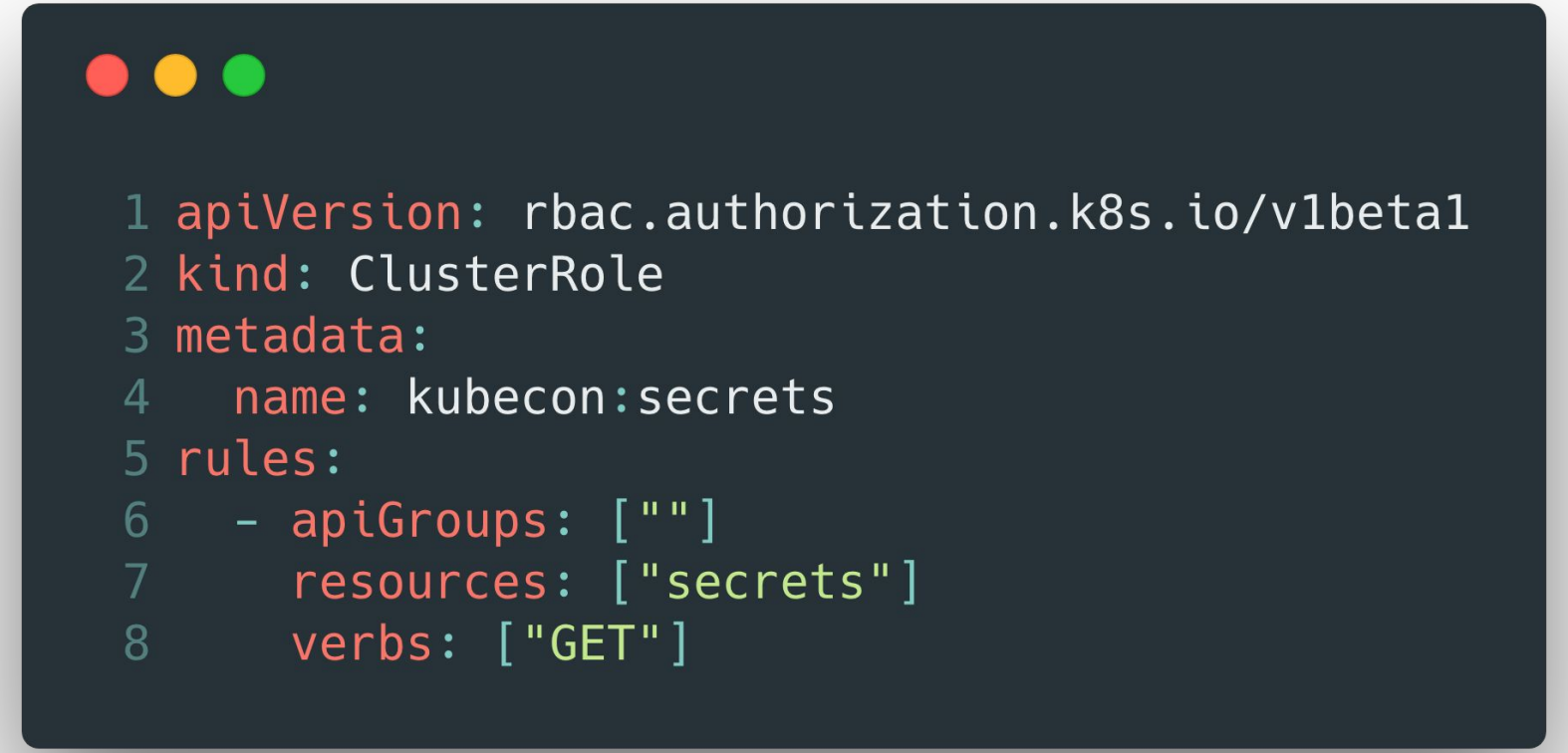
@jelmersnoeck



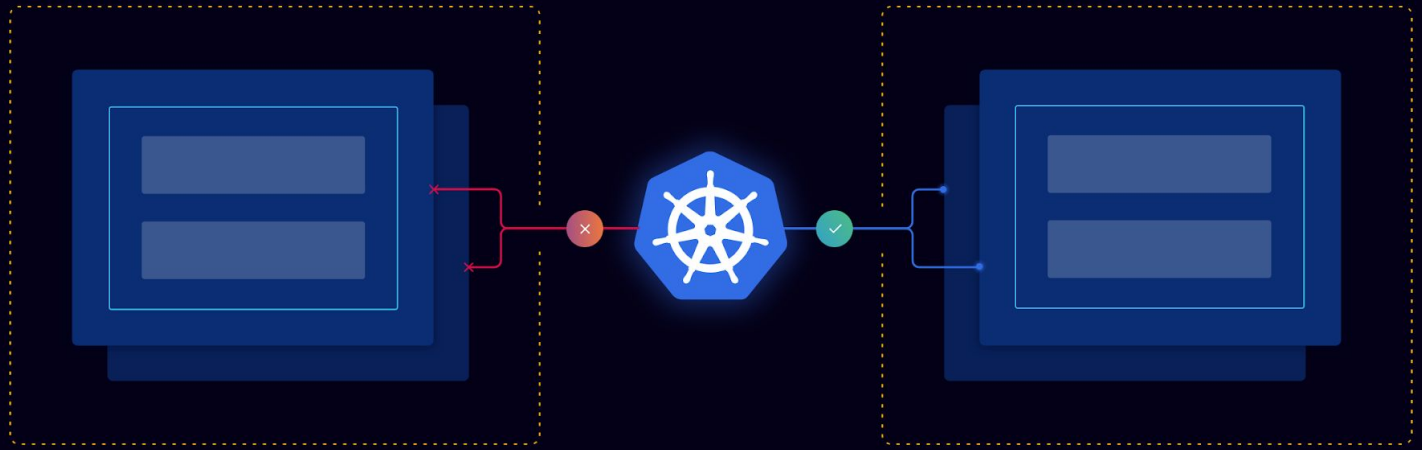
manifold

@jelmersnoeck

or...

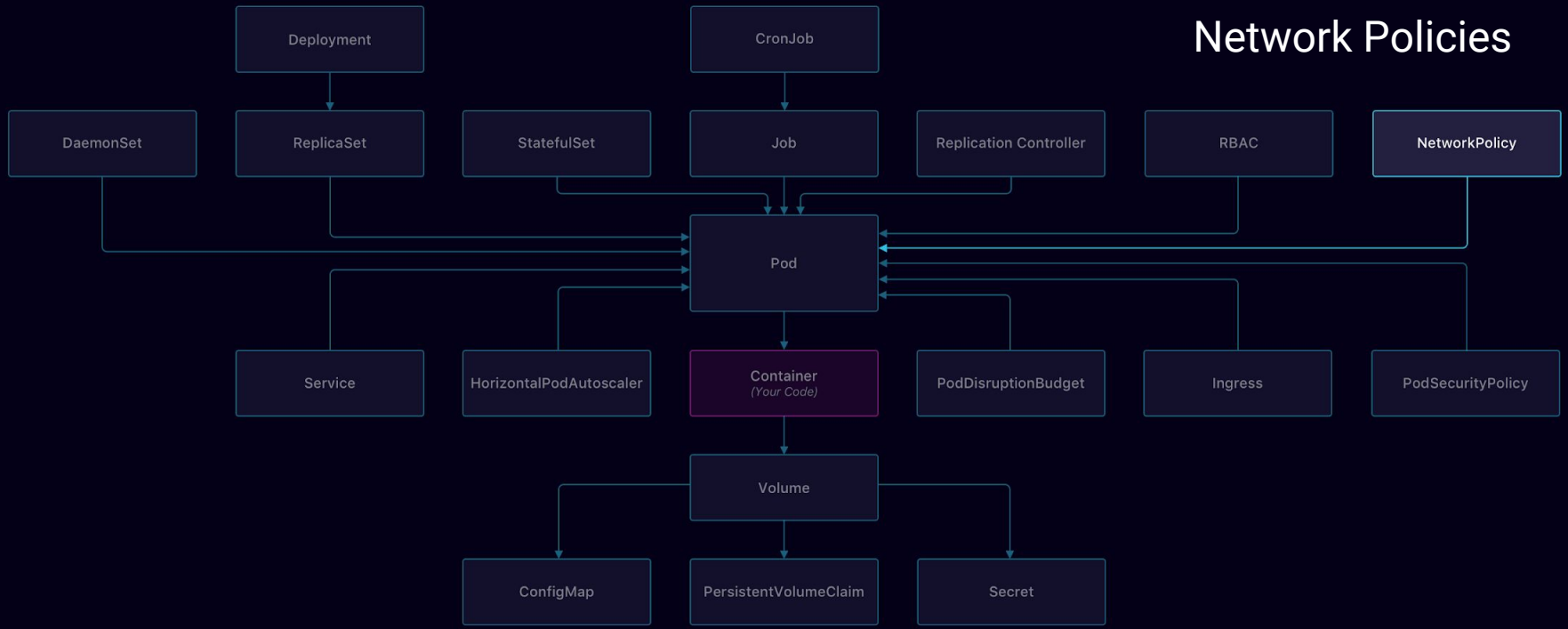


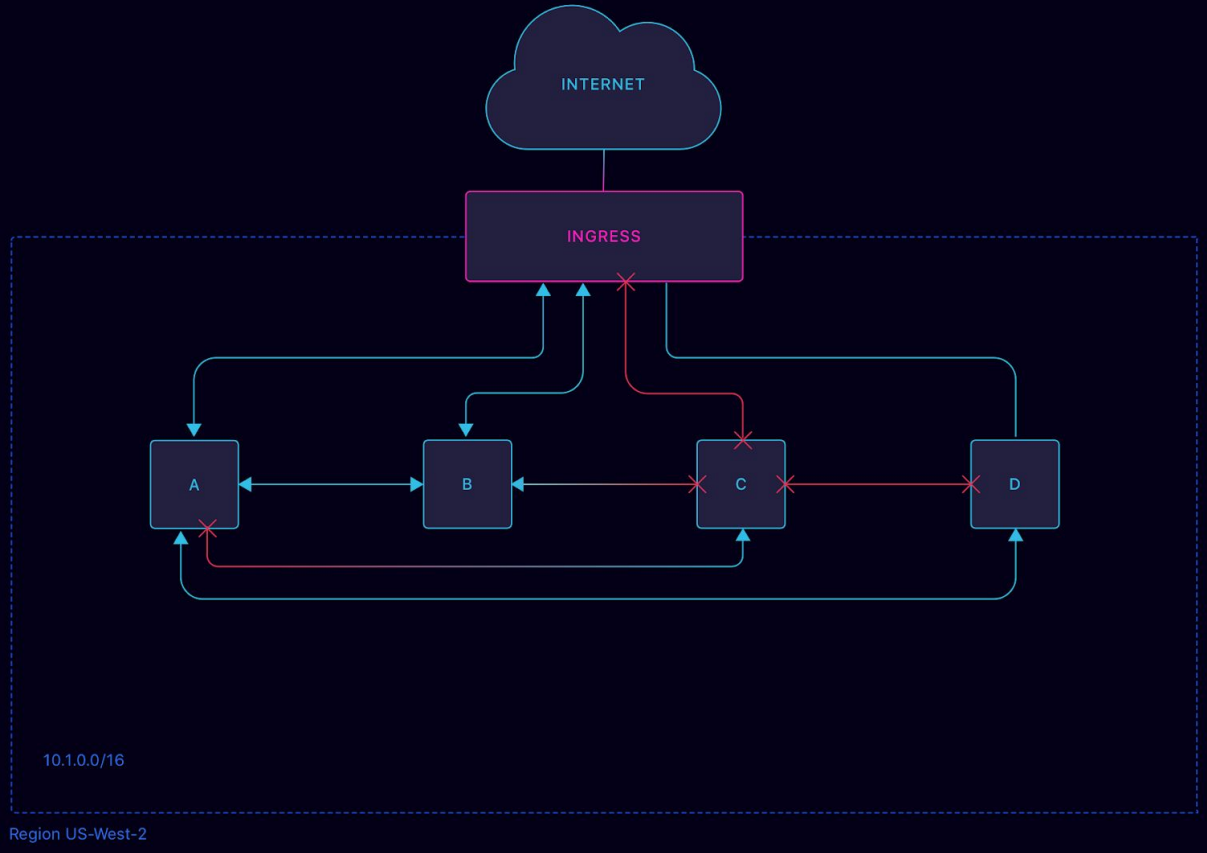
```yaml
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: kubecon:secrets
rules:
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["GET"]
```

