Kubernetes schrittweise zum Selbstbedienungsladen für Entwickler ausbauen

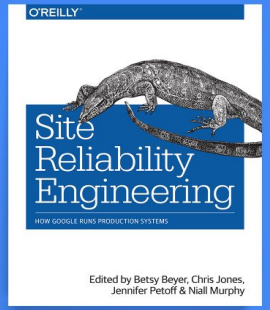
November, 2019

Google - Site Reliability Engine

landing.google.com/sre/books/

Google

What is SRE? In Conversation Resources Read SRE Books



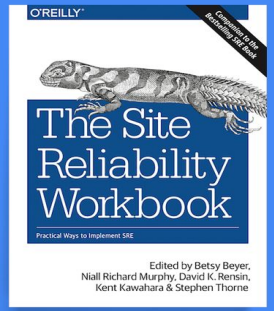
Site Reliability Engineering

Edited by Betsy Beyer, Chris Jones, Jennifer Petoff and Niall Richard Murphy

Members of the SRE team explain how their engagement with the entire software lifecycle has enabled Google to build, deploy, monitor, and maintain some of the largest software systems in the world.

READ ONLINE FOR FREE

BUY FROM GOOGLE BOOKS



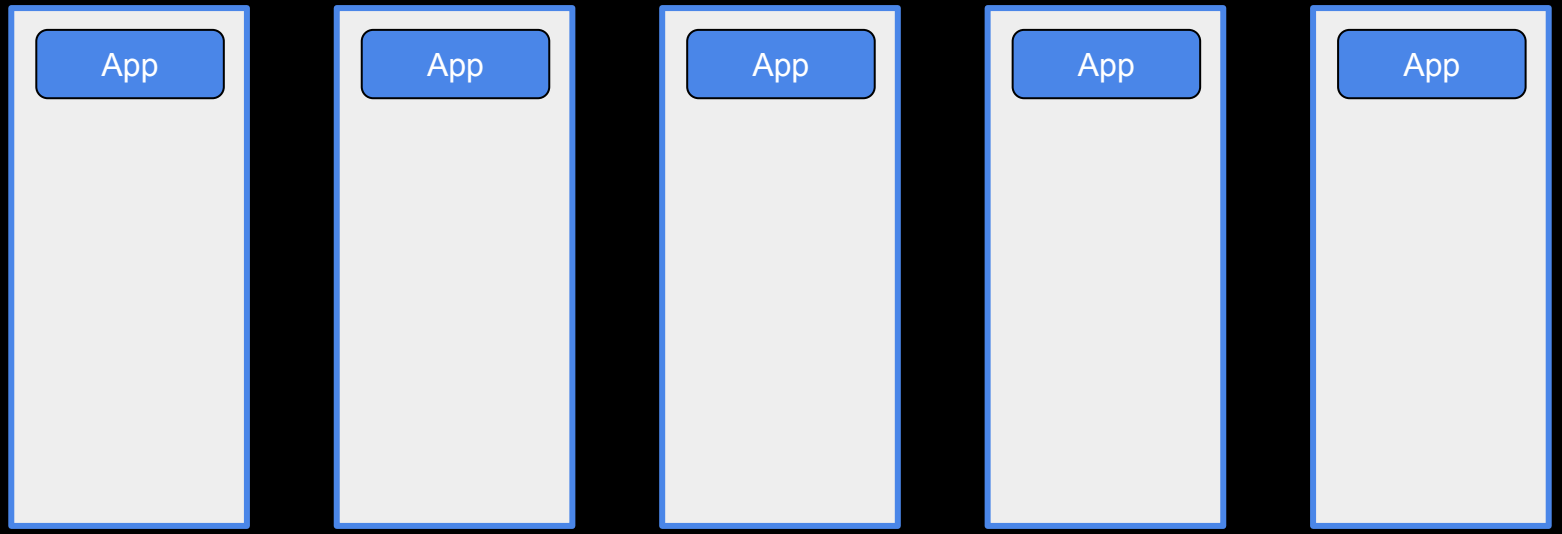
The Site Reliability Workbook

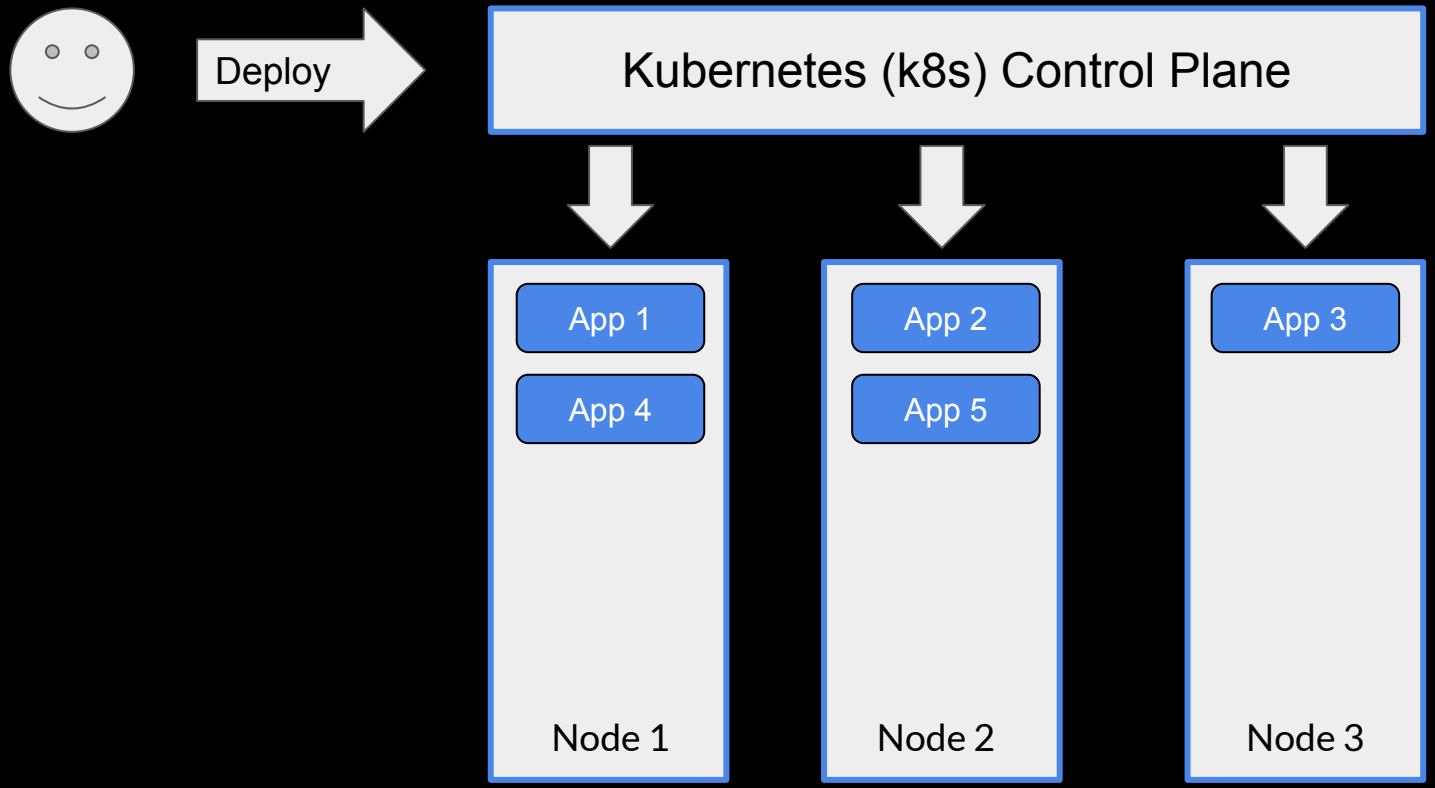
Edited by Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara and Stephen Thorne

The Site Reliability Workbook is the hands-on companion to the bestselling Site Reliability Engineering book and uses concrete examples to show how to put SRE principles and practices to work. This book contains practical examples from Google's experiences and case studies from Google's Cloud Platform customers. Evernote, The Home Depot, The New York Times, and other companies outline hard-won experiences of what worked for them and what didn't.

