# CKA

Certified Kubernetes Administrator

# Einführung

# Information

Der CKA-Test ist rund um das Managed und Administrieren von Kubernetes.

Hier sind ein paar nützliche Links:

CKA Zertifikat:

https://www.cncf.io/certification/cka/

CKA Curriculum (Zusammenfassung der Themen):

https://github.com/cncf/curriculum

CKA Curriculum (KodeKloud edition)

https://github.com/kodekloudhub/certified-kubernetes-administrator-course

CKA Teilnehmer Handbuch:

https://docs.linuxfoundation.org/tc-docs/certification/lf-handbook2

CKA Test Tips & Tricks:

https://docs.linuxfoundation.org/tc-docs/certification/tips-cka-and-ckad

Use the code - **KODEKLOUD20** - while registering for the CKA or CKAD exams at Linux Foundation to get a 20% discount.

Kodekloud Slack:

https://kodekloud.com/pages/community

# Grund Konzept

# Cluster Architektur

# Control Nodes (Controllplane / Master Node)

## Controlplane Komponenten

Diese Komponenten können, müssen aber nicht in form von Containern auf den Control Nodes laufen.

### ETCD

Dies ist ein Key-Value Store (Datenbank) welche die Informationen speichert.

### Scheduler

Dies ist eine Applikation, welche die Container und Deployments auf die entsprechende Node schiebt.

### Controller-Manager

#### Node-Controller

Kümmert sich um das Hinzufügen, Entfernen und Status spezifische handeln von nodes. Bsp. wenn eine node abstürzt.

### Replication-Controller

Kümmert sich um das Erstellen, Löschen und verschieben von Containern auf den Unterschiedlichen Nodes.

## Kube-Apiserver

Kümmert sich um die Kommunikation zwischen den unterschiedlichen Komponenten und veröffentlicht die eigentliche Kuberentes API welche mittels Kubectl usw. angesteuert wird.

## Conatiner Runtime Engine

Die Master nodes haben optional eine Container Runtime Engine (Bsp.: Docker, Containerd, Rocket), der Master kann die Controlplane Komponenten in Containern laufen lassen, diese können jedoch auch als applikation auf dem Host direkt laufen.

# Worker Nodes

## Conatiner Runtime Engine

Jede Worker node hat eine Container Tunrtime Engine, diese wird benötigt um die Container auf der jeweiligen Node auszuführen.