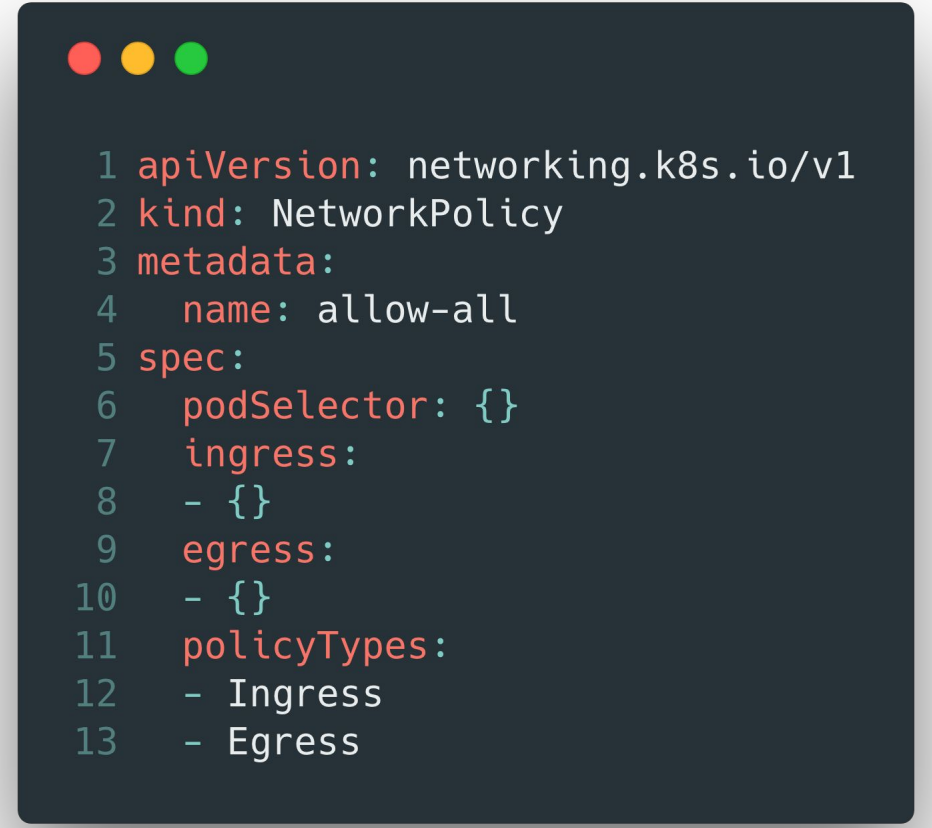


A KUBERNETES MICROSERVICE

Network Policies



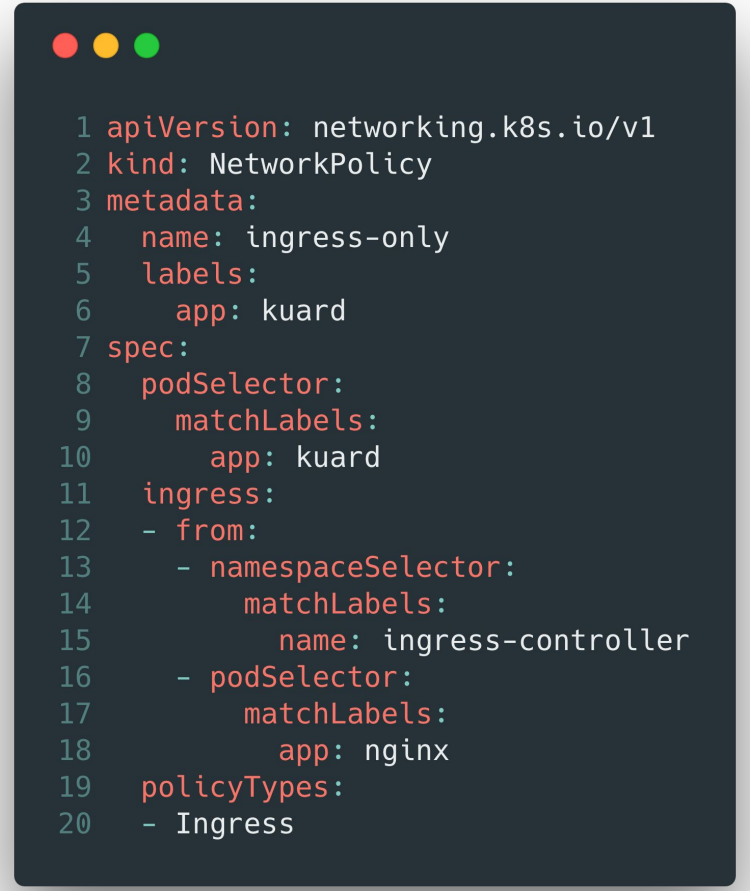

INTERNET
INGRESS
A
B
C
D
10.1.0.0/16
Region US-West-2



```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all
spec:
  podSelector: {}
  ingress:
  - {}
  egress:
  - {}
  policyTypes:
  - Ingress
  - Egress
```

manifold

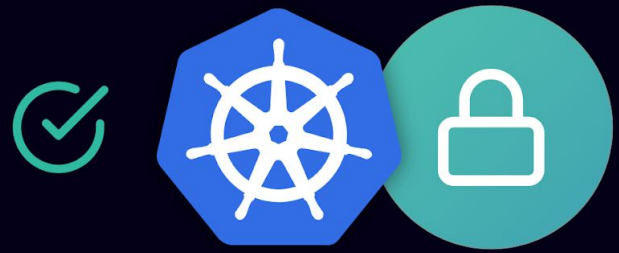
@jelmersnoeck



```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ingress-only
  labels:
    app: kuard
spec:
  podSelector:
    matchLabels:
      app: kuard
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          name: ingress-controller
    - podSelector:
        matchLabels:
          app: nginx
  policyTypes:
  - Ingress
```

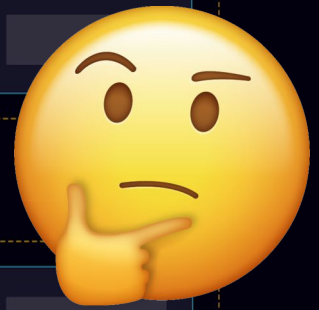
Caveat: availability depends on your networking plugin

Security

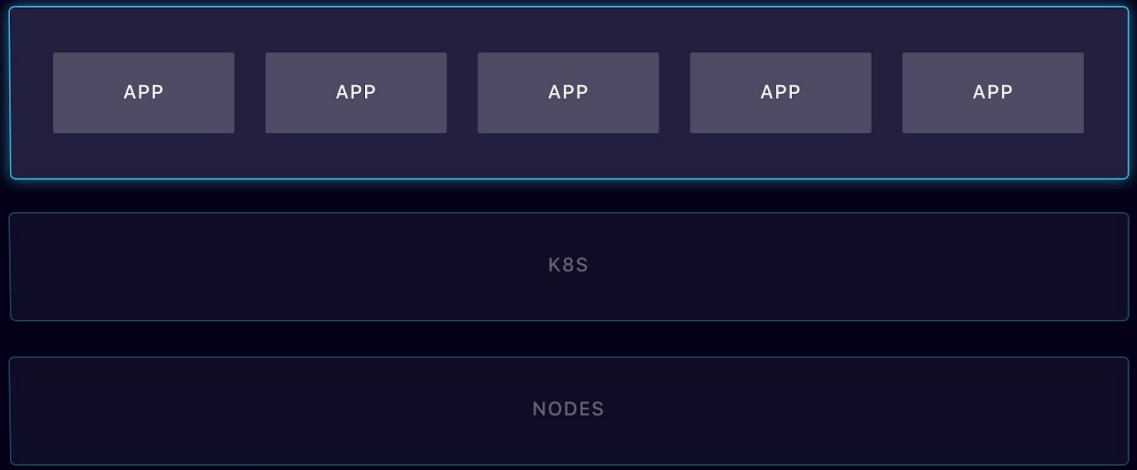


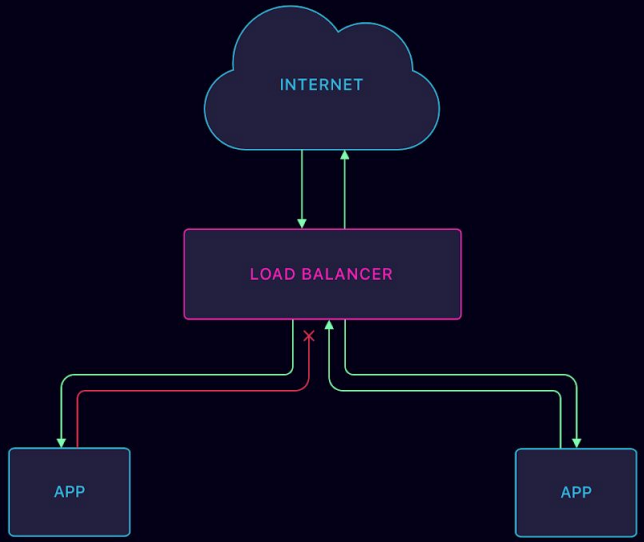

High Availability

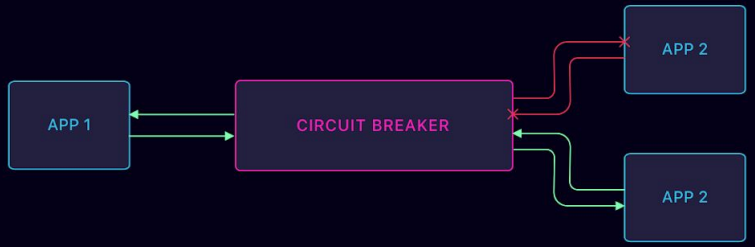
What even is High Availability?

