cloudflight

CertOps

Certificate Lifecycle Management done Cloud Native

23. Cloud Native Computing Linz Meetup - May 30, 2023

Martin Strigl

cloudflight

Ehlo!

Martin Strigl

Master of disaster (internal IT + SRE)
Not afraid of touching php
Using linux container technics since 2010

Why this talk ?

- I was asked if I have an idea covering the areas cloud native and security
- Thinking about it I realized that quite some non necessary problems in the last years derived from certificate handling or to be more specific from missing and/or manual certificate lifecycle management
- This good read also showed that I' not alone: https://venafi.com/blog/why-manual-certificate-management-really-hurts/

Picture Source: https://p1.pxfuel.com/preview/382/198/335/grinding-maintenance-labor-work.jpg

Picture Source:https://upload.wikimedia.org/wikipedia/commons/9/99/ABB_Schweissroboter_Technisches_Museum_Wien_Februar_2013_File2.JPG

cloudflight

Topics covered

01

History & Terms used

02

The/A Solution

03

The Howto get running

04

The Main Use Cases

05

The creative Solutions

06

Showtime

History & Terms used

Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011 (RFC 6176)
SSL 3.0	1996	Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	Deprecated in 2021 (RFC 8996)
TLS 1.1	2006	Deprecated in 2021 (RFC 8996)
TLS 1.2	2008	In use since 2008
TLS 1.3	2018	In use since 2018

- **Digital certificates**
  - A digital certificate certifies the ownership of a public key by the named subject of the certificate, and indicates certain expected usages of that key. This allows others (relying parties) to rely upon signatures or on assertions made by the private key that corresponds to the certified public key
  - As a consequence of choosing X.509 certificates, certificate authorities and a public key infrastructure are necessary to verify the relation between a certificate and its owner, as well as to generate, sign, and administer the validity of certificates
  - https://en.wikipedia.org/wiki/Transport_Layer_Security#SSL_1.0,_2.0,_and_3.0
  - https://en.wikipedia.org/wiki/X.509

- **2012**
  - The Let's Encrypt project was started in 2012 by two Mozilla employees, Josh Aas and Eric Rescorla, together with Peter Eckersley at the Electronic Frontier Foundation and J. Alex Halderman at the University of Michigan. Internet Security Research Group, the company behind Let's Encrypt, was incorporated in May 2013
- **2015**
  - On January 28, 2015, the ACME protocol was officially submitted to the IETF for standardization. On April 9, 2015, the ISRG and the Linux Foundation declared their collaboration. The root and intermediate certificates were generated in the beginning of June. On June 16, 2015, the final launch schedule for the service was announced, with the first certificate expected to be issued sometime in the week of July 27, 2015, followed by a limited issuance period to test security and scalability. General availability of the service was originally planned to begin sometime in the week of September 14, 2015.
  - On October 19, 2015, the intermediate certificates became cross-signed by IdenTrust, causing all certificates issued by Let's Encrypt to be trusted by all major browsers
- https://en.wikipedia.org/wiki/Let%27s_Encrypt

cloudflight

Topics covered

01

History & Terms used

02

The/A Solution

03

The Howto get running

04

The Main Use Cases

05

The creative Solutions

06

Showtime

The/A Solution

- **cert-manager.io**
  - cert-manager is a powerful and extensible X.509 certificate controller for Kubernetes and OpenShift workloads. It will obtain certificates from a variety of Issuers, both popular public Issuers as well as private Issuers, and ensure the certificates are valid and up-to-date, and will attempt to renew certificates at a configured time before expiry.
  - Really good documentation at: https://cert-manager.io/docs/

The/A Solution

The/A Solution

cert-manager is a CNCF member project

CLOUD NATIVE COMPUTING FOUNDATION

cert-manager was created by JETSTACK

It was proudly donated to CNCF in 2020.

