

Securing Ingress Traffic Using the Cert-manager Operator



Nigel Brown

@n_brownuk www.windsock.io



Module Outline



Learn how to configure TLS for ingress routes

Become familiar with the cert-manager operator

Configure custom resource definitions for certificate management

Demonstrate how to automate certificate acquisition and renewal

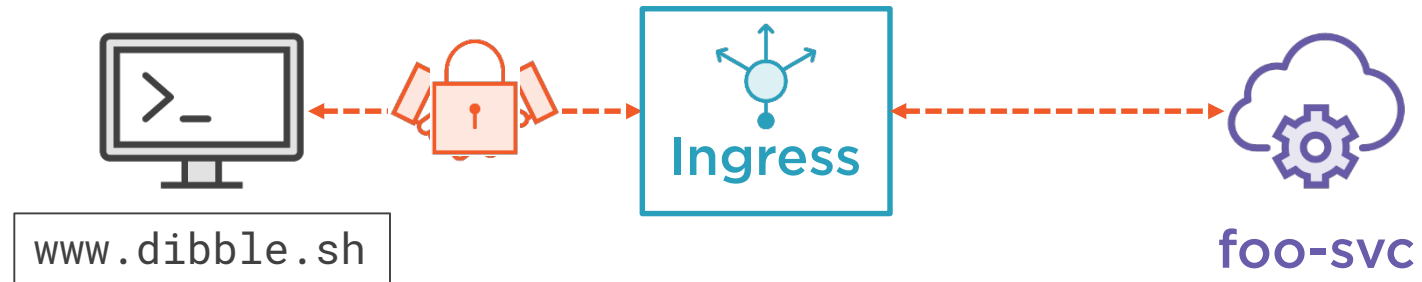


Secure Ingress

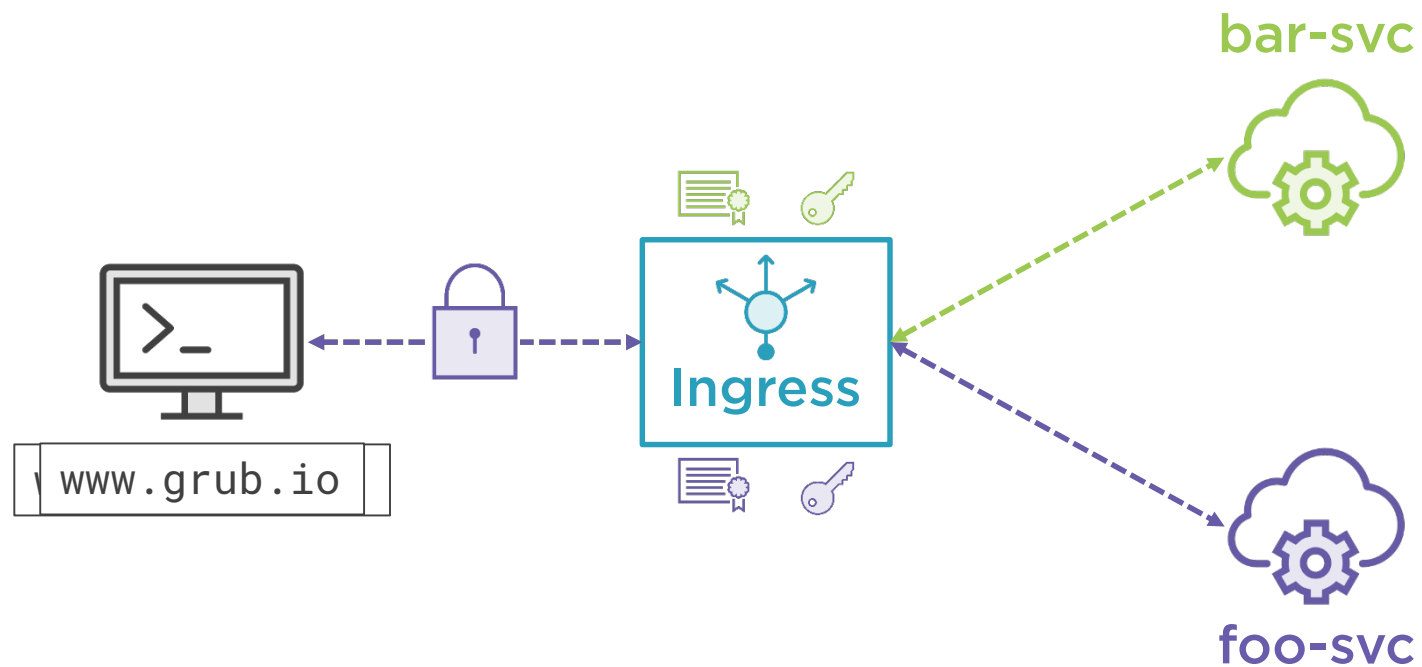
The Ingress API allows us to configure secure connections using Transport Layer Security (TLS)



