CLOUD NATIVE + OPEN SOURCE

Virtual Summit China 2020

Putting an Invisible Shield on Kubernetes Secrets

Kailun Qin, Ant Group

Agenda

- K8s Secrets: Overview
- TEE-based K8s Secrets Protection: Solution
- Production Experience @ Ant Group
- Demo
- Summary & Plan

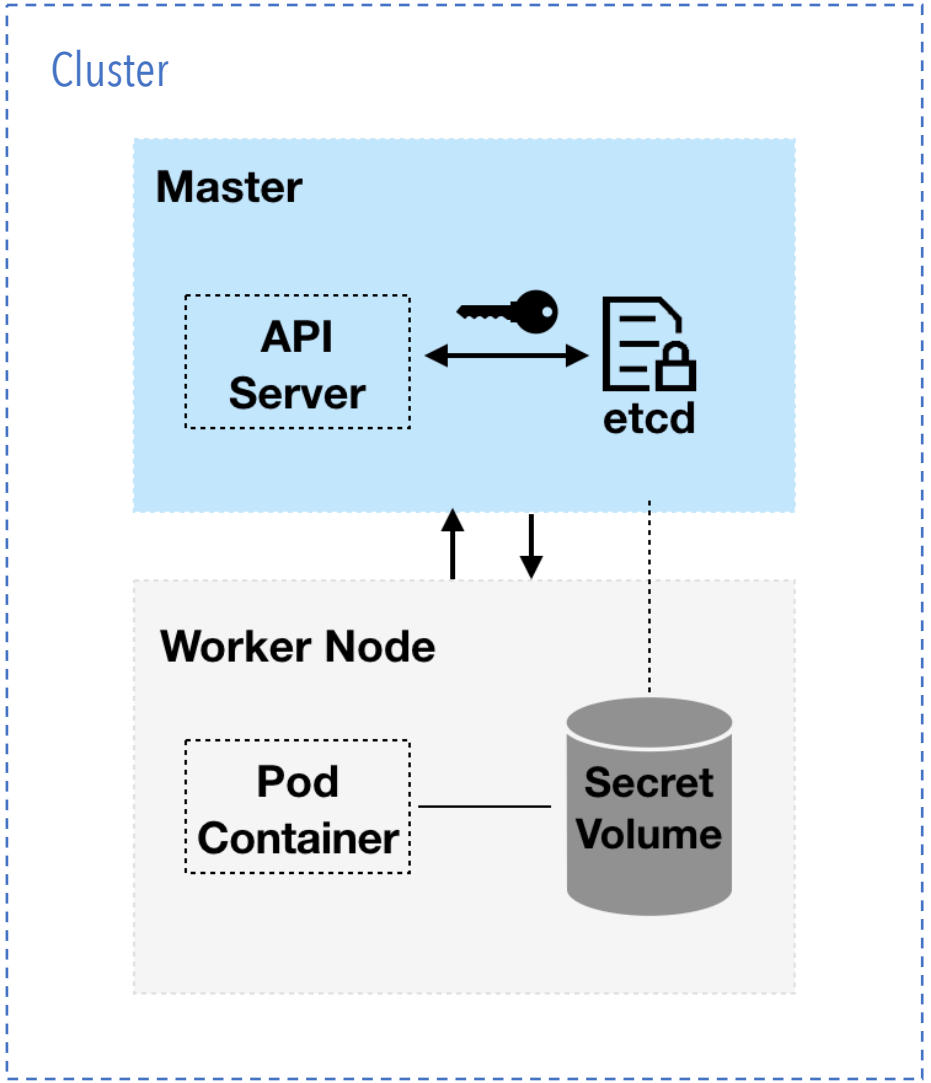
CLOUD NATIVE + OPEN SOURCE

Virtual Summit China 2020

K8s Secrets: Overview

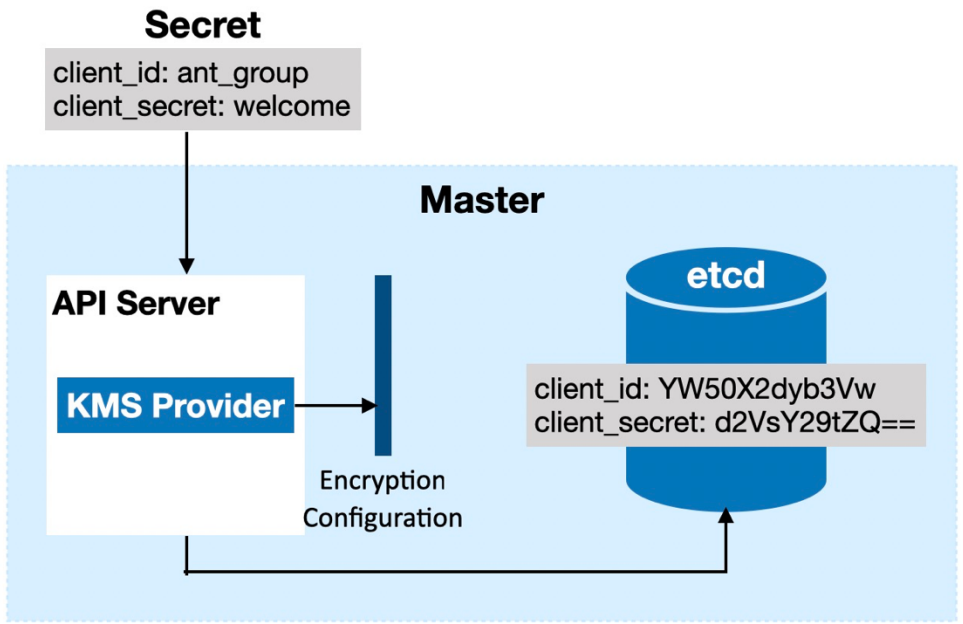
Background: K8s Secrets

- **What they are?**
  - Sensitive information
    - Passwords
    - OAuth tokens
    - ssh keys etc.
  - Stored in etcd
    - distributed Key-Value data store
- **How about their security?**
  - Default K8s setup
    - etcd contents not encrypted (only base64 encoded)
  - > K8s 1.7+
    - at-rest encryption for etcd (local + remote)



