KubeCon | CloudNativeCon

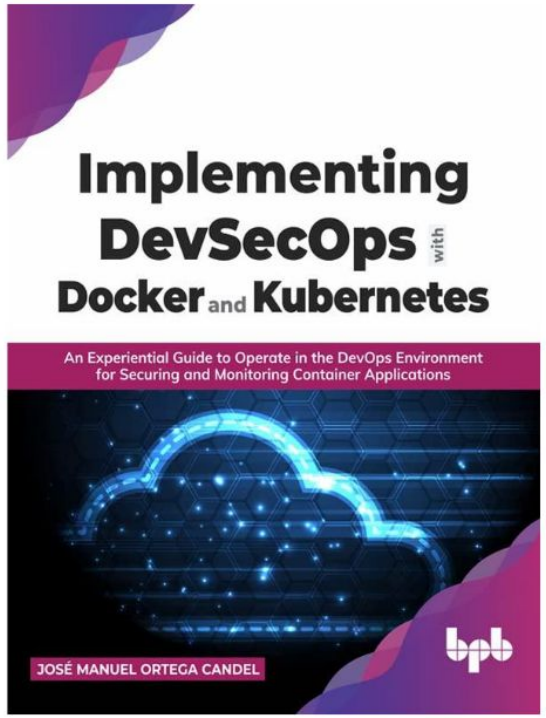
Europe 2022

WELCOME TO VALENCIA

KubeCon | CloudNativeCon
Europe 2022

Implementing cert-manager in K8s

Jose Manuel Ortega, Freelance



1. Getting Started with DevSecOps
2. Container Platforms
3. Managing Containers and Docker Images
4. Getting Started with Docker Security
5. Docker Host Security
6. Docker Images Security
7. Auditing and Analyzing Vulnerabilities in Docker Containers
8. Managing Docker Secrets and Networking
9. Docker Container Monitoring
10. Docker Container Administration
11. Kubernetes Architecture
12. Kubernetes Security
13. Auditing and Analyzing Vulnerabilities in Kubernetes
14. Observability and Monitoring in Kubernetes

KubeCon CloudNativeCon Europe 2022

Jose Manuel Ortega

Software engineer,

Freelance

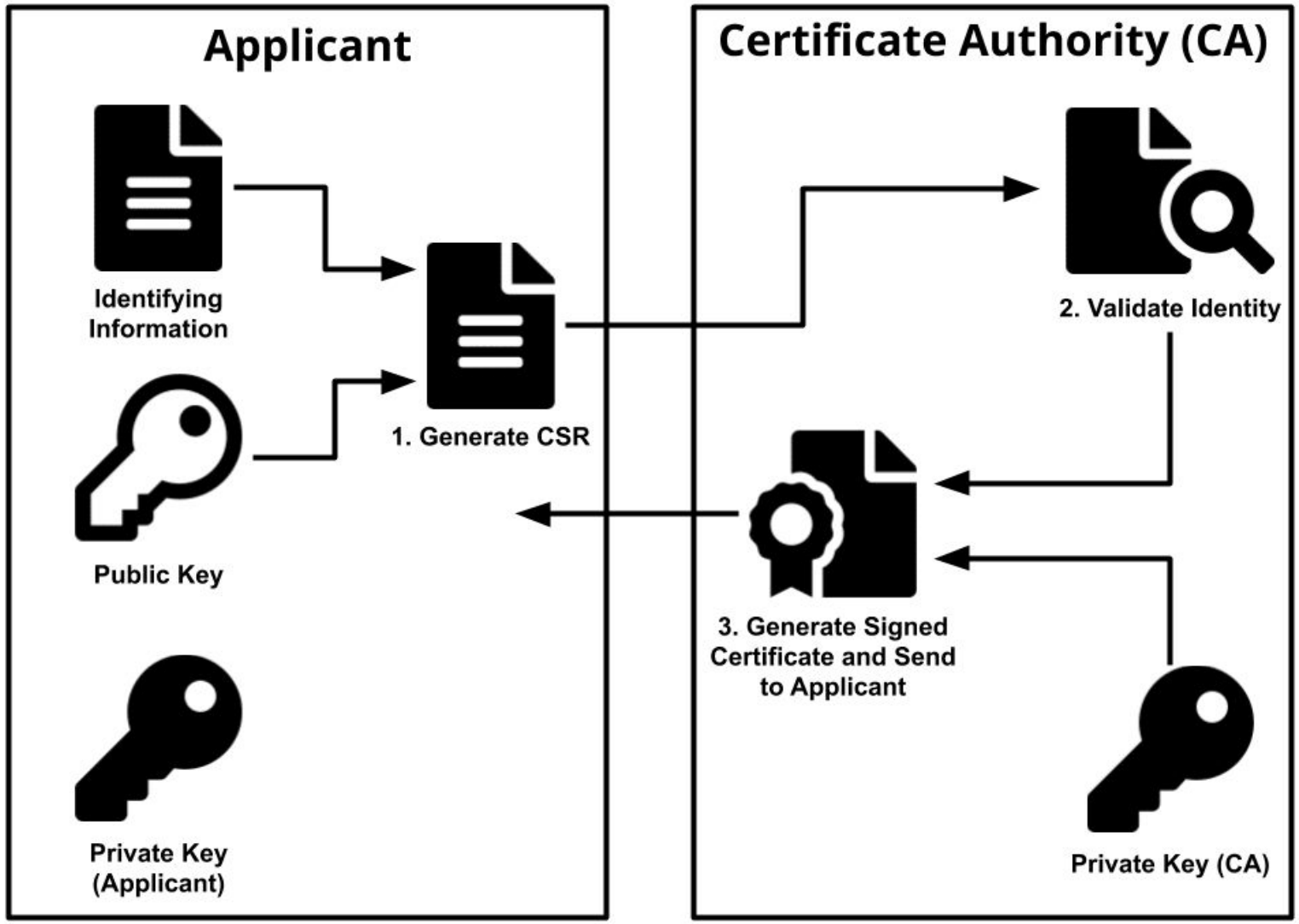
Grafana
RANCHER
docker
podman
clair
Falco
kube-hunter
aqua kube-bench
kubernetes
trivy