KEY FEATURES

- Automated issuance and renewal of certificates to secure Ingress with TLS
- Fully integrated Issuers from recognised public and private Certificate Authorities
- Secure pod-to pod communication with mTLS using private PKI Issuers
- Supports certificate use cases for web facing and internal workloads
- Open source add-ons for enhanced cloud native service mesh security
- Backed by major cloud service providers and distributions

cloudflight

Topics covered

01

History & Terms used

02

The/A Solution

03

The Howto get running

04

The Main Use Cases

05

The creative Solutions

06

Showtime

The Howto get running

Release	Release Date	End of Life	Supported Kubernetes versions	Supported OpenShift versions
1.12	May 19, 2023	End of September, 2024	1.22 → 1.27	4.9 → 4.14
1.11	Jan 11, 2023	Release of 1.13	1.21 → 1.27	4.8 → 4.14

- **Cloud Compatibility**
  - See https://cert-manager.io/docs/installation/compatibility/
- **Possible ways to install**
  - kubectl apply (https://cert-manager.io/docs/installation/kubectl/)
    - kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.12.0/cert-manager.yaml
  - Helm (https://cert-manager.io/docs/installation/helm/)
    - helm repo add jetstack https://charts.jetstack.io
    - helm repo update
    - helm install cert-manager jetstack/cert-manager --namespace cert-manager --create-namespace --version v1.12.0 --set installCRDs=true
  - Operatorhub - OLM (https://cert-manager.io/docs/installation/operator-lifecycle-manager/)
    - operator-sdk olm install
    - kubectl krew install operator
    - kubectl operator install cert-manager -n operators --channel stable --approval Automatic
    - kubectl get events -w -n operators
    - Limited configuration can be done via editing subscription and/or ClusterServiceVersion

The Howto get running

- **Issuers**
  - Issuer
    - Is a namespaced ressource
    - Cannot be used from different namespace
  - ClusterIssuer
    - Basically the same as Issuer - just no namespacing and therefore can be used to issue certificates across all namespaces
    - If you reference a secret from a ClusterIssuer this secret has to exist in the Cluster Resource Namespace of cert-manager (default: cert-manager, can be changed by flag --cluster-resource-namespace of cert-manager controller)
- **Issuer Types**
  - SelfSigned
  - CA
  - Vault
  - Venafi
  - External (https://cert-manager.io/docs/configuration/external/)
    - e.g.: step-issuer (https://github.com/smallstep/step-issuer)
  - ACME (Automated Certificate Management Environment)
    - HTTP01
    - DNS01: ACMEDNS / Akamai / AzureDNS / Cloudflare / DigitalOcean / Google ClouDNS / RFC-2136 / Route53 / Webhook

The Howto get running

- **Example Issuer using Route53 and LE**

```
[mstrigl@PCATLNZPC0MAB1G cert-manager]$ cat cm-le-prod-route53.yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: cm-clusterissuer-letsencrypt-prod-dns01-route53
spec:
  acme:
    email: martin.strigl@cloudflight.io
    preferredChain: ISRG Root X1
    privateKeySecretRef:
      key: tls.key
      name: cm-clusterissuer-letsencrypt-prod-dns01-route53-account-secret
    server: 'https://acme-v02.api.letsencrypt.org/directory'
    solvers:
      - dns01:
          route53:
            region: eu-central-1
            accessKeyID: AKXXXXXXXXXXXXXXXXX
            hostedZoneID: Z0XXXXXXXXXXXXXXXXXX
            secretAccessKeySecretRef:
              name: cm-clusterissuer-letsencrypt-prod-dns01-route53-credentials
              key: secretKey
        selector:
          dnsZones:
            - mstdemo.openshift-sandbox.
```

The Howto get running

- **Example Issuer using ACMEDNS and LE**

```
[mstrigl@PCATLNZPC0MAB1G cert-manager]$ cat cm-le-prod-acmedns.yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: cm-clusterissuer-letsencrypt-prod-dns01
spec:
  acme:
    email: martin.strigl@cloudflight.io
    preferredChain: ISRG Root X1
    privateKeySecretRef:
      key: tls.key
      name: cm-clusterissuer-letsencrypt-prod-dns01-account-secret
    server: 'https://acme-v02.api.letsencrypt.org/directory'
    solvers:
      - dns01:
          acmeDNS:
            accountSecretRef:
              key: acmedns.json
              name: cm-clusterissuer-letsencrypt-prod-dns01-acmedns-secret
            host: 'https://acme-dns.xxxxxxxxx.xx'
          cnameStrategy: Follow
        selector:
          dnsZones:
            - demo.openshift-sandbox.[redacted]
```

```
[mstrigl@PCATLNZPC0MAB1G cert-manager]$ cat acmedns.json
{
        "demo.openshift-sandbox.[redacted]": {
            "username": "af6ba2f8-[redacted]",
            "password": "[redacted]",
            "fulldomain": "5c4c21a9-4f2c-[redacted]",
            "subdomain": "5c4c21a9-4f2c-[redacted]",
            "allowfrom": []
        }
}
```