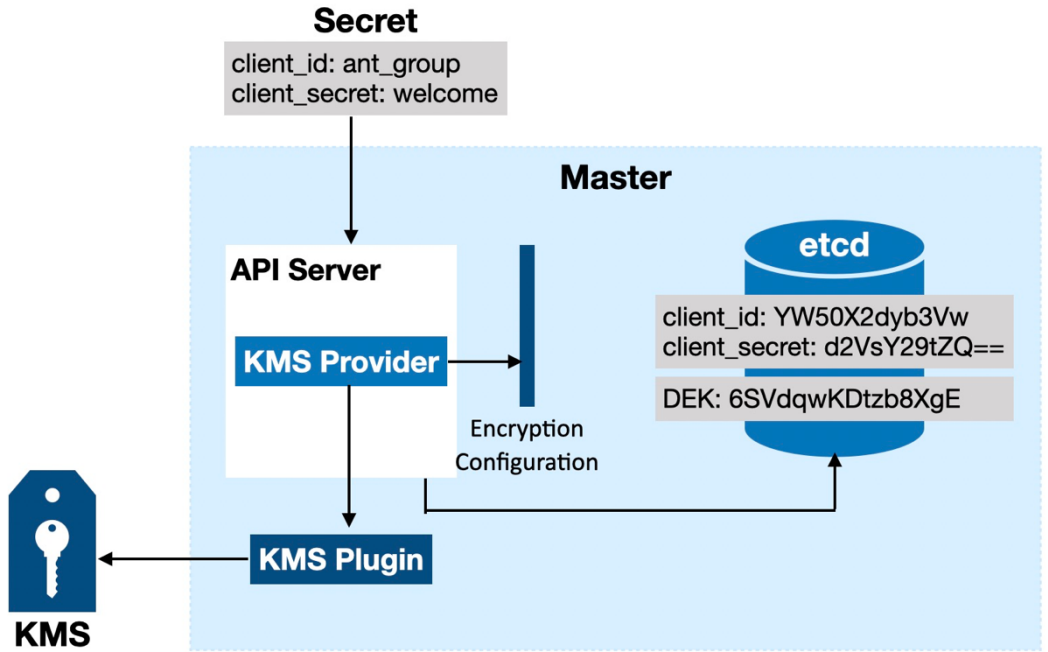
Background: K8s Secrets

Local Encryption Provider



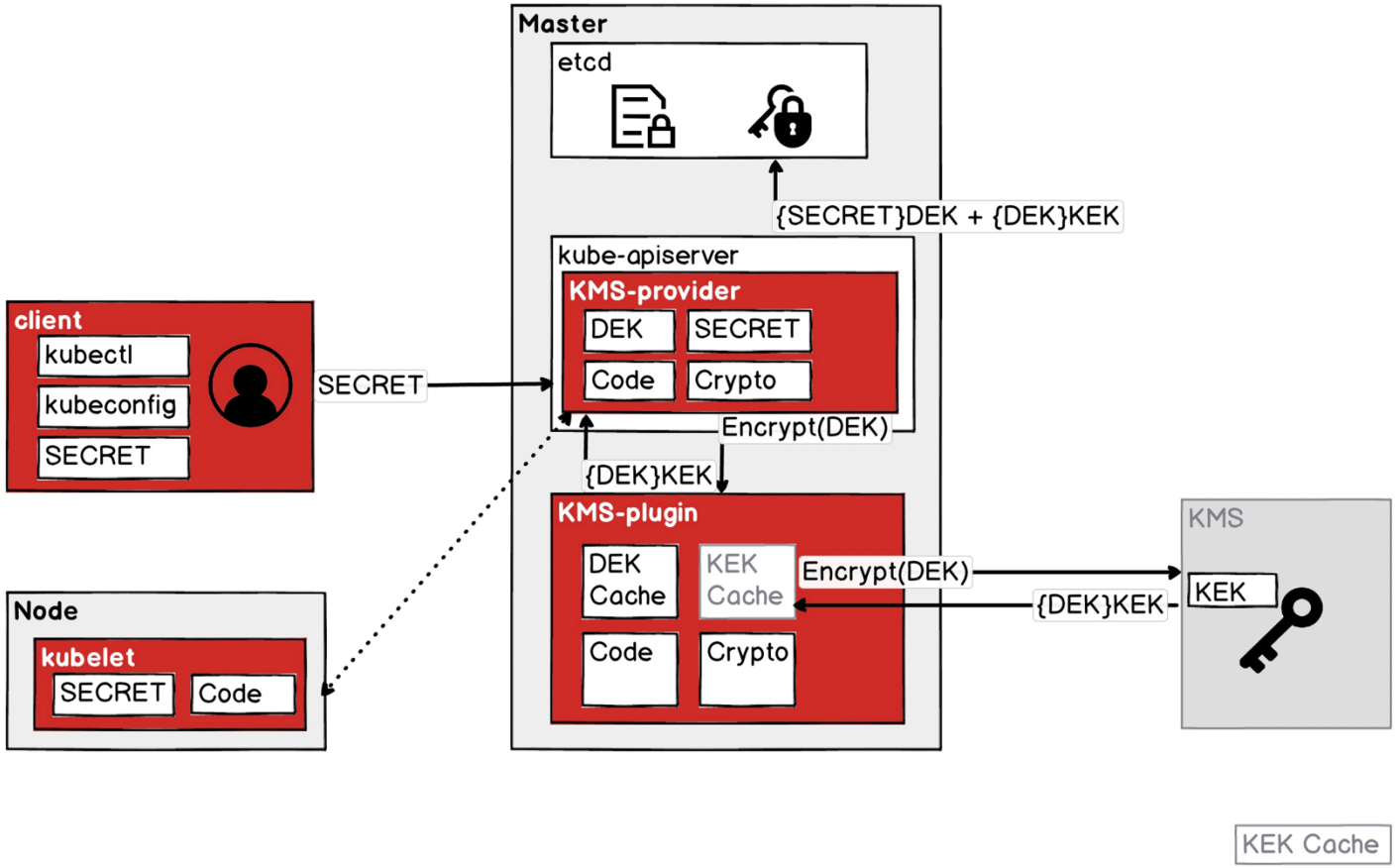
- Encryption Keys stored on API Server
- Secrets encrypted prior to storage in etcd
- Secrets decrypted on API Server prior to use

KMS Encryption Provider



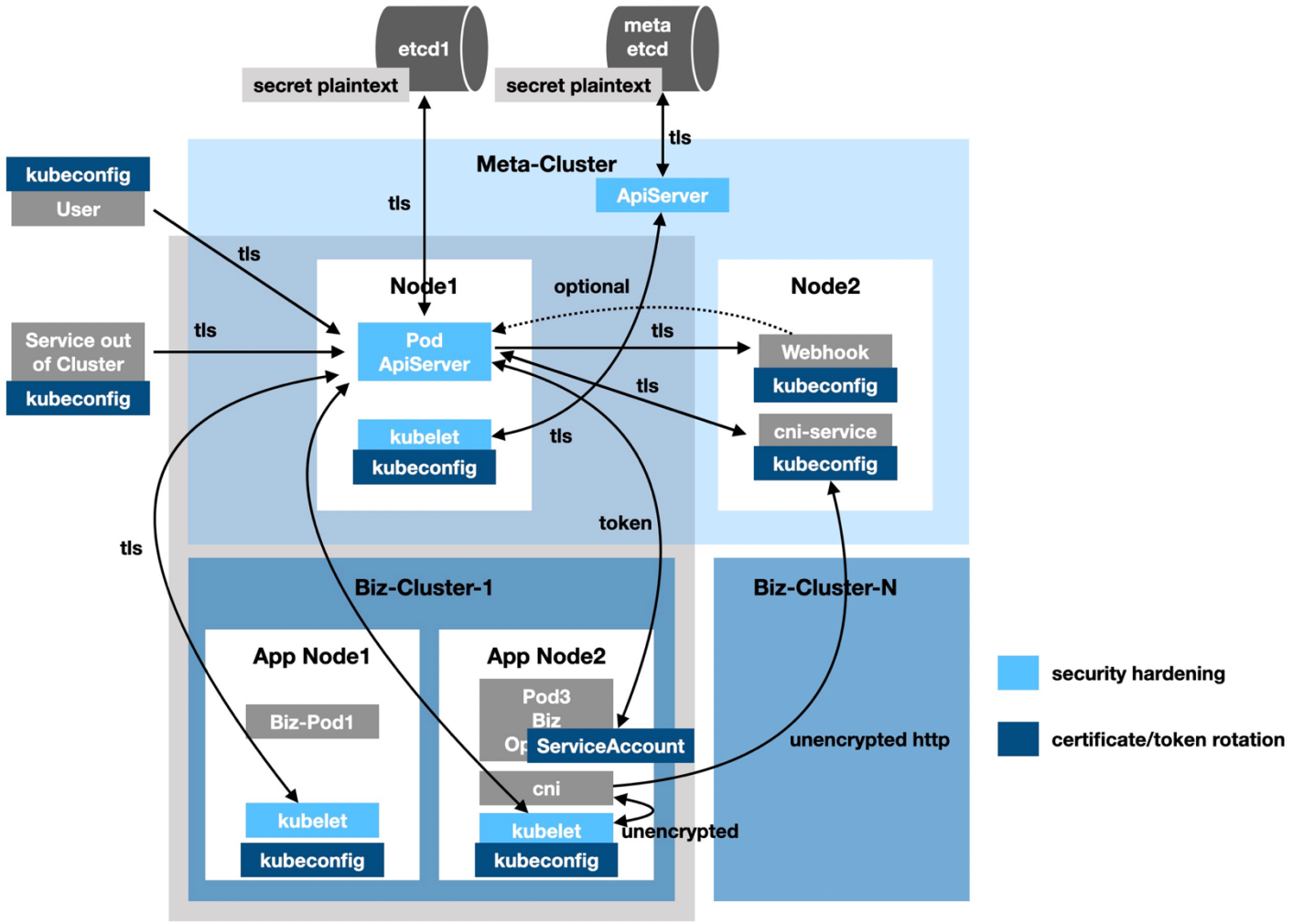
- Encryption keys stored in a remote KMS
- Use envelope encryption scheme
  - DEK & KEK

Motivation: K8s Secrets Protection



- **Performance & latency**
  - Network
- **Security**
  - DEK in the clear in memory
  - Secret in the clear in memory
  - kubeconfig in the clear in memory
  - KEK in the clear in memory
    - ✓ leak ALL DEKs
    - ✓ leak ALL secrets
    - ✓ **trust collapse!**
  - DEK decryption interfaces invoked by fake users

Motivation: K8s Secrets Protection



- Kube-on-Kube [1]
  - ✓ Components
    => **too many!**
  - ✓ Interactions
    => **complicated!**
  - ✓ User access management
    => **raw and extensive!**
  - ✓ Secrets management
    => **crucial!**
- Financial-grade security

[1] KubeCon China 2018: Node Operator: Kubernetes Node Management Made Simple - Joe Chen, Ant Financial

CLOUD NATIVE + OPEN SOURCE

Virtual Summit China 2020

TEE-based Secrets Protection: Solution

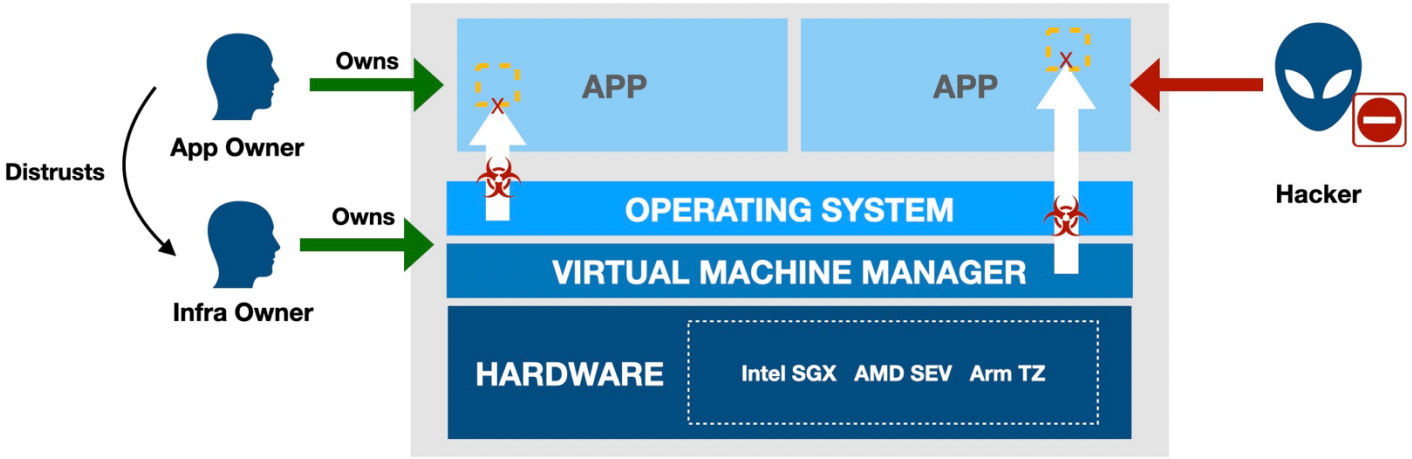
Confidential Computing

A **Trusted Execution Environment** (TEE) is

- a secure area protected by the processor (aka. Enclave)

