



KubeCon



CloudNativeCon

Europe 2026

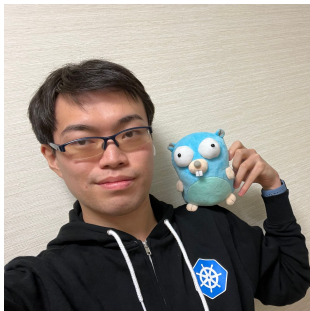
#KubeCon #CloudNativeCon

Exploring NRI for Automated CA Trust Injection

**Tsuzuki Tsuchiya & Kento Kubo,
LY Corporation**

March 25, 2026 | Amsterdam, Netherlands





Tsuzuki Tsuchiya

Software Engineer at LY Corporation, Japan
Developing and operating internal Kubernetes as a Service

GitHub: tsuzu / X: @_tsuzu_ / LinkedIn: tsuzu



Kento Kubo

Software Engineer at LY Corporation, Japan
Developing and operating internal Kubernetes as a Service

GitHub: KentoKubo / LinkedIn: kento-kubo-ken8

Agenda



KubeCon



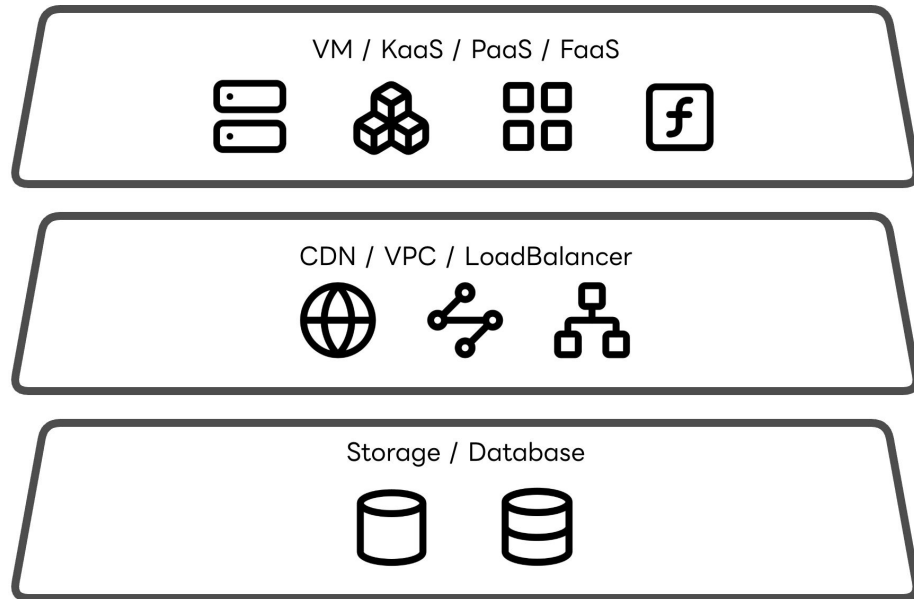
CloudNativeCon

Europe 2026

- Background: Managing Private CA
- Challenges: Injecting CA to Containers
- Introduction to NRI (Node Resource Interface)
- Our solution with NRI
- Comparison
- Summary

Our On-Premises Private Cloud

- Using OpenStack for VMs
- Using Athenz for IAM
 - <https://github.com/AthenZ>
- KaaS (Kubernetes as a Service)
 - 1000+ Kubernetes clusters
 - Powered by Cluster API