SCAN ME

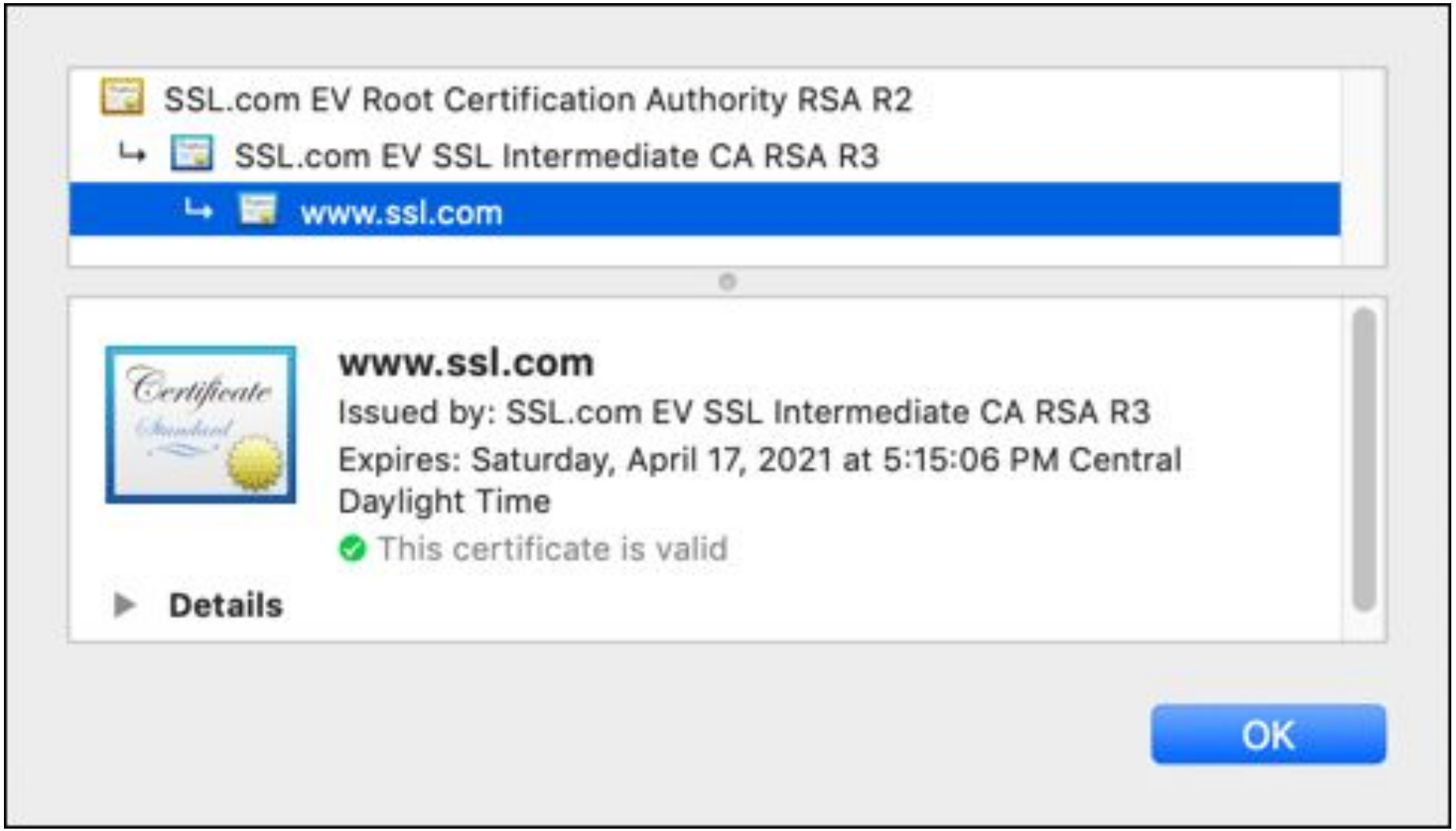
CERT MANAGER

INDEX

Introduction to certificates and certification authorities (CA)

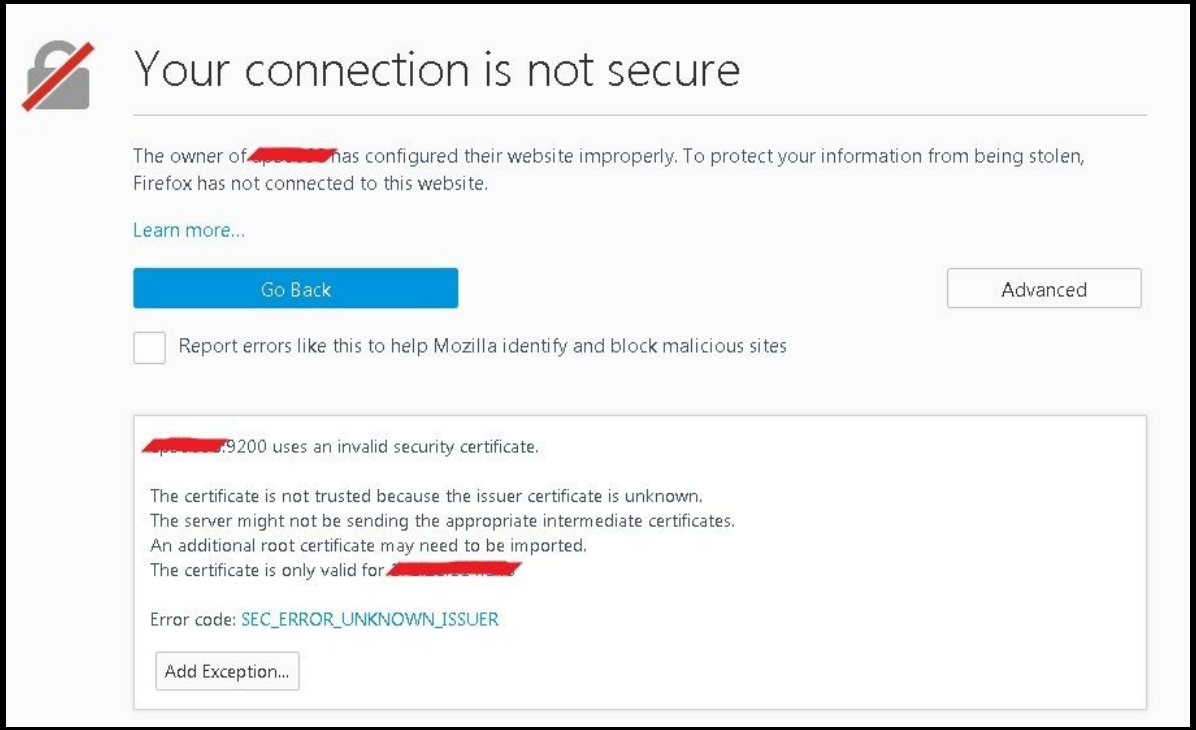


Introduction to certificates and certification authorities (CA)