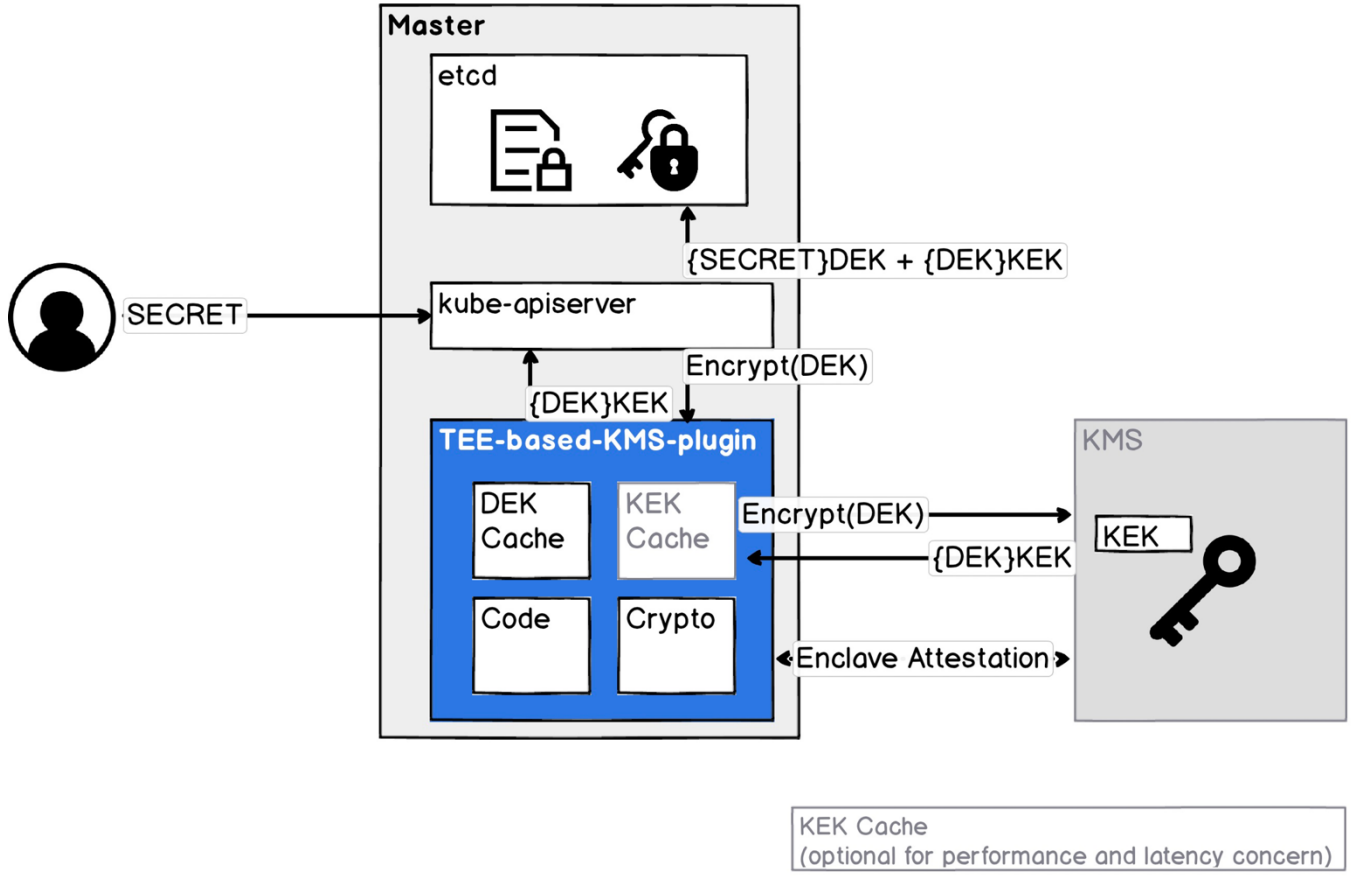
Example: Intel SGX

- Strong isolation
- Encrypted memory
- SW/HW attacks prevented



TEE-based KMS Plugin [1]

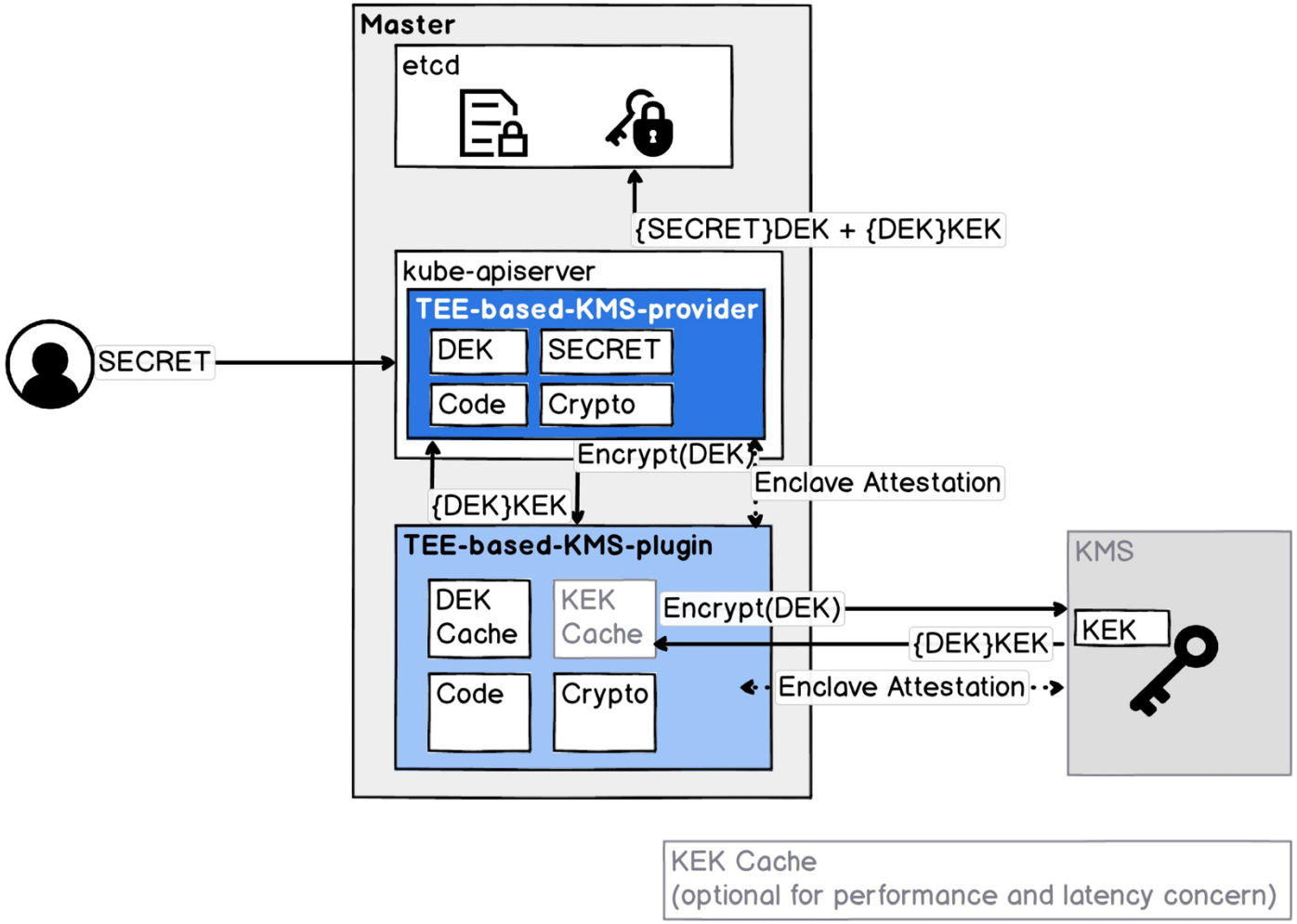
- Address performance & latency concerns
  - Reduce / minimize remote KMS interactions w/o compromising security
- Address security threats
  - etcd compromise
  - Host (KMS plugin) compromise
    - ➢ leak DEKs
    - ➢ leak KEKs



[1] KubeCon NA 2019: "TEE-based KMS Plugin for encryption of Kubernetes Secrets", by Raghu Yeluri & Haidong Xia, Intel Corp.