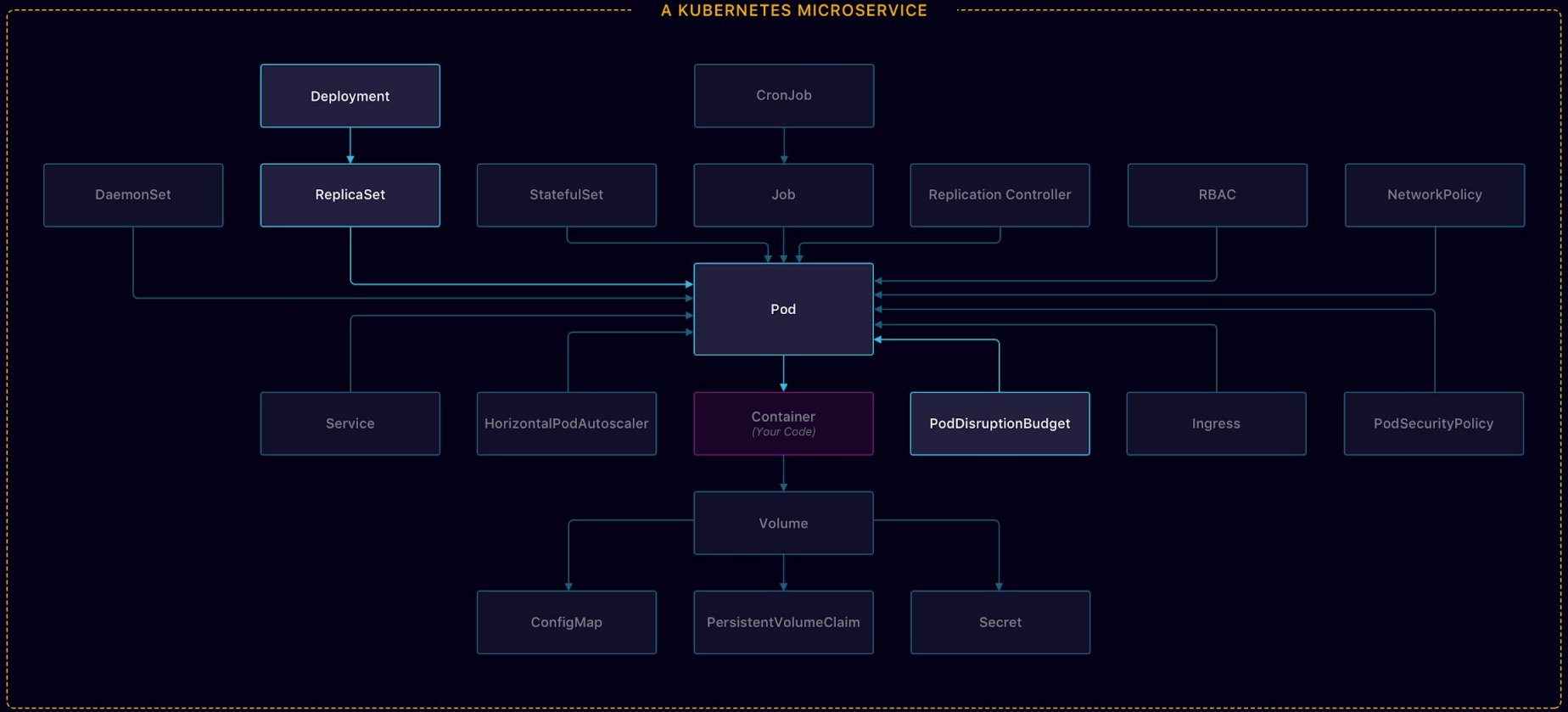


This talk is not about High Availability for nodes

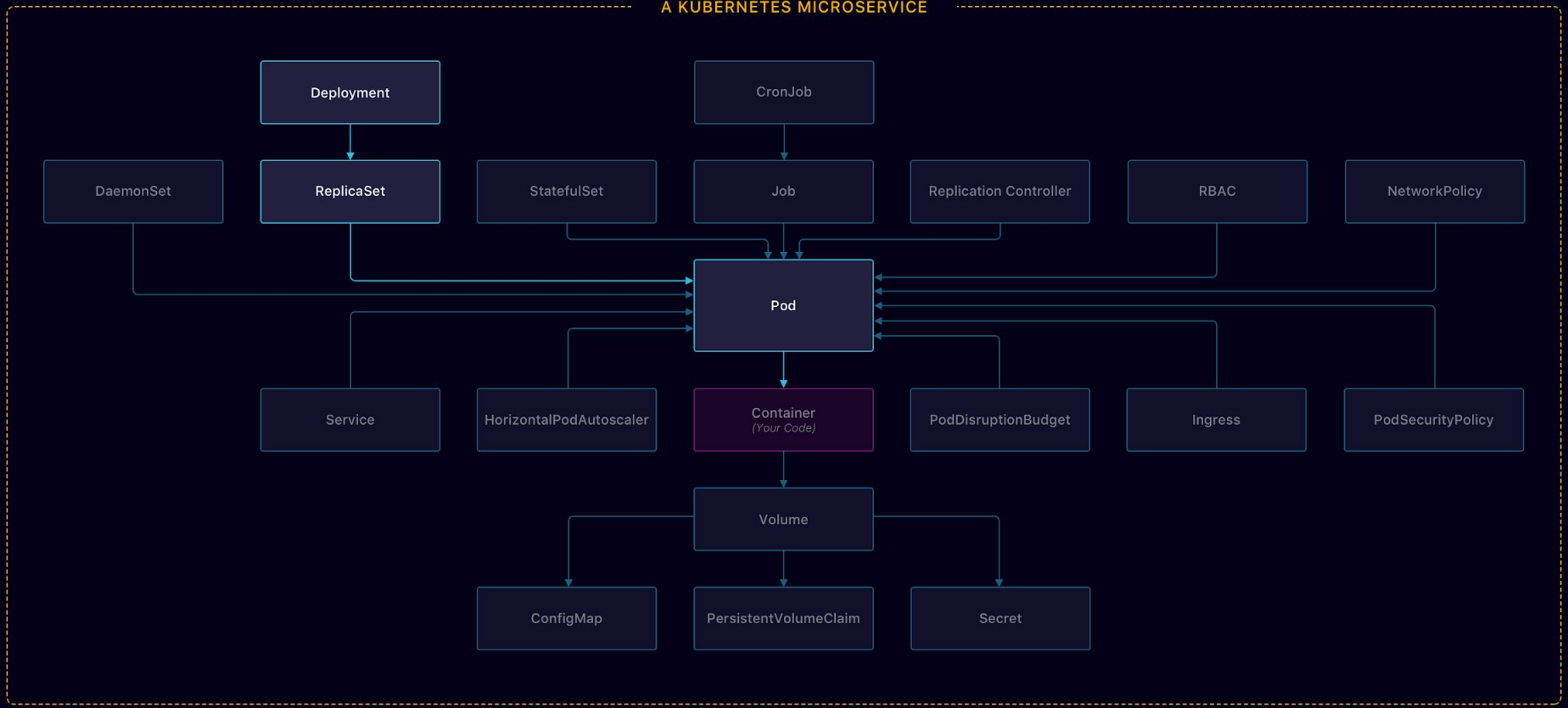



INTERNET
LOAD BALANCER
APP
APP


APP 1
CIRCUIT BREAKER
APP 2
APP 2

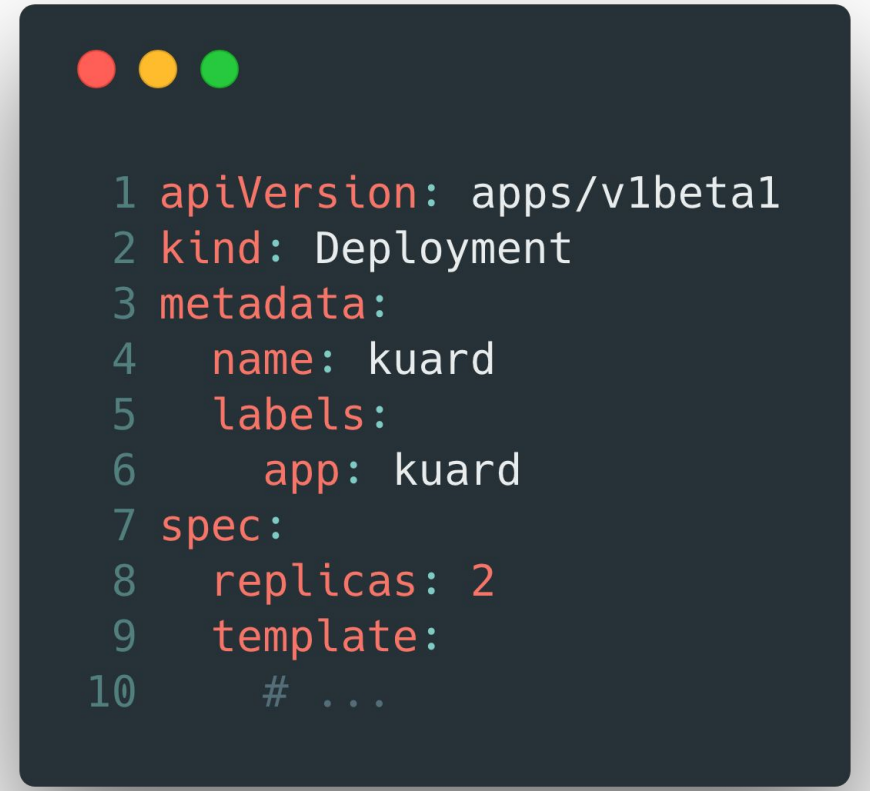


@jelmersnoeck

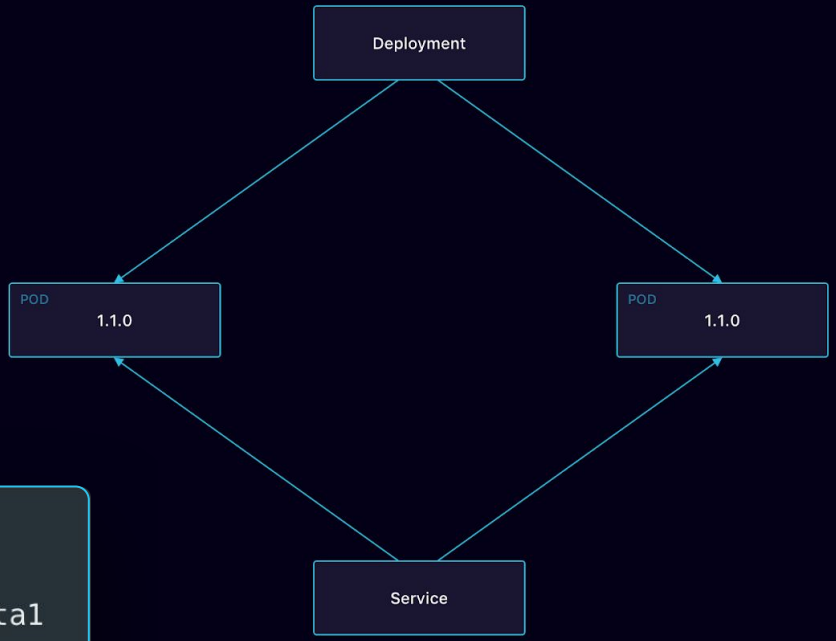


@jelmersnoeck

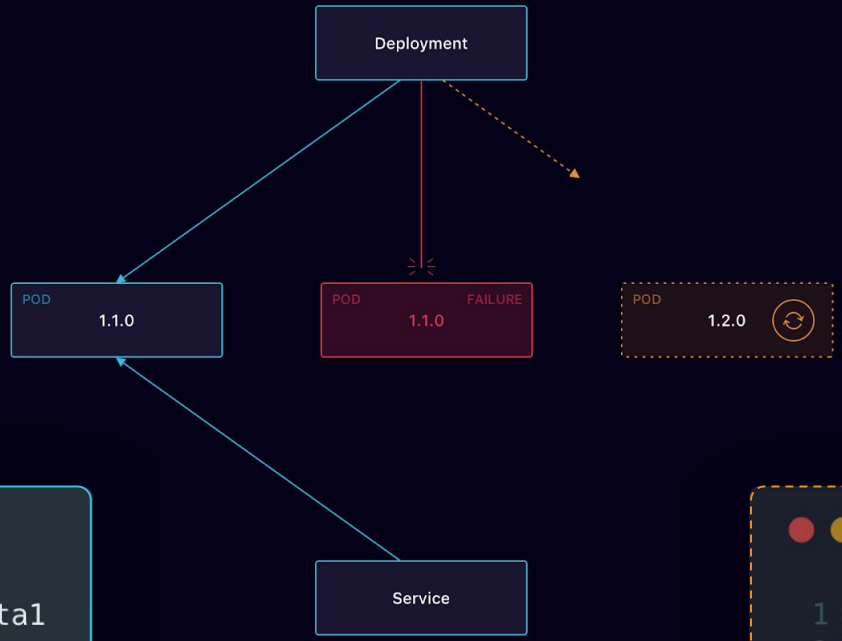
Replicas + UpdateStrategy



```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: kuard
  labels:
    app: kuard
spec:
  replicas: 2
  template:
    # ...
```




```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.1.0
```




```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.1.0
```

```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.2.0
```

