Confidential (Encoded) Processing

There is no safety without security

Christof Fetzer , TU Dresden

Objectives

- Move **mission-/safety critical functionality** to the cloud
  - to reduce **costs**,
  - to increase **security**, and
  - to increase **availability**
  - to increase **safety**,

focus: cloud-native application

Use confidential computing

Use features of (untrusted) Kubernetes

Use encoded processing

Healthcare Confidential Computing

- reusable for other critical infrastructures -

Threat Model

- outsourcing changes the threat model -

Threat Model

Perfect forward security…

A1) **Unprivileged Software Adversary**

A2) **System Software Adversary**

A3) **Startup Code/SMM Software Adversary**

A4) **Network Adversary**

A5) **Software Side-Channel/Covert-Channel Adversary**

A6) **Simple Hardware Adversary**

*A7) **Roll-Back State Adversary***

*A8) **CVE Adversary***

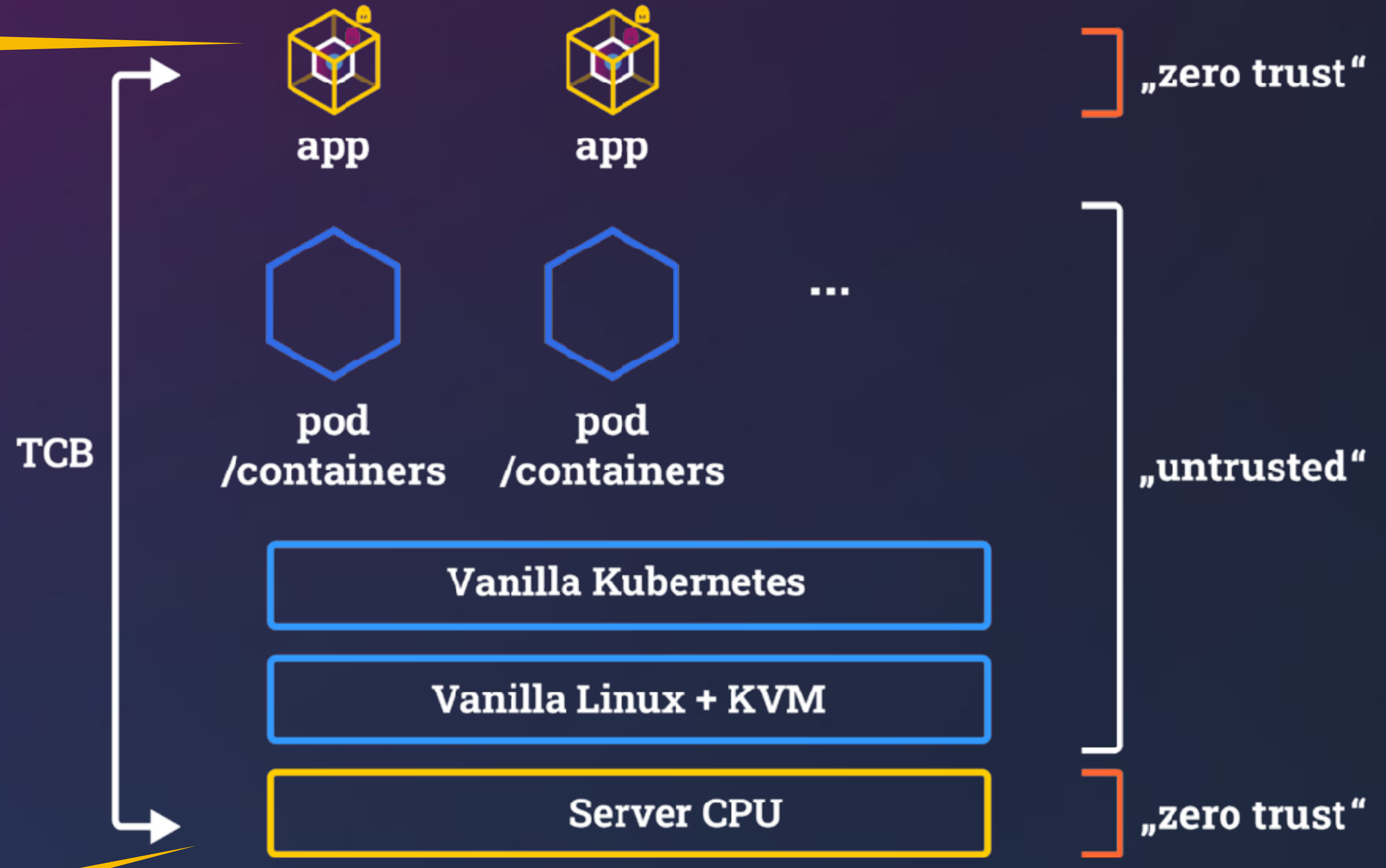
*A9) **Insider attacks from Security Team***