Securing Connections with TLS



Securing Multiple Backend Services



```
spec:
  tls:
  - hosts:
    - dibble.sh
    secretName: dibble-sh-tls
  - hosts:
    - grub.io
    secretName: grub-io-tls
```

Configuring TLS for Ingress Traffic

The `tls` key defines one or more TLS configurations



TLS Secret Considerations



Secret must contain a PEM encoded certificate and a private key



Intermediate certificates must be bundled with public key certificate



TLS secrets are referenceable by ingress objects in same namespace



Managing X.509 Certificates

cert-manager

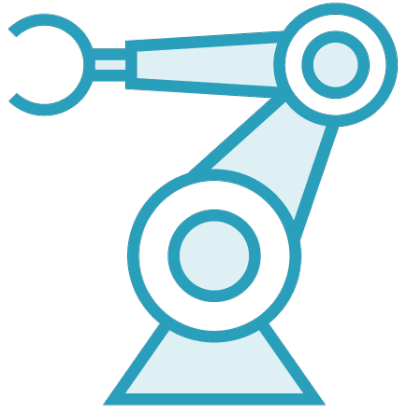


cert-manager

An application that automates the management of the lifecycle of TLS certificates within a Kubernetes cluster

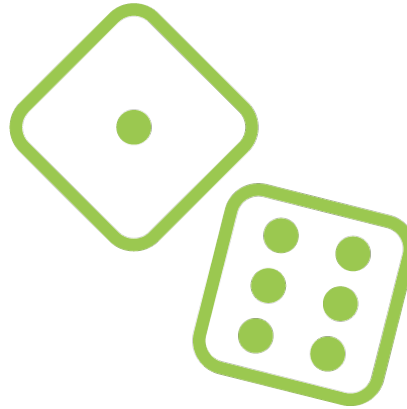


Benefits of Automated X.509 Certificates



Better Automation

Allows us to focus on other important tasks



Minimize Risk

Removing human element reduces risk

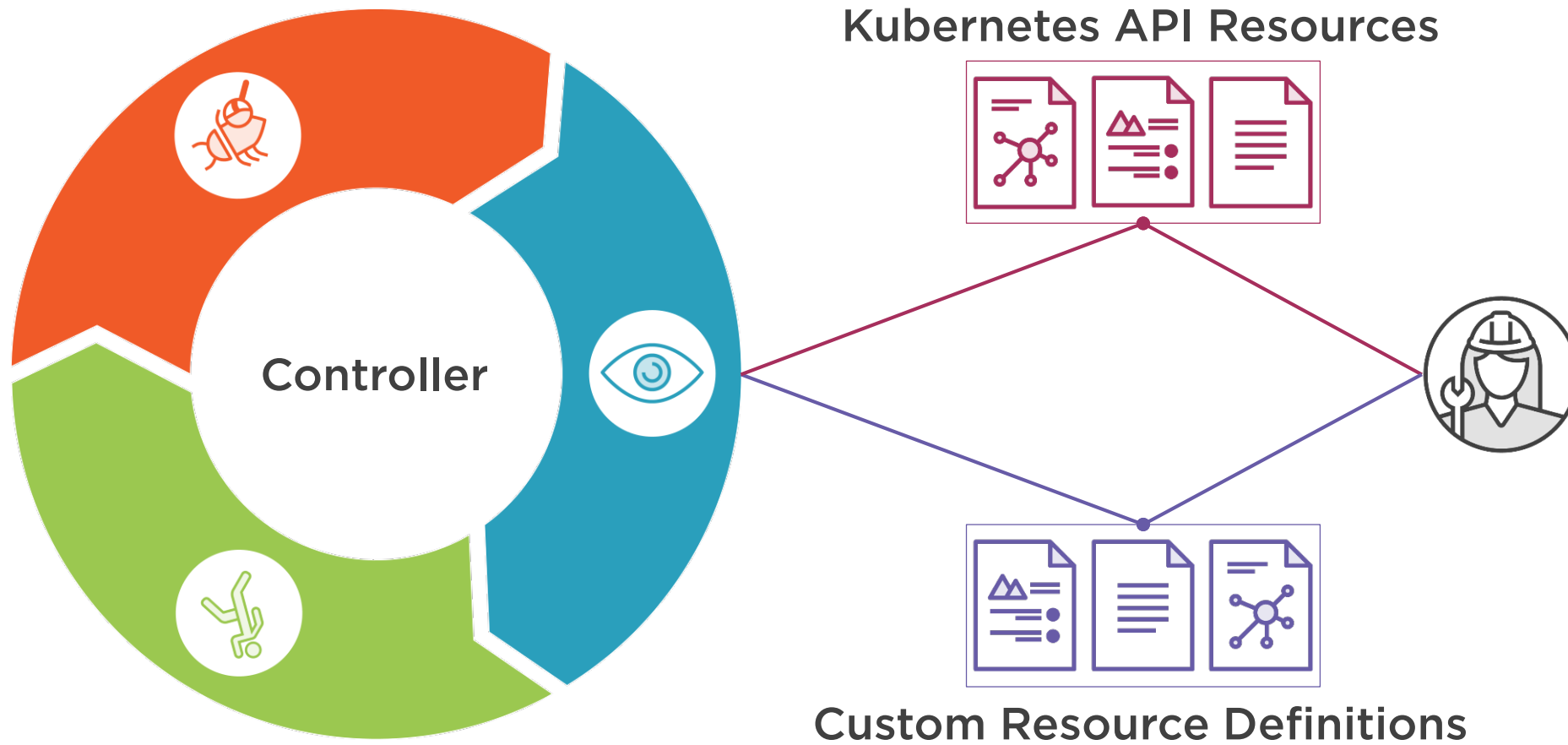


Reliable Outcomes