@jelmersnoeck


```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.1.0
```

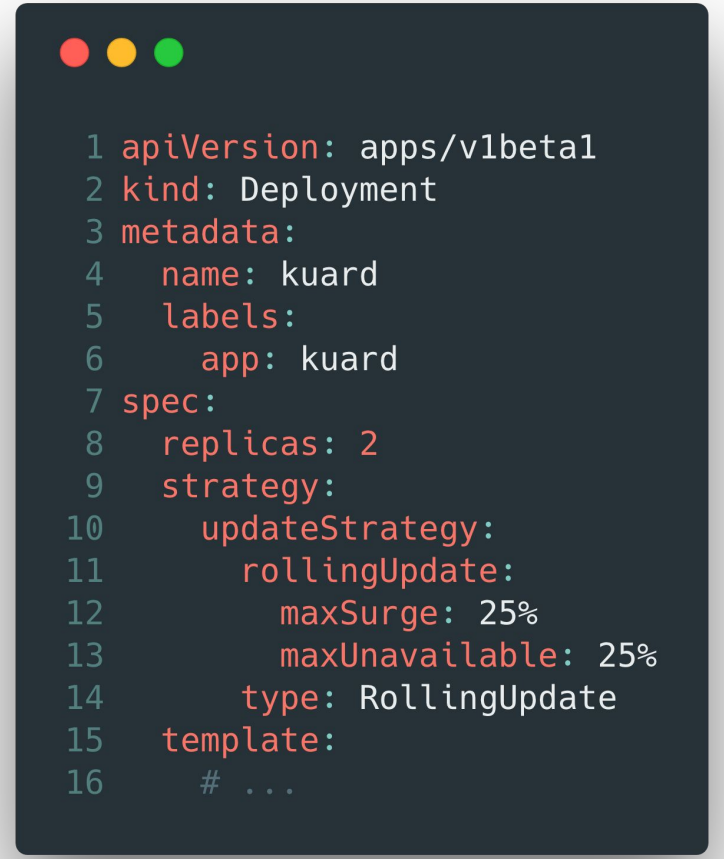
```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.2.0
```

@jelmersnoeck

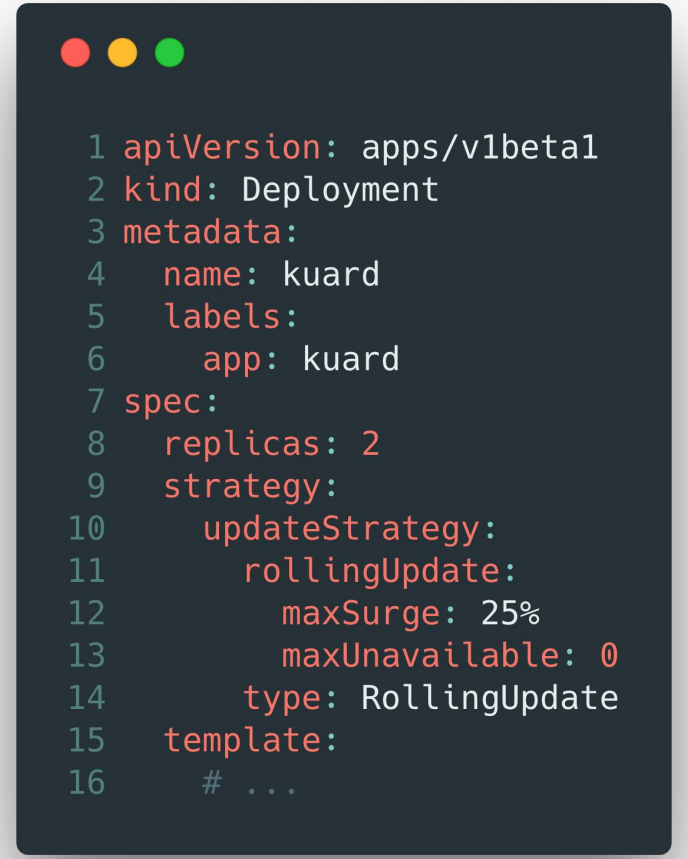

```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.1.0
```

```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.2.0
```

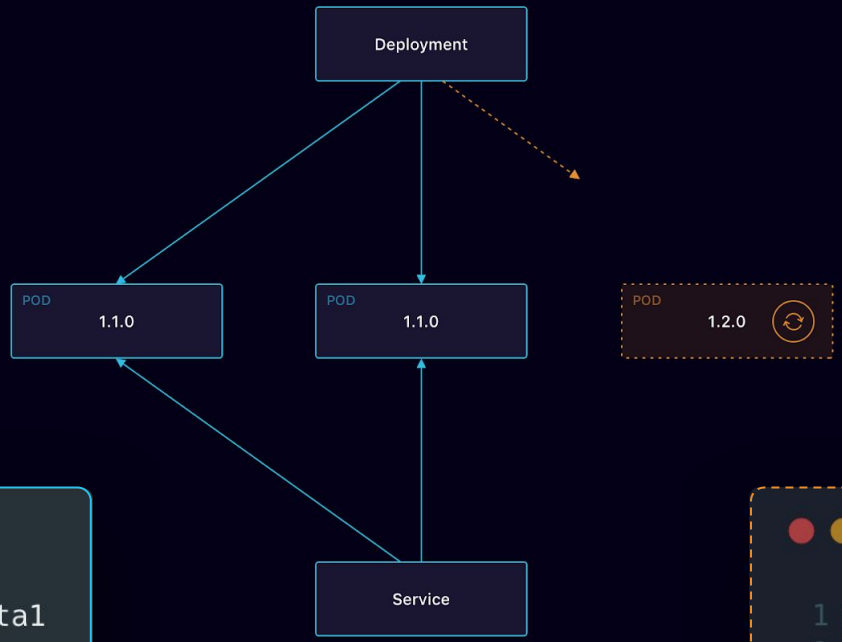
@jelmersnoeck



```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: kuard
  labels:
    app: kuard
spec:
  replicas: 2
  strategy:
    updateStrategy:
      rollingUpdate:
        maxSurge: 25%
        maxUnavailable: 25%
      type: RollingUpdate
  template:
    # ...
```



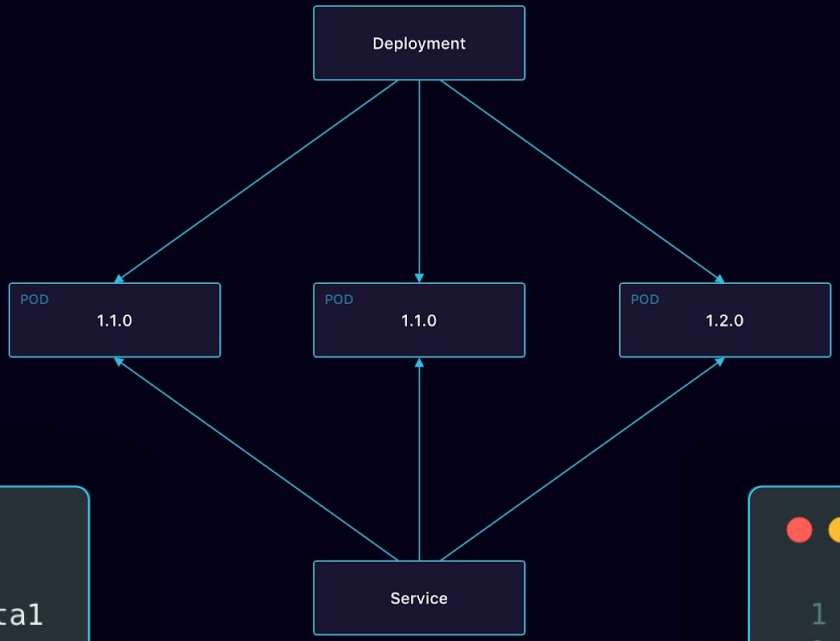
```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: kuard
  labels:
    app: kuard
spec:
  replicas: 2
  strategy:
    updateStrategy:
      rollingUpdate:
        maxSurge: 25%
        maxUnavailable: 0
      type: RollingUpdate
  template:
    # ...
```




```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.1.0
```

```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.2.0
```

@jelmersnoeck



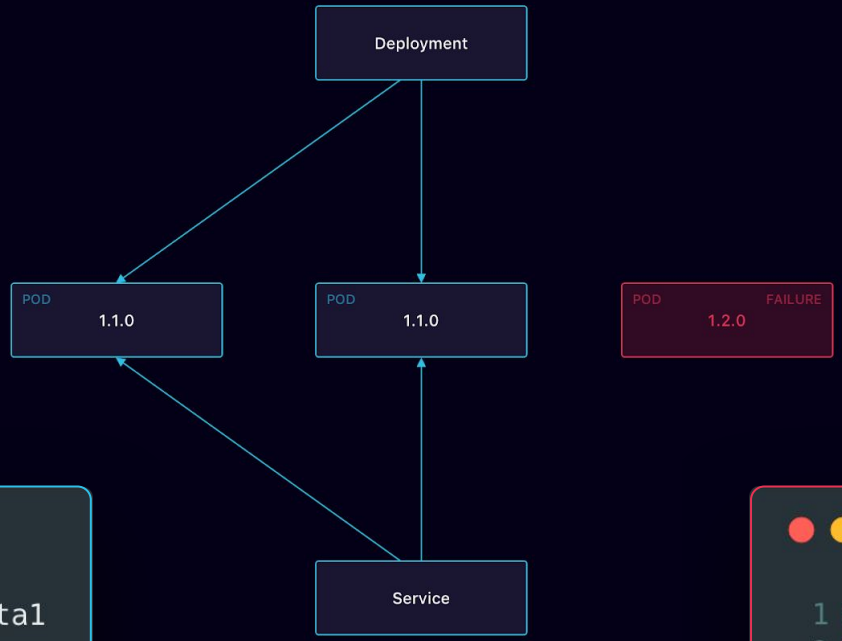

```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.1.0
```

```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.2.0
```


```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.1.0
```

```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.2.0
```