Summary: Skilled adversary that has root access to the cluster and knows all CVEs

https://www.intel.com/content/www/us/en/security/security-practices/secure-development-practices/threat-modeling.html

Approach: Small TCB

- CPU with TEE support
  - **trust after attestation & verification**
- Services of application
  - **trust after attestation & verification**



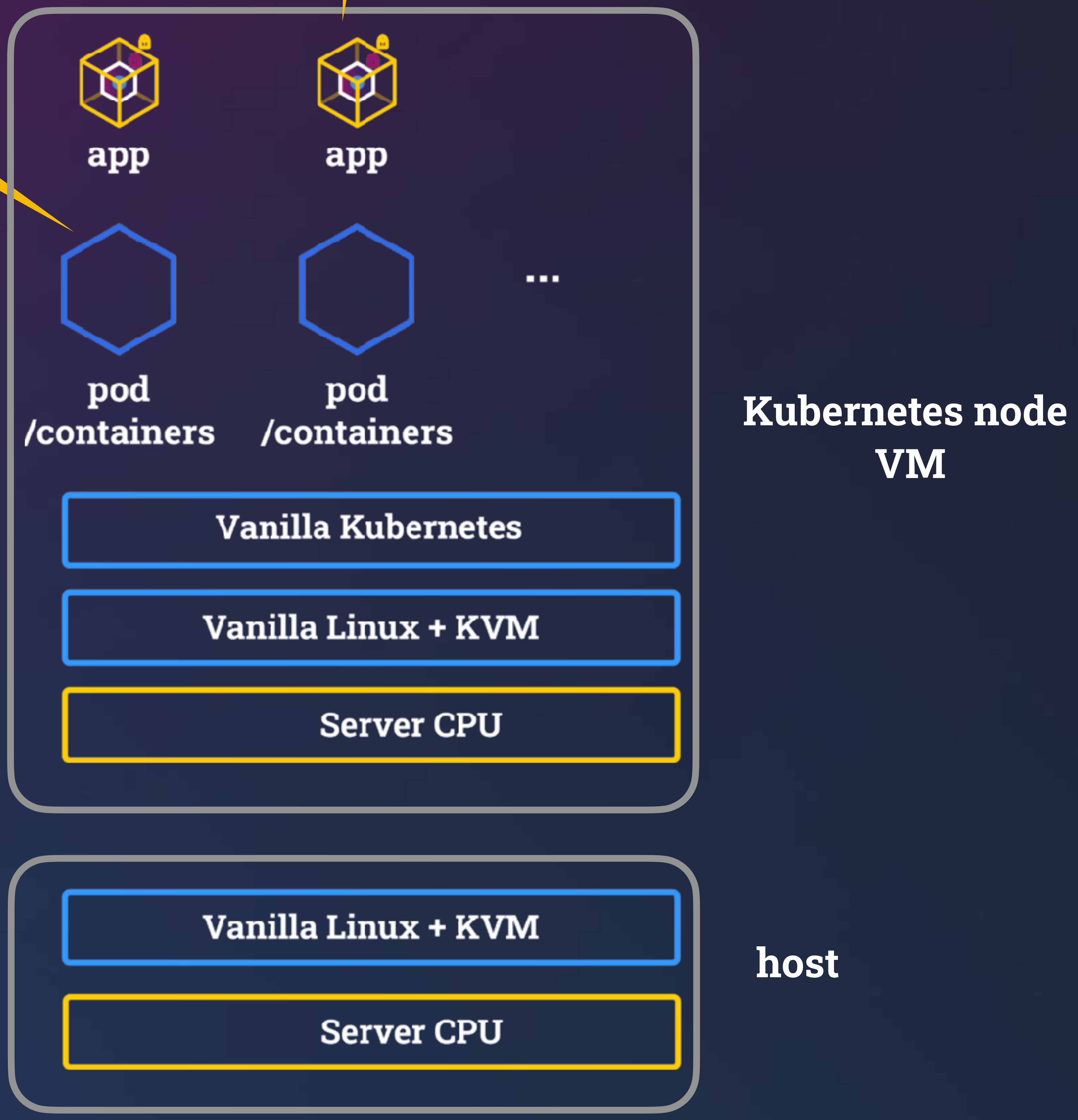
Attestation: checks for CVEs

Attestation: checks for CVEs

„zero trust“ = trusted after measurement
all components are periodically measured