Private CAs for Service-to-Service Authentication



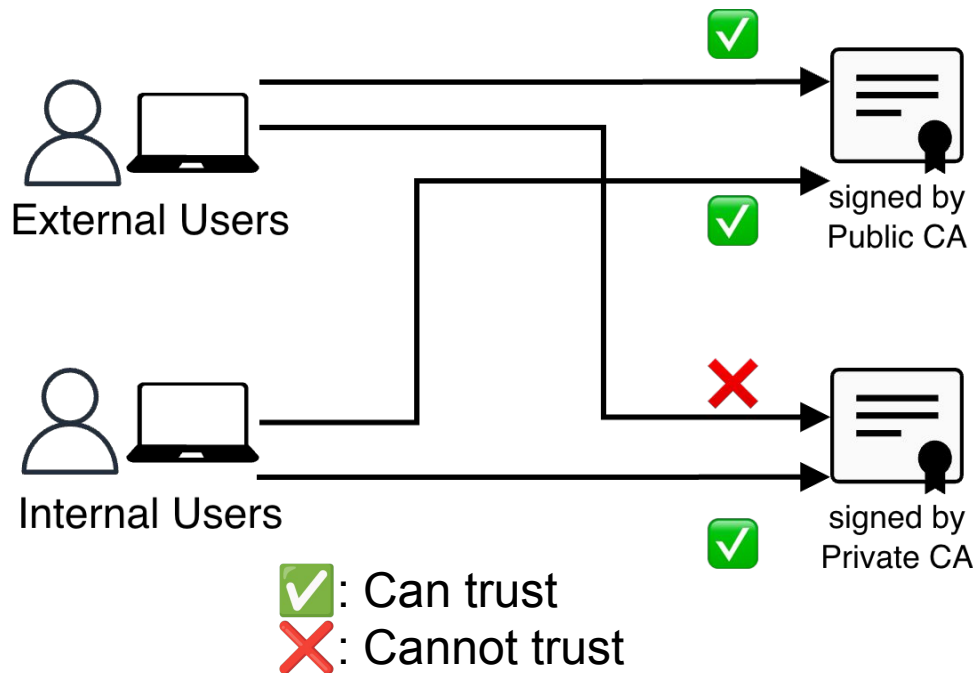
KubeCon



CloudNativeCon

Europe 2026

- We have our internal PKI (Public Key Infrastructure) for the flexible certificate management.
- PKI is a foundational technology for Zero Trust, enabling strong identities and mutual authentication.
- Purpose
 - Enforce security policies
 - mTLS with Athenz
 - Server certs for internal services (not public to the Internet)



```
root@33b83ae95742:~# go run main.go
panic: Get "https://example.com": tls: failed to verify certificate: x509:
certificate signed by unknown authority
```



How to install public CA certificates

- Install certs with a package manager

```
RUN apt-get install -y ca-certificates
```

How to install private CA certificates

- **Writing Dockerfiles:** Prepare CI pipeline and push to internal registries.
- **Init-Containers:** Copy private CA certs at init-containers.
- **Mounting static files:** Mount static files from ConfigMaps, hostPath, etc....

```
FROM debian:12-slim AS certs

COPY private-ca-cert.pem /usr/local/share/ca-certificates/
RUN apt-get update && \
    apt-get install -y --no-install-recommends ca-certificates && \
    update-ca-certificates

FROM debian:12-slim

COPY --from=certs /etc/ssl/certs/ca-certificates.crt \
    /etc/ssl/certs/ca-certificates.crt
```

Manually copy certificates for each image....

✗ Issue

Needs to maintain image repositories, CI pipelines, Dockerfiles

```
initContainers:
- name: build-ca-bundle
  image: ubuntu:24.04
  command: ["/bin/sh", "-c"]
  args:
  - |
    cp /path/to/my-company-ca.crt \
      /usr/local/share/ca-certificates/my-company-ca.crt
    update-ca-certificates
  volumeMounts:
  - name: ca-work
    mountPath: /work
containers:
- name: app
  image: your-image:tag
  volumeMounts:
  - name: ca-work
    mountPath: /etc/ssl/certs/ca-certificates.crt
    subPath: ca-certificates.crt
    readOnly: true
```

Share CA certificates with emptyDir

✗ Issue

- Needs to maintain complex Pod templates
- Takes longer time to launch

Mounting static files

```
...
spec:
  containers:
    - name: app
      image: your-image:tag
      volumeMounts:
        - name: ca-cert
          mountPath: /etc/ssl/certs
          readOnly: true
  volumes:
    - name: ca-cert
      projected:
        sources:
          - clusterTrustBundle:
              name: ca-certificates
              path: ca-certificates.pem
---
apiVersion: certificates.k8s.io/v1alpha1
kind: ClusterTrustBundle
metadata:
  name: ca-certificates
spec:
  trustBundle: "<... PEM data ...>"
```

Mount static files from...

- ConfigMaps / Secrets
- hostPath
- ClusterTrustBundle (KEP-3257)

This looks very good, but...

Trust Stores Are Not Uniform Across OSes

Distribution Family	Default CA Bundle Location	Individual CA Directory	Update Command
Debian / Ubuntu	<code>/etc/ssl/certs/ca-certificates.crt</code>	<code>/usr/local/share/ca-certificates/</code>	<code>update-ca-certificates</code>
RHEL / CentOS / Rocky / Alma / Fedora	<code>/etc/pki/tls/certs/ca-bundle.crt</code>	<code>/etc/pki/ca-trust/source/anchors/</code>	<code>update-ca-trust</code>
openSUSE / SLES	<code>/etc/ssl/ca-bundle.pem</code>	<code>/etc/pki/trust/anchors/</code>	<code>update-ca-certificates</code>
Alpine Linux	<code>/etc/ssl/certs/ca-certificates.crt</code>	<code>/usr/local/share/ca-certificates/</code>	<code>update-ca-certificates</code>