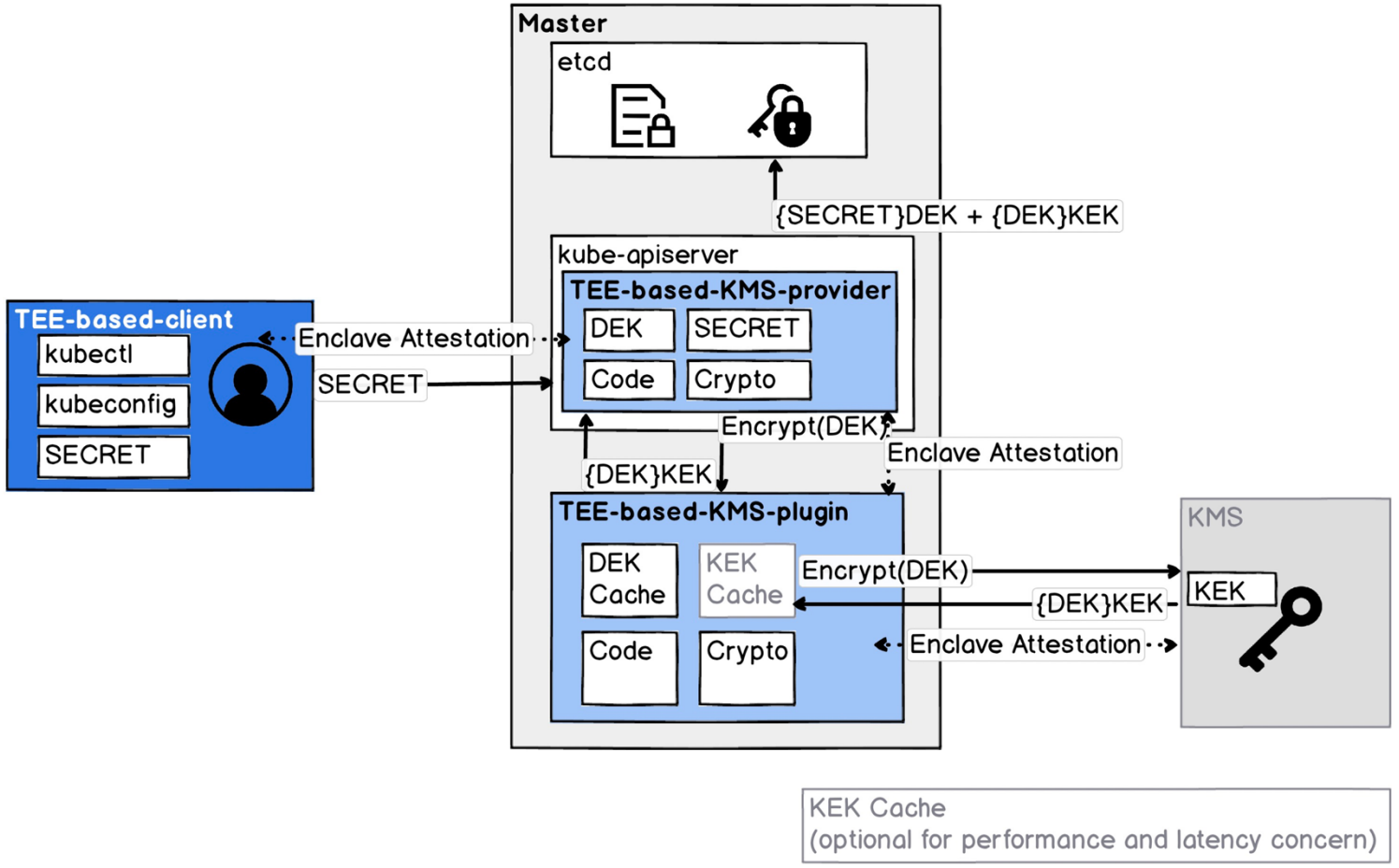
TEE-based KMS Provider

- Address security threats
  - Host (KMS provider) compromise
    - leak DEKs
    - leak Secrets
  - Fraudsters calling DEK decryption interfaces



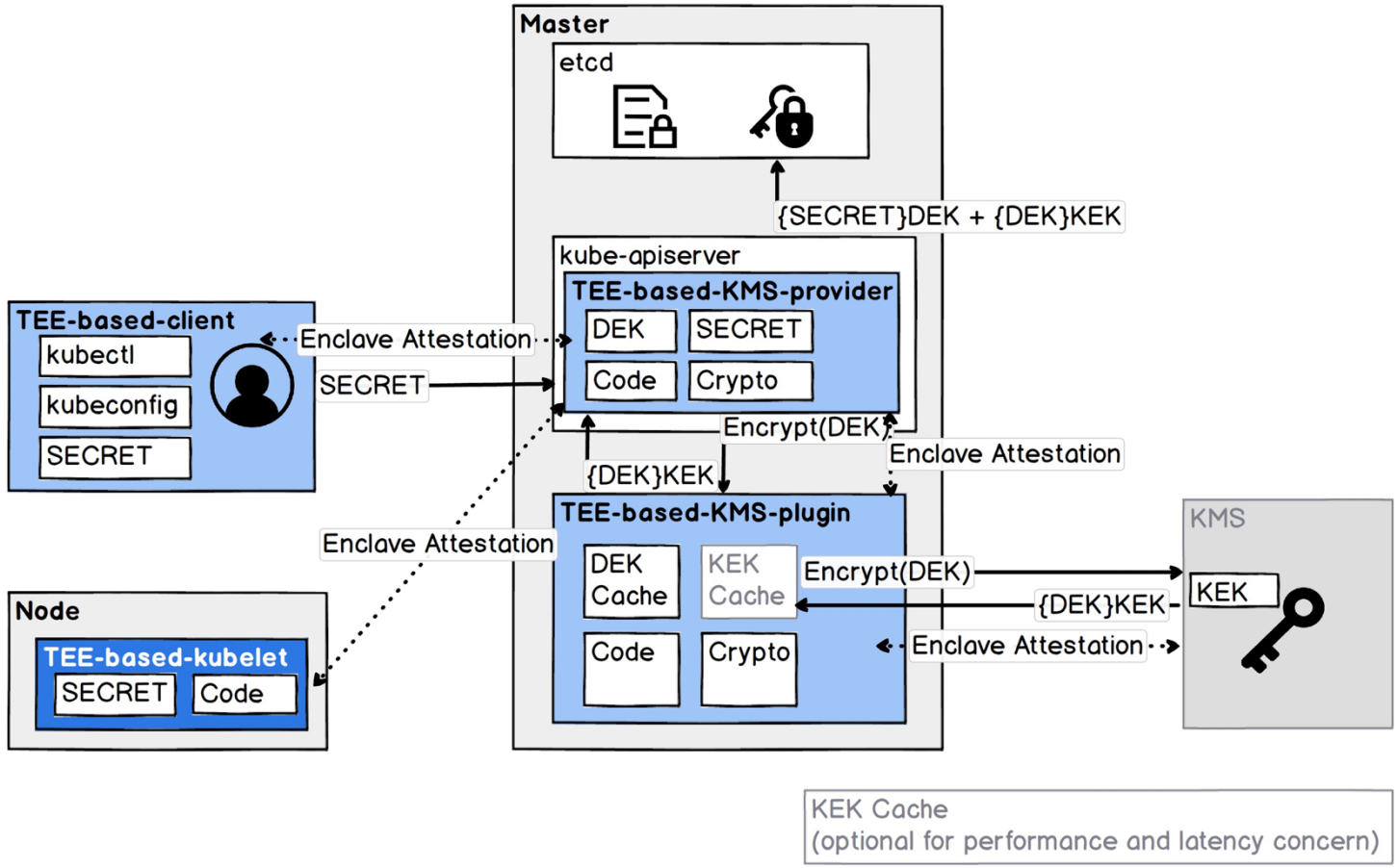
TEE-based Kubectl

- Address security threats
  - Client compromise
    - ➢ kubeconfig maliciously reused by attackers
    - ➢ kubeconfig in the clear in clients' memory
    - ➢ leak users' secrets
  - Sending to / receiving from malicious software entity (logic)

