

Release H: Cert-Manager

- [Introduction](#)
- [Setup](#)
 - [Install](#)
 - [Create Issuer](#)
 - [Create Certificate](#)
- [CA injector](#)
- [Kafka](#)
- [Prometheus & Grafana](#)
- [Links](#)

Introduction

cert-manager provides X.509 certificate management on Kubernetes.

Setup

Install

Install cert-manager on your cluster by following the instruction in the link below.

You can use following command: `kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.11.0/cert-manager.yaml`

Once started you should see the following 3 pods running:

cert-manager					
\$ kubectl get pods -n cert-manager					
NAME	READY	STATUS	RESTARTS	AGE	
cert-manager-5b65cb968c-d2zbv	1/1	Running	0	5h46m	
cert-manager-cainjector-56b88bcd7-7gbj6	1/1	Running	0	5h46m	
cert-manager-webhook-c784c79c7-6d57m	1/1	Running	0	5h46m	

Create Issuer

Create a cluster-issuer and a certificate/secret for the self signed root CA

ClusterIssuer

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: selfsigned-rootca-cluster-issuer
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: selfsigned-rootca
  namespace: default
spec:
  isCA: true
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  commonName: selfsigned-rootca
  subject:
    organizations:
      - oran
    organizationalUnits:
      - oran
    countries:
      - Ireland
    localities:
      - Dublin
    streetAddresses:
      - Main Street
  secretName: cm-cluster-issuer-rootca-secret
  privateKey:
    rotationPolicy: Always
    algorithm: RSA
    encoding: PKCS1
    size: 2048
  issuerRef:
    name: selfsigned-rootca-cluster-issuer
    kind: ClusterIssuer
    group: cert-manager.io
  dnsNames:
    - localhost
    - minikube
  ipAddresses:
    - 127.0.0.1
    - 192.168.49.2
  emailAddresses:
    - ca@mail.com
```

Create an issuer for the self signed root CA

Issuer

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: cm-ca-issuer
  namespace: default
spec:
  ca:
    secretName: cm-cluster-issuer-rootca-secret
```

Create Certificate

Create a server key/certificate/keystore/truststore

server

```
apiVersion: v1
kind: Secret
metadata:
  name: cm-keycloak-jwk-pw
  namespace: default
type: Opaque
data:
  password: Y2hhbmdlaXQ=
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: keycloak-server-cert
  namespace: default
spec:
  secretName: cm-keycloak-server-certs
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  subject:
    organizations:
      - oran
    organizationalUnits:
      - oran
    countries:
      - IE
    localities:
      - Dublin
    streetAddresses:
      - Main Street
  commonName: keycloak
  isCA: false
  keystores:
    jks:
      create: true
      passwordSecretRef:
        name: cm-keycloak-jwk-pw
        key: password
  privateKey:
    algorithm: RSA
    encoding: PKCS1
    size: 2048
  usages:
    - server auth
  dnsNames:
    - keycloak.default
    - keycloak
    - keycloak.est.tech
  emailAddresses:
    - server@mail.com
  issuerRef:
    name: cm-ca-issuer
    kind: Issuer
    group: cert-manager.io
```

This certificate creates a secret "cm-keycloak-server-certs" containing 5 data items: tls.key (private key), tls.crt (Corresponding certificate), ca.crt (CA certificate), keystore.jks (keystore) and truststore.jks (truststore)

The keystore and truststore can be used to start keycloak over https.

