

# cert-manager für Kubernetes

## Installieren des cert-managers

### Erstellen des jetstack helm repo für den cert-manager

```
helm repo add jetstack https://charts.jetstack.io
```

### Installation des cert-manager inklusive CRDs Support

```
helm install cert-manager jetstack/cert-manager -n cert-manager --set crds.enabled=true --create-namespace
```

## Konfigurieren eines ClusterIssuer Objektes

Ein ClusterIssuer Objekt enthält das CA und den Schlüssel der eigenen rootCA. Mit Hilfe dieser Informationen werden dann passende Server Zertifikate erstellt. Wie man sich eine eigene CA erstellen kann habe [ich hier unter anderem dokumentiert](#).

Mit Hilfe der CA oder besser noch einer Intermediate CA und dem passendem keyfile wird ein ClusterIssuer Objekt erstellt, welches am besten im Namespace vom cert-manager erstellt wird.

Zu erst erstellen wir das Secret mit der CA und dem Secret

```
kubectl -n cert-manager create secret generic tuxnet-ca --from-file=tls.crt --from-file=tls.key --dry-run=client -o yaml > tuxnetlan-ca.yml
kubectl apply -f tuxnetlan-ca.yml
```

Nun können wir das passende ClusterIssuer Objekt mit verweis auf das Secret anlegen

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: tuxnetlan-clusterissuer
spec:
  ca:
    secretName: tuxnet-ca
```

## Erstellen von Certificate Objekte

Ein Certificate Objekte sorgt nun dafür das ein passendes Zertifikat als secret Objekt im korrekten Namespace erstellt wird.

```
apiVersion: cert-manager.io/v1
```

```
kind: Certificate
metadata:
  name: argocd-cert
  namespace: argocd
spec:
  secretName: argocd-server-tls
  issuerRef:
    name: tuxnetlan-clusterissuer
    kind: ClusterIssuer
  dnsNames:
    - argocd.tuxnet.lan
```

Oder

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: pdftools-cert
  namespace: stirlingpdf-production
spec:
  secretName: stirlingpdf-tls
  issuerRef:
    name: tuxnetlan-clusterissuer
    kind: ClusterIssuer
  dnsNames:
    - pdftools.tuxnet.lan
```

## Neuladen von Apps auf Kubernetes

Wir müssen die jeweilige Anwendung im Kubernetes Cluster jedes Mal neu laden, wenn sich das TLS Secret durch den cert-manager ändert.

Darum kann sich zum Beispiel der Stakater Reloader kümmern. Der muss aber erstmal mittels Helm in unserem Cluster ausgerollt werden.

```
$ helm repo add stakater https://stakater.github.io/stakater-charts
$ helm install application-reloader stakater/reloader -n cert-manager
```

Um den automatischen Neustart des Pods mit Stakater Reloader zu ermöglichen, müssen wir das Deployment der jeweiligen Anwendung mit `secret.reloader.stakater.com/reload` **annotieren**. Die Annotation sollte den Namen vom TLS Secret enthalten, der das Neuladen der Anwendung auslöst.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: stirlingpdf
  annotations:
    secret.reloader.stakater.com/reload: "stirlingpdf-tls"
```

*spec* :

[ ... ]

From:

<https://www.cooltux.net/> - TuxNet DokuWiki

Permanent link:

<https://www.cooltux.net/doku.php?id=it-wiki:kubernetes:cert-manager&rev=1729582246>

Last update: **2024/10/22 07:30**