READ ONLINE FOR FREE

BUY FROM GOOGLE BOOKS





Pods

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp
spec:
  containers:
    - name: nginx
      image: nginx:1.7.9
      ports:
        - containerPort: 80
```

Demo 1: Deploy Pod

Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
```

Skalieren und Updates

```yaml
spec:
  strategy:
    rollingUpdate:
      maxUnavailable: 50%
    replicas: 2
```

Probes

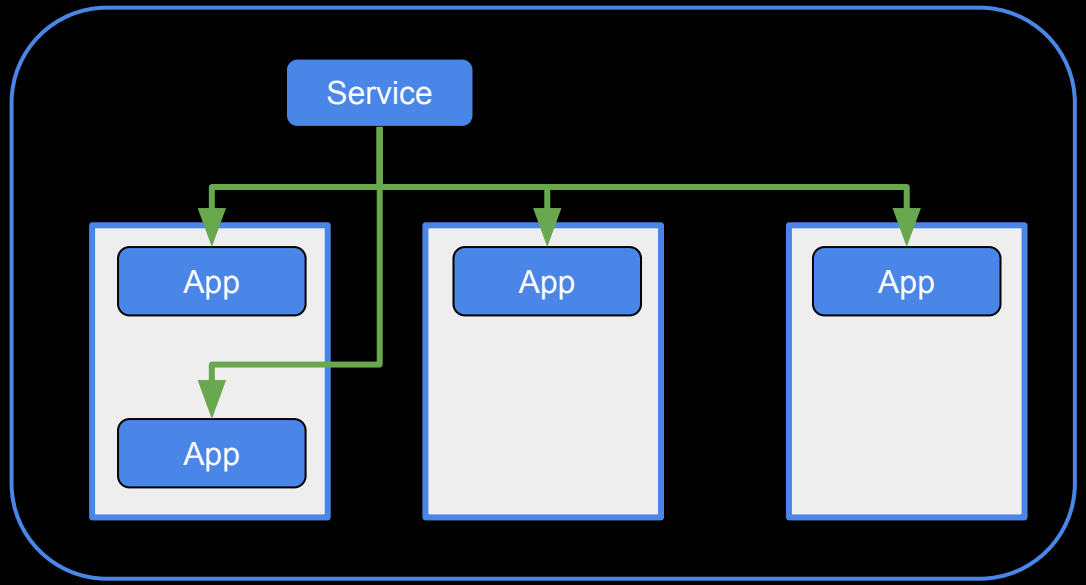
```yaml
spec:
  template:
    spec:
      containers:
      - name: nginx
        livenessProbe:
          tcpSocket:
            port: 80
          timeoutSeconds: 3
          periodSeconds: 30
        readinessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 30
```