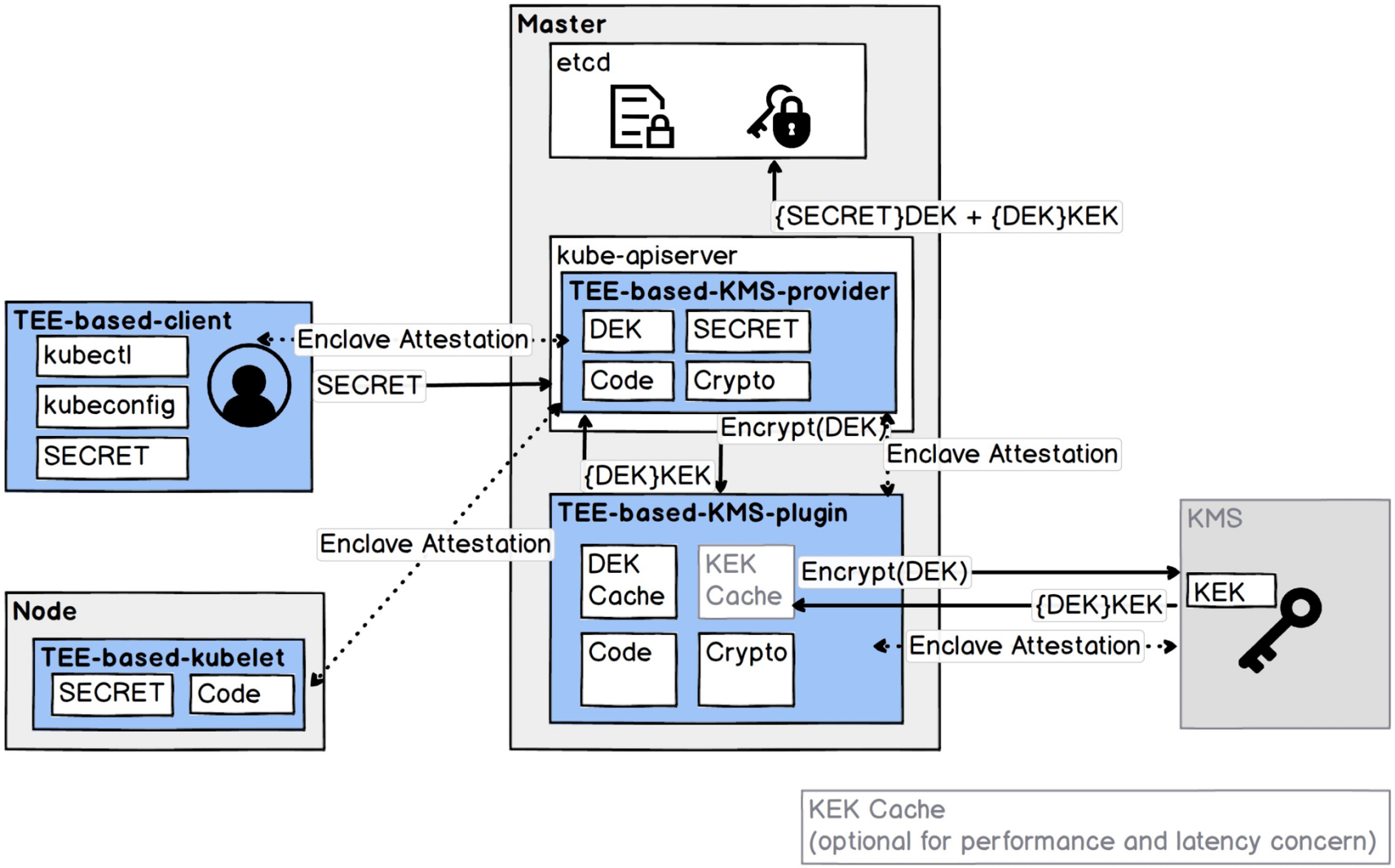


TEE-based Kubelet

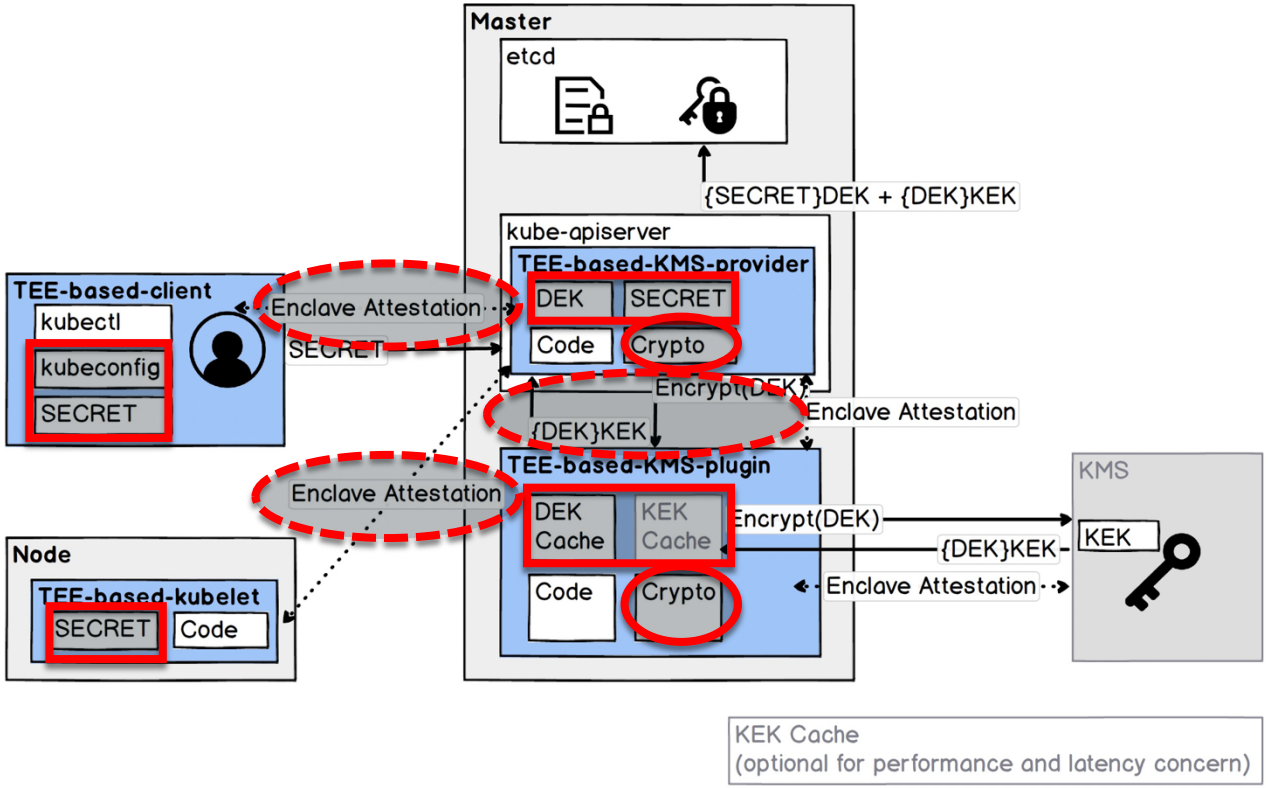
- Address security threats
  - Node (kubelet) compromise
    - leak secrets on consumption



TEE-based Secrets Protection



TEE-based Secrets Protection (cont.)



In a nutshell…

- Protecting DEKs / KEKs / Secrets / kubeconfig in memory
- Protecting the encryption / decryption of DEKs / Secrets
- Introducing mutual (remote / local) attestations between entities

CLOUD NATIVE + OPEN SOURCE

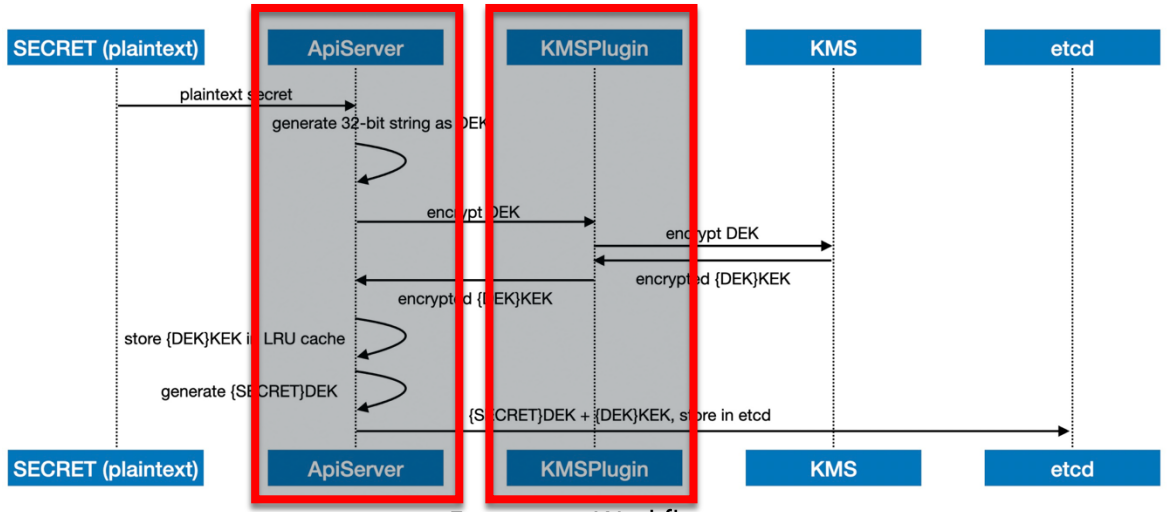
Virtual Summit China 2020

Production Experience @ Ant Group

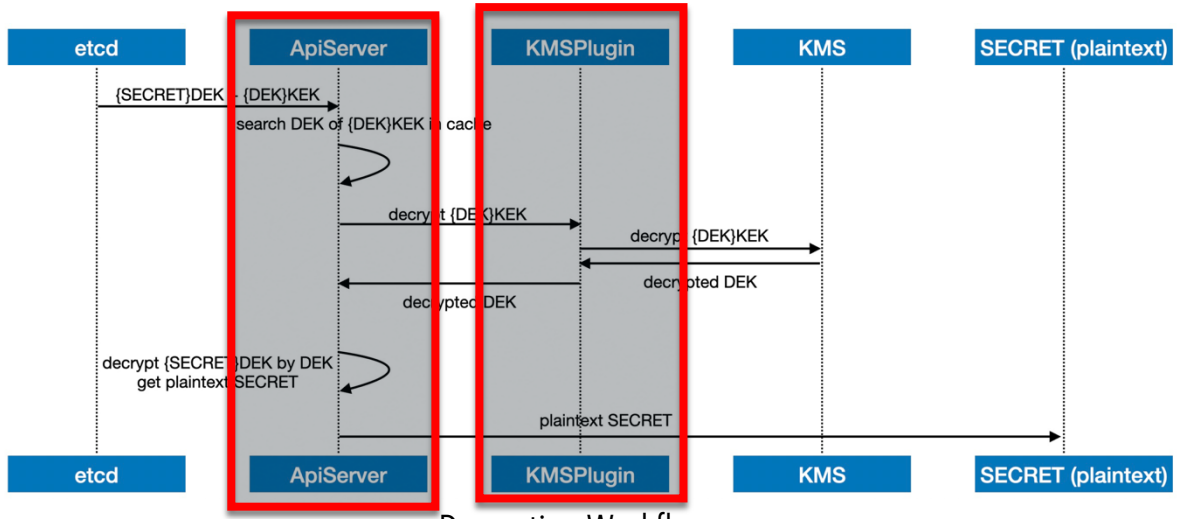
KMS Plugin

- **Workflow**
  - Encryption
  - Decryption

- **Engineering decisions**
  - apiserver is responsible for
    - DEK generation
    - Secret en/decryption
  - kms-plugin
    - keeps KEK cache
    - only en/decrypts DEK, not secrets



Encryption Workflow



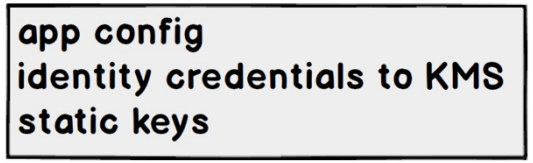
Decryption Workflow

KMS Plugin (cont.)

- **Deployment Modes**
  - One kms-plugin container per Master Node: sidecar to apiserver
  - Use Annotation to enable encrypted secret read / write
  - LivenessProbe for health check

- **Configurations**
  - kms-plugin
  - apiserver

- **Caching**
  - API server
    - Set up Encrypted(DEK) => DEK mapping
  - KMS plugin
    - Set up SecretKeyName:SecretKeyVersion => SecretKeyData mapping



kms-plugin config

```yaml
apiVersion: apiserver.config.k8s.io/v1
kind: EncryptionConfig
resources:
  - resources:
    - secrets
    providers:
    - kms:
        name: kms-plugin
        endpoint: unix:///var/run/kmsplugin/kms-plugin.sock
        cachesize: 100
        timeout: 3s
    - identity: {}
```