Trust Stores Are Not Uniform Across Languages/Libraries

Language / Libraries	Default Trust Source	Uses OS Trust Store?	Custom CA Injection Method
Go (crypto/x509)	OS trust store	✓ Yes	Install CA into OS store
Java (OpenJDK / Oracle JDK)	<code>cacerts</code> (JKS / PKCS12) inside JRE	✗ No	<code>keytool -import</code> into <code>cacerts</code>
Python (requests)	<code>certifi</code> bundle	✗ No	Replace <code>certifi</code> bundle or set <code>REQUESTS_CA_BUNDLE</code>
Python (ssl stdlib)	OS trust store	✓ Yes (on Linux)	Install CA into OS store
Node.js	Built-in Mozilla CA bundle	✗ No	<code>NODE_EXTRA_CA_CERTS</code> env var

Discussion: Existing Approaches



KubeCon



CloudNativeCon

Europe 2026

Points	Writing Dockerfiles	Init-containers	Mounting static files
No cost to manage images	✗	✓	✓
Pod-template Simplicity	✓	✗	✓ Needs some volumes/volumeMounts
Delay before Starting Containers	✓	⚠ Longer than other approaches	✓
OS-aware / Language-aware injection	⚠ Operators need to care the base image.	⚠ Operators need to care the base image.	✗

Node Resource Interface (NRI)



KubeCon



CloudNativeCon

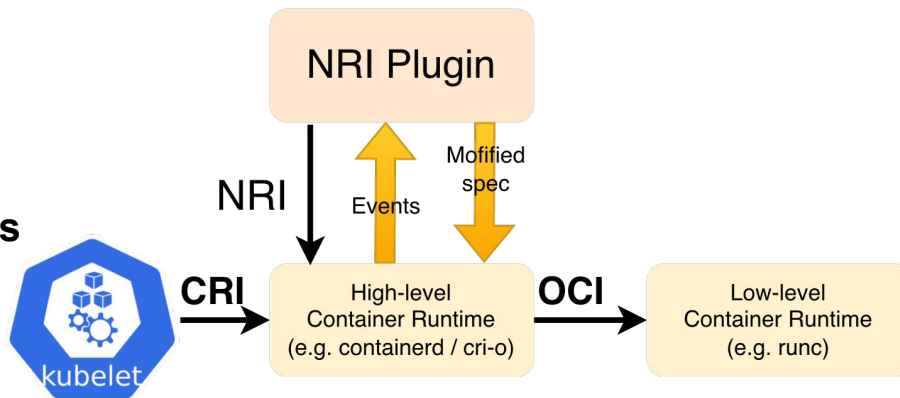
Europe 2026

A plugin API for container runtimes that lets external plugins customize container lifecycle and resource management.

<https://github.com/containerd/nri>

Key Points

- **Intercept container lifecycle events**
(Create, Start, Post-Start, etc.).
- **Mutate container configuration**
(env, mounts, devices, args).
- **Enable consistent behavior across runtimes**
(e.g., containerd, CRI-O).
 - *Enabled by default from containerd 2.0*