cloudflight

Topics covered

01

History & Terms used

02

The/A Solution

03

The Howto get running

04

The Main Use Cases

05

The creative Solutions

06

Showtime

The Main Use Cases

- **Securing Ingress**
  - For individual hostnames

```
[mstrigl@PCATLNZPC0MAB1G cert-manager]$ cat ingress-mst-demo-httpd-le-prod-route53.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    cert-manager.io/duration: 4320h
    cert-manager.io/issuer: cm-clusterissuer-letsencrypt-prod-dns01-route53
    cert-manager.io/issuer-group: cert-manager.io
    cert-manager.io/issuer-kind: ClusterIssuer
    cert-manager.io/renew-before: 2160h
    route.openshift.io/termination: edge
  name: mst-demo-httpd-le-prod-route53
  namespace: mst-demo
spec:
  tls:
    - hosts:
        - demo1.mstdemo.openshift-sandbox.[redacted]
      secretName: ingress-mst-demo-httpd-le-prod-route53
  rules:
  - host: demo1.mstdemo.openshift-sandbox.[redacted]
    http:
      paths:
      - backend:
          service:
            name: mst-demo-httpd
            port:
              number: 80
        path: /
        pathType: Prefix
```

The Main Use Cases

- **Securing Ingress**
  - With a wildcard record certificate and reusing that cert in an ingress ressource
  - Wildcard cert could also be used as default certificate for ingress controller

```
[mstrigl@PCATLNZPC0MAB1G cert-manager]$ cat cert-mst-demo-plain10.yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: mst-demo-plain10
  namespace: mst-demo
spec:
  secretName: secret-mst-demo-plain10-tls
  secretTemplate:
    annotations:
      my-secret-annotation-1: "foo"
      my-secret-annotation-2: "bar"
    labels:
      my-secret-label: foo
  duration: 2160h # 60d
  renewBefore: 360h # 15d
  subject:
    organizations:
      - cloudflight
  isCA: false
  privateKey:
    algorithm: RSA
    encoding: PKCS1
    size: 2048
  usages:
    - server auth
    - client auth
  # At least one of a DNS Name, URI, or IP address is required.
  dnsNames:
    - "*.demo.openshift-sandbox.[redacted]"
  issuerRef:
    name: cm-clusterissuer-letsencrypt-staging-dns01
    kind: ClusterIssuer
    group: cert-manager.io
```

```
[mstrigl@PCATLNZPC0MAB1G cert-manager]$ cat ingress-reusing-cm-cert.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    route.openshift.io/termination: edge
  name: mst-demo-httpd
  namespace: mst-demo
spec:
  tls:
  - hosts:
    - mst-demo-httpd.demo.openshift-sandbox.[redacted]
    secretName: secret-mst-demo-plain10-tls
  rules:
  - host: mst-demo-httpd.demo.openshift-sandbox.[redacted]
    http:
      paths:
      - backend:
          service:
            name: mst-demo-httpd
            port:
              number: 80
        path: /
        pathType: Prefix
```

The Main Use Cases

- **Securing Pod Communication by establishing mTLS through csi-driver**
  - The goal for this plugin is to seamlessly request and mount certificate key pairs to pods. This is useful for facilitating mTLS, or otherwise securing connections of pods with guaranteed present certificates whilst having all of the features that cert-manager provides.
  - Needs to be installed
    - helm repo add jetstack https://charts.jetstack.io --force-update
    - helm upgrade -i -n cert-manager cert-manager-csi-driver jetstack/cert-manager-csi-driver –wait
  - Must not be used with public CA's since you will quite fast hit their rate limits
    - Either use Issue Types CA or SelfSigned or External (e.g.: step-issuer)

```
apiVersion: v1
kind: Pod
metadata:
  name: my-csi-app
  namespace: sandbox
  labels:
    app: my-csi-app
spec:
  containers:
    - name: my-frontend
      image: busybox
      volumeMounts:
      - mountPath: "/tls"
        name: tls
      command: [ "sleep", "1000000" ]
  volumes:
    - name: tls
      csi:
        driver: csi.cert-manager.io
        volumeAttributes:
              csi.cert-manager.io/issuer-name: ca-issuer
              csi.cert-manager.io/dns-names: ${POD_NAME}.${POD_NAMESPACE}.svc.cluster.local
```