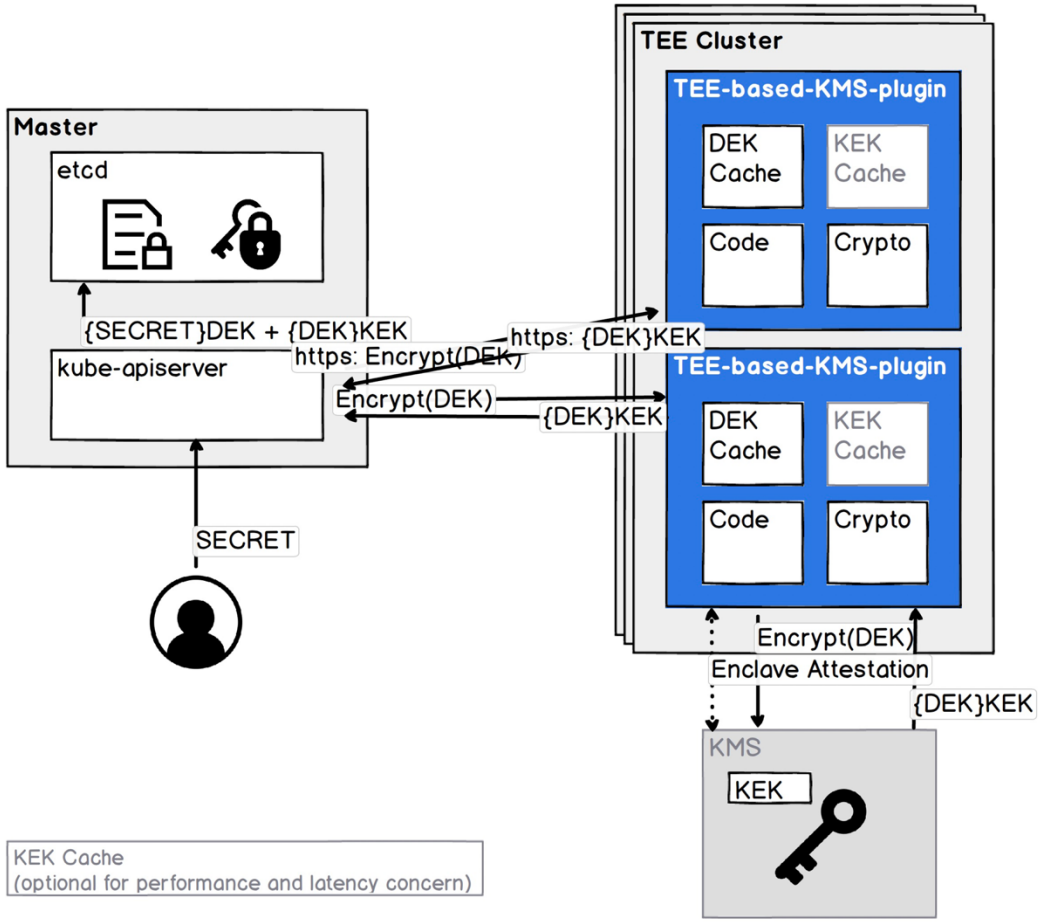
apiserver config

KMS Plugin (cont.)

- **Gray release**
  - extensions-webhook: /mutating-secret
  - Annotation: /storage-transform-disable=<bool>
- **Emergency management**
  - High Availability guarantee
    - KMS
    - API server & kms-plugin
  - Cron job backup for KEKs (from KMS)
  - Static key configuration support in kms-plugin
  - One click decryption
  - Key force update
  - Liveness probe
- **Monitoring**
  - Integration w/ Prometheus
  - Metrics including
    - latency of en/decryption
    - failure times of en/decryption
    - KMS health check
- **Ops tooling**
  - kms-plugin-tools

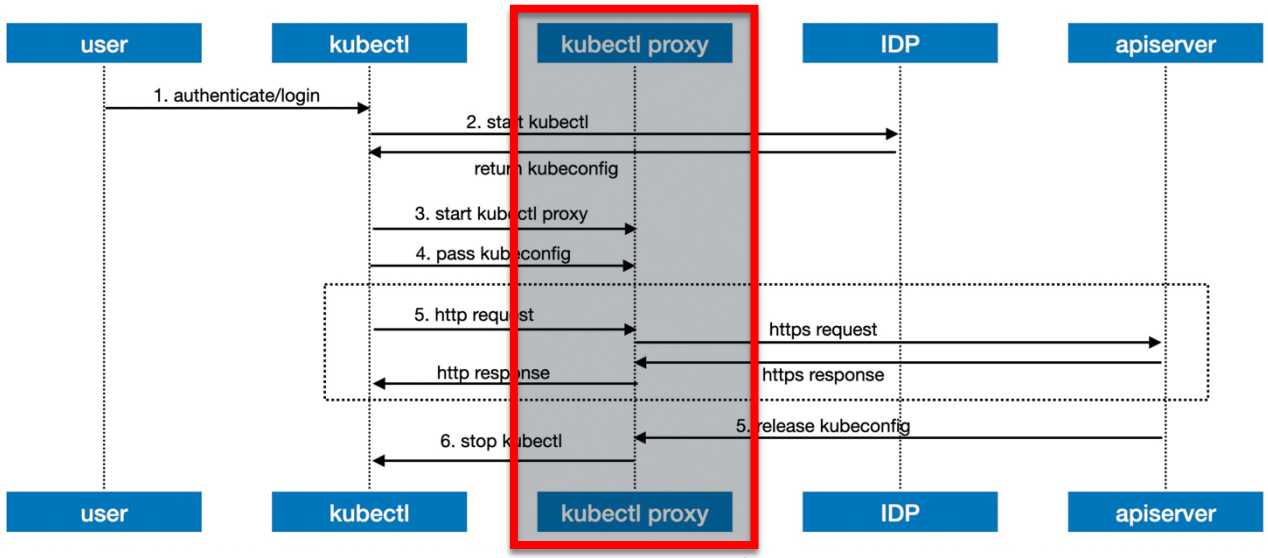
KMS Plugin as a Service

- **Motivation**
  - SGX physical servers do not meet API servers' performance requirements
- **Solution**
  - Same TEE-based KMS-plugin runtime
  - Deployment modes
    - N (>=3) SGX servers deployed w/ sgx-device-plugin daemonset [1]
    - kms-plugins deployed as deployment
  - Interfaces
    - https + connection reuse
    - certificate: similar to apiserver ⇔ etcd (X.509)
  - Version-based key synchronization
  - Adaption
    - apiserver KMS provider endpoint to support https endpoint
    - KMS plugin to support https



[1] https://github.com/AliyunContainerService/sgx-device-plugin

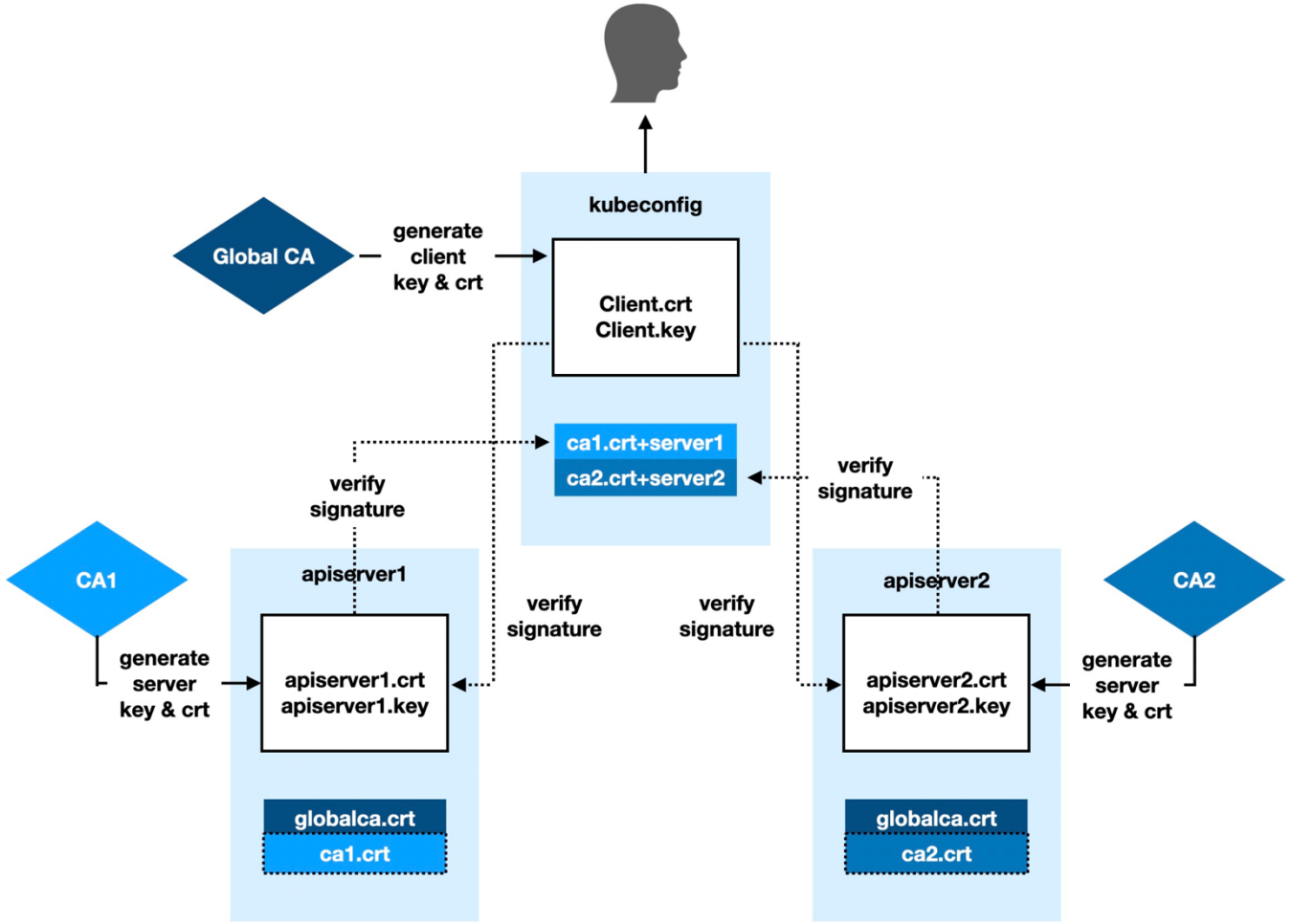
Secure Kubectl



Secure Kubectl Workflow

- **Design goal**
  - kubconfig transparent to kubectl users
  - kubeconfig credentials binding w/ identity
  - kubeconfig only in memory
  - TEE as an option
- **Solution**
  - Get kubeconfig
    - Relay server mode
    - Non-relay server mode
  - Keep kubeconfig in memory
    - proxy mode
    - kubectl ⇔ http/uds ⇔ proxy ⇔ https ⇔ apiserver
  - X.509 or OIDC Token

Secure Kubectl (cont.)