TASKTOP

Demo 2: Deployments skalieren

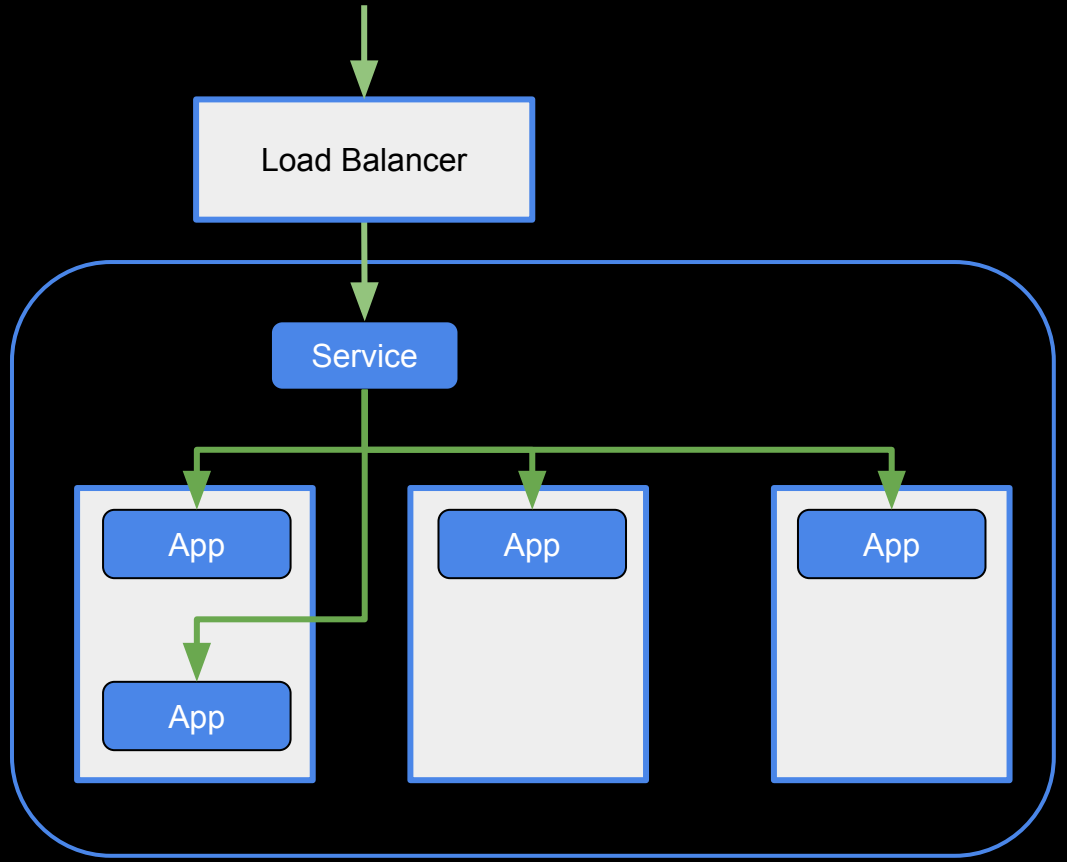
Services

```yaml
apiVersion: v1
kind: Service
apiVersion: v1
metadata:
  name: myapp
spec:
  selector:
    app: myapp
  ports:
  - port: 80
```



Load Balancer

```yaml
apiVersion: v1
kind: Service
apiVersion: v1
metadata:
  name: myapp
spec:
  type: LoadBalancer
  selector:
    app: myapp
  ports:
  - port: 80
```