Increases confidence in achieving our goals



The Operator Pattern



Custom Resource Definitions

Issuer

Represents a certificate authority from which X.509 certificates can be obtained

CertificateRequest

Contains a PEM encoded certificate request which is sent to the CA issuer that it references

Challenge

Manages the lifecycle of the challenge presented by an ACME server for authorization

Certificate

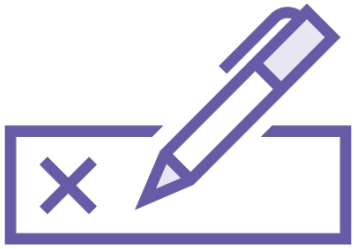
Defines the attributes of an X.509 certificate to be retrieved from the CA associated with an Issuer

Order

Used to manage the order process for an X.509 certificate placed on an ACME server



Issuer Types



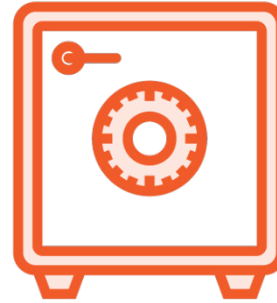
Self-Signing

Creates self-signed certificates



CA

Backed by a signing key pair



Vault

Uses Vault's PKI Secrets Engine



ACME

Defines a remote ACME server



```
apiVersion: cert-manager.io/v1alpha3
kind: Issuer
metadata:
  name: letsencrypt
```

Defining an Issuer

An Issuer object is scoped to a namespace ...