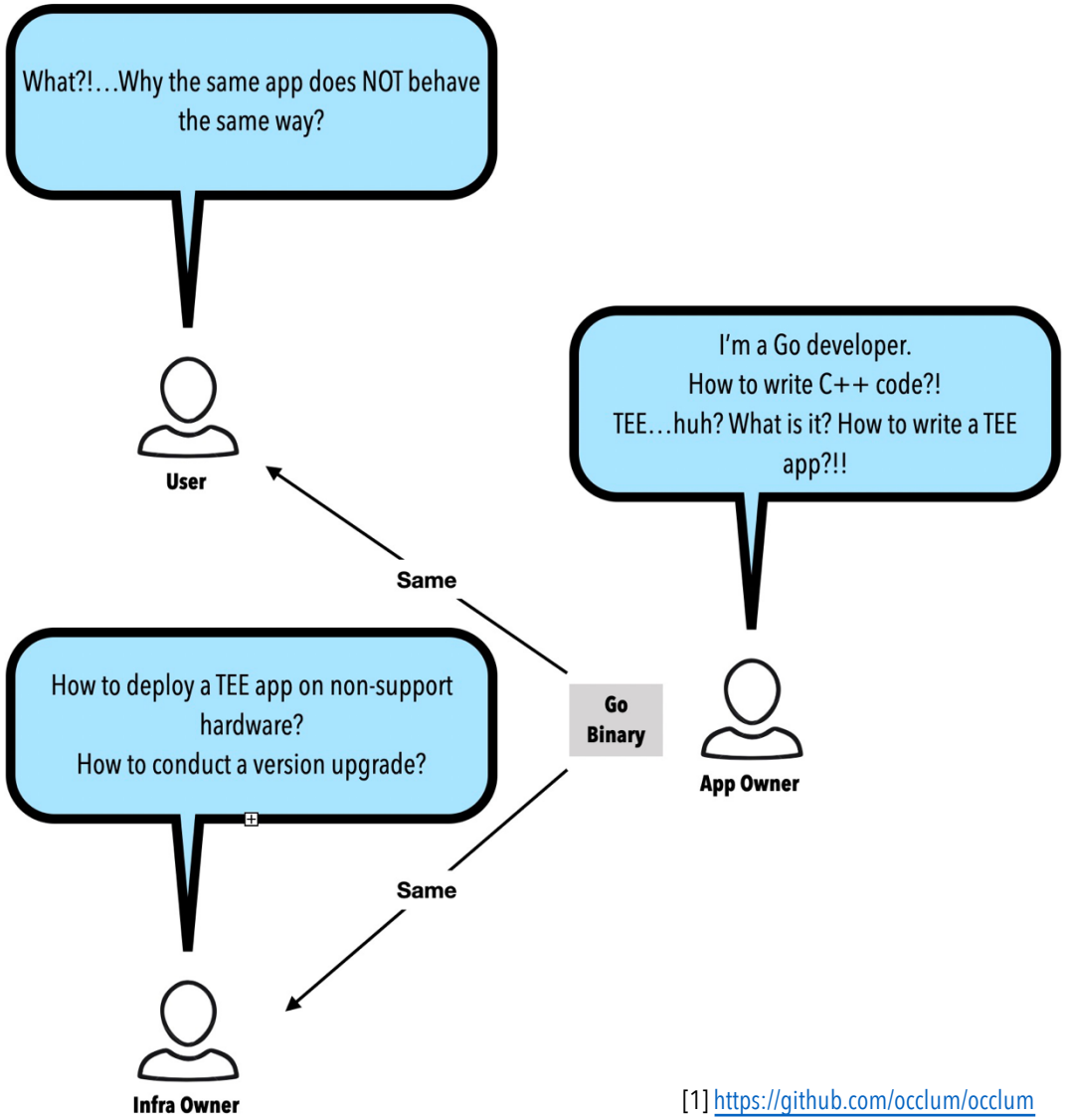


- Global CA
- One kubeconfig for multiple clusters

One binary: TEE Transparency

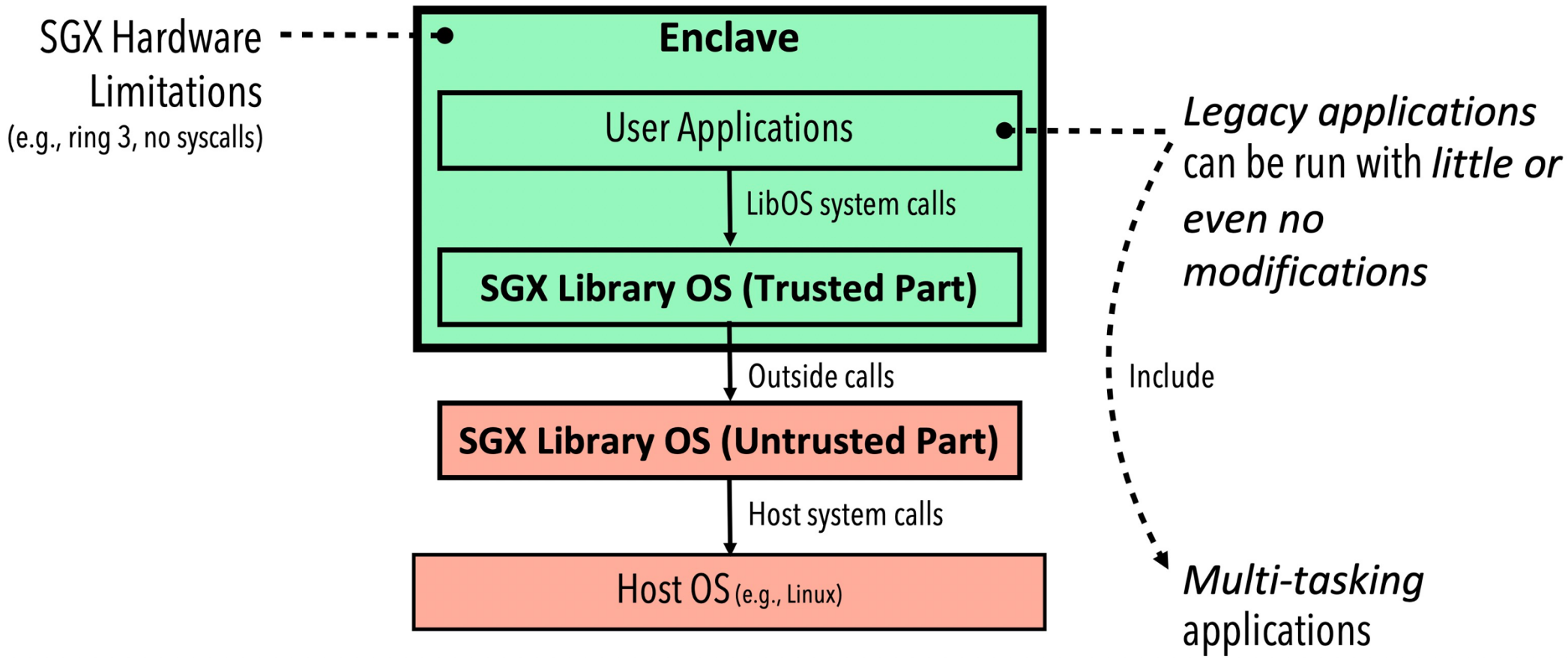
- **Motivation**
  - Leverage the same code base, thus the same
    - APIs, logic, iteration plan for developers
    - Experience for users/operators
  - TEE as an option, en/disable based on
    - Hardware configurations
    - Biz scenarios

- **Solutions evaluated**
  - Go KMS Plugin + C Go SGX Functions
  - C++ KMS Plugin
  - Rust KMS Plugin
  - ✓ Go KMS Plugin + TEE LibOS (Occlum [1])