@jelmersnoeck

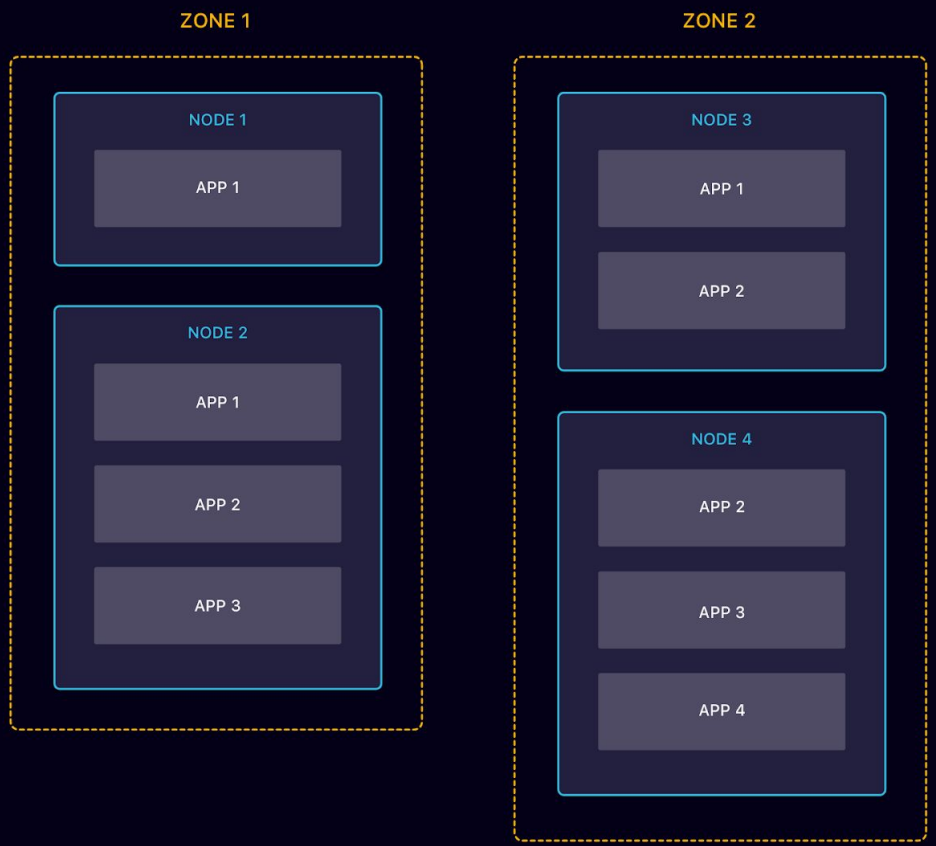


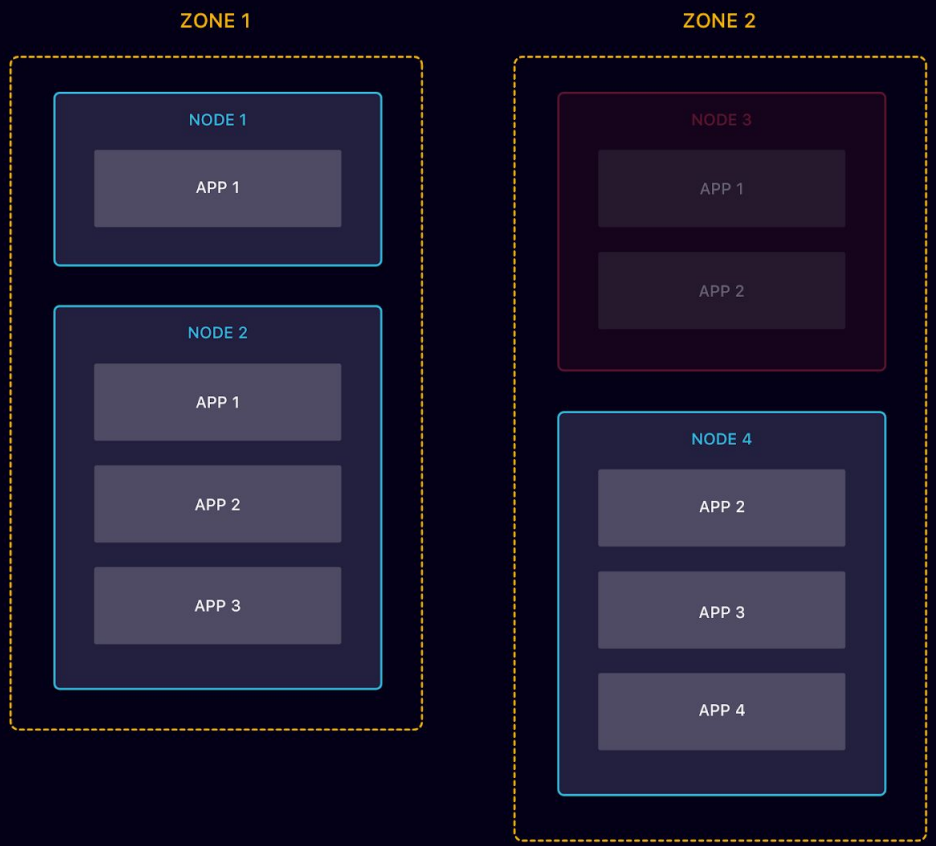

```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.1.0
```

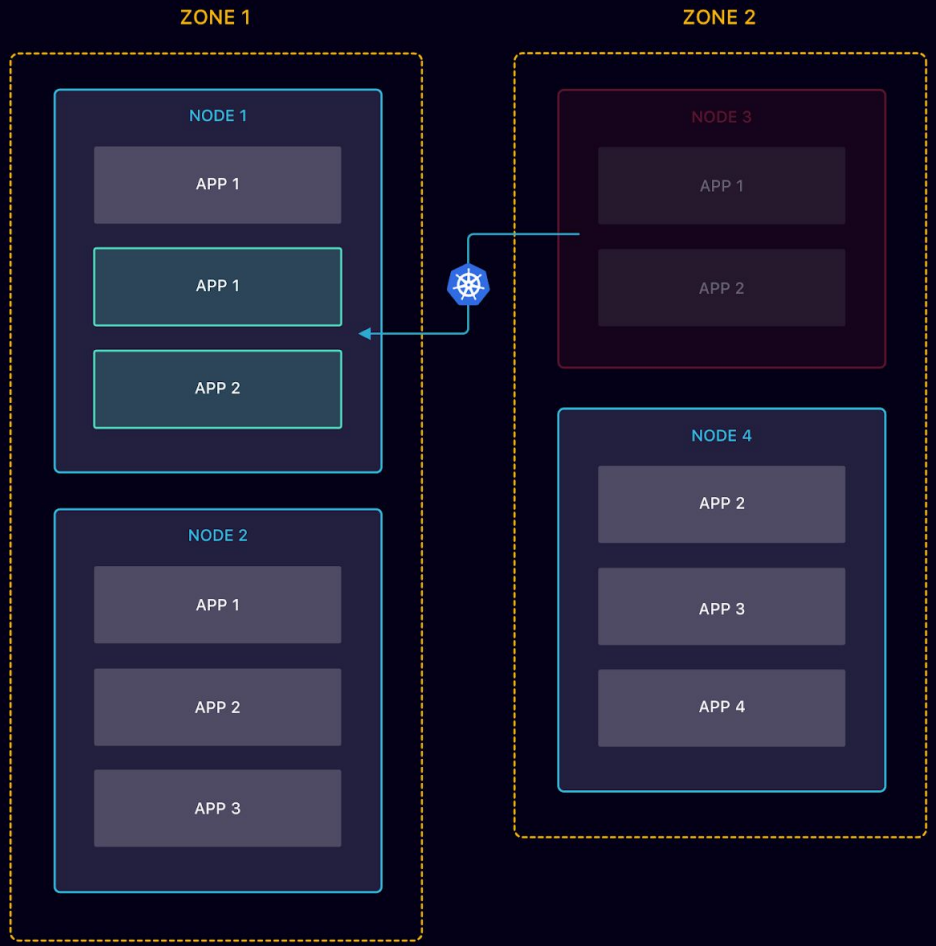
```yaml
apiVersion: apps/v1beta1
kind: Deployment
# ...
  image: kubecon:1.2.0
```