cloudflight

Topics covered

01

History & Terms used

02

The/A Solution

03

The Howto get running

04

The Main Use Cases

05

The creative Solutions

06

Showtime

The creative Solutions

- **Own PKI**
  - CA
    - Can be used for mTLS by csi driver
    - Can be used for already established private Company PKI
    - Has the "disadvantage" that the (intermediate) CA certificate and private key is accessible to some persons with enough permissions
  - StepIssuer
    - Can be used for mTLS by csi driver
    - Can be used for already established private Company PKI
    - Has the advantage that (intermediate) CA certificate and private key is NOT directly accessible since communication to step ca service can be encapsulated by using foreign endpoint (step ca is not running inside the cluster)
    - See
      - https://github.com/smallstep/step-issuer
      - https://github.com/smallstep/certificates

- **Monitoring**
  - Regular

```yaml
spec:
  template:
    spec:
      containers:
        - name: cert-manager-controller
          ports:
            - containerPort: 9402
              name: http
              protocol: TCP
```

```yaml
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: cert-manager
  namespace: cert-manager
  labels:
    app: cert-manager
    app.kubernetes.io/name: cert-manager
    app.kubernetes.io/instance: cert-manager
    app.kubernetes.io/component: "controller"
spec:
  jobLabel: app.kubernetes.io/name
  selector:
    matchLabels:
      app: cert-manager
      app.kubernetes.io/name: cert-manager
      app.kubernetes.io/instance: cert-manager
      app.kubernetes.io/component: "controller"
  podMetricsEndpoints:
    - port: http
      honorLabels: true
```

  - https://gitlab.com/uneeq-oss/cert-manager-mixin

The creative Solutions

- **Delegated Domains**

```yaml
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  ...
spec:
  acme:
    ...
    solvers:
    - selector:
        dnsZones:
        - 'example.com'
      dns01:
        # Valid values are None and Follow
        cnameStrategy: Follow
        route53:
          region: eu-central-1
          accessKeyID: <Access ID for less-privileged.example.org here>
          hostedZoneID: <Zone ID for less-privileged.example.org here>
          secretAccessKeySecretRef:
            ...
```

```
_acme-challenge.example.com     IN  CNAME   _acme-challenge.less-privileged.example.org.
_acme-challenge.www.example.com IN  CNAME   _acme-challenge.less-privileged.example.org.
_acme-challenge.foo.example.com IN  CNAME   _acme-challenge.less-privileged.example.org.
_acme-challenge.bar.example.com IN  CNAME   _acme-challenge.less-privileged.example.org.
```

```go
// Present performs the work to configure DNS to resolve a DNS01 challenge.
func (s *Solver) Present(ctx context.Context, issuer v1.GenericIssuer, ch *cmacme.Challenge) error {

    slv, providerConfig, err := s.solverForChallenge(ctx, issuer, ch)

    fqdn, err := util.DNS01LookupFQDN(ch.Spec.DNSName, followCNAME(providerConfig.CNAMEStrategy), s.DNS01Nameservers...)

    return slv.Present(ch.Spec.DNSName, fqdn, ch.Spec.Key)
}
```

```go
// Present creates a TXT record using the specified parameters
func (r *DNSProvider) Present(domain, fqdn, value string) error {
    value = `"` + value + `"`
    return r.changeRecord(route53.ChangeActionUpsert, fqdn, value, route53TTL)
}
```

```go
// Present creates a TXT record to fulfil the dns-01 challenge
func (c *DNSProvider) Present(domain, fqdn, value string) error {
    if account, exists := c.accounts[domain]; exists {
        // Update the acme-dns TXT record.
        return c.client.UpdateTXTRecord(account, value)
    }

    return fmt.Errorf("account credentials not found for domain %s", domain)
}
```