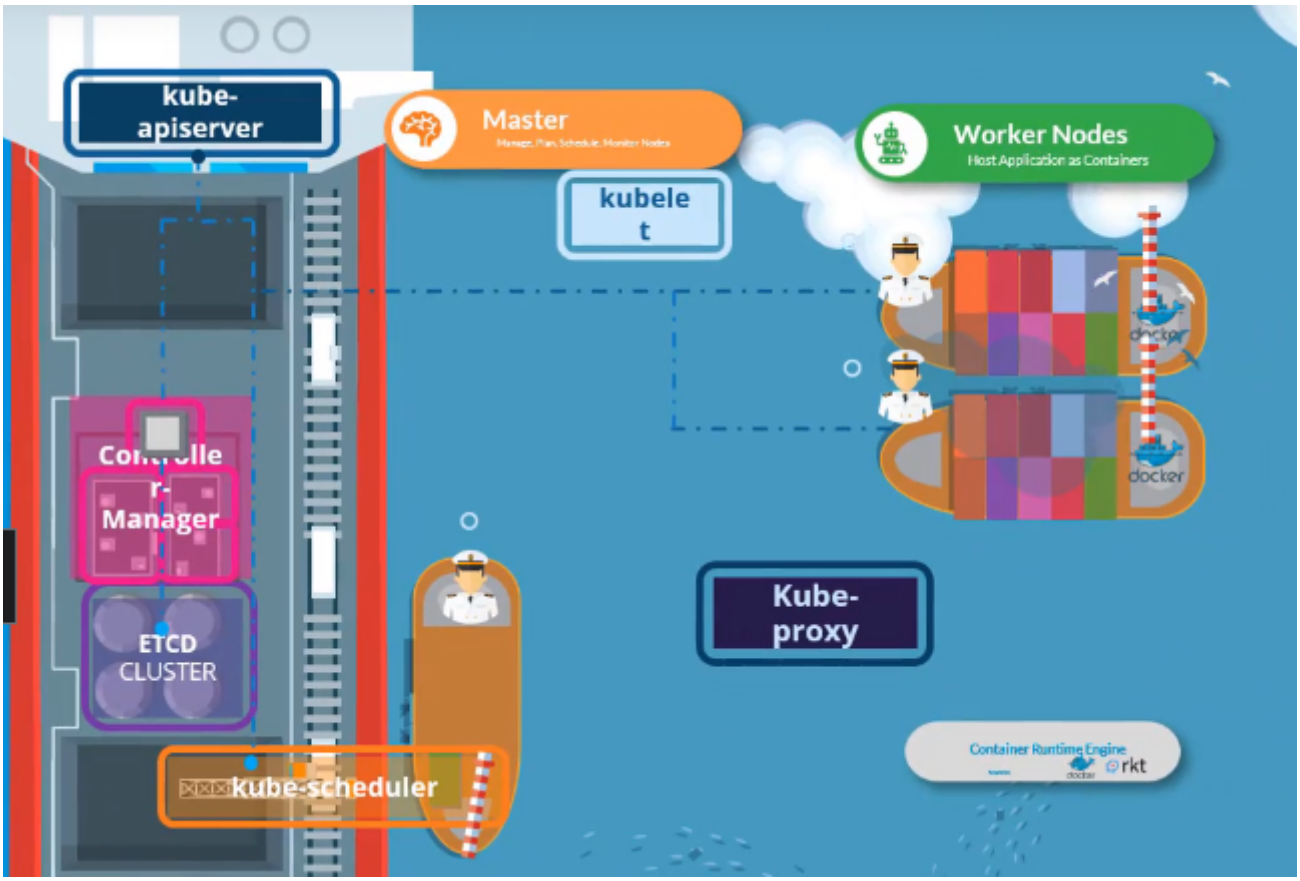
## Kubelet

Die Kubelet ist der "Kapitän" der Nodes, diese Applikation ist dafür verantwortlich die Kommunikation mit der Masternode aufrecht zu halten und die Anforderungen entsprechend umzusetzen (Bsp.: neue Container aus zu führen)

## Kube-Proxy

Der Kube-Proxy ist für die Kommunikation zwischen Container auf unterschiedlichen Nodes verantwortlich.

# Übersicht


kube-
apiserver
Master
kubele
t
Worker Nodes
Host Application as Containers
Controlle
r-
Manager
ETCD
CLUSTER
kube-scheduler
Kube-
proxy
Container Runtime Engine
docker
rkt

# Kube Controller Manager

Der Kube-Controller-Manager ist ein Manager von mehreren Kubernetes Controllern. In Kubernetes ist ein Kontroller dazu da, ein Cluster zu überwachen und anpassungen basierend auf den definierten Einstellungen zu tätigen.

# Node-Controller

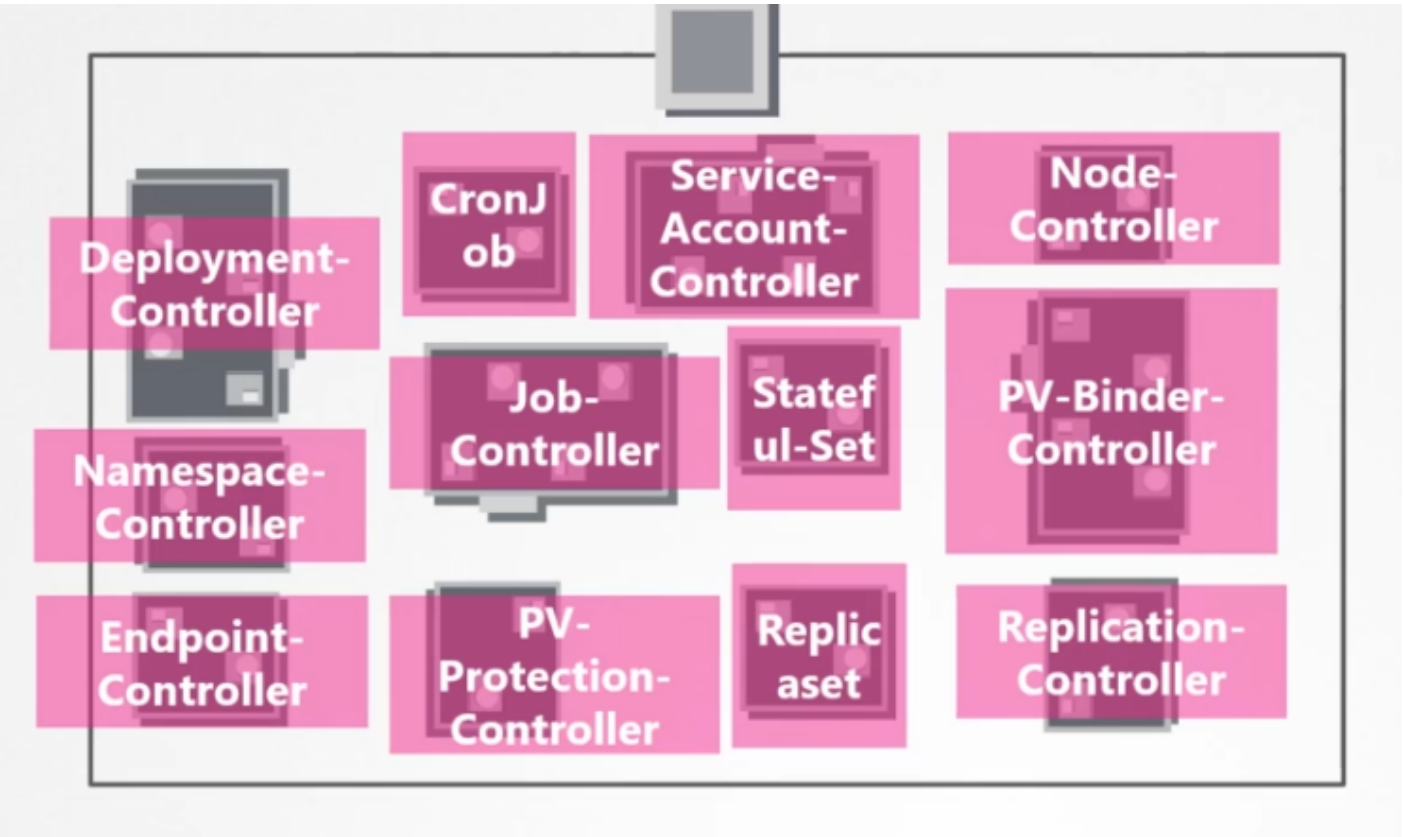
Der Node Kontroller überprüft jede node alle 5 Sekunden, wenn der Node-Controller 40 Sekunden lange keine Information von der Node erhält, wird die Node als UNREACHABLE markiert. Nach 5 Minuten ohne Kontakt werden alle Pods auf eine andere Node verschoben (POD Eviction Timeout)