... whilst a ClusterIssuer is scoped cluster wide



```
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: bette@dibble.sh
    privateKeySecretRef:
      name: acme-letsencrypt
```

Configuring an ACME Issuer

The ACME challenge can be either http01 or dns01

- <https://letsencrypt.org/how-it-works/>



```
spec:  
  acme:  
    [...]  
    solvers:  
      - http01:  
        ingress:  
          class: nginx
```

Providing ACME Challenge Details

ACME challenges which use HTTP, rely on an ingress controller



ACME Account Registration

```
$ kubectl describe issuer letsencrypt
```

```
[...]
```

```
Status:
```

```
Acme:
```

```
Uri: https://acme-v02.api.letsencrypt.org/acme/acct/8292286
```

```
Conditions:
```

```
Last Transition Time: 2019-02-19T16:02:00Z
```

```
Message: The ACME account was registered with  
the ACME server
```

```
Reason: ACMEAccountRegistered
```

```
Status: True
```

```
Type: Ready
```



```
apiVersion: cert-manager.io/v1alpha3
kind: Certificate
metadata:
  name: dibble-sh
```

Defining a Certificate

A Certificate object is scoped to a namespace



Mapping Certificate to Issuer

```
apiVersion: cert-manager.io/v1alpha3
kind: ClusterIssuer
metadata:
  name: letsencrypt
spec:
  acme:
    server: https://acme .....
    email: bette@dibble.sh
    privateKeySecretRef:
      name: acme-letsencrypt
    solvers:
      - http01:
          ingress:
            class: nginx
```

Issuer Object

```
apiVersion: cert-manager.io/v1alpha3
kind: Certificate
metadata:
  name: dibble-sh
spec:
  secretName: dibble-sh-tls
  issuerRef:
    kind: ClusterIssuer
    name: letsencrypt
  dnsNames:
    - www.dibble.sh
```

Certificate Object



```
spec:  
  commonName: dibble.sh  
  dnsNames:  
  - www.dibble.sh  
  secretName: dibble-sh-tls  
  issuerRef:  
    name: letsencrypt  
    kind: ClusterIssuer
```

Configuring the Certificate Request

The issuerRef field assumes an Issuer kind ...

... and a ClusterIssuer kind must be explicitly defined



The ACME Challenge



ingress-shim

Watches Ingress objects, and for those that have the appropriate annotations, will automatically provision X.509 certificates



```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: nginxhello-ingress
  annotations:
    cert-manager.io/issuer: letsencrypt
```

Simplifying ACME Certificate Acquisition

Required annotation:

- `cert-manager.io/issuer`, or
- `cert-manager.io/cluster-issuer`



Certificate Renewals



A period of certificate validity can be specified when configuring a Certificate object (`duration`)



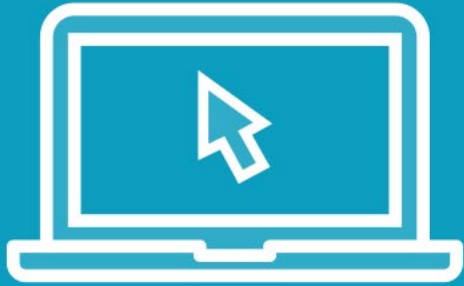
A renewal window can be defined by specifying time before expiration to attempt renewal (`renewBefore`)



Default certificate validity is 90 days, with renewals attempted 30 days before expiration



Demo



Deploy cert-manager to a cluster

Configure an ACME Issuer object for Let's Encrypt

Define a Certificate object

Set up automatic certificate acquisition using ingress annotations



Module Summary



Ingress supports configuration of secure routes

Cert-manager operator provides in-cluster certificate management

Certificate lifecycles can be automated with ingress and cert-manager