The creative Solutions

- **Approver Policy**

```
helm upgrade cert-manager jetstack/cert-manager \
  --install \
  --create-namespace \
  --namespace cert-manager \
  --version REPLACE-WITH-YOUR-CERT-MANAGER-VERSION \
  --set installCRDs=true \
  --set extraArgs={--controllers='*\,-certificaterequests-approver'}
```

```
$ helm upgrade -i -n cert-manager cert-manager-approver-policy jetstack/cert-manager-approver-p
  --set app.approveSignerNames="{\
issuers.cert-manager.io/*,clusterissuers.cert-manager.io/*,\
googlecasclusterissuers.cas-issuer.jetstack.io/*,googlecasissuers.cas-issuer.jetstack.io/*,\
awspcaclusterissuers.awspca.cert-manager.io/*,awspcaissuers.awspca.cert-manager.io/*\
}"
```

```yaml
apiVersion: policy.cert-manager.io/v1alpha1
kind: CertificateRequestPolicy
metadata:
  name: test-policy
spec:
  allowed:
    commonName:
      value: "hello.world"
      required: true
  selector:
    # Select all IssuerRef
    issuerRef: {}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cert-manager-policy:hello-world
rules:
  - apiGroups: ["policy.cert-manager.io"]
    resources: ["certificaterequestpolicies"]
    verbs: ["use"]
    # Name of the CertificateRequestPolicies to be used.
    resourceNames: ["test-policy"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cert-manager-policy:hello-world
roleRef:
# ClusterRole or Role _must_ be bound to a user for the policy to be considered.
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cert-manager-policy:hello-world
subjects:
# The users who should be bound to the policies defined.
# Note that in the case of users creating Certificate resources, cert-manager
# is the entity that is creating the actual CertificateRequests, and so the
# cert-manager controller's
# Service Account should be bound instead.
- kind: Group
  name: system:authenticated
  apiGroup: rbac.authorization.k8s.io
```

The creative Solutions

- **Webhook Issuer**
  - See https://github.com/topics/cert-manager-webhook
  - e.g. Hetzner: https://github.com/vadimkim/cert-manager-webhook-hetzner
    - helm repo add cert-manager-webhook-hetzner https://vadimkim.github.io/cert-manager-webhook-hetzner
    - helm install --namespace cert-manager cert-manager-webhook-hetzner cert-manager-webhook-hetzner/cert-manager-webhook-hetzner --set groupName=acme.yourdomain.tld

```yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-staging
spec:
  acme:
    # The ACME server URL
    server: https://acme-staging-v02.api.letsencrypt.org/directory

    # Email address used for ACME registration
    email: mail@example.com # REPLACE THIS WITH YOUR EMAIL!!!

    # Name of a secret used to store the ACME account private key
    privateKeySecretRef:
      name: letsencrypt-staging

    solvers:
      - dns01:
          webhook:
            # This group needs to be configured when installing the helm package, otherwise the webh
            groupName: acme.yourdomain.tld
            solverName: hetzner
            config:
              secretName: hetzner-secret
              zoneName: example.com # (Optional): When not provided the Zone will searched in Hetzne
              apiUrl: https://dns.hetzner.com/api/v1
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: hetzner-secret
  namespace: cert-manager
type: Opaque
data:
  api-key: your-key-base64-encoded
```