CVE Adversary

- **Requirements**:
  - Must fix all CVEs within **D** days (**D** small)
  - Must **not** stop the application to fix CVEs
- **Approach**:
  - patch all CPUs, host OS, VMs, app without stopping app!
  - stop application if not updated within D days
- **Challenges**
  - updates change expected measurement, seal keys, …



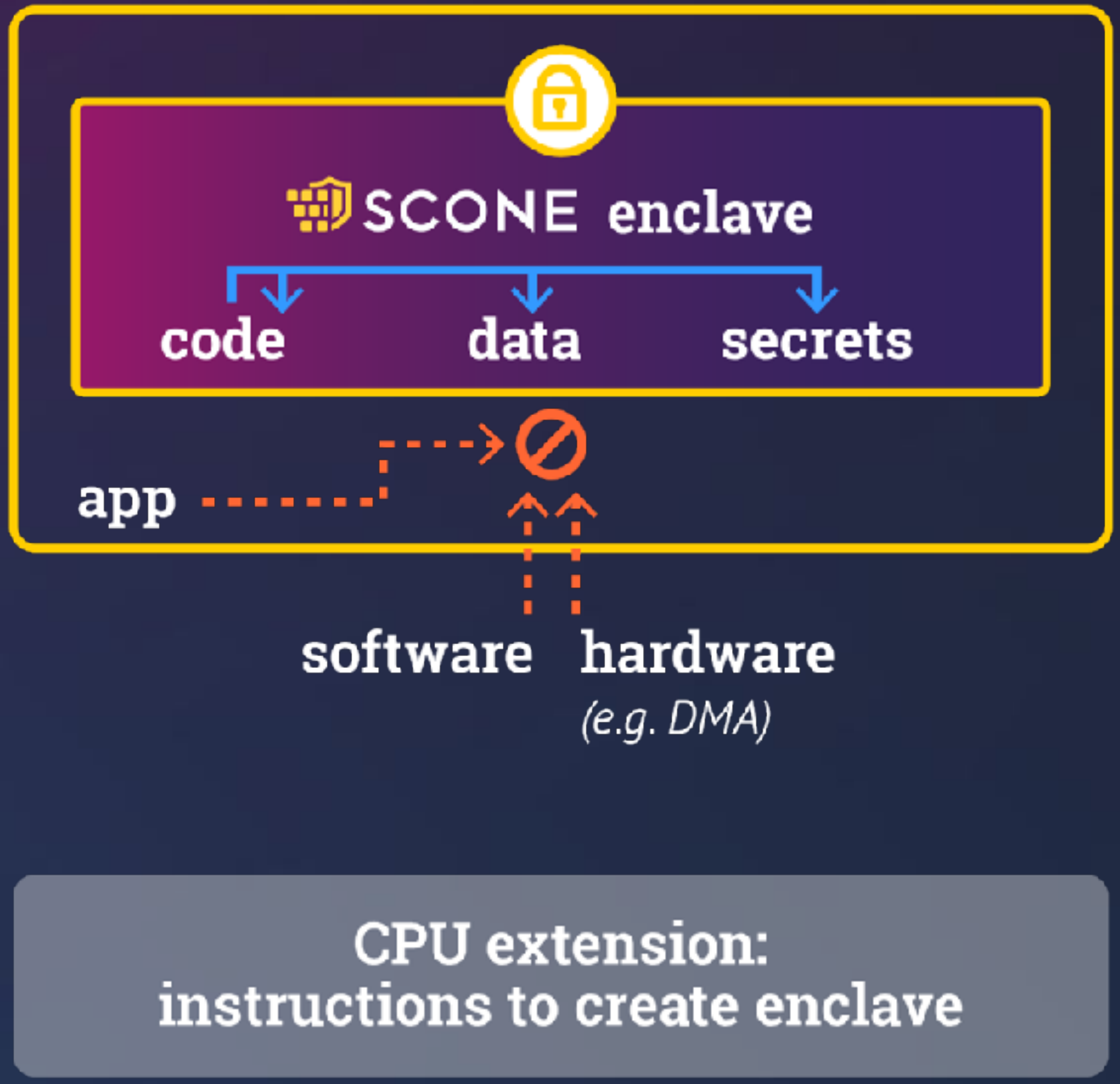
Runs in „enclave“

Might use SGX, TDX, SEV SNP, …

Enclaves

- **Protect data/code/secrets in use** (i.e, in main memory):
  - run application code in encrypted memory region (aka **enclave**)
  - only code in enclave can access memory region

Some keys tied to CPU - cannot be migrated!

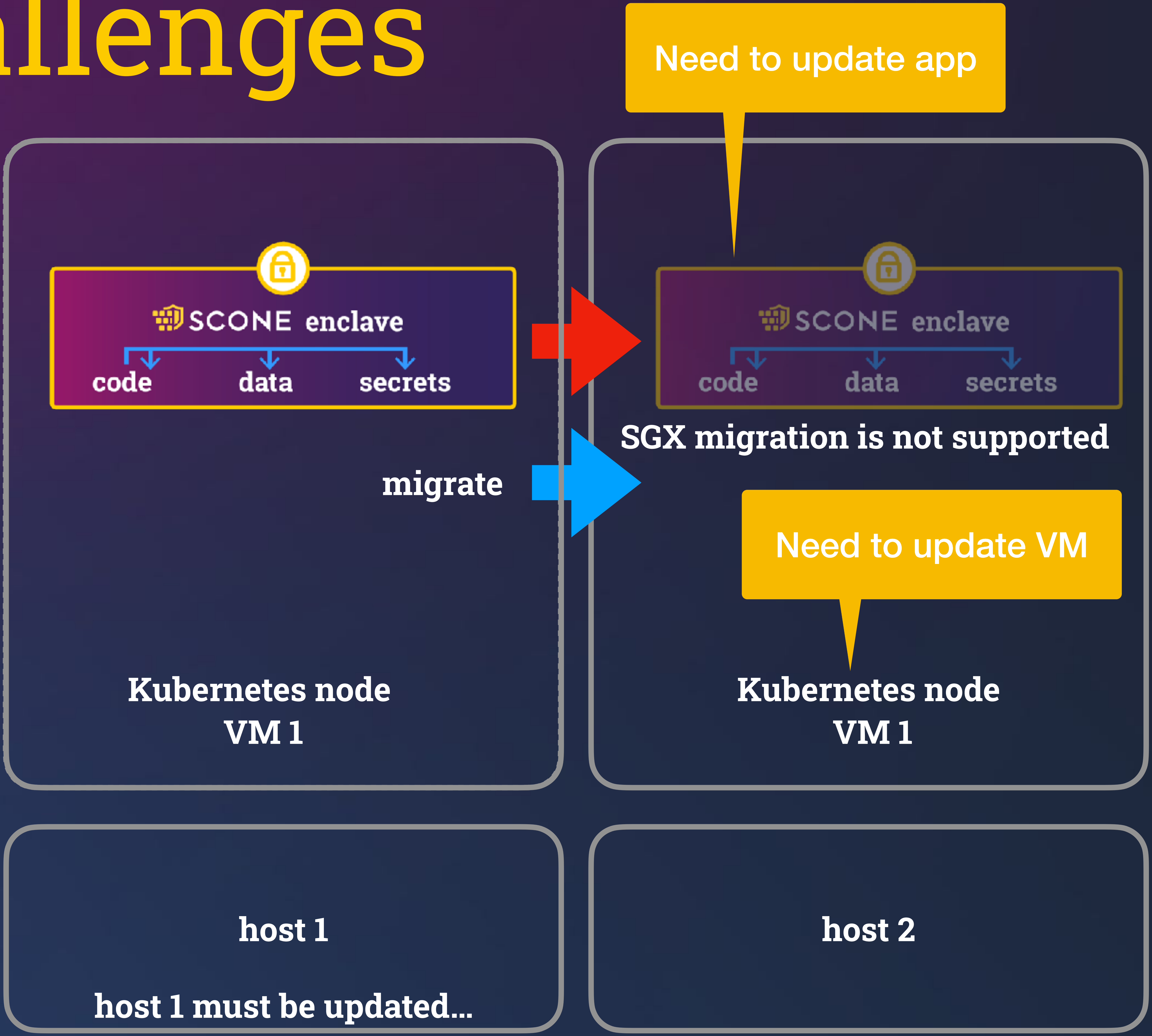


SCONE: Secure CONtainer Environment
- platform for Confidential Computing

https://sconedocs.github.com

Challenges

- **Cloud approach**
  - To **update a host**, migrate all VMs to different host
- **Challenges**
  - Intel SGX prevents migration
  - for other TEEs, we prefer VMs not to be migratable
- **Observation**
  - VM migration does not help in upgrading VM or app itself



Need to update app

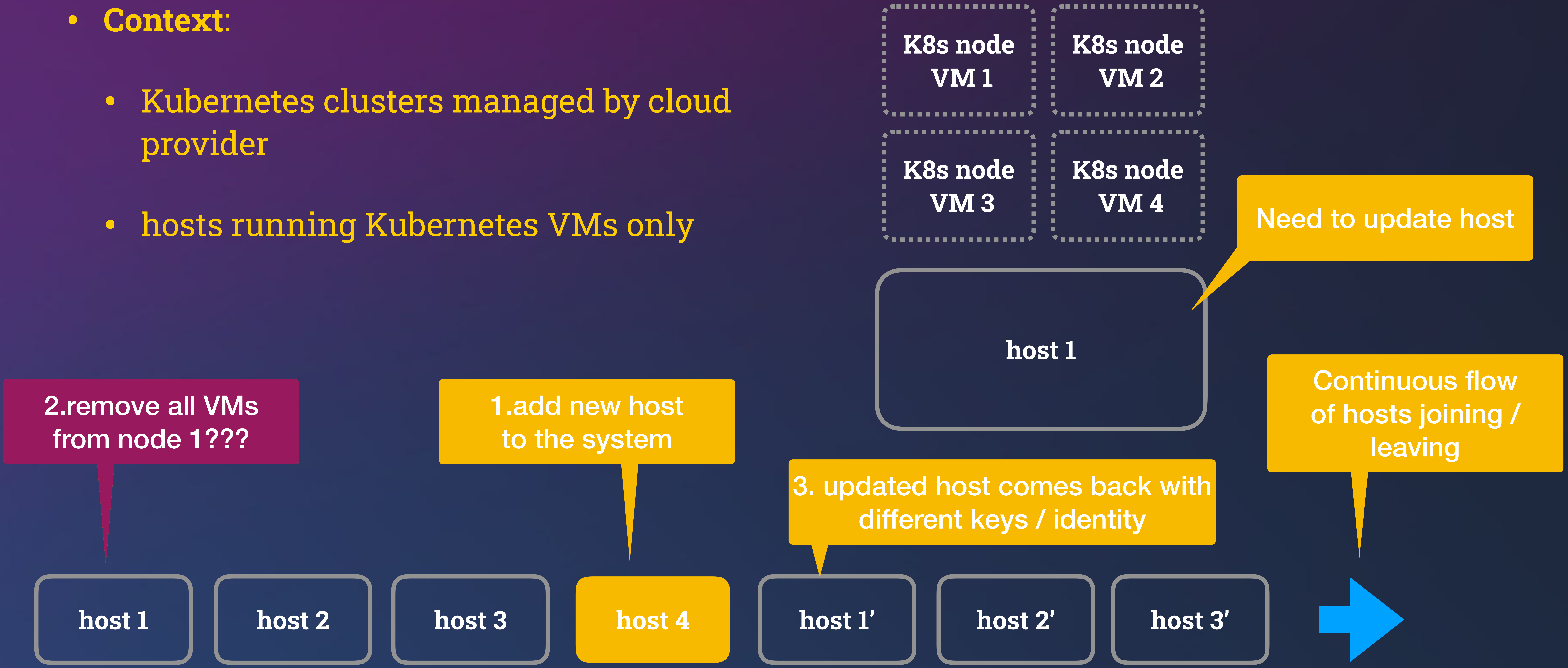
SGX migration is not supported

Need to update VM

host 1 must be updated…

Host Updates

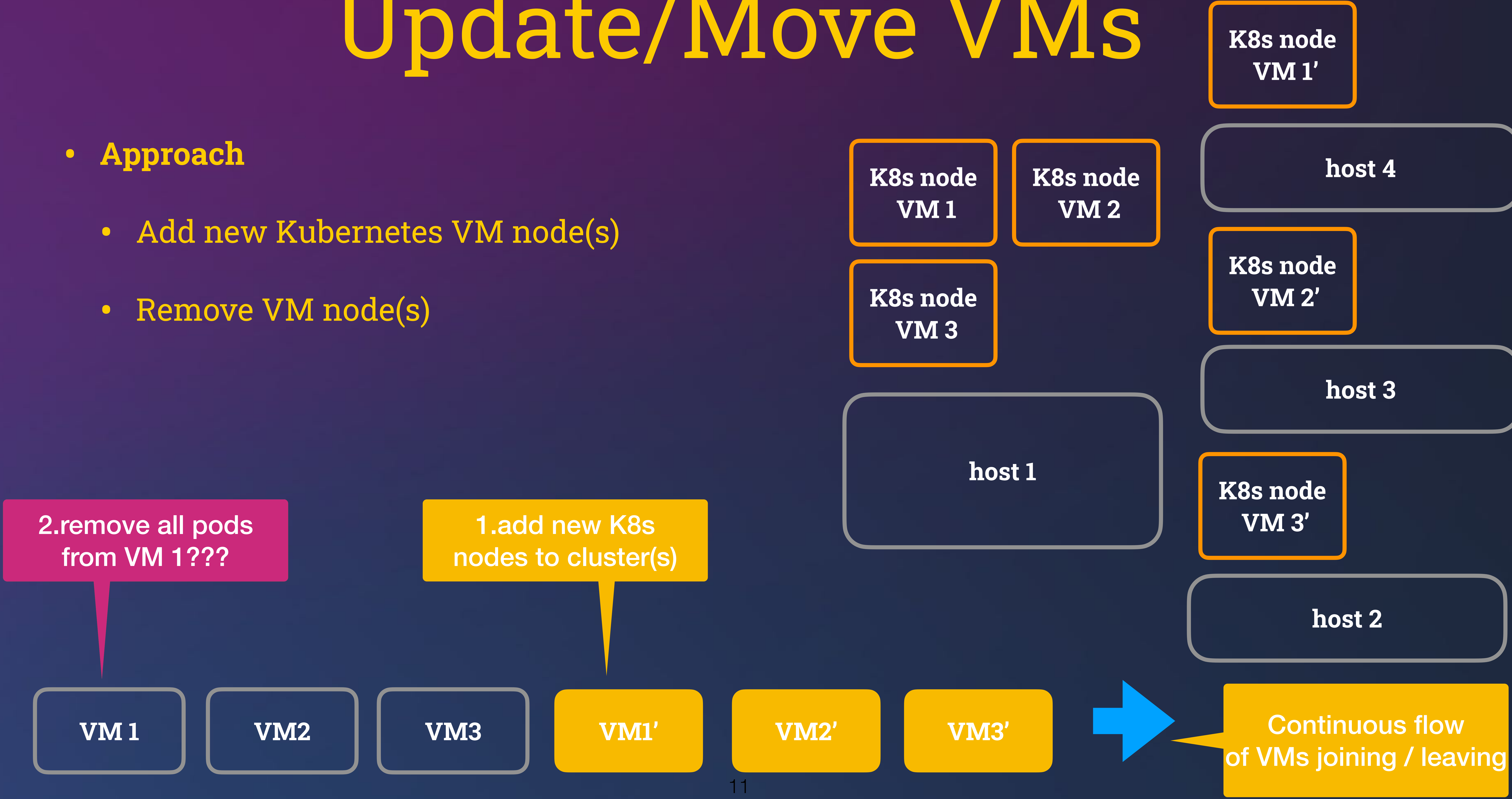
- **Context**:
  - Kubernetes clusters managed by cloud provider
  - hosts running Kubernetes VMs only



K8s node VM 1
K8s node VM 2
K8s node VM 3
K8s node VM 4
host 1
Need to update host
Continuous flow of hosts joining / leaving
2.remove all VMs from node 1???
1.add new host to the system
3. updated host comes back with different keys / identity
host 1
host 2
host 3
host 4
host 1'
host 2'
host 3'

Update/Move VMs

- **Approach**
  - Add new Kubernetes VM node(s)
  - Remove VM node(s)



K8s node VM 1’

host 4

K8s node VM 2’

host 3

K8s node VM 3’

host 2

K8s node VM 1

K8s node VM 2

K8s node VM 3

host 1

2.remove all pods from VM 1???

1.add new K8s nodes to cluster(s)

VM 1

VM2

VM3

VM1’

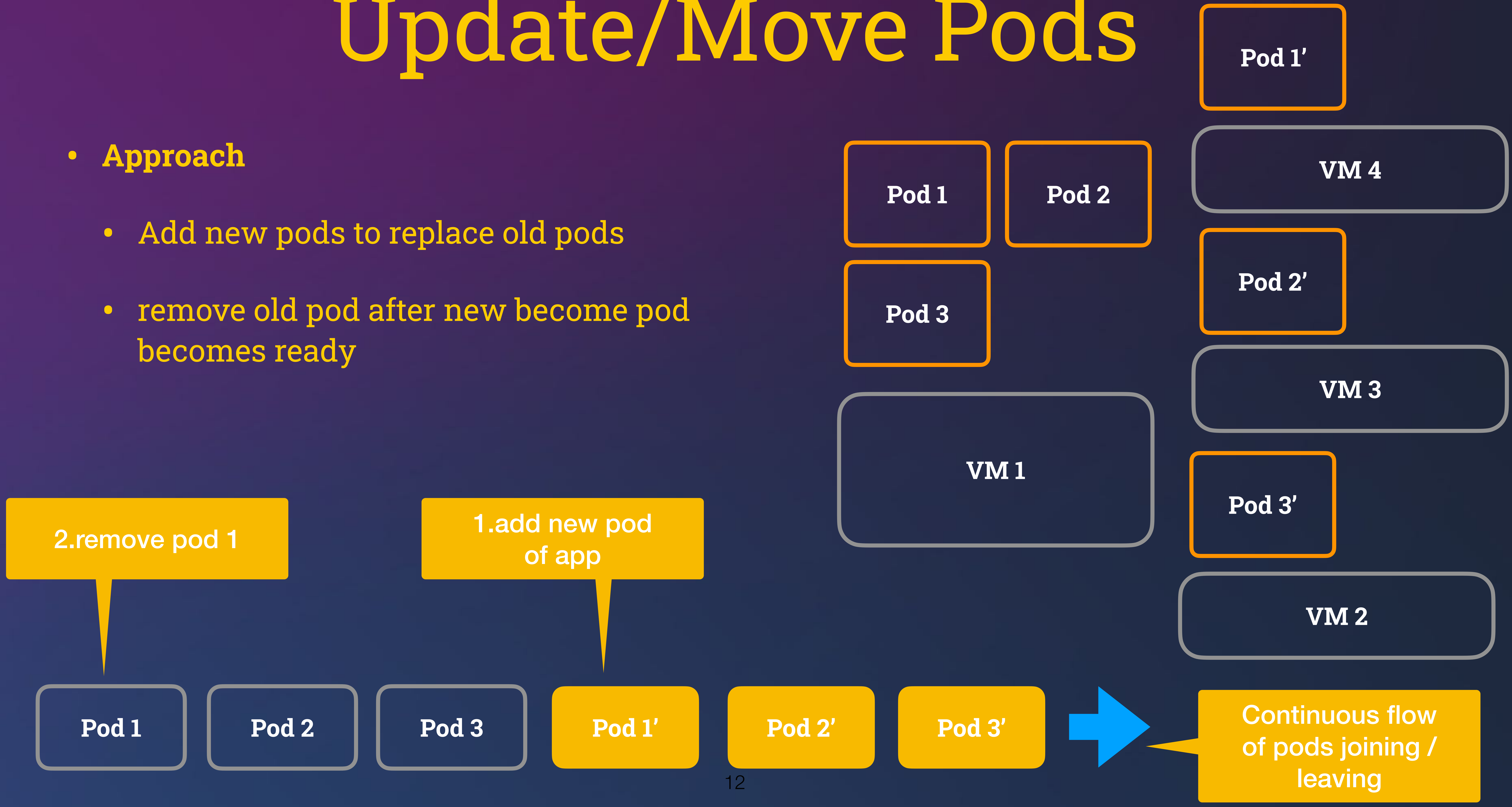
VM2’

VM3’

Continuous flow of VMs joining / leaving

Update/Move Pods

- **Approach**
  - Add new pods to replace old pods
  - remove old pod after new become pod becomes ready

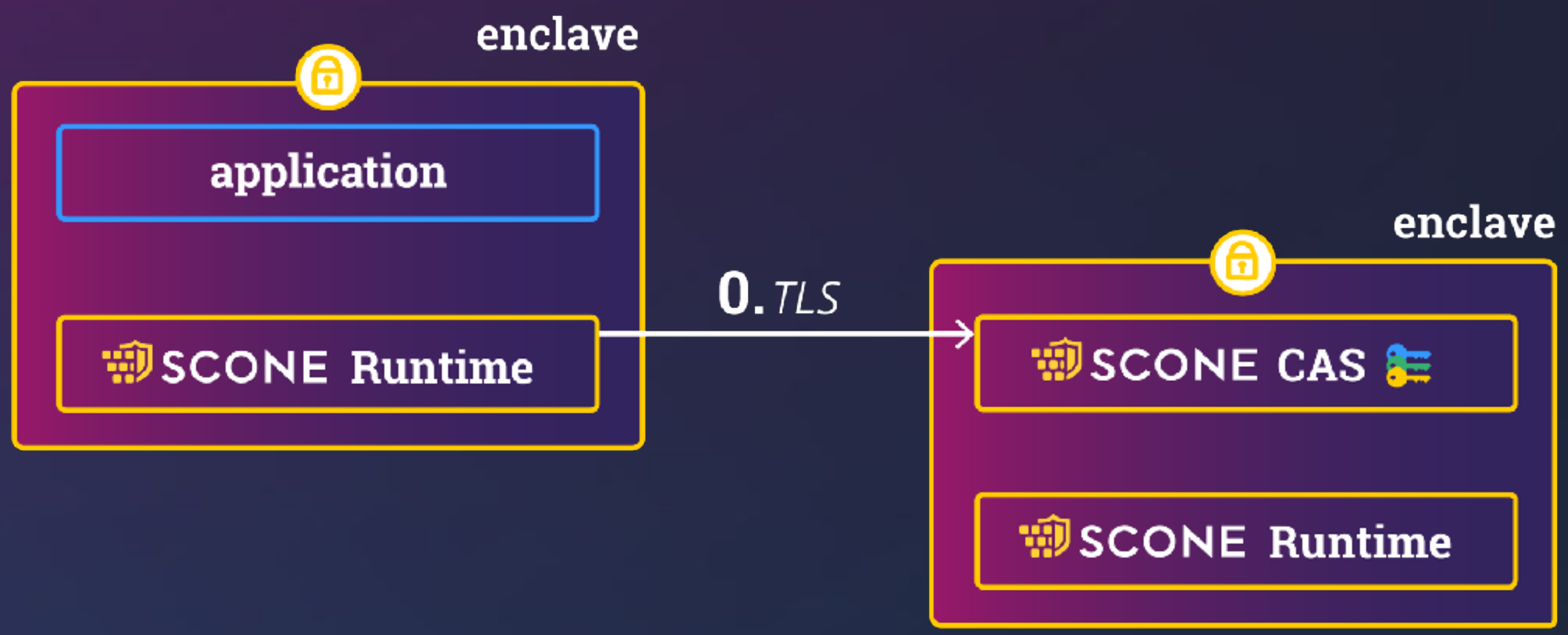


How to provision secrets to restarted pods?

- e.g., keys, configuration files, … -

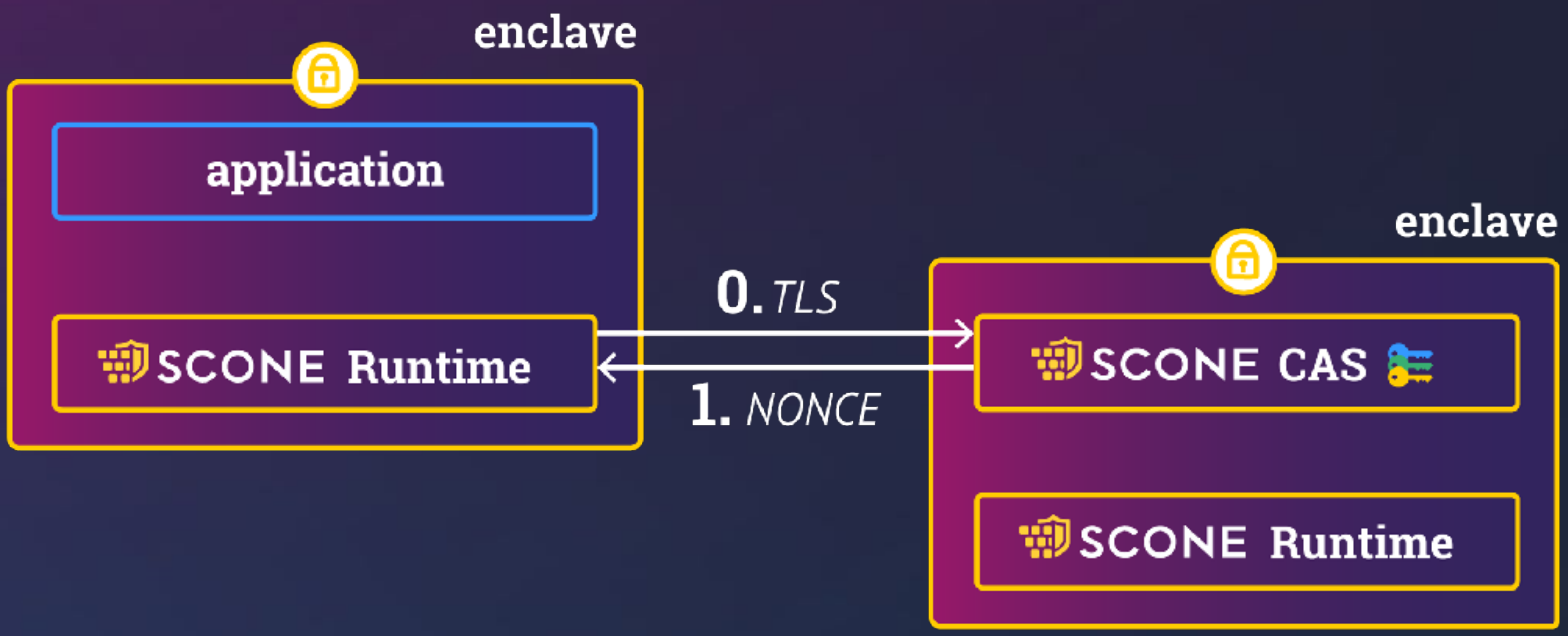
SCONE CAS (Configuration and Attestation)

- **SCONE**:
  - no need to change application
- **Attestation flow:**
  - transparently performed by SCONE runtime
  - application gets configuration
    - arguments
    - environment variables
    - configuration files