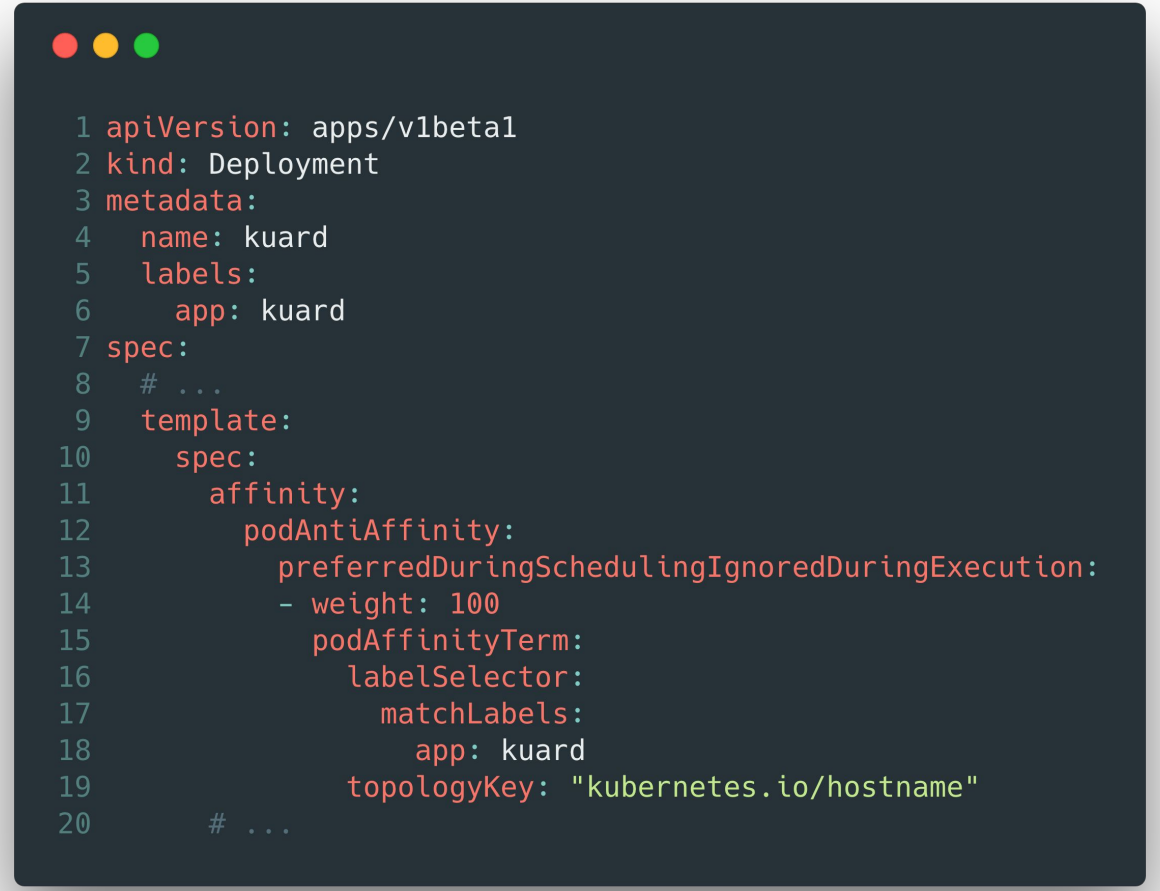
@jelmersnoeck

(Anti)Affinity

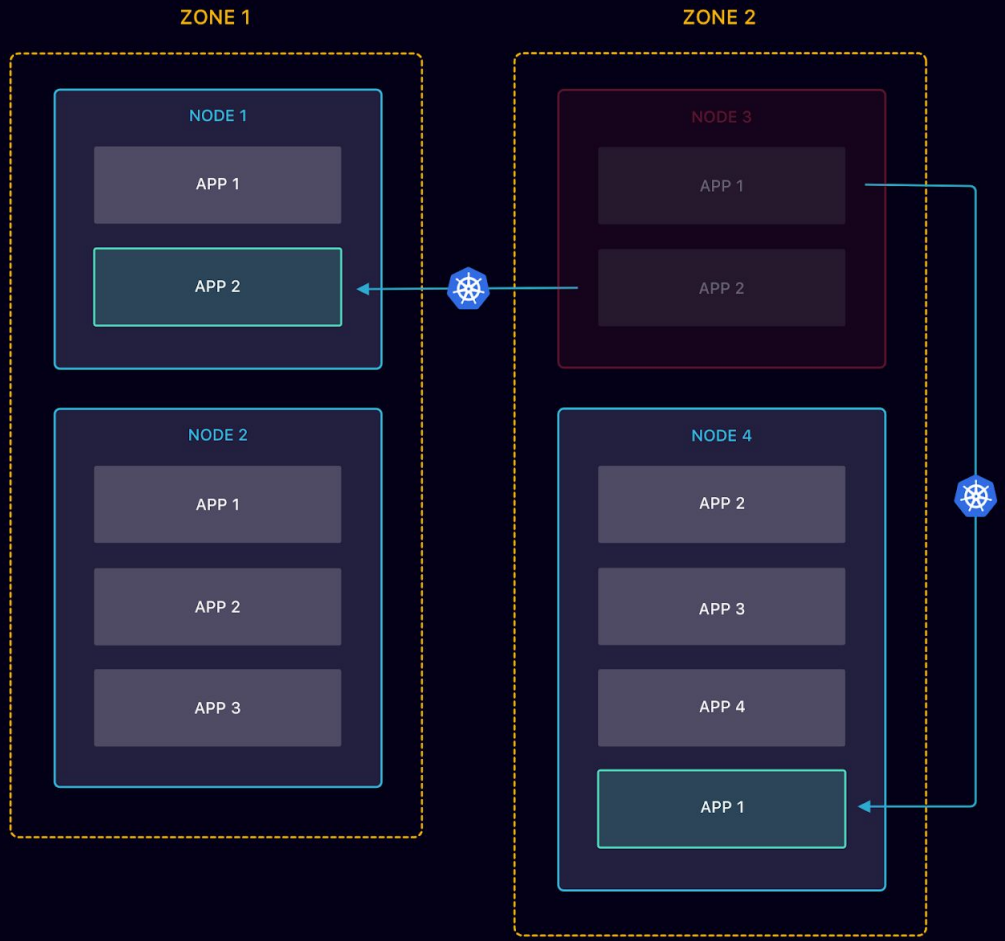

ZONE 1
NODE 1
APP 1
NODE 2
APP 1
APP 2
APP 3
ZONE 2
NODE 3
APP 1
APP 2
NODE 4
APP 2
APP 3
APP 4


ZONE 1
NODE 1
APP 1
NODE 2
APP 1
APP 2
APP 3
ZONE 2
NODE 3
APP 1
APP 2
NODE 4
APP 2
APP 3
APP 4

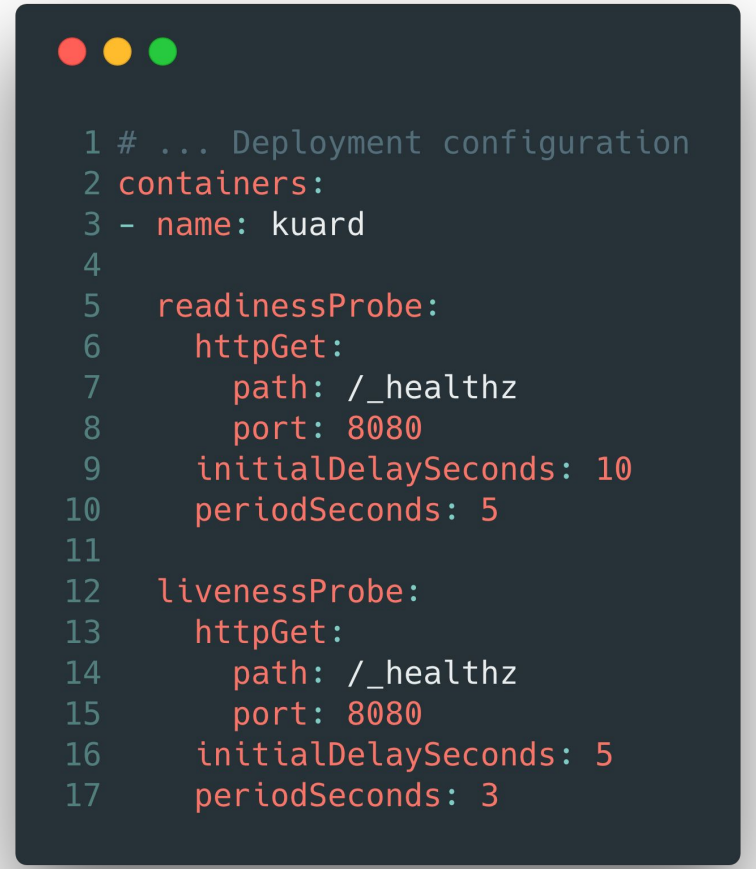

ZONE 1
ZONE 2
NODE 1
APP 1
APP 1
APP 2
NODE 2
APP 1
APP 2
APP 3
NODE 3
APP 1
APP 2
NODE 4
APP 2
APP 3
APP 4



```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: kuard
  labels:
    app: kuard
spec:
  # ...
  template:
    spec:
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 100
            podAffinityTerm:
              labelSelector:
                matchLabels:
                  app: kuard
              topologyKey: "kubernetes.io/hostname"
      # ...
```

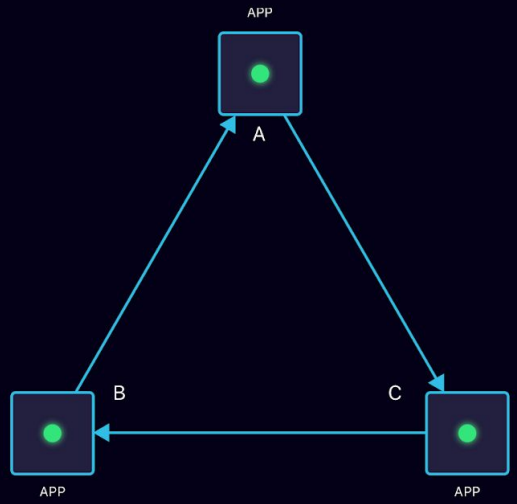


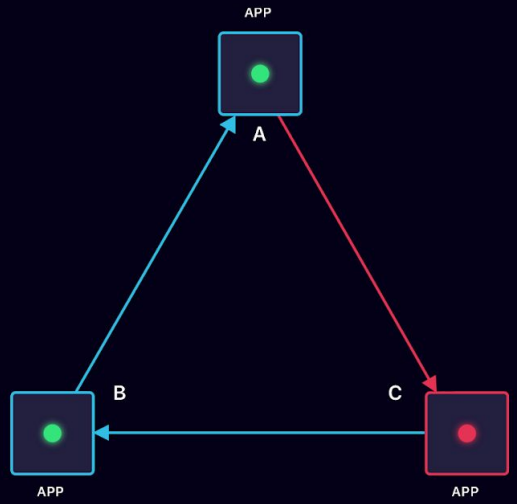
Probes

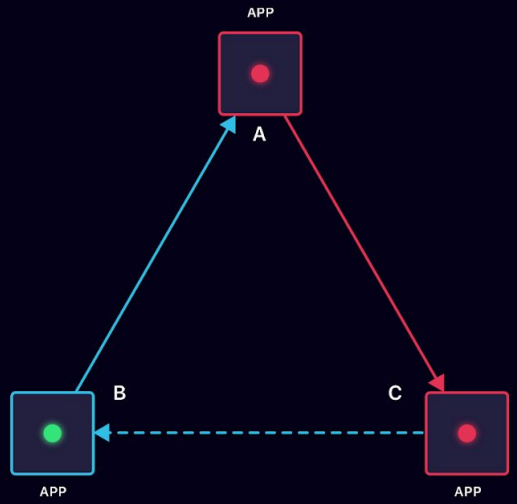


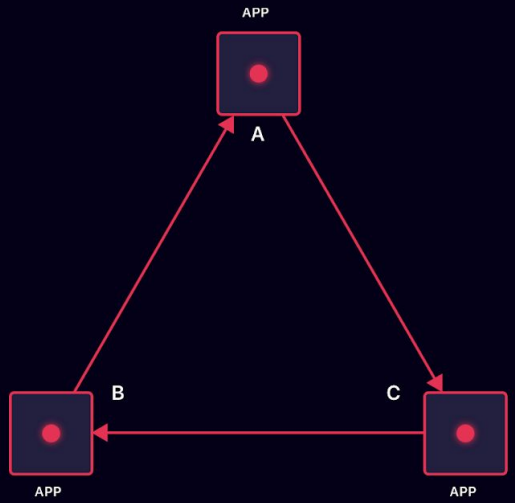
```yaml
# ... Deployment configuration
containers:
- name: kuard

  readinessProbe:
    httpGet:
      path: /_healthz
      port: 8080
    initialDelaySeconds: 10
    periodSeconds: 5

  livenessProbe:
    httpGet:
      path: /_healthz
      port: 8080
    initialDelaySeconds: 5
    periodSeconds: 3
```