K8s ingress with HTTPS

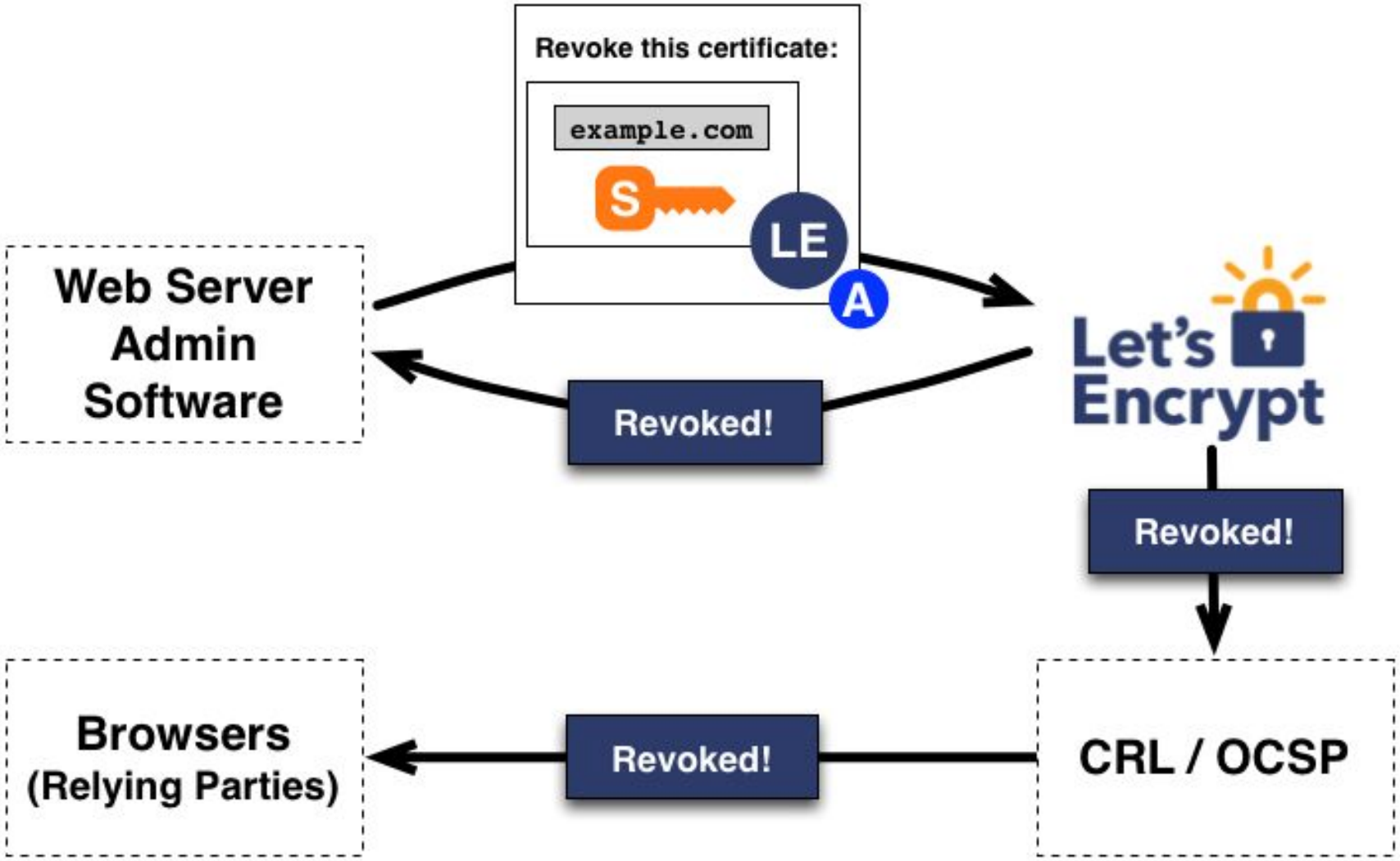

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: wordpress
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: wordpress
                port:
                  number: 80
  tls:
    - hosts:
      - domain.com
```



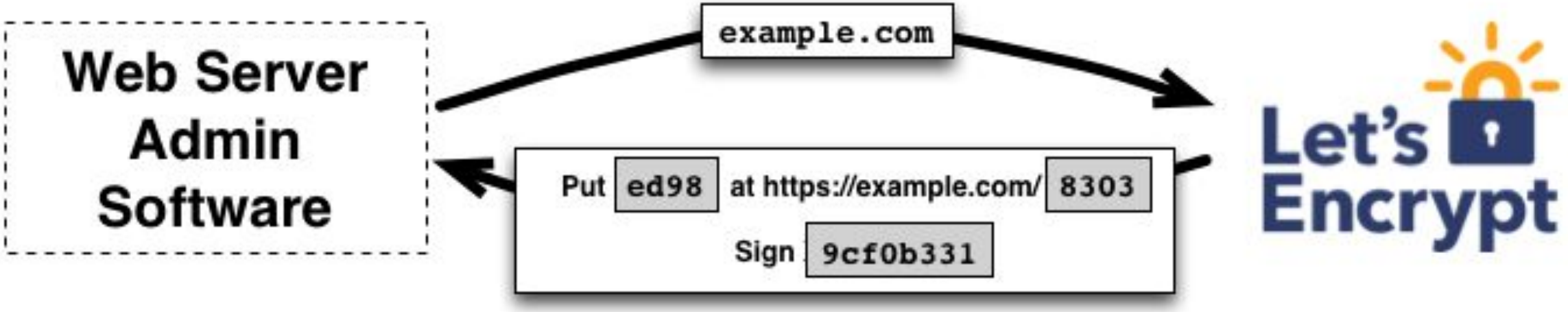
Certificates

- Self-Signed Certificates
- Purchase an SSL Certificate
- Use Let’s Encrypt Certificate

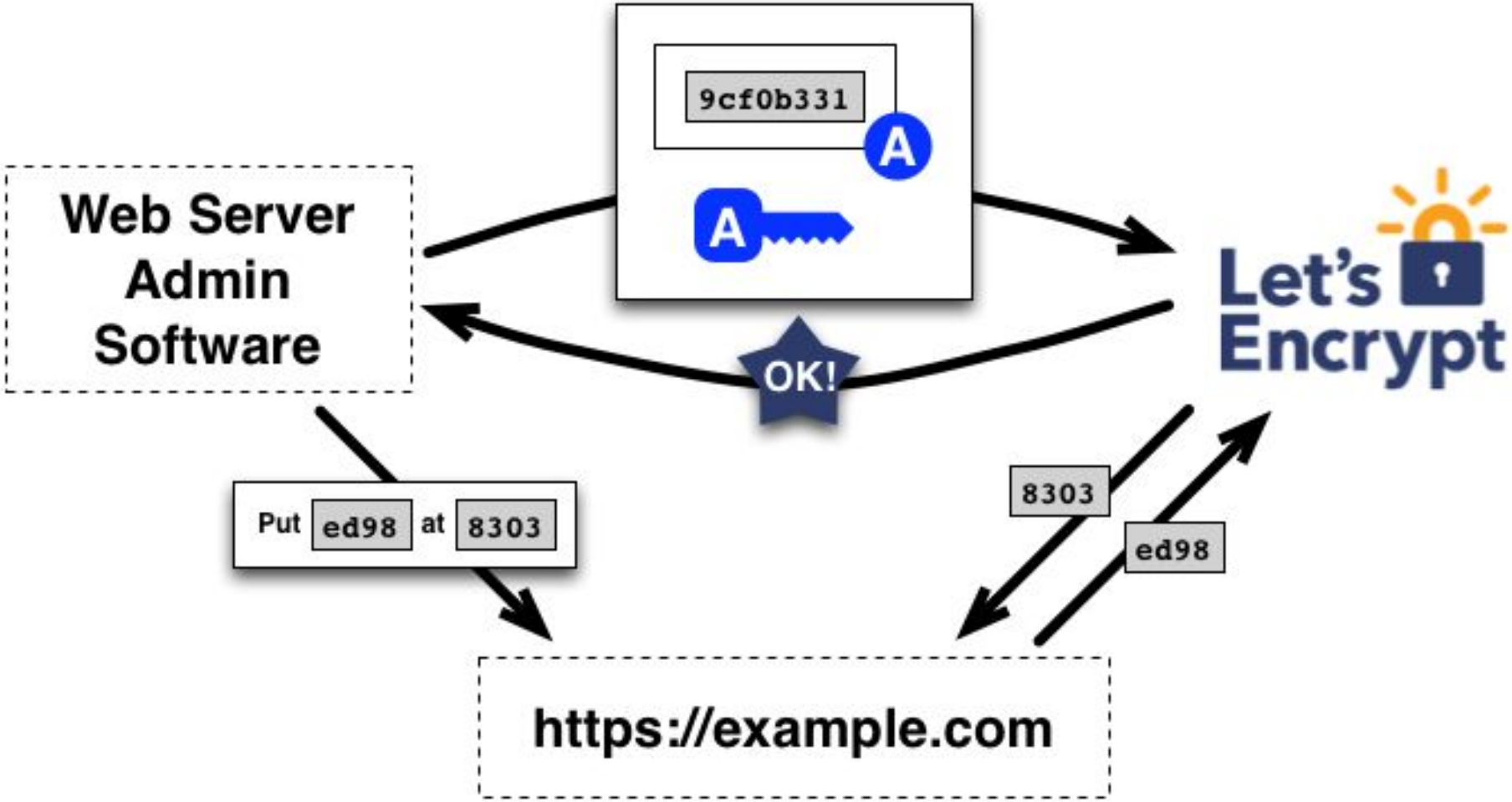
Let’s Encrypt as CA



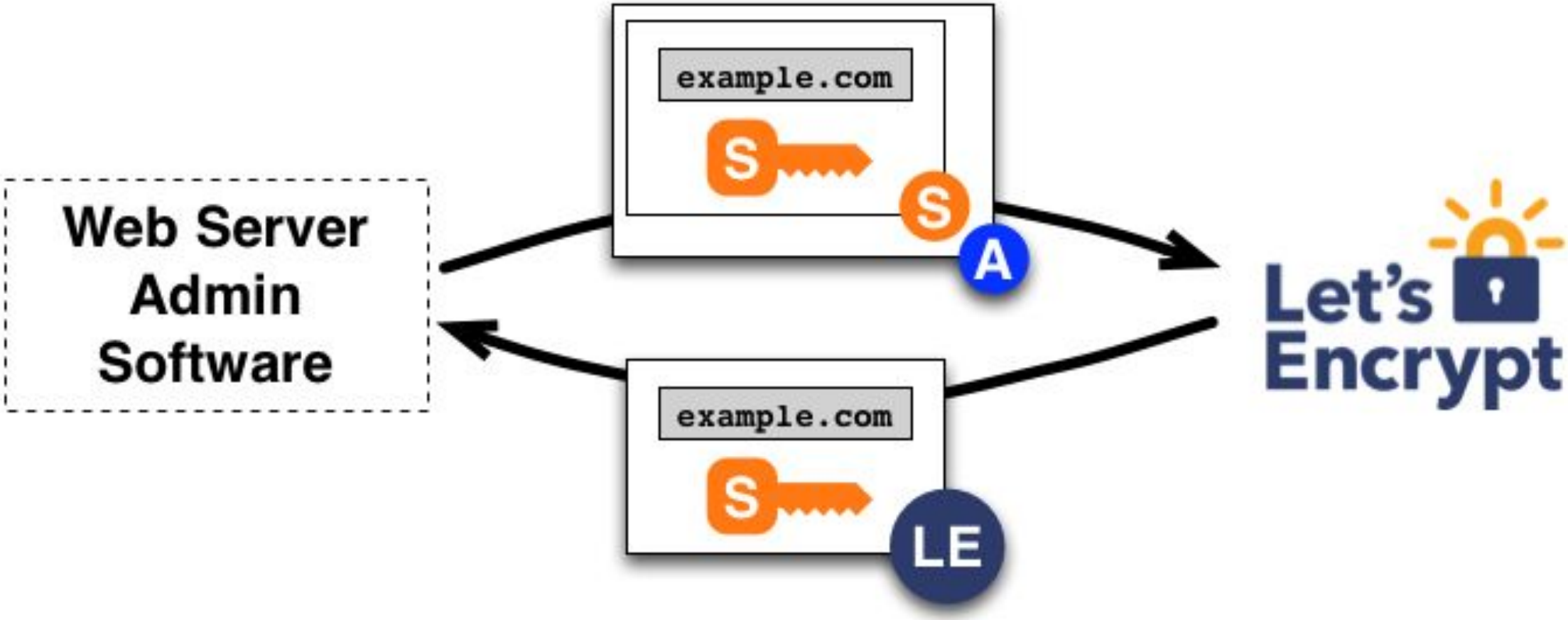
Let’s Encrypt



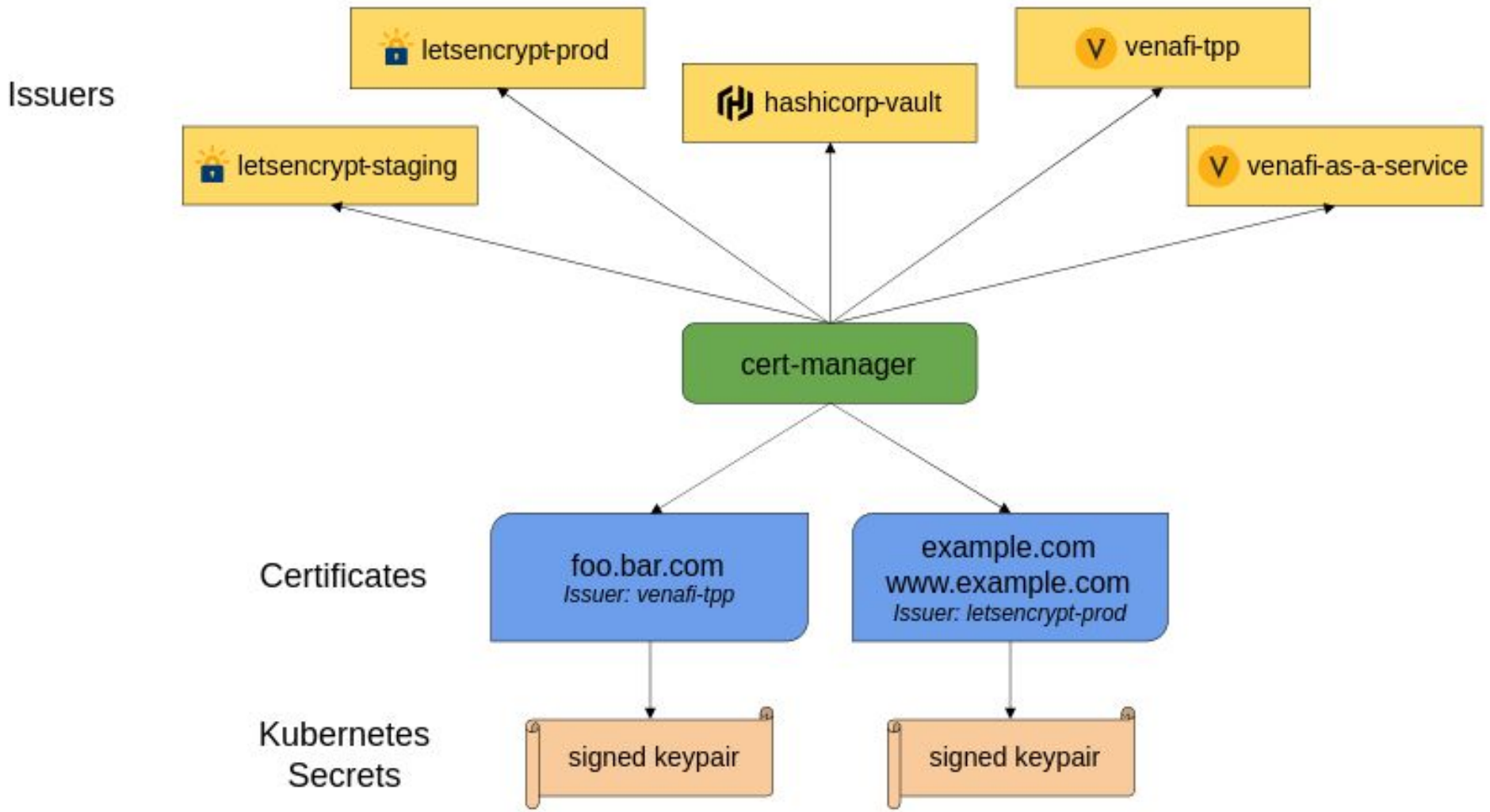
Let’s Encrypt



Let’s Encrypt



Introduction to cert-manager



Cert-manager repository

https://github.com/cert-manager/cert-manager

https://github.com/cert-manager/cert-manager/releases/

Releases | Tags

Tags

v1.8.0 Verified
26 days ago · e466a52 · zip · tar.gz · Notes · Downloads

v1.8.0-beta.0 Verified
on 1 Apr · e466a52 · zip · tar.gz · Notes · Downloads

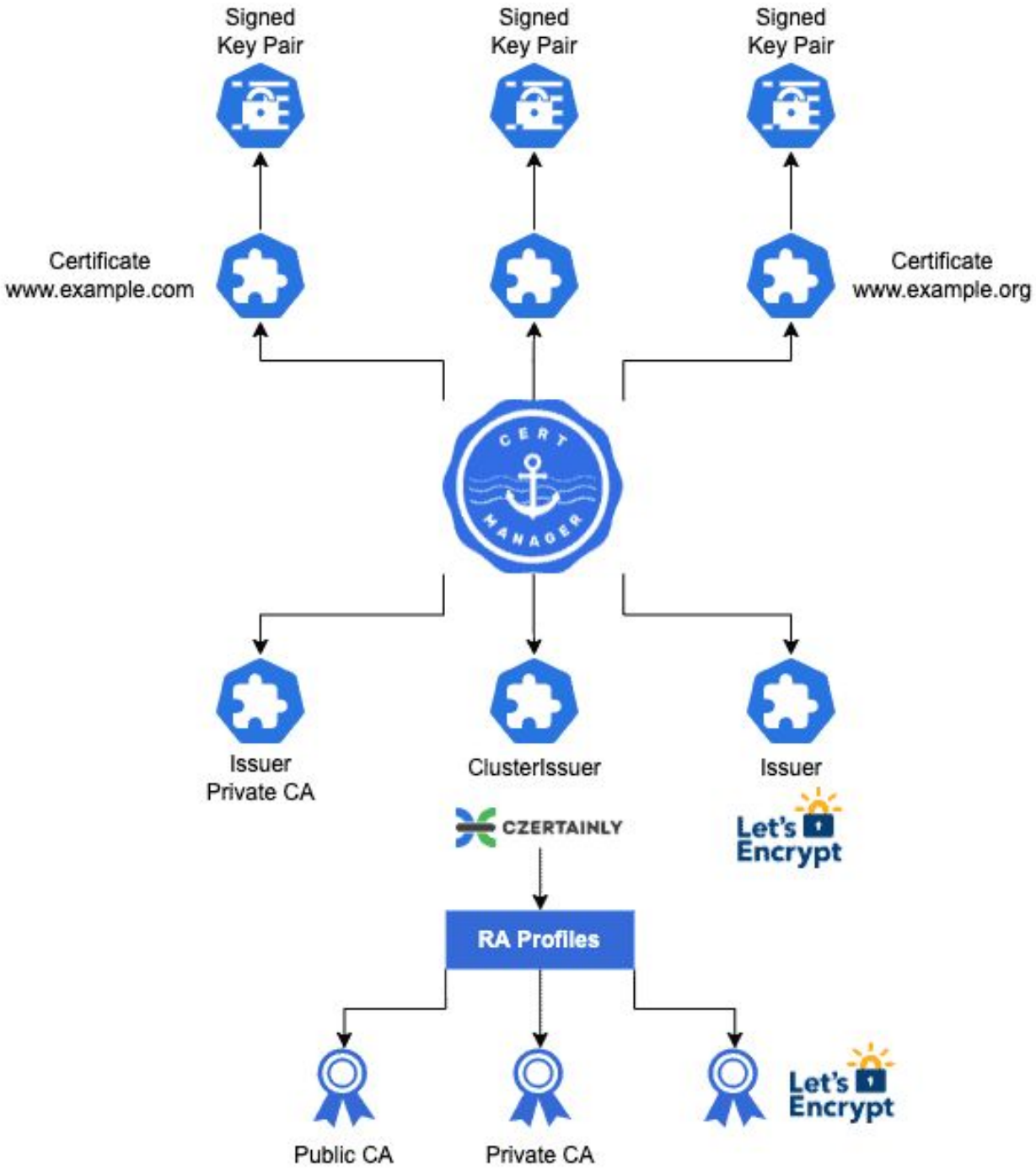
v1.8.0-alpha.2 Verified
on 31 Mar · aed1692 · zip · tar.gz · Notes · Downloads