The creative Solutions

- **Bring it together in tanka / jsonnet**

```
service: k.util.serviceFor(self.deployment),
ingress: k.util.ingressFor(self.service, $._config.grafana.host, port=4180)
         + k.networking.v1.ingress.metadata.withAnnotationsMixin({
           'cert-manager.io/issuer': 'step-cluster-issuer',
           'cert-manager.io/issuer-kind': 'StepClusterIssuer',
           'cert-manager.io/issuer-group': 'certmanager.step.sm',
           'cert-manager.io/duration': '4320h',
           'cert-manager.io/renew-before': '2160h',
         })
         + k.networking.v1.ingress.spec.withTls([
           {
             hosts: [$._config.grafana.host],
             secretName: $._config.grafana.name + '-tls',
           },
         ]),
```

```
autoApproveRole: {
  apiVersion: 'rbac.authorization.k8s.io/v1',
  kind: 'ClusterRole',
  metadata: {
    name: 'cert-manager-controller-approve:step-issuer',
  },
  rules: [
    {
      apiGroups: [
        'cert-manager.io',
      ],
      resources: [
        'signers',
      ],
      verbs: [
        'approve',
      ],
      resourceNames: [
        'stepclusterissuers.certmanager.step.sm/*',
        'stepissuers.certmanager.step.sm/*',
      ],
    },
  ],
},
autoApproveRoleBinding: {
  apiVersion: 'rbac.authorization.k8s.io/v1',
  kind: 'ClusterRoleBinding',
  metadata: {
    name: 'cert-manager-controller-approve:step-issuer',
  },
  roleRef: {
    apiGroup: 'rbac.authorization.k8s.io',
    kind: 'ClusterRole',
    name: 'cert-manager-controller-approve:step-issuer',
  },
  subjects: [
    {
      kind: 'ServiceAccount',
      name: 'cert-manager',
      namespace: 'openshift-cert-manager',
    },
  ],
},
```

```
local k = import '                    /prelude.libsonnet';
local secrets = import 'secrets.json';
local tk = import 'tk';

(import 'step-issuer.libsonnet')
+ {
  secret: k.core.v1.secret.new('ca01', {})
          + k.core.v1.secret.withStringData({
            password: secrets.provisionerPassword,
          }),
  clusterIssuer: {
    apiVersion: 'certmanager.step.sm/v1beta1',
    kind: 'StepClusterIssuer',
    metadata: {
      name: 'step-cluster-issuer',
    },
    spec: {
      caBundle: std.base64(|||
        -----BEGIN CERTIFICATE-----
        ...
        -----END CERTIFICATE-----
      |||),
      provisioner: {
        kid: 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
        name: 'operator@                ',
        passwordRef: {
          key: 'password',
          name: 'ca01',
          namespace: 'step-issuer',
        },
      },
      url: 'https://ca01.                ',
    },
  },
```