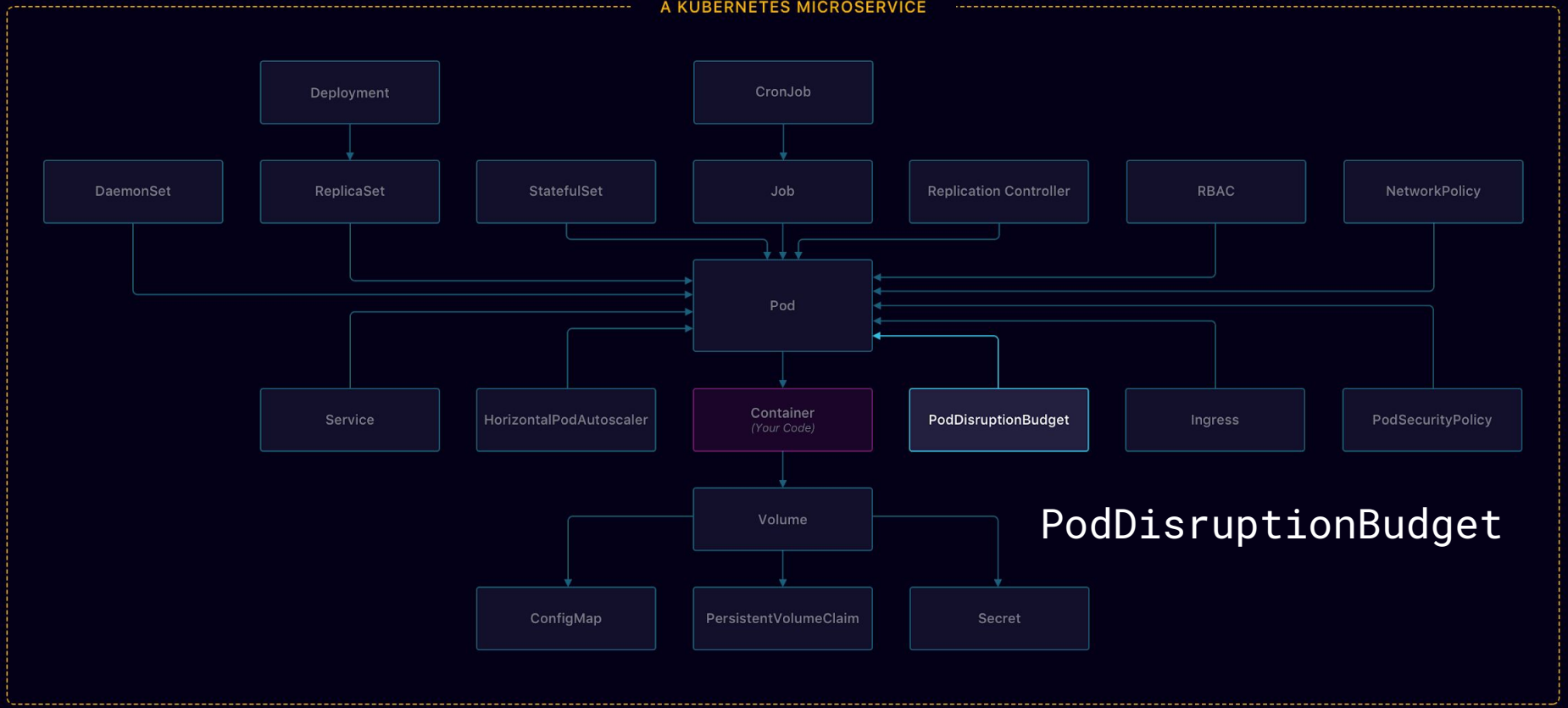
Caveat: Circular Dependencies


APP
A
B
C
APP
APP


APP
A
B
C
APP
APP

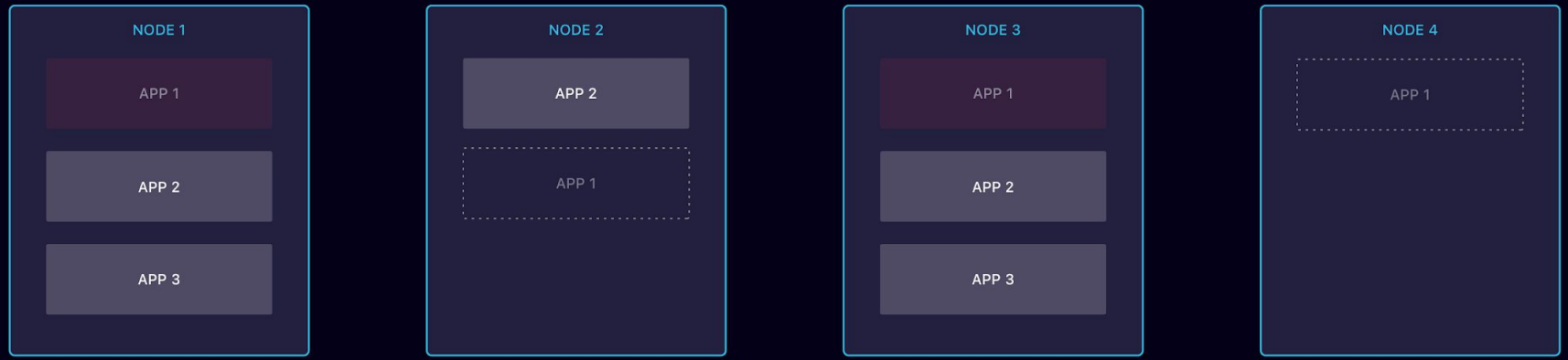

APP
A
B
C
APP
APP

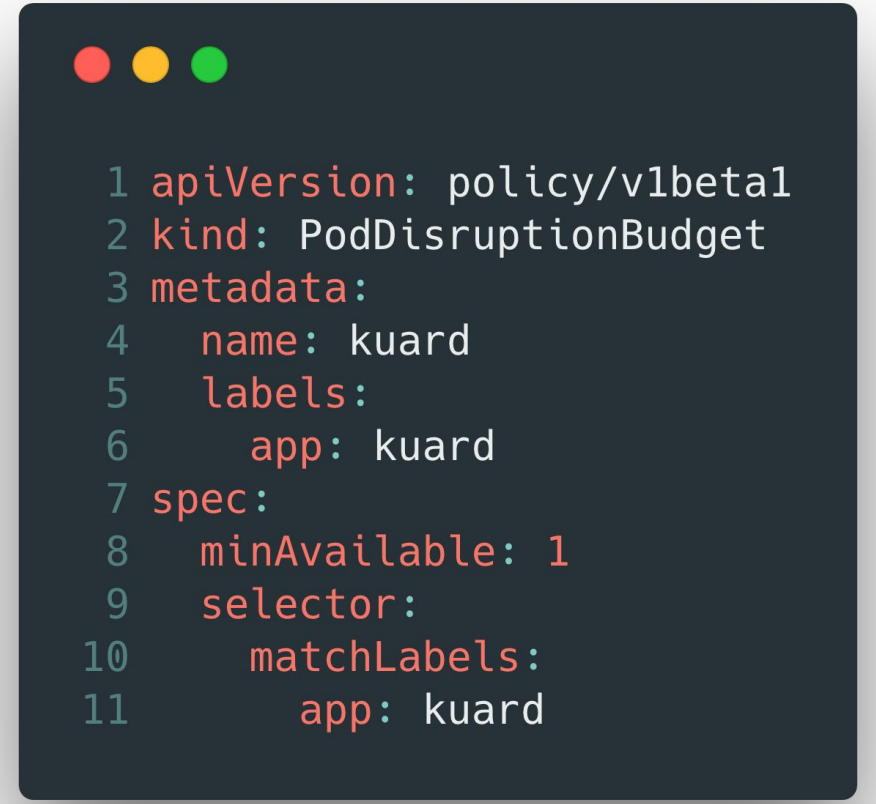

APP
A
B
C
APP
APP



@jelmersnoeck


NODE 1
APP 1
APP 2
APP 3
NODE 2
APP 2
NODE 3
APP 1
APP 2
APP 3
NODE 4

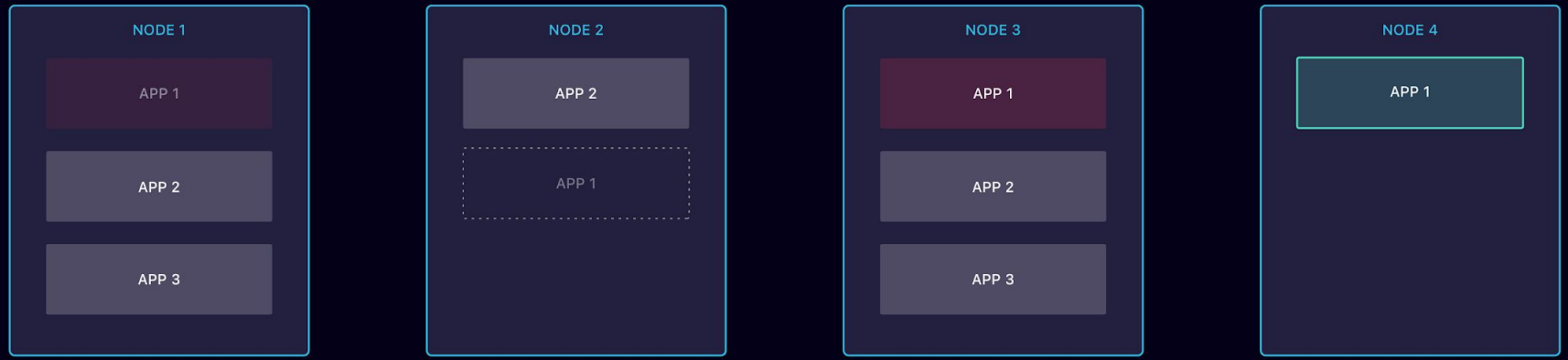

NODE 1
APP 1
APP 2
APP 3
NODE 2
APP 2
APP 1
NODE 3
APP 1
APP 2
APP 3
NODE 4
APP 1



```yaml
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
metadata:
  name: kuard
  labels:
    app: kuard
spec:
  minAvailable: 1
  selector:
    matchLabels:
      app: kuard
```


NODE 1
APP 1
APP 2
APP 3
NODE 2
APP 2
NODE 3
APP 1
APP 2
APP 3
NODE 4


NODE 1
APP 1
APP 2
APP 3
NODE 2
APP 2
NODE 3
APP 1
APP 2
APP 3
NODE 4
APP 1

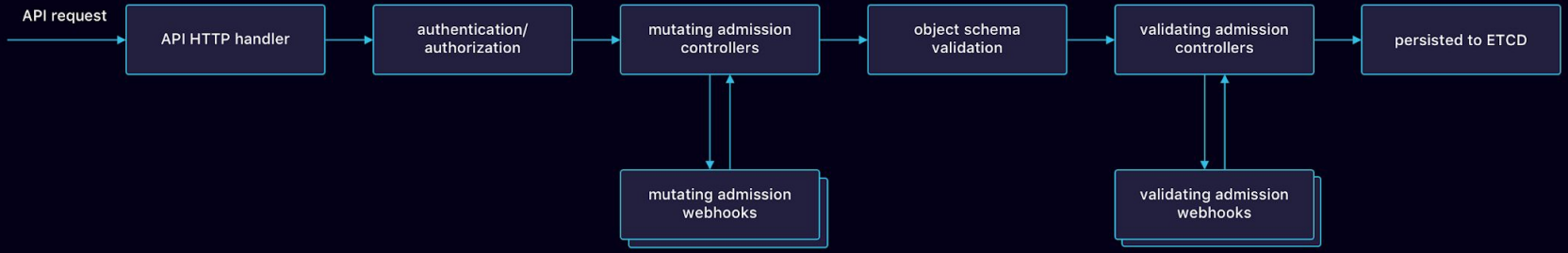

NODE 1
APP 1
APP 2
APP 3
NODE 2
APP 2
APP 1
NODE 3
APP 1
APP 2
APP 3
NODE 4
APP 1

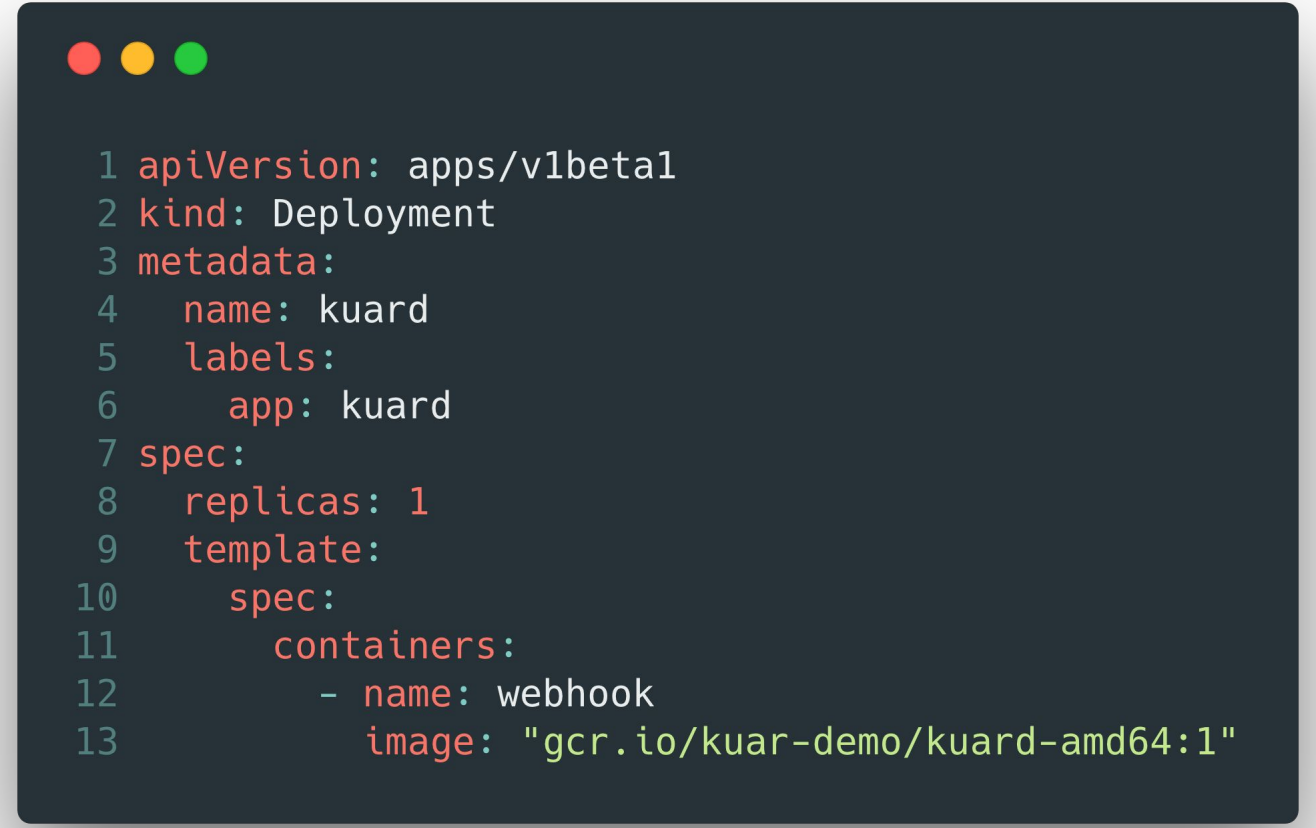
Prevent misconfiguration

Linting?