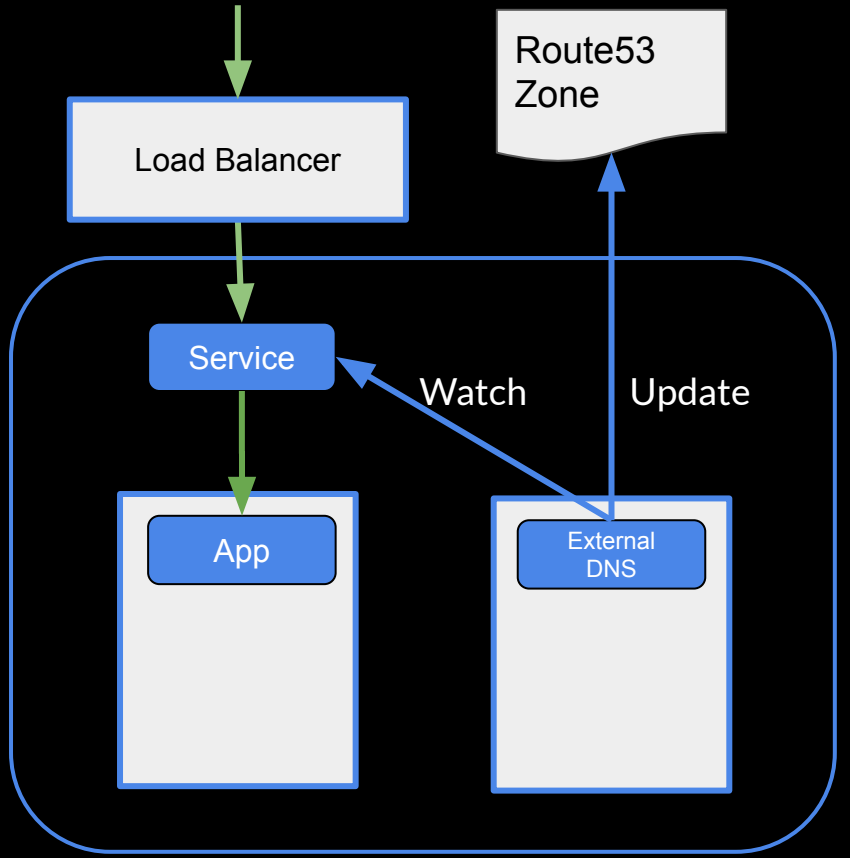
TLS

Es können eigene Zertifikate oder Cloud Provider (z.B. AWS) Zertifikate verwendet werden

```
apiVersion: v1
kind: Service
apiVersion: v1
metadata:
  name: myapp
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-ssl-cert: |
      arn:aws:acm:eu-central-1:1234:certificate/abcd
    service.beta.kubernetes.io/aws-load-balancer-backend-protocol: http
    service.beta.kubernetes.io/aws-load-balancer-ssl-ports: "443"
```

External DNS

```
apiVersion: v1
kind: Service
apiVersion: v1
metadata:
  name: myapp
  annotations:
    external-dns.alpha.kubernetes.io/hostname: |
      myapp.example.com
```