cloudflight

Topics covered

01

History & Terms used

02

The/A Solution

03

The Howto get running

04

The Main Use Cases

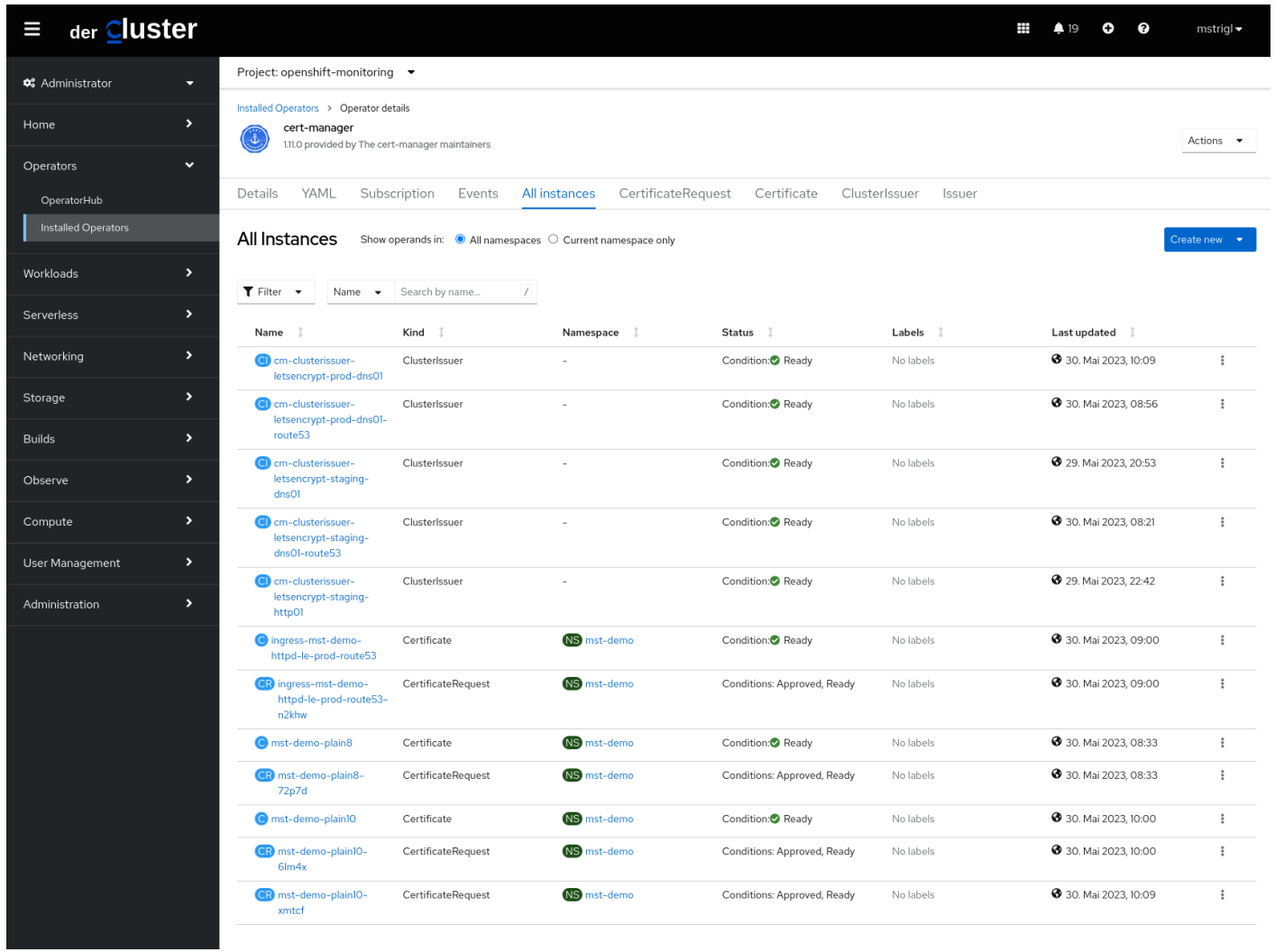
05

The creative Solutions

06

Showtime

Showtime


der cluster
19
mstrigl
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Serverless
Networking
Storage
Builds
Observe
Compute
User Management
Administration
Project: openshift-monitoring
Installed Operators > Operator details
cert-manager
1.11.0 provided by The cert-manager maintainers
Actions
Details YAML Subscription Events All instances CertificateRequest Certificate ClusterIssuer Issuer
All Instances
Show operands in: All namespaces Current namespace only
Create new
Filter Name Search by name... /
Name Kind Namespace Status Labels Last updated
cm-clusterissuer-letsencrypt-prod-dns01 ClusterIssuer - Condition: Ready No labels 30. Mai 2023, 10:09
cm-clusterissuer-letsencrypt-prod-dns01-route53 ClusterIssuer - Condition: Ready No labels 30. Mai 2023, 08:56
cm-clusterissuer-letsencrypt-staging-dns01 ClusterIssuer - Condition: Ready No labels 29. Mai 2023, 20:53
cm-clusterissuer-letsencrypt-staging-dns01-route53 ClusterIssuer - Condition: Ready No labels 30. Mai 2023, 08:21
cm-clusterissuer-letsencrypt-staging-http01 ClusterIssuer - Condition: Ready No labels 29. Mai 2023, 22:42
ingress-mst-demo-httpd-le-prod-route53 Certificate mst-demo Condition: Ready No labels 30. Mai 2023, 09:00
ingress-mst-demo-httpd-le-prod-route53-n2khw CertificateRequest mst-demo Conditions: Approved, Ready No labels 30. Mai 2023, 09:00
mst-demo-plain8 Certificate mst-demo Condition: Ready No labels 30. Mai 2023, 08:33
mst-demo-plain8-72p7d CertificateRequest mst-demo Conditions: Approved, Ready No labels 30. Mai 2023, 08:33
mst-demo-plain10 Certificate mst-demo Condition: Ready No labels 30. Mai 2023, 10:00
mst-demo-plain10-6lm4x CertificateRequest mst-demo Conditions: Approved, Ready No labels 30. Mai 2023, 10:00
mst-demo-plain10-xmtcf CertificateRequest mst-demo Conditions: Approved, Ready No labels 30. Mai 2023, 10:09

cloudflight

Thank you for your attention