# Replication-Controller

Der Replication Kontroller überprüft die Deployments, ob alle die Entsprechenden replikationen haben, wenn nicht wird dies angepasst.

# Andere Controller

Es gibt unzählige unterschiedliche Controller, diese sind alle teil von Kube-Controller-Manager. Sämtliche Einstellungen bezüglich Timeouts und Eviction times werden bei dieser Binary definiert.


Deployment-Controller
CronJob
Service-Account-Controller
Node-Controller
Job-Controller
Stateful-Set
PV-Binder-Controller
Namespace-Controller
Endpoint-Controller
PV-Protection-Controller
Replicaset
Replication-Controller

# Kube Scheduler

Der Kube Scheduler ist zuständig um die Pods auf die Nodes zu verteilen, der Kube Scheduler trifft nur die Entscheidung, die Pods werden anschliessend von dem Kubelet ausgeführt. Die Verteilung basiert auf diversen Kriterien wie aktuelle auslastung, insgesamt Ressourcen und mehr.

# Kubelet

Das Kubelet wird auf jeder Node ausgeführt und ist zuständig um die Container auf den Nodes auszuführen. Das Kubelet kommuniziert mit dem Kube-Api server und fragt, was auf der Entsprechenden Node ausgeführt werden sollte.

# Kube Proxy

In einem Kubernetes Cluster kann jeder Pod jeden anderen Pod erreichen, dies geschieht in Kubernetes via Kube Proxy. Kube Proxy ist ein Daemon set, welches auf jeder node läuft und iptable rules erstellt um den traffic zwischen den verschiedenen Nodes weiterzuleiten. Services sind auch virtuelle objekte welche auf dem Kube Proxy existieren, ansonsten aber nicht existent sind, alles was zu einem Service gerouted wird geht über den Kube Proxy, dieser leitet es dann an das entsprechende backend weiter.

# Kubernetes Ressourcen

# Pods

Ein Pod in Kubernetes ist ein container mit der Instanz der Applikation welche wir ausführen wollen, in einigen fällen beinhaltet ein Pod zusätzlich zum Applikationscontainer auch noch ein oder mehrere Helper Container, welche weitere aufgaben im selben Pod ausführen. Alle Container in einem Pod haben den selben storage und netzwerkspace.

Example YAML:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
```

# Replica Controller

Der Replikations Kontroller ist ein Kontroller, welcher die Anzahl von Applikations instanzen im Kubernetes cluster managed. In einem ReplicaController wird der Pod als Template hinterlegt und anschliessend deployed.

Beseispiels YAML:

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-rc
  labels:
    app: nginx
    type: webserver
spec:
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx
        type: webserver
    spec:
      containers:
      - name: nginx-container
        image: nginx
  replicas: 3
```

# Replica Set

Ein Replica Set ist ähnlich wie ein Replica Controller, jedoch managed das Replica Set bereits erstellte Pods und erstellt wenn nötig neue wohingegen der Replica Controller für neue Pods genutzt wird. Kurz gesagt, das Replica Set kann bereits existierende Ressourcen Managen, dies geschieht über ein selector field.

Beispiels YAML:

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
  labels:
    app: nginx
    type: webserver
spec:
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx
        type: webserver
    spec:
      containers:
      - name: nginx
        image: nginx
  replicas: 5
  selector:
    matchLabels:
      type: webserver
```

# Glossar

| Abkürzung | Ausgeschrieben | Beschreibung |
| --- | --- | --- |
| CRI | Container Runtime Interface | Das ist das Interface welches kubernetes nutzt um mit den Container Runtime Engines (Containerd, Rocket) zu sprechen |