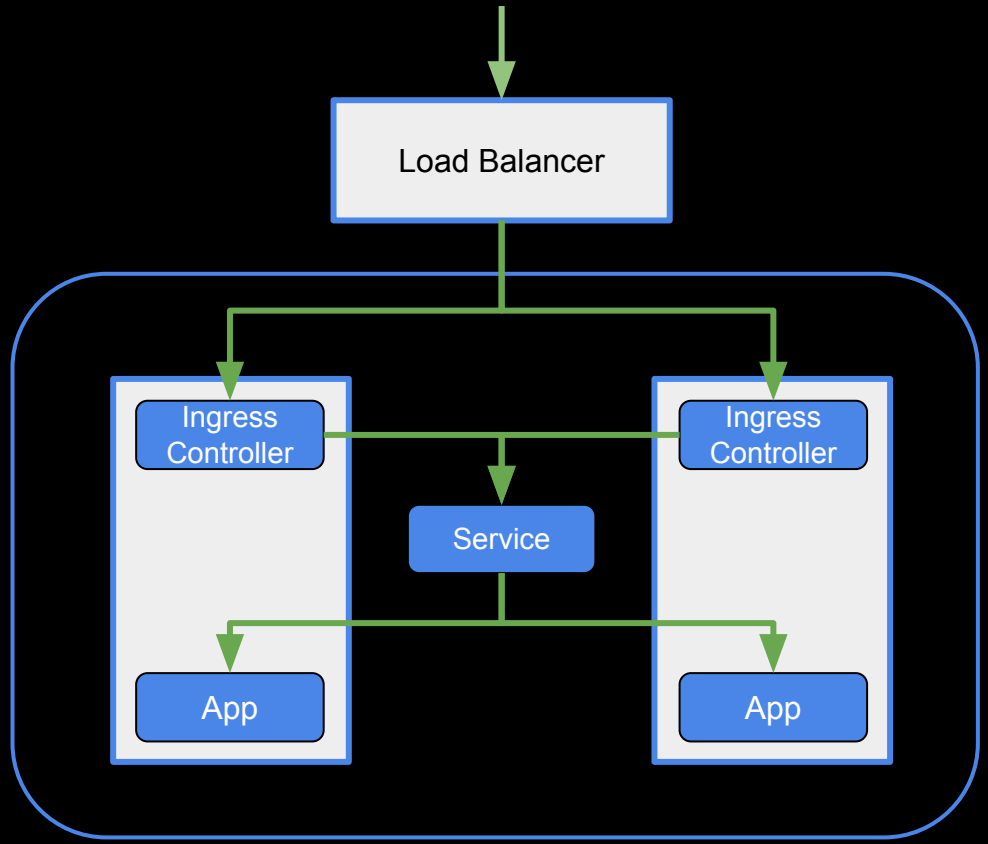


TASKTOP

Demo 3: Services und Load Balancer

Ingress

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: myapp
spec:
  rules:
    - host: myapp.example.com
      http:
        paths:
        - path: /
          backend:
            serviceName: myapp
            servicePort: 80
```



Persistenz

```yaml
apiVersion: apps/v1
kind: StatefulSet
spec:
  serviceName: myapp
  replicas: 1
  volumeClaimTemplates:
  - metadata:
      name: myapp-pages
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: "1Gi"
```

```yaml
template:
  metadata:
    labels:
      app: myapp
  spec:
    containers:
    - name: nginx
      image: nginx:1.7.9
      ports:
      - containerPort: 80
      volumeMounts:
        - name: myapp-pages
          mountPath: /usr/share/nginx/html
```

TASKTOP

Demo 4: Ingress, Persistenz und Helm

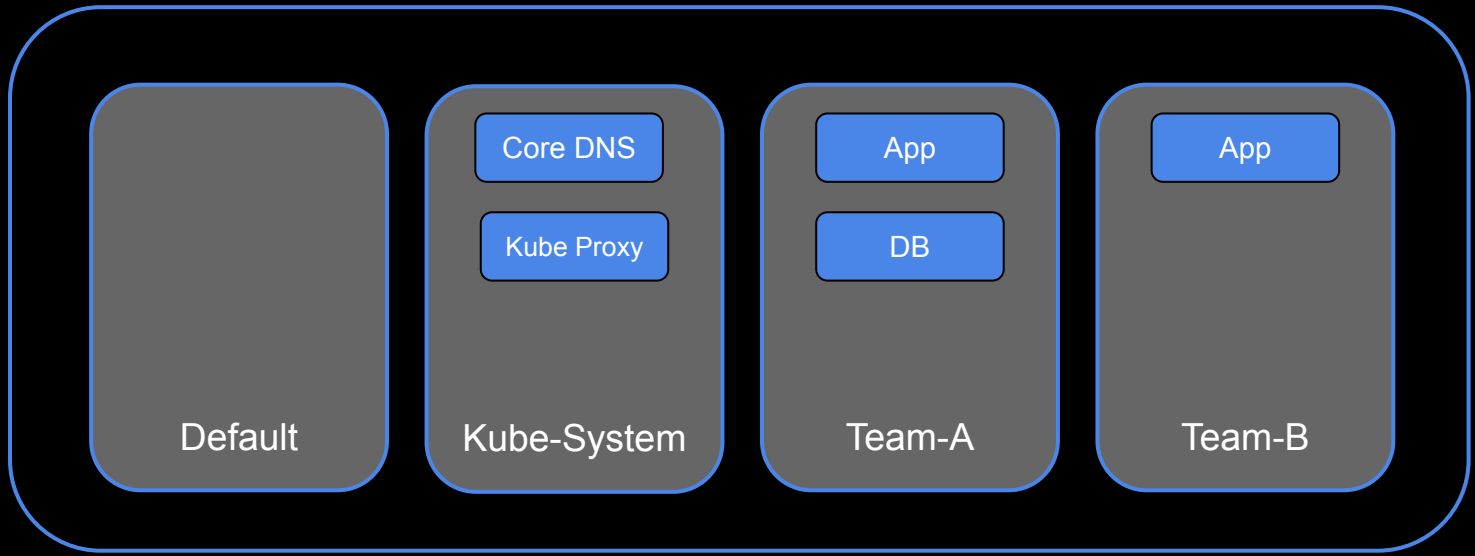
Zwischenstand

Entwickler können mit den gezeigten Techniken Applikation deployen, die folgende Anforderungen erfüllen:

- Load balancing über mehrere Instanzen
- Ausfallsicherheit und Verfügbarkeit während Updates
- Endpunkte sind mit TLS terminiert
- Endpunkte sind per DNS auflösbar
- Daten werden persistiert

Namespaces

Namespaces ermöglichen mehreren Teams oder Projekten isoliert voneinander auf dem selben Cluster zu arbeiten.



Berechtigungen

Role based access control (RBAC)

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: developers
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
- kind: Group
  name: developers
```