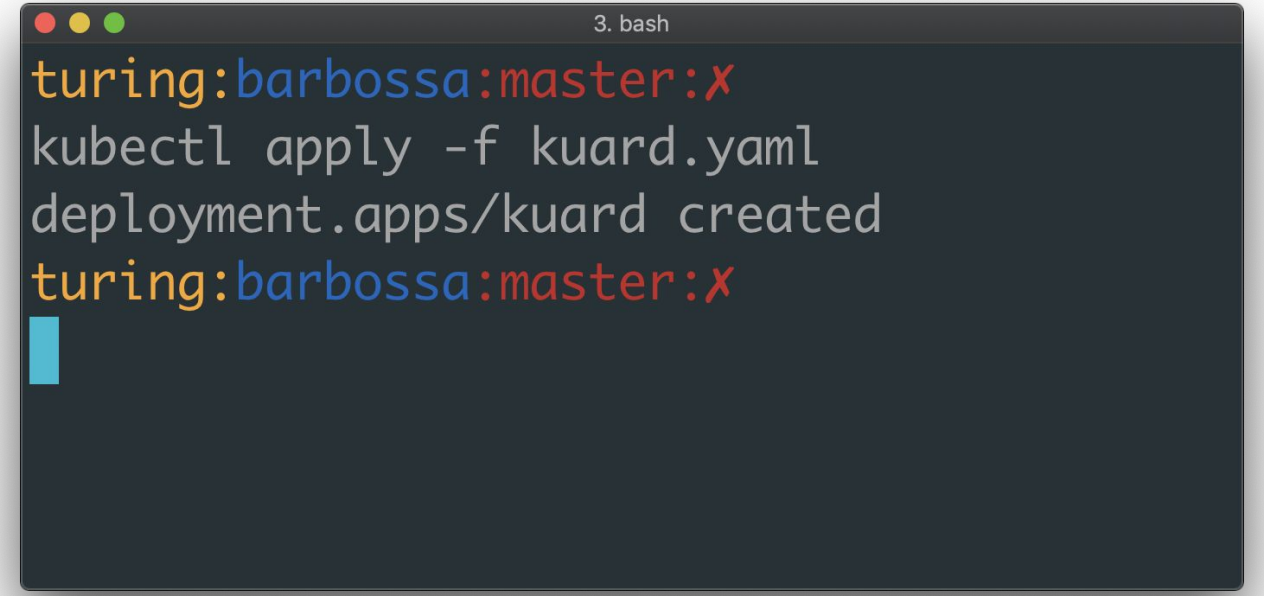
Webhooks!

manifold

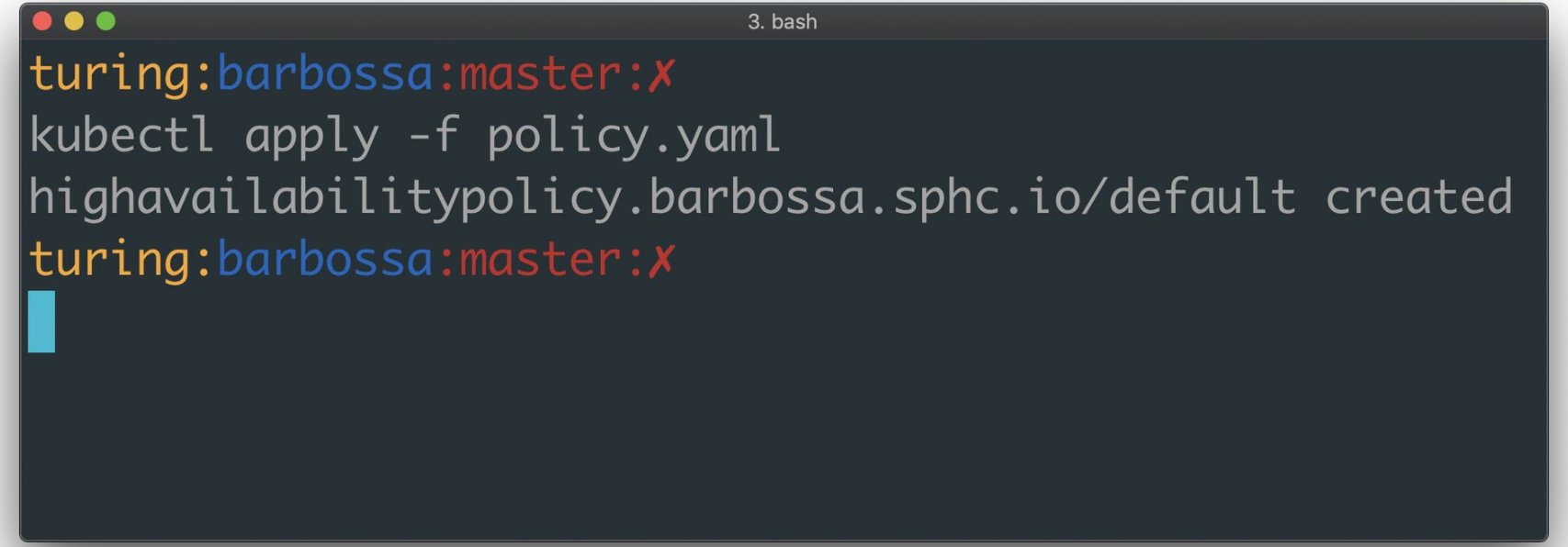

API request
API HTTP handler
authentication/ authorization
mutating admission controllers
object schema validation
validating admission controllers
persisted to ETCD
mutating admission webhooks
validating admission webhooks

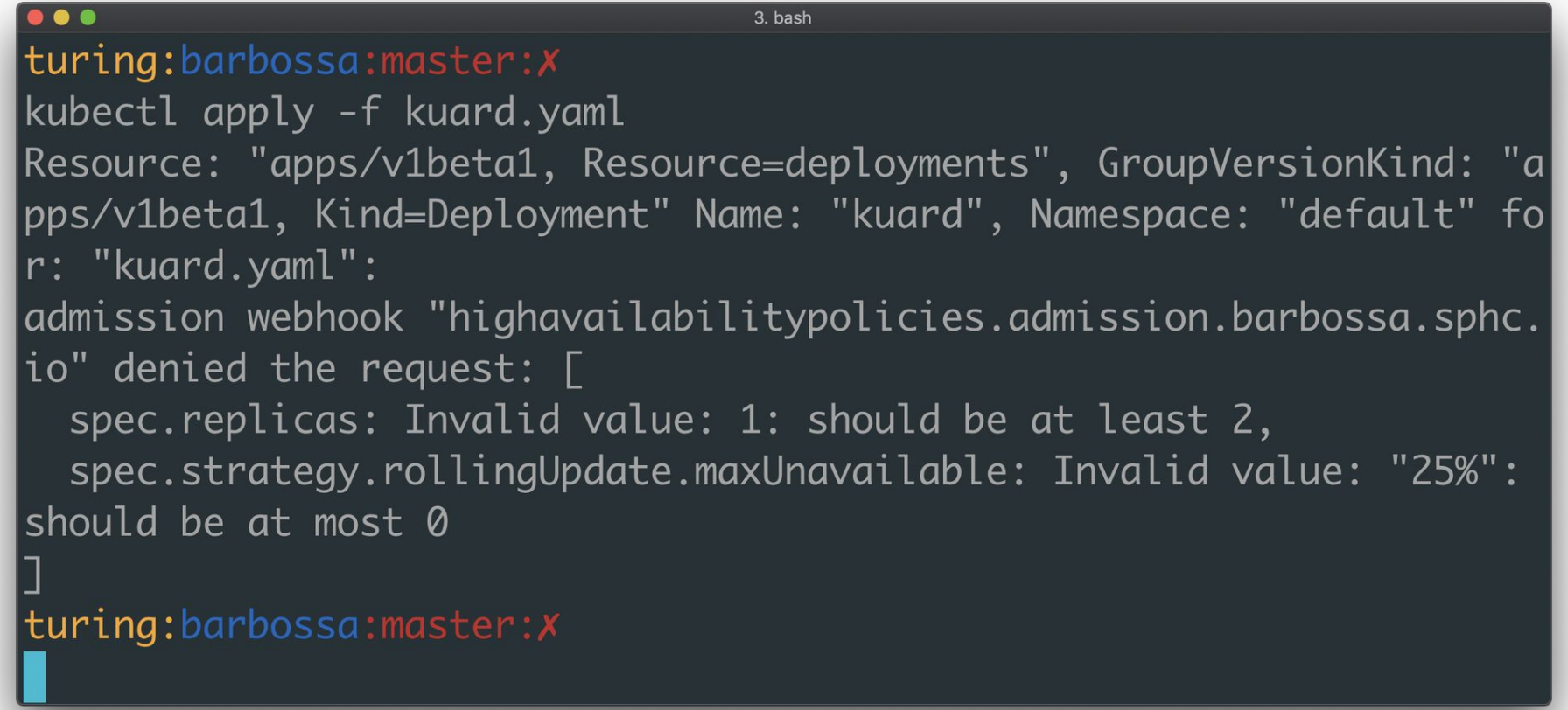


```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: kuard
  labels:
    app: kuard
spec:
  replicas: 1
  template:
    spec:
      containers:
        - name: webhook
          image: "gcr.io/kuar-demo/kuard-amd64:1"
```


3. bash
turing:barbossa:master:✗
kubectl apply -f kuard.yaml
deployment.apps/kuard created
turing:barbossa:master:✗

```yaml
apiVersion: barbossa.sphc.io/v1alpha1
kind: HighAvailabilityPolicy
metadata:
  name: default
  namespace: default
spec:
  selector: {}
  replicas:
    minimum: 2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      minSurge: 25%
      maxSurge: 50%
      maxUnavailable: 0
```


3. bash
turing:barbossa:master:✗
kubectl apply -f policy.yaml
highavailabilitypolicy.barbossa.sphc.io/default created
turing:barbossa:master:✗


3. bash
turing:barbossa:master:✗
kubectl apply -f kuard.yaml
Resource: "apps/v1beta1, Resource=deployments", GroupVersionKind: "a
pps/v1beta1, Kind=Deployment" Name: "kuard", Namespace: "default" fo
r: "kuard.yaml":
admission webhook "highavailabilitypolicies.admission.barbossa.sphc.
io" denied the request: [
spec.replicas: Invalid value: 1: should be at least 2,
spec.strategy.rollingUpdate.maxUnavailable: Invalid value: "25%":
should be at most 0
]
turing:barbossa:master:✗

Webhooks

- Barbossa
- Azure Kubernetes Policy Controller (OPA)
- Anchore Engine
- …

Thanks

I'll be around for questions

FIND ME

 github.com/jelmersnoeck

 twitter.com/jelmersnoeck

SPECIAL THANKS TO

 twitter.com/megthesmith

 manifold

Resources

- https://hackernoon.com/deploying-rock-solid-applications-with-kubernetes-230fd9bb61f4
- https://thenewstack.io/myth-cloud-native-portability
- https://kubernetes.io/docs/concepts/policy/pod-security-policy/
- https://kubernetes.io/docs/reference/access-authn-authz/rbac/
- https://kubernetes.io/docs/concepts/services-networking/network-policies/
- https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
- https://container-solutions.com/kubernetes-deployment-strategies/
- https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity
- https://kubernetes.io/docs/concepts/workloads/pods/disruptions/
- https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/
- https://banzaicloud.com/blog/k8s-admission-webhooks/
- https://github.com/jelmersnoeck/barbossa
- https://github.com/Azure/kubernetes-policy-controller