Create a client key/certificate

client

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: keycloak-client-cert
  namespace: default
spec:
  secretName:
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  subject:
    organizations:
      - oran
    organizationalUnits:
      - oran
    countries:
      - IE
    localities:
      - Dublin
    streetAddresses:
      - Main Street
  commonName: keycloak
  isCA: false
  privateKey:
    algorithm: RSA
    encoding: PKCS1
    size: 2048
  usages:
    - client auth
  dnsNames:
    - keycloak.default
    - keycloak
    - keycloak.est.tech
  emailAddresses:
    - client@mail.com
  issuerRef:
    name: cm-ca-issuer
    kind: Issuer
    group: cert-manager.io
```

This certificate creates a secret "cm-keycloak-client-certs" containing 3 data items: tls.key (private key), tls.crt (Corresponding certificate) and ca.crt (CA certificate)

These certs can be used to communicate with the keycloak server over https.

Note: email addresses appear in the subject's alternative name rather than the distinguished name

CA injector

cainjector is used to configure the CA certificates for Mutating Webhooks - see link below.

Kafka

You can use your own certificates and keys with Kafka by adding the following section in your configuration:

clusterCa

```
clusterCa:
  generateCertificateAuthority: false
```

If this is set to false you need to setup your own cluster secrets containing the keys and certificates prior to starting the cluster.

Please refer to [8.3. Installing your own CA certificates](#)

You can do the same for the client CA.

For the listeners you can also configure your own server certificates in the `brokerCertChainAndKey` section.

brokerCertChainAndKey

```
- name: external
  port: 9098
  type: nodeport
  tls: true
  authentication:
    type: tls
  configuration:
    brokerCertChainAndKey:
      secretName: cm-kafka-server-certs
      certificate: tls.crt
      key: tls.key
    bootstrap:
      alternativeNames:
        - localhost
        - 192.168.49.2
        - my-cluster-kafka-external-bootstrap.kafka
        - my-cluster-kafka-external-bootstrap.kafka.svc
        - my-cluster-kafka-external-0.kafka
        - my-cluster-kafka-external-0.kafka.svc
```

Please refer to the [brokerCertChainAndKey](#) section in the Strimzi documentation for more information.

Prometheus & Grafana

cert-manager provides a metrics endpoint which can be scraped by Prometheus.

scrape config