Gilt nur fuer den Namespace in dem das RoleBinding deployed ist.
ClusterRoleBinding kann für globale Berechtigungen verwendet werden.

Eigene Rollen

Berechtigungen lassen sich feingranular auf Resourcenebene festlegen.

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: list-namespaces
rules:
  - apiGroups:
      - ""
    resources:
      - namespaces
    verbs:
      - get
      - list
```

TASKTOP

Demo 5: Namespaces und Berechtigungen

Pod Security Policies

Pod Security Policies ermöglichen Einschränkungen für das Starten und Aktualisieren von Pods festzulegen:

- Erzwingen von docker security profilen
- Capabilities beschränken
- Starten als root bzw. sudo unterbinden
- Volumetypen beschränken

Pod Security Policy

Docker profile schränken container ein, z.B. Zugriffe auf /proc oder /sys

```yaml
apiVersion: extensions/v1beta1
kind: PodSecurityPolicy
metadata:
  name: development
  annotations:
    seccomp.security.alpha.kubernetes.io/defaultProfileName:  'docker/default'
    apparmor.security.beta.kubernetes.io/defaultProfileName:  'runtime/default'
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: 'docker/default'
    apparmor.security.beta.kubernetes.io/allowedProfileNames: 'runtime/default'
```

Pod Security Policy

- Keine Privilegierten Pods
- Keine Zugriff auf den Host:
  - Verzeichnisse
  - Netzwerk
  - Prozesse

```
spec:
  privileged: false
  # Required to prevent escalations to root.
  allowPrivilegeEscalation: false
  requiredDropCapabilities:
    - ALL
  volumes:
    - 'configMap'
    - 'emptyDir'
    - 'projected'
    - 'secret'
    - 'downwardAPI'
    - 'persistentVolumeClaim'
  hostNetwork: false
  hostIPC: false
  hostPID: false
```

Pod Security Policy

Ausführen von Prozessen in Containern nicht als root.

```yaml
spec:
  runAsUser:
    # Require the container to run without root privileges.
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'MustRunAs'
    ranges:
      # Forbid adding the root group.
      - min: 1
        max: 65535
  fsGroup:
    rule: 'MustRunAs'
    ranges:
      # Forbid adding the root group.
      - min: 1
        max: 65535
```

Network Policies

Network Policies definieren Regeln zur Kommunikation von Pods.

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: dns
spec:
  podSelector:
    matchLabels: {}
  egress:
  # allow all pods to resolve DNS names
  - to:
    - namespaceSelector:
        matchLabels:
          name: kube-system
    ports:
    - protocol: UDP
      port: 53
```

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: kafka-broker
spec:
  podSelector:
    matchLabels:
      app: kafka
  ingress:
  # incoming connections from kafka clients
  - from:
    - podSelector:
        matchLabels:
          kafka-client: true
    ports:
    - protocol: TCP
      port: 9092
```

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: kafka-client
spec:
  podSelector:
    matchLabels:
      kafka-client: true
  egress:
  # outgoing connections to kafka
  - to:
    - podSelector:
        matchLabels:
          app: kafka
    ports:
    - protocol: TCP
      port: 9092
```

TASKTOP

Demo 6: Security und Network Policies

Misc

Deployment

- CI/CD für automatisierte Deployments
- Helm zur Paketverwaltung

Monitoring

- SLOs helfen die Robustheit zu messen und zu priorisieren

Sicherheit

- Kommerzielle Werkzeuge zur aktiven Überwachung
- Vault um Geheimnisse zu Verwalten

TASKTOP

Fragen?