[1] https://github.com/occlum/occlum

Occlum: SGX Dev Made Easy



Occlum: Major Features

Usability

Container-Inspired Interface

Performance

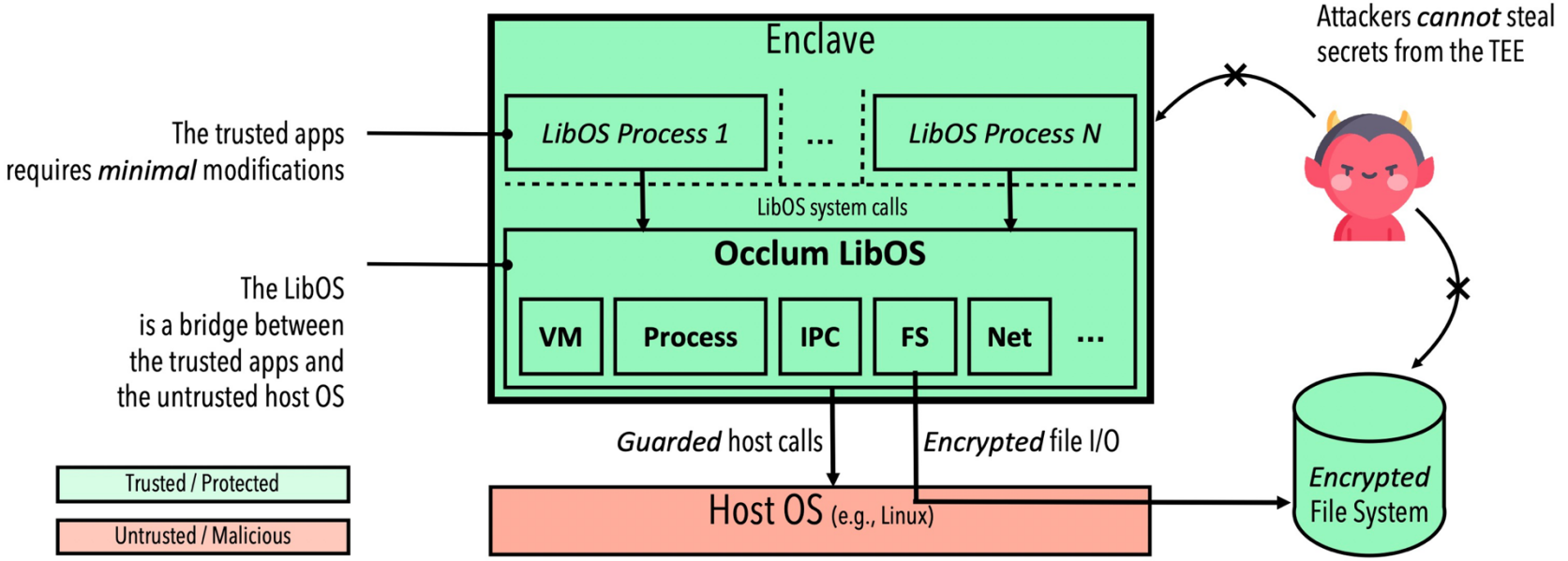
Efficient Multitasking

Functionality

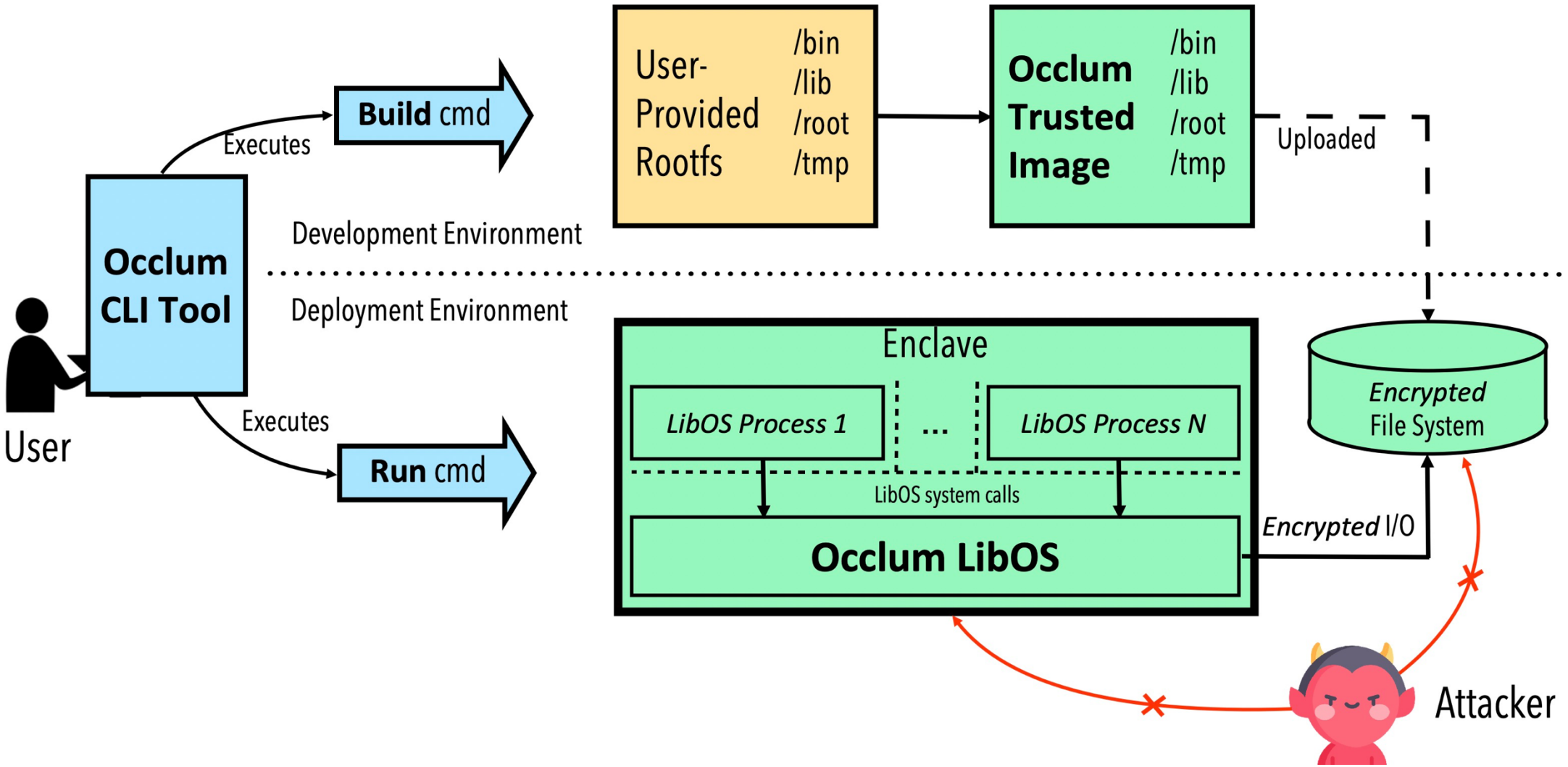
Multiple File Systems

Security

Memory Safety



Occlum: Container-Inspired Interface



CLOUD NATIVE + OPEN SOURCE

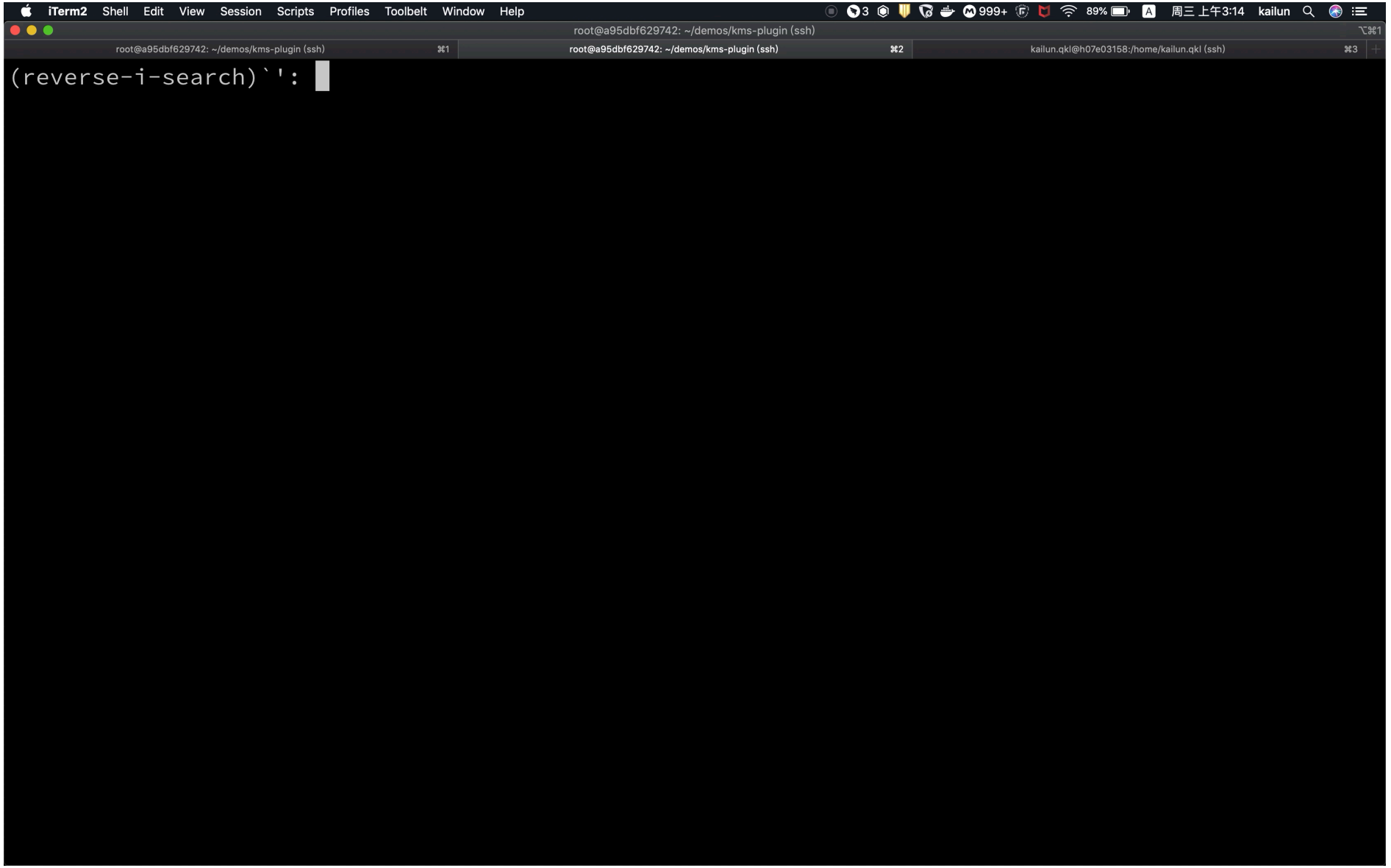
Virtual Summit China 2020

Demo

Demo

- The purpose of this demo is to
  - Demonstrate TEE Transparency w/ Occlum's Golang support
  - Showcase the confidentiality guaranteed by TEE

Demo


iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
周三 上午3:14 kailun
root@a95dbf629742: ~/demos/kms-plugin (ssh)
root@a95dbf629742: ~/demos/kms-plugin (ssh) ⌘1
root@a95dbf629742: ~/demos/kms-plugin (ssh) ⌘2
kailun.qkl@h07e03158:/home/kailun.qkl (ssh) ⌘3
(reverse-i-search)`':

CLOUD NATIVE + OPEN SOURCE

Virtual Summit China 2020

Summary & Plan

Summary & Next Steps

- **Summary**
  - A TEE-based E2E solution aiming to guard K8s secrets while in use, at rest, and in transit
  - TEE transparency empowered by LibOS (e.g. Occlum)

- **Where we are?**
  - TEE-based KMS plugin on the way to production
  - TEE-based KMS plugin as a Service - production in plan
  - Secure kubectl already in production; TEE-based secure kubectl completed PoC
  - API server and the rest changes for the TEE-based - PoC stage

- **Where we go?**
  - To keep the production practice
  - To explore more potential intersections between Cloud-Native and Confidential Computing
  - To submit KEPs to upstream for related changes

Special Thanks

We'd like to thank the contributors to this work:

Yinyong Zheng, Wei Zhang, Yong Zhang, Jun Chen, Yang Yang, Ai Jing, Huanxin Shi, Xiaoyun Mao, Ziteng Xu, Junxian Xiao, Shoumeng Yan and the whole Occlum Team

CLOUD NATIVE + OPEN SOURCE

Virtual Summit China 2020