```
scrape_configs:
- job_name: cert-manager-job
  metrics_path: /metrics
  scheme: http
  static_configs:
  - targets: ['cert-manager.cert-manager:9402']
```

Once the collection starts you can view these metrics in Prometheus.

☐ Use local time

☐ Enable query history

☒ Enable autocomplete

☐ Use experimental editor

☒ Enable highlighting

☒ Enable linter

Q

{job=~".*.cert.*"}

⌵

⌵

Execute

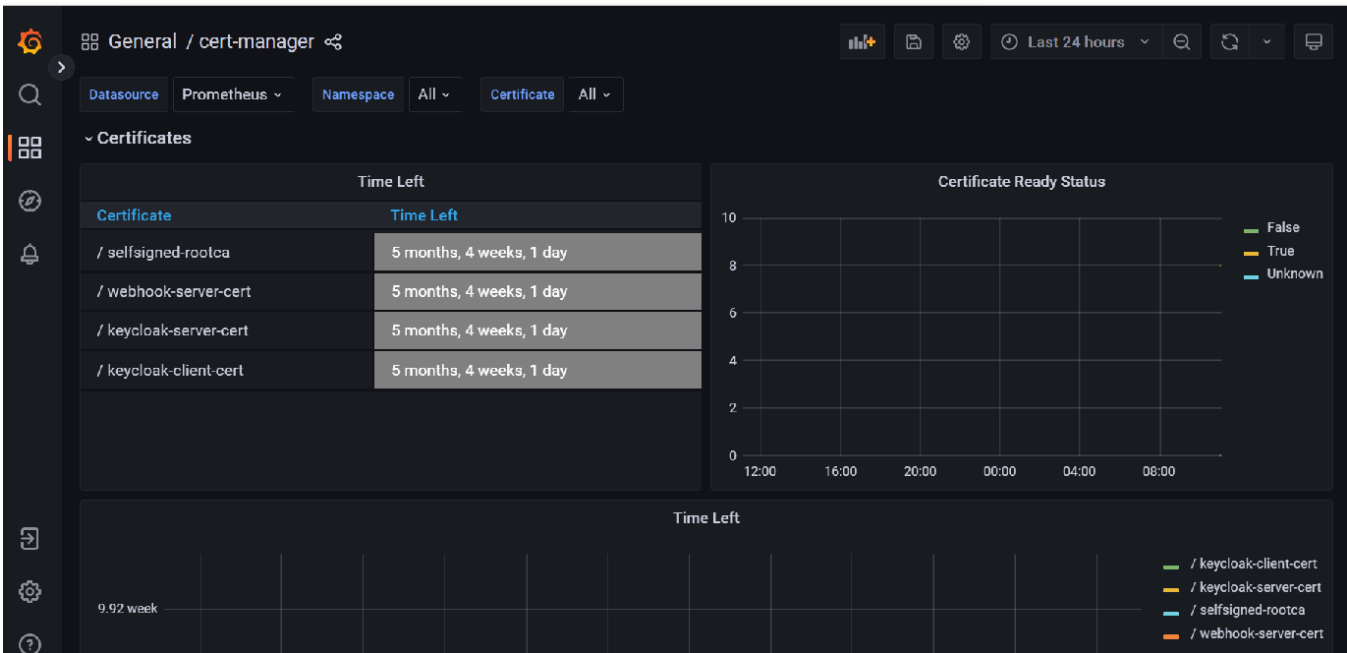
TableGraph

Load time: 19ms Resolution: 14s Result series: 43

<Evaluation time>

certmanager_certificate_expiration_timestamp_seconds(instance="cert-manager:cert-manager:9402", issuer_group="cert-manager.io", issuer_kind="ClusterIssuer", issuer_name="selfsigned-rootca-cluster-issuer", job="cert-manager-job", name="selfsigned-rootca", namespace="default")	1689633862
certmanager_certificate_expiration_timestamp_seconds(instance="cert-manager:cert-manager:9402", issuer_group="cert-manager.io", issuer_kind="Issuer", issuer_name="cm-ca-issuer", job="cert-manager-job", name="keycloak-client-cert", namespace="default")	1689633867
certmanager_certificate_expiration_timestamp_seconds(instance="cert-manager:cert-manager:9402", issuer_group="cert-manager.io", issuer_kind="Issuer", issuer_name="cm-ca-issuer", job="cert-manager-job", name="keycloak-server-cert", namespace="default")	1689633867
certmanager_certificate_expiration_timestamp_seconds(instance="cert-manager:cert-manager:9402", issuer_group="cert-manager.io", issuer_kind="Issuer", issuer_name="cm-ca-issuer", job="cert-manager-job", name="webhook-server-cert", namespace="default")	1689633867
certmanager_certificate_ready_status(condition="false", instance="cert-manager:cert-manager:9402", issuer_group="cert-manager.io", issuer_kind="ClusterIssuer", issuer_name="selfsigned-rootca-cluster-issuer", job="cert-manager-job", name="selfsigned-rootca", namespace="default")	0
certmanager_certificate_ready_status(condition="false", instance="cert-manager:cert-manager:9402", issuer_group="cert-manager.io", issuer_kind="Issuer", issuer_name="cm-ca-issuer", job="cert-manager-job", name="keycloak-client-cert", namespace="default")	0
certmanager_certificate_ready_status(condition="false", instance="cert-manager:cert-manager:9402", issuer_group="cert-manager.io", issuer_kind="Issuer", issuer_name="cm-ca-issuer", job="cert-manager-job", name="keycloak-server-cert", namespace="default")	0

Grafana also provides a dashboard for these metrics: [cert-manager dashboard](#)



Links

[Installation](#)

[Issuer](#)

[SelfSigned](#)

[trust-manager](#)

[Github trust-manager](#)

[Certificate Resources](#)

[API Reference](#)

[Istio Integration](#)

[CA Injector](#)

[Prometheus Metrics](#)