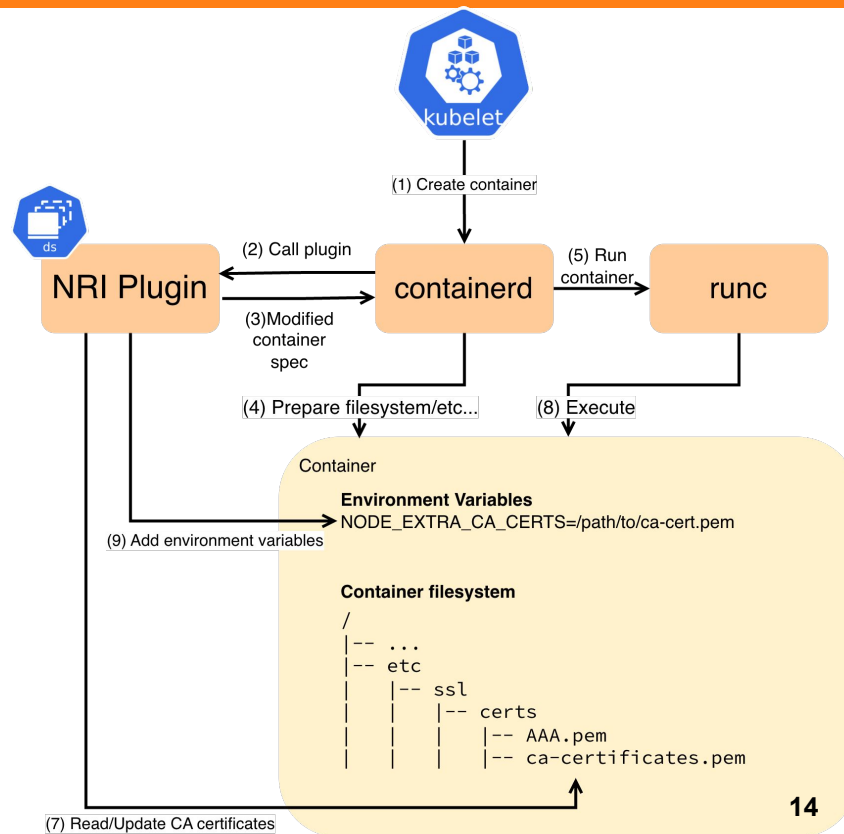
Automated CA Trust Injection Using NRI

During the container startup...

- Detects OS and languages
 - Debian / RHEL / Alpine / etc...
 - Node.js / python / etc...
- Appends CA certificates
 - e.g. Modifies /etc/ssl/certs/ca-certificates.pem
- Modifies environment variables if needed
 - `$NODE_EXTRA_CA_CERTS`, etc...

Advantages:

- No need to care the differences of base images / languages
- Can be enabled with a single Pod annotation




```
> bash ./generate-certs.sh
Certificates are generated!
  CA certificate: /tmp/cainjekt-demo-ca.crt
  Server certificate: /tmp/cainjekt-demo-server.key
  Server key: /tmp/cainjekt-demo-server.crt
> █
```

```
> bash ./generate-certs.sh
Certificates are generated!
  CA certificate: /tmp/cainjekt-demo-ca.crt
  Server certificate: /tmp/cainjekt-demo-server.key
  Server key: /tmp/cainjekt-demo-server.crt
> bash ./run-https-server.sh
secret/https-server-tls created
pod/https-server created
service/https-server created
> █
```

```
> bash ./generate-certs.sh
Certificates are generated!
CA certificate: /tmp/cainjekt-demo-ca.crt
Server certificate: /tmp/cainjekt-demo-server.key
Server key: /tmp/cainjekt-demo-server.crt
```

```
> bash ./run-https-server.sh
secret/https-server-tls created
pod/https-server created
service/https-server created
```

```
> kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
https-server	ClusterIP	10.96.128.209	<none>	8443/TCP	3s

```
> █
```

```
> cat client-plain.yaml
apiVersion: v1
kind: Pod
metadata:
  name: curl-plain
spec:
  containers:
  - name: curl
    image: ghcr.io/tsuzu/debian-curl
    command:
    - sh
    - -lc
    - |
      curl --no-progress-meter https://https-server:8443/healthz || true
      sleep 3600
> █
```

```
> cat client-plain.yaml
apiVersion: v1
kind: Pod
metadata:
  name: curl-plain
spec:
  containers:
  - name: curl
    image: ghcr.io/tsuzu/debian-curl
    command:
    - sh
    - -lc
    - |
      curl --no-progress-meter https://https-server:8443/healthz || true
      sleep 3600
> kubectl apply -f client-plain.yaml
pod/curl-plain created
> kubectl logs curl-plain
curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: https://curl.se/docs/sslcerts.html
```

curl failed to verify the legitimacy of the server and therefore could not establish a secure connection to it. To learn more about this situation and how to fix it, please visit the web page mentioned above.

```
> kubectl -n kube-system create configmap cainjekt-ca-bundle --from-file=ca-bundle.  
pem=/tmp/cainjekt-demo-ca.crt  
configmap/cainjekt-ca-bundle created  
> kubectl -n kube-system apply -f rbac.yaml -f daemonset.yaml  
serviceaccount/cainjekt unchanged  
clusterrole.rbac.authorization.k8s.io/cainjekt unchanged  
clusterrolebinding.rbac.authorization.k8s.io/cainjekt unchanged  
daemonset.apps/cainjekt created  
> █
```

```
> kubectl -n kube-system create configmap cainjekt-ca-bundle --from-file=ca-bundle.
pem=/tmp/cainjekt-demo-ca.crt
configmap/cainjekt-ca-bundle created
> kubectl -n kube-system apply -f rbac.yaml -f daemonset.yaml
serviceaccount/cainjekt unchanged
clusterrole.rbac.authorization.k8s.io/cainjekt unchanged
clusterrolebinding.rbac.authorization.k8s.io/cainjekt unchanged
daemonset.apps/cainjekt created
> kubectl -n kube-system get daemonsets -l app=cainjekt
```

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
cainjekt	1	1	1	1	1	<none>	9s