v1.8.0-alpha.1 Verified
on 28 Mar · bfcc204 · zip · tar.gz · Notes · Downloads

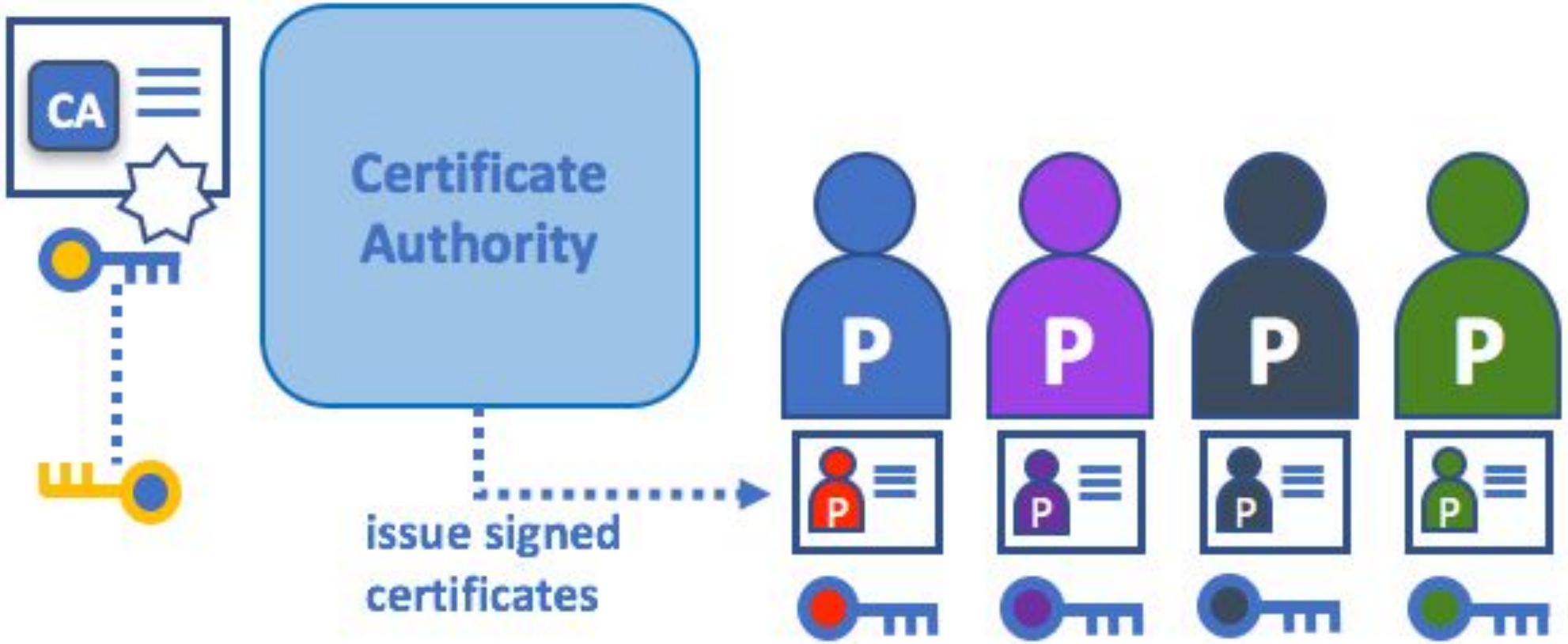
Cert-manager features

- cert-manager can use multiple Issuers, including:
  - self-signed
  - cert-manager acting as a CA
  - the ACME protocol (used by Let's Encrypt)
  - HashiCorp Vault
- Multiple issuers can be configured simultaneously
- Issuers can be available in a single namespace, or in the whole cluster (then we use the ClusterIssuer CRD)

Concepts


Signed
Key Pair
Signed
Key Pair
Signed
Key Pair
Certificate
www.example.com
Certificate
www.example.org
CERT
MANAGER
Issuer
Private CA
ClusterIssuer
Issuer
CZERTAINLY
Let's
Encrypt
RA Profiles
Public CA
Private CA
Let's
Encrypt

Certification authorities (CA) issuer



cert-manager in action

- We will install cert-manager
- We will create a ClusterIssuer to obtain certificates with Let's Encrypt (this will involve setting up an Ingress Controller)
- We will create a Certificate request and cert-manager will create a TLS Secret

Install Cert-manager with HELM


```
$ helm repo add jetstack https://charts.jetstack.io
$ helm repo update
$ helm install cert-manager jetstack/cert-manager
--namespace cert-manager --create-namespace --set
installCRDs=true
```

Install Cert-manager with HELM


```
$ kubectl cert-manager help

kubectl cert-manager is a CLI tool manage and configure cert-manager resources for Kubernetes

Usage: kubectl cert-manager [command]

Available Commands:
  approve      Approve a CertificateRequest
  check        Check cert-manager components
  convert      Convert cert-manager config files between different API versions
  create       Create cert-manager resources
  deny         Deny a CertificateRequest
  experimental Interact with experimental features
  help         Help about any command
  inspect      Get details on certificate related resources
  renew        Mark a Certificate for manual renewal
  status       Get details on current status of cert-manager resources
  version      Print the cert-manager CLI version and the deployed cert-manager version
```

Install & configure Cert-manager


```
$ kubectl create namespace cert-manager

$ kubectl apply --validate=false -f
https://github.com/cert-manager/cert-manager/releases/download/v1.7.2/cert-manager.yaml
```

Install & configure Cert-manager


```
customresourcedefinition.apiextensions.k8s.io/certificaterequests.cert-manager.io
created
customresourcedefinition.apiextensions.k8s.io/certificates.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/challenges.acme.cert-manager.io
created
customresourcedefinition.apiextensions.k8s.io/clusterissuers.cert-manager.io created
. . .
deployment.apps/cert-manager-webhook created
mutatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook
created
validatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook
created
```

Install & configure Cert-manager

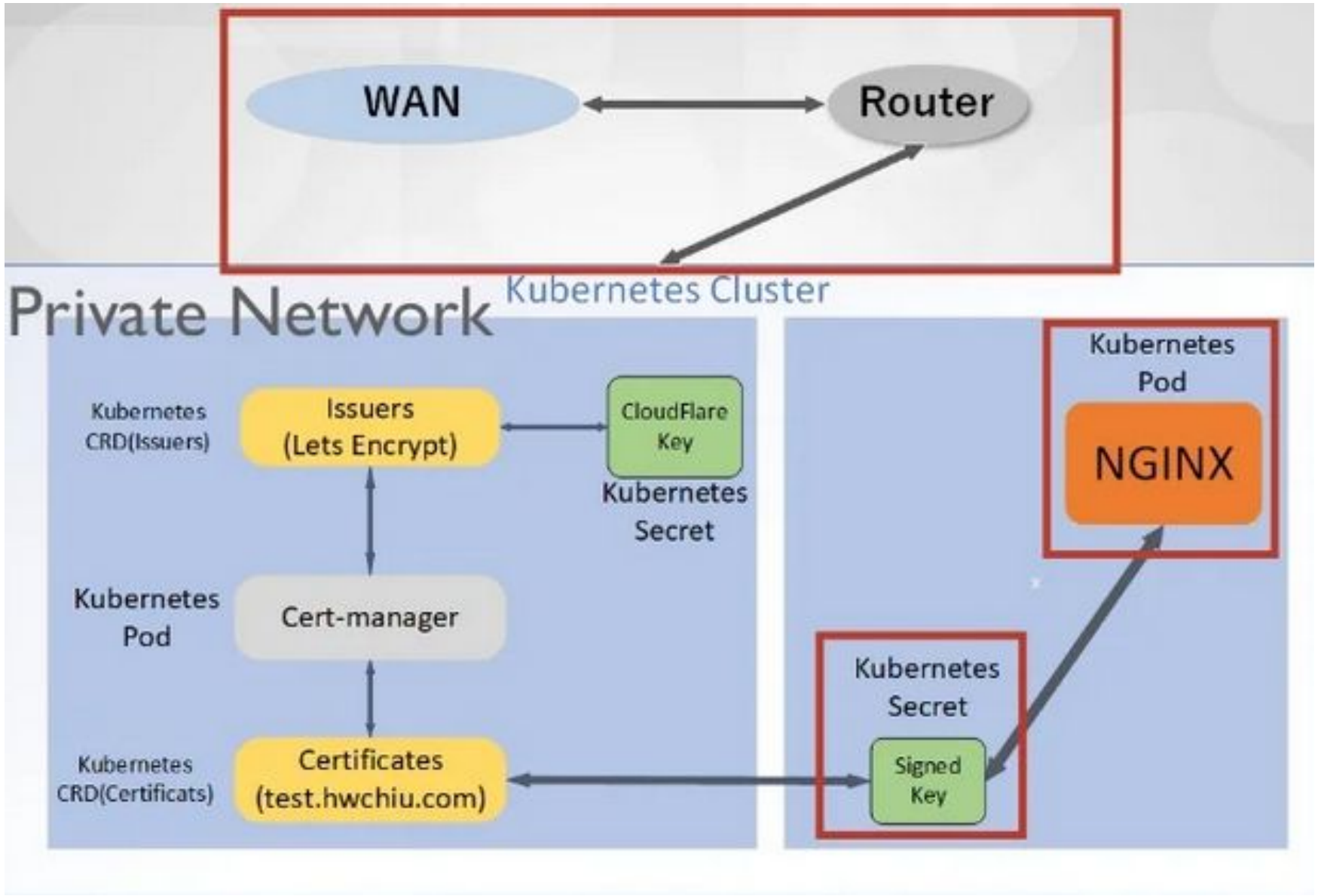

```
$ kubectl get pods --namespace cert-manager

NAME                                        READY   STATUS    RESTARTS   AGE
cert-manager-5c47f46f57-jknnx               1/1     Running   0          27s
cert-manager-cainjector-6659d6844d-j8cbg    1/1     Running   0          27s
cert-manager-webhook-547567b88f-qks44       1/1     Running   0          27s
```

Issuers

- Issuers (and ClusterIssuers) represent a certificate authority from which signed x509 certificates can be obtained, such as Let’s Encrypt.

- You will need at least one Issuer or ClusterIssuer to begin issuing certificates within your cluster.

Let’s Encrypt



Issuer

https://cert-manager.io/docs/concepts/issuer/

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ca-issuer
  namespace: mesh-system
spec:
  ca:
    secretName: ca-key-pair
```

Issuer vs ClusterIssuers

https://cert-manager.io/docs/concepts/issuer/

- Issuers only works on its Kubernetes cluster
- ClusterIssuers works for all namespaces

Working with LetsEncryt staging


```yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-staging
  namespace: cert-manager
spec:
  acme:
    # Email address used for ACME registration
    email: your-email-id-here
    server: https://acme-staging-v02.api.letsencrypt.org/directory
    privateKeySecretRef:
      # Name of a secret used to store the ACME account private key
      name: letsencrypt-staging-private-key
    # Add a single challenge solver, HTTP01 using nginx
    solvers:
    - http01:
        ingress:
          class: nginx
```

Working with LetsEncryt production

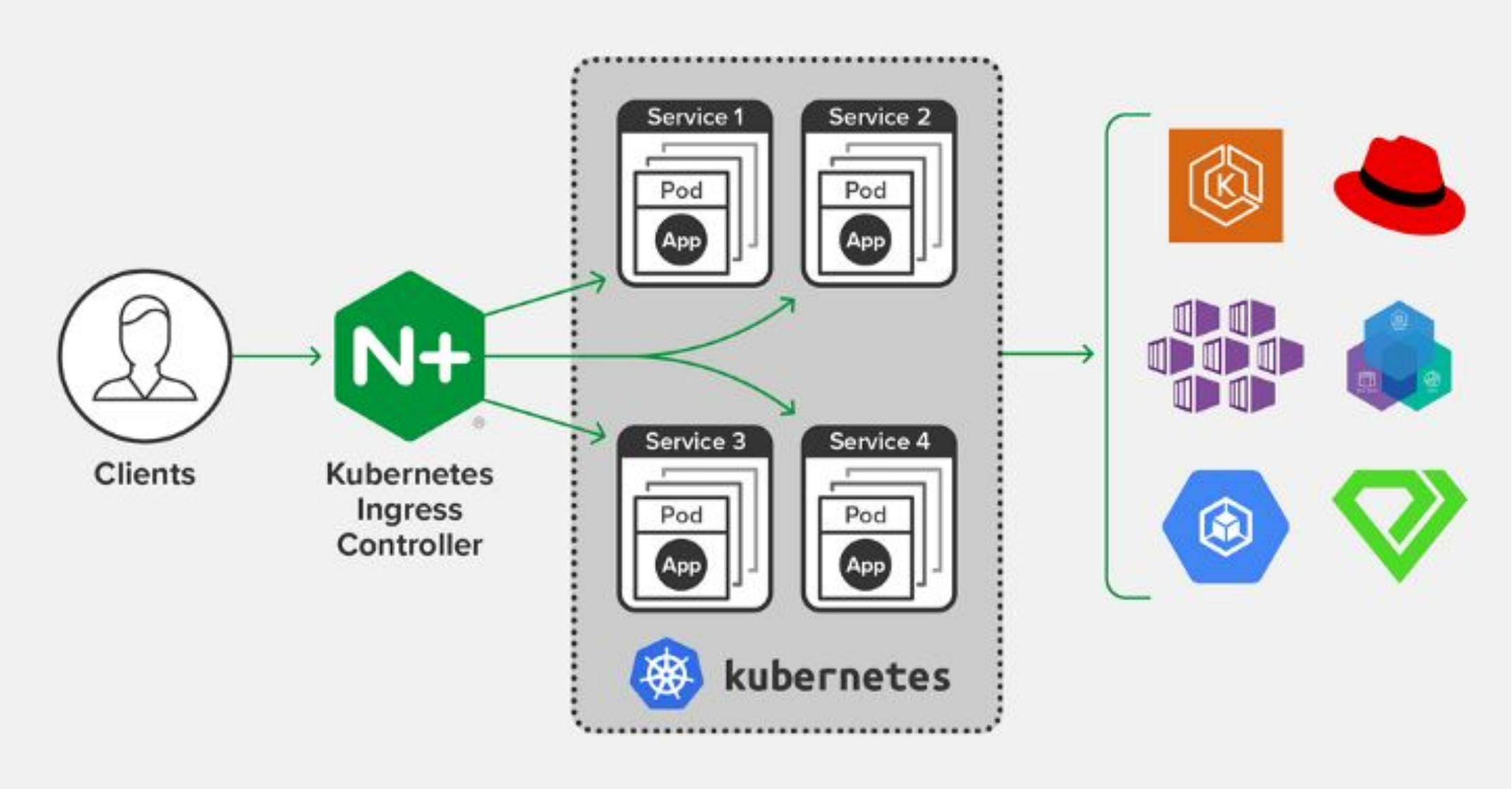

```yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-production
  namespace: cert-manager
spec:
  acme:
    # Email address used for ACME registration
    email: your-email-id-here
    server: https://acme-staging-v02.api.letsencrypt.org/directory
    privateKeySecretRef:
      # Name of a secret used to store the ACME account private key
      name: letsencrypt-production-private-key
    # Add a single challenge solver, HTTP01 using nginx
    solvers:
    - http01:
        ingress:
          class: nginx
```

Creating ClusterIssuer


```
$ kubectl apply -f staging_issuer.yaml

clusterissuer.cert-manager.io/letsencrypt-staging created
```

NGINX Ingress controller

https://github.com/kubernetes/ingress-nginx



Adding Ingress TLS/SSL support

- Create a Kubernetes secret with **server.crt** certificate and **server.key** private key file.
- Add the TLS block to the ingress resource

Kubernetes TLS Secret


```
$ kubectl create secret tls app-tls \
    --namespace dev \
    --key server.key \
    --cert server.crt
```

Add TLS block to Ingress Object


```
tls:
  - hosts:
    - your-domain.com
    secretName: app-tls
```

Ingress && Cert-manager


```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: cert-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
    cert-manager.io/cluster-issuer: "letsencrypt-staging"
spec:
  tls:
  - hosts:
    - your-domain.com
    secretName: app-tls
```

Install & configure Cert-manager


```
$ kubectl apply -f cert_ingress.yaml

ingress.networking.k8s.io/echo-ingress configured
```

Install & configure Cert-manager


```
$ kubectl get secrets

NAME                TYPE                  DATA   AGE
app-tls             kubernetes.io/tls     3      1m
```

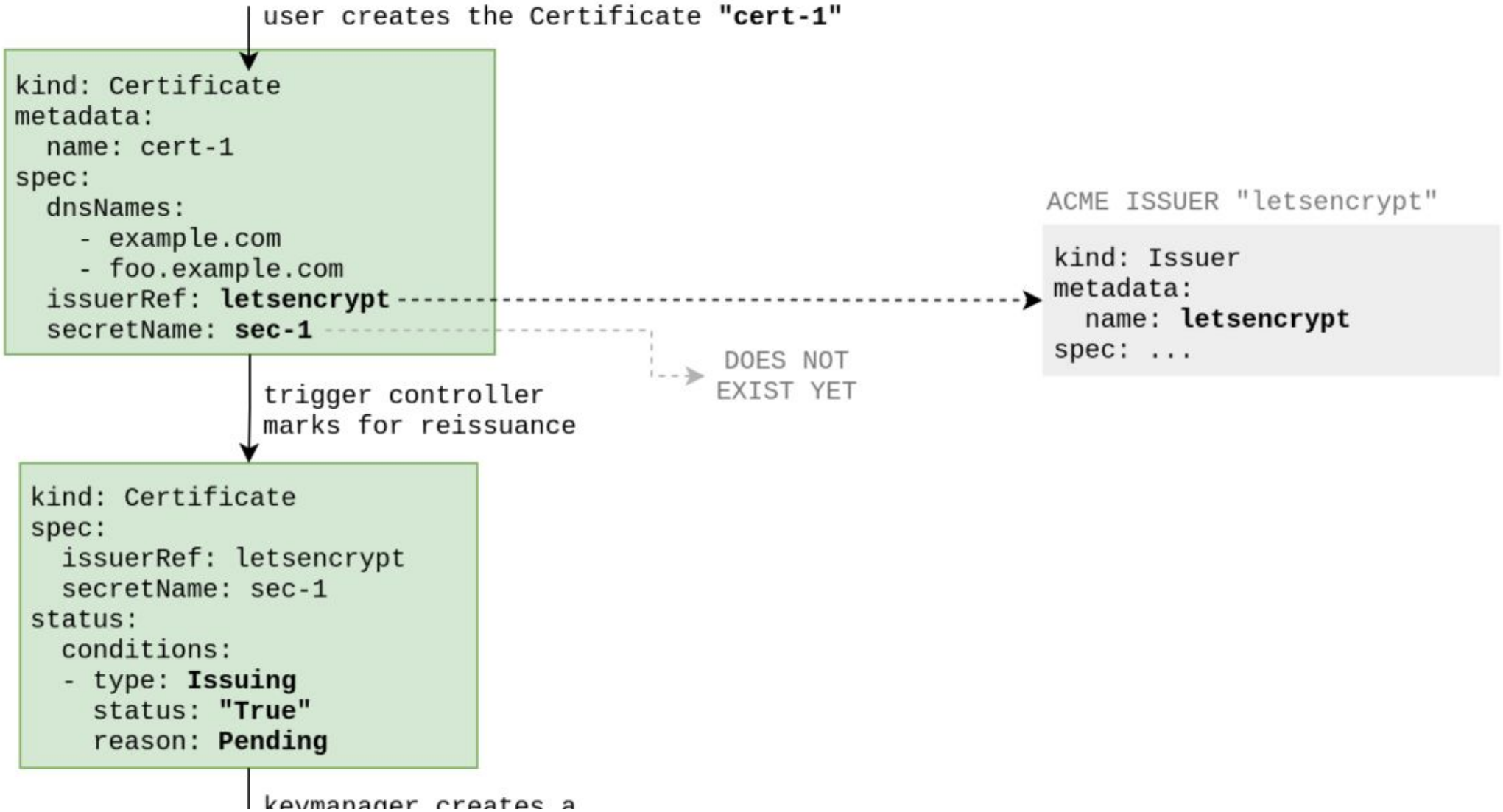
Install & configure Cert-manager


```
$ kubectl get certificates

NAME     READY  SECRET AGE
app-tls  True   app-tls  1m
```

Install & configure Cert-manager


```
$ kubectl describe certificate

Events:
  Type    Reason          Age   From          Message
  ----    ------          ----  ----          -------
  Normal  GeneratedKey    2m12s cert-manager  Generated a new private key
  Normal  Requested       2m12s cert-manager  Created new CertificateRequest resource "echo-tls-3768100355"
  Normal  Issued          47s   cert-manager  Certificate issued successfully
```

Certificate Lifecycle



Certificate Lifecycle

keymanager creates a temporary private key

```
kind: Certificate
spec:
  issuerRef: letsencrypt
  secretName: sec-1
status:
  nextPrivateKeySecret: sec-1-01ab4f
  conditions:
  - type: Issuing
    status: "True"
    reason: Pending
```

TEMPORARY SECRET

```
kind: Secret
metadata:
  name: sec-1-01ab4f
stringData:
  tls.key: |
    -----BEGIN PRIVATE KEY-----
    AaBbCcDd0
    -----END PRIVATE KEY-----
```

(a) requestmanager creates CertificateRequest with revision = "1" since its revision is nil

(b) requestmanager signs the CSR using the private key

Certificate Lifecycle

(a) requestmanager creates CertificateRequest
with revision = "1" since its revision is nil

(b) requestmanager signs the CSR
using the private key

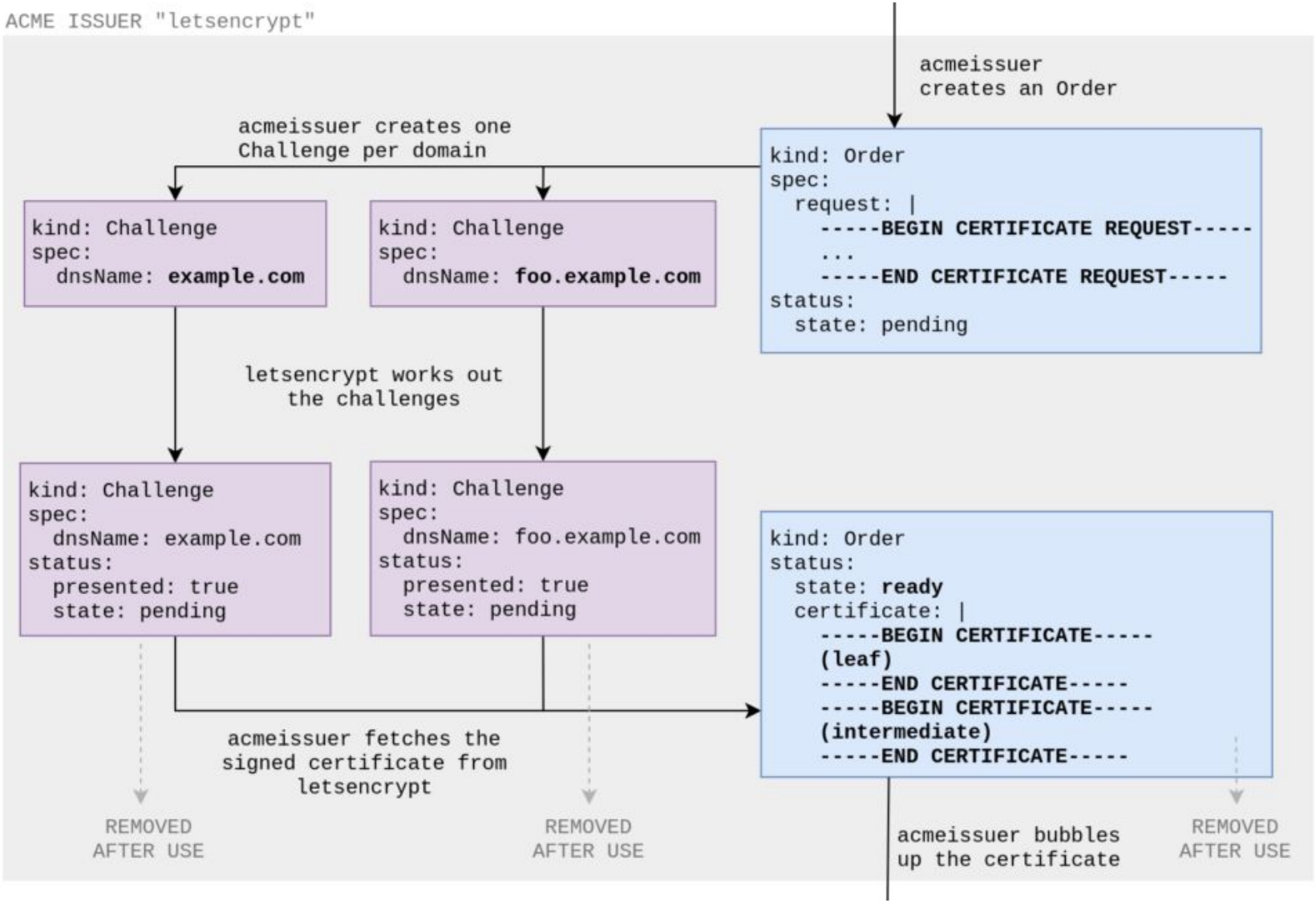
```
kind: Certificate
spec:
  issuerRef: letsencrypt
  secretName: sec-1
status:
  nextPrivateKeySecret: sec-1-01ab4f
  revision: nil
  conditions:
  - type: Issuing
    status: "True"
    reason: Pending
```

(a)

```
kind: CertificateRequest
metadata:
  name: cert-1-ab0123
  annotations:
    cert-manager.io/certificate-revision: "1"
spec:
  issuerRef: letsencrypt
  dnsNames: [example.com, foo.example.com]
  request: |
    -----BEGIN CERTIFICATE REQUEST-----
    ...
    -----END CERTIFICATE REQUEST-----
```

(b)

Certificate Lifecycle



Certificate Lifecycle


```
kind: Certificate
spec:
  issuerRef: letsencrypt
  secretName: sec-1
status:
  nextPrivateKeySecret: sec-1-01ab4f
  revision: 1
  conditions:
    - type: Issuing
      status: "False"
      reason: Issued
  certificate: |
    -----BEGIN CERTIFICATE-----
    (leaf)
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    (intermediate)
    -----END CERTIFICATE-----
```

(a) issuing controller sets revision = 1

(b) issuing controller bubbles up the certificate

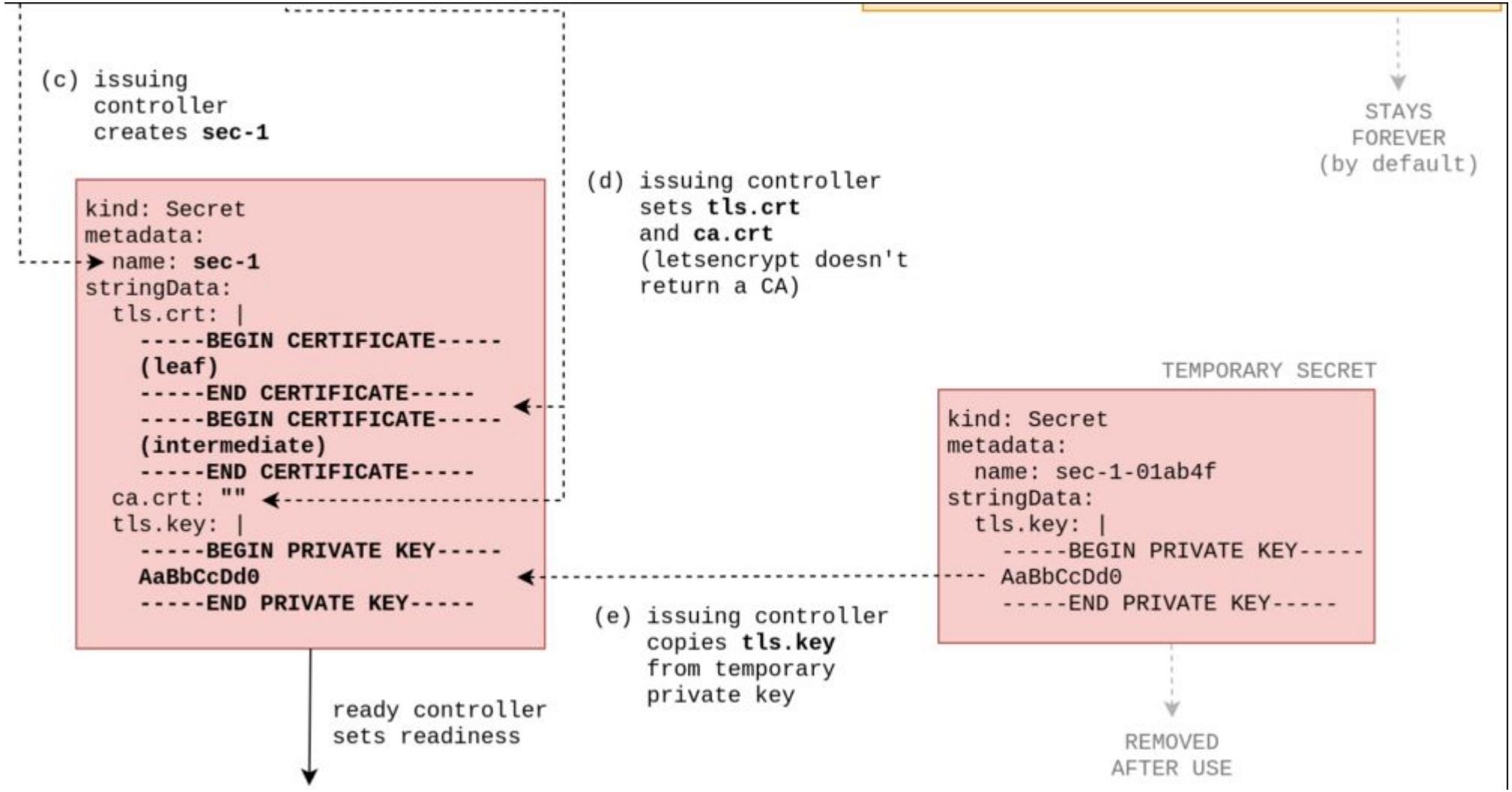
```
kind: CertificateRequest
metadata:
  name: cert-1-ab0123
  annotations:
    cert-manager.io/certificate-revision: "1"
status:
  conditions:
    - type: Ready
      status: "True"
      reason: Issued
  certificate: |
    -----BEGIN CERTIFICATE-----
    (leaf)
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    (intermediate)
    -----END CERTIFICATE-----
```

(c) issuing controller creates **sec-1**

STAYS FOREVER (by default)

(d) issuing controller

Certificate Lifecycle



Certificate Lifecycle

ready controller sets readiness

```
kind: Certificate
spec:
  issuerRef: letsencrypt
  secretName: sec-1
status:
  revision: 1
  conditions:
    - type: Ready
      status: "True"
      reason: Issued
    - type: Issuing
      status: "False"
      reason: Issued
  certificate: |
    -----BEGIN CERTIFICATE-----
    (leaf)
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    (intermediate)
    -----END CERTIFICATE-----
```

DEMO

KubeCon | CloudNativeCon Europe 2022

https://www.katacoda.com/lynnfrank/scenarios/vault-kubernetes-cert-manager

O'REILLY Katacoda

KATACODA OVERVIEW & SOLUTIONS

CLAIM

Welcome!

Configure Vault as a Certificate Manager in Kubernetes with Helm

Difficulty: intermediate

Estimated Time: 30 minutes

Kubernetes configured to use Vault as a certificate manager enables your services to establish their identity and communicate securely over the network with other services or clients internal or external to the cluster.

Jetstack's cert-manager enables Vault's PKI secrets engine to dynamically generate X.509 certificates within Kubernetes through an Issuer interface.

In this guide, you setup Vault with the Vault Helm chart, configure the PKI secrets engine and Kubernetes authentication. Then install Jetstack's cert-manager, configure it to use Vault, and request a certificate.

https://learn.hashicorp.com/vault

START SCENARIO

Conclusions

- Cert-manager facilitates certificate signing through the Kubernetes API:
  - we create a Certificate object.
  - cert-manager creates a private key
  - it signs that key …
  - ... or interacts with a certificate authority to obtain the signature
  - it stores the resulting key+cert in a Secret resource
- These Secret resources can be used in many places (Ingress, mTLS, ...)

Survey

https://bit.ly/3s3XfS5

KubeCon CloudNativeCon Europe 2022

Cloud Native certificate management

X.509 certificate management for Kubernetes and OpenShift

cert-manager community user survey

cert-manager is a CNCF Sandbox project that is applying for Incubation status. This short anonymous survey is designed to help with that process by providing insights into the way cert-manager is being used in production Kubernetes environments. The results will be shared with the community and provided to the CNCF to help further cert-manager's journey to becoming a fully Graduated CNCF project. We appreciate your time, thank you!