JETSTACK by Venafi

Rotate Roots Right Round

Using cert-manager for Safer Private PKI

Ashley Davis **Senior Software Engineer**

KubeCon Amsterdam

Ashley Davis

Senior Software Engineer

✉️ ashley.davis@jetstack.io

🐦 @SgtCoDFish

🐘 @SgtCoDFish@infosec.exchange

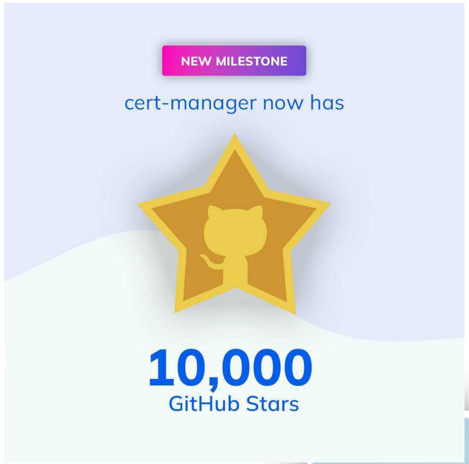
github.com/SgtCoDFish

cert-manager

"cert-manager is the easiest way to automatically manage certificates in Kubernetes clusters.

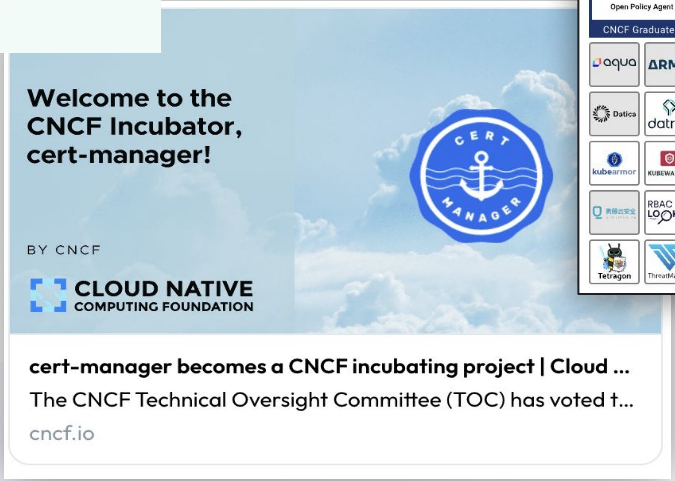
— Literally us, the cert-manager team

CLOUD NATIVE
COMPUTING FOUNDATION


NEW MILESTONE

cert-manager now has

10,000
GitHub Stars


Welcome to the CNCF Incubator, cert-manager!

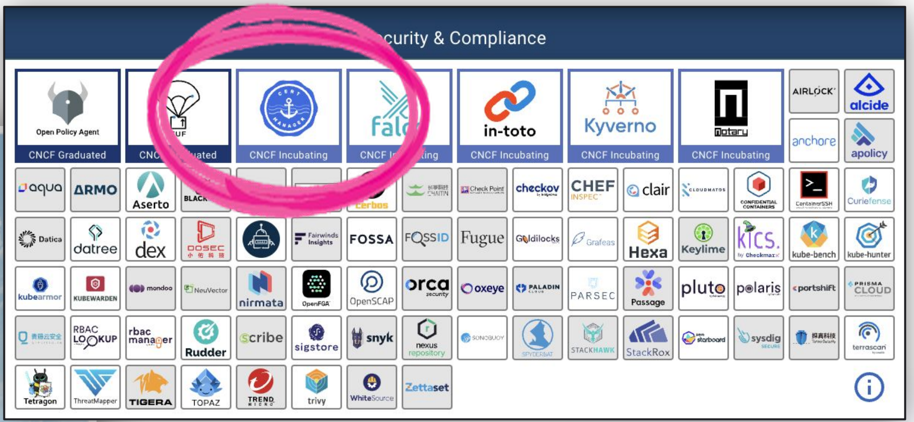
BY CNCF

CLOUD NATIVE COMPUTING FOUNDATION

cert-manager becomes a CNCF incubating project | Cloud ...

The CNCF Technical Oversight Committee (TOC) has voted t...

cncf.io



```
kind: ClusterIssuer
apiVersion: cert-manager.io/v1
metadata: { name: my-ca-issuer }
spec:
  ca: { secretName: root-secret }
```

HashiCorp Vault

Venafi Platform

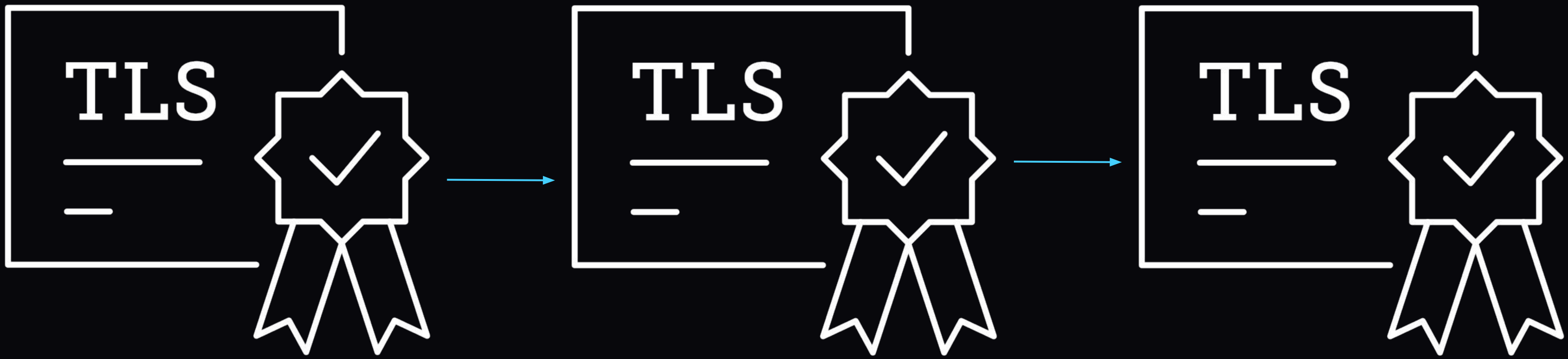
Let's Encrypt

What’s PKI?

- PKI stands for **P**ublic **K**ey **I**nfrastructure
- PKI = Chains of certificates
- Private PKI: Private certificates controlled by you

- Private PKI:
  - Is cost-effective (free or at least cheap)
  - Gives total control over certs
  - Has no artificial rate limits

- Certificates are the things you need for TLS / SSL
- Many / most services today need certificates
- Enables encryption in transit.
- That applies for Kubernetes too!



Root

Certificate

Intermediate

Certificate

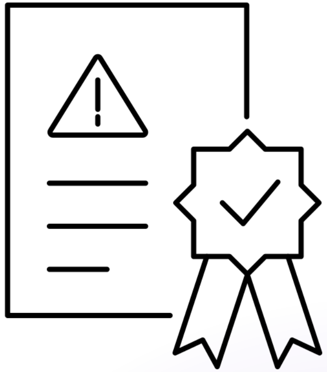
Leaf

Certificate

- Traditionally, intermediates issue leaf certificates
- Roots are stored offline
- Roots are only used to issue new intermediates

- This architecture isn’t required!
- Roots could issue directly
- What’s best depends on your architecture

Risks of Private PKI



- Private PKI is no free lunch
- There are serious issues to consider
- Some risks are specific to cert-manager
- Some are true for PKI generally

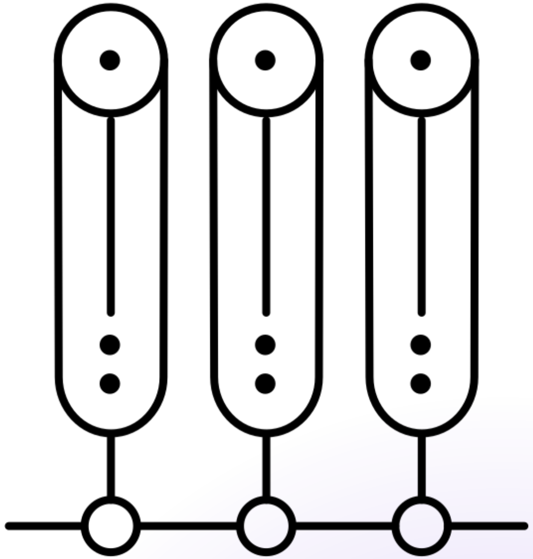
- **Risk #1:** Not Locking Down
- CA issuers will issue anything
- Including other CA certificates!
- Can be mitigated with approver-policy

- **Risk #2:** Root Rotation!
- Lots of gotchas, easy to get wrong
- Requires a plan; this is unavoidable
- Generally tricky to automate
- Luckily, this talk will cover this!

- **Risk #3:** Trust
- Private CAs aren’t useful if nothing trusts them
- Subtleties with rotation
- trust-manager can help
- You still need to prepare

- We’ll mitigate these risks
- And we’ll produce a sustainable plan
- Must be production ready and easy to maintain

Safe
Private PKI



- There's a repo for this!
- https://github.com/SgtCoDFish/rotate-roots
- Don't copy from slides!
- Link shared again later

Issuance

- Simplest approach is entirely in-cluster
- Could just use cert-manager
- Safer to use approver-policy and trust-manager
- Other methods are available!

```yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata: { name: root-certificate, namespace: cert-manager }
spec:
  isCA: true
  commonName: root-certificate
  secretName: root-secret
  duration: 219000h # ~25y
  issuerRef: <selfsigned issuer ref>
```

- Roots can have longer lifetimes
- As long as rotation policy is in place
- Feel free to use more descriptive names!

```yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata: { name: root-issuer }
spec:
  ca: { secretName: root-secret }
```

Policy

- Policy prevents several vulnerabilities:
  - Issuing unexpected CA certificates
  - Issuing for privileged domains
  - Dangerously long lifetimes

```yaml
apiVersion: policy.cert-manager.io/v1alpha1
kind: CertificateRequestPolicy
metadata: { name: ca-issuer-policy }
spec:
  allowed:
    isCA: false
    commonName: { value: "*", required: true }
    dnsNames: { values: ["*"], required: true }
  constraints: { maxDuration: 48h }
  selector: { issuerRef: { name: root-issuer, … } }
```

- In prod: restrict allowed dnsNames
- Max duration should be as low as possible
- Really shouldn’t run private PKI without something like this!

Trust

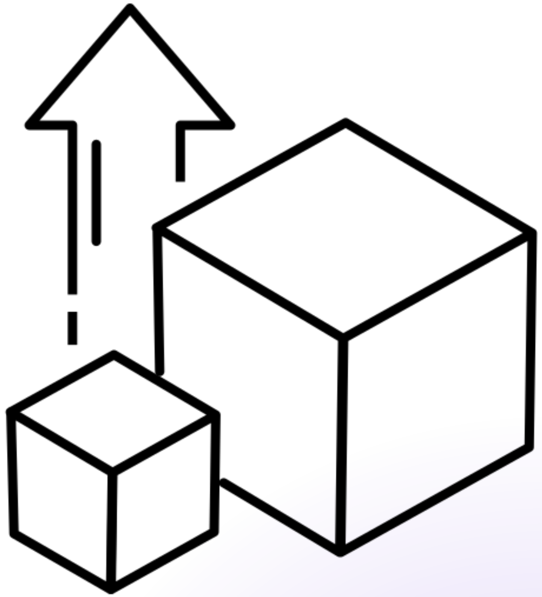
- How do we use our root?
- Pods need to trust it
- Pods also need to trust public certificates
- Enter trust-manager!

```yaml
apiVersion: trust.cert-manager.io/v1alpha1
kind: Bundle
metadata: { name: trust-bundle }
spec:
  sources:
  - useDefaultCAs: true
  - configMap: { name: "cluster-root", key: "root.pem" }
  target: { configMap: { key: "root-certs.pem" } }
```

- Pods mount the resulting ConfigMap and trust our CA
- Bundle means we only have one thing to update
- Avoid rebuilding every container if there’s a new bundle!

- **Warning:** *Don’t use “ca.crt” from the root, directly!*
- If the cert is changed in-place, trust will be updated immediately
- This would break running applications
- Copy the cert into a ConfigMap instead

Scaling Up and Operationalizing



Root Rotation 🐘

- “Rotating intermediates is easier!”
- True - but context matters
- Rotating intermediates in a disaster is hard
- So you need to practice rotating roots!

- What about revocation?
- Very, *very* hard to get right
- Unlikely you'll defend against all attacks
- Lukewarm take: Revocation is not worth it

- Assumptions:
    - No downtime for regular rotation
    - Automated where possible

1. Create new root certificate
2. Add root to Bundle
3. Ensure everything uses new trust bundle
4. Shift issuance to new root
5. Remove old root

1. Create new root certificate
2. Add root to Bundle
3. Ensure everything uses new trust bundle ⚠️
4. Shift issuance to new root
5. Remove old root

- Almost impossible to automate this generally
- Within k8s, redeploy everything (thanks, trust-manager!)
- Outside k8s? Your org needs to keep track

```yaml
apiVersion: trust.cert-manager.io/v1alpha1
kind: Bundle
metadata: { name: trust-bundle }
spec:
  sources:
  - useDefaultCAs: true
  - configMap: { name: "cluster-root", key: "root.pem" }
  - configMap: { name: "cluster-root-2", key: "root.pem" }
  target: { configMap: { key: "root-certs.pem" } }
```

There's No One Answer

- **Key point #1:** Your Architecture Matters
- Designing PKI depends on your architectural choices
- Probably default to one root per workload / one per cluster

- **Key point #2:** Rotation Is Key
- Architecture aside, you need to plan rotation
- Rotating roots is fairly simple with trust-manager
- Needs regular practice
- Might need to be done in an emergency

- **Key point #3:** Revocation Isn’t Reliable
- You need a rotation plan anyway, so…
- Lean on your rotation plan!

- **Key point #4:** Try Private PKI
- At the very least you’ll learn!
- Maybe you’ll save some money or time, too!

https://github.com/SgtCoDFish/rotate-roots

JETSTACK by Venafi

Thank you! 🚀

Ashley Davis **Senior Software Engineer**

KubeCon Amsterdam