```
> █
```


How Cert Injection Works?



KubeCon



CloudNativeCon

Europe 2026

3 steps in our implementation

- **NRI plugin:** CreateContainer hook
- **OCI Hook:** createRuntime hook
- **Container entrypoint:** env modifier



<https://github.com/tsuzu/cainjekt>

How Cert Injection Works?: (1) NRI plugin



KubeCon



CloudNativeCon

Europe 2026

Subscribes to **CreateContainer** events

On a CreateContainer event:

- Checks annotations on Pods
 - Whether to enable the plugin
- Adds an **OCI hook**
- Inject the **Env modifier**
 - Mounts the env modifier executable
 - Insert the env modifier to args

No files/envs are edited in this step

```
hook := &api.Hook{
    Path: self,
    Env: []string{
        config.EnvHookMode + "=" + config.ModeCreateRT,
        config.EnvCAFile + "=" + caFileForHook,
        config.EnvFailPolicy + "=" + config.FailPolicyOpen,
        config.EnvHookContextFile + "=" + config.HookContextFile,
    },
    Timeout: api.Int(config.DefaultHookTimeoutSec),
}
```

```
adjustment := &api.ContainerAdjustment{}
adjustment.AddHooks(&api.Hooks{CreateRuntime: []*api.Hook{hook}})
```

```
adjustment.AddMount(&api.Mount{
    Destination: config WrapperPath,
    Type:        "bind",
    Source:      self,
    Options:     []string{"bind", "ro"},
})
if args := ctr.GetArgs(); len(args) > 0 && args[0] != config WrapperPath {
    adjustment.UpdateArgs(append([]string{config WrapperPath}, args...))
}
return adjustment, nil, nil
```

How Cert Injection Works?: (2) OCI Hook



KubeCon



CloudNativeCon

Europe 2026

What are OCI Hooks?

- A standardized extension mechanism defined by the OCI Runtime Specification.
 - For low-level container runtimes: runc, crun, youki, etc...
- Several hook points exist in the container lifecycle.

On createRuntime event:

- Detects OS and languages by reading files on container images
- Updates CA certificates files in containers
 - e.g. /etc/ssl/certs/ca-certificates.pem

```
info, err := readOSRelease(ctx.Rootfs)
if err != nil {
    // ...
}
if p.matches(info) {
    return hookapi.DetectResult{
        Applicable: true,
        Priority:    p.priority,
    }
}
```

How Cert Injection Works?: (3) Container entrypoint



KubeCon



CloudNativeCon

Europe 2026

In containers, **env modifier** is executed as the entrypoint.

Env modifier does

- Modifies environment variables
- Calls `execve(2)` system call and start the container

```
individualCAPath, ok := ctx.Facts.Get(hookapi.FactIndividualCAPath)
if !ok {
    return nil
}
individualCAPath = strings.TrimSpace(individualCAPath)
if individualCAPath == "" {
    return nil
}
ctx.Env = upsertEnv(ctx.Env, "NODE_EXTRA_CA_CERTS", individualCAPath)
return nil
```

```
containerd
└─ container-shim
    └─ runc
        └─ ↓ execve(2)
            └─ L env modifier
                └─ ↓ execve(2)
                    └─ L original entrypoint
```

Comparison: Comparison with Existing Approaches



KubeCon



CloudNativeCon

Europe 2026

Points	Writing Dockerfiles	Init-containers	Mounting static files	NRI plugin
No cost to manage images	✗	✓	✓	✓
Pod-template Simplicity	✓	✗	✓	✓
Delay before Starting Containers	✓	⚠	✓	✓
OS-aware / Language-aware injection	⚠	⚠	✗	✓
Without Node-level privileges	✓	✓	✓	✗ *

* To mitigate the risk, we'll add a sandbox with RESOLVE_IN_ROOT for openat2 (2).



- Environment variables aren't modified on `kubectl exec`
 - Env modifier is executed for container entrypoints
- Not available for read-only filesystem
 - Trying to resolve it by adding bind mounts for replacement files
- Integration with Cluster Trust Bundles (KEP-3257)
 - Standard way to distribute CA certificates to workloads in Kubernetes.

Summary and Outlook: NRI for Operational Automation

- NRI-based approach enables OS- / language-aware CA trust injection.
- Suggests that NRI can be a useful extension point for broader automation.
- Aiming for deep integration with Kubernetes standards, including Cluster Trust Bundles

Feedback is welcome!



<https://sched.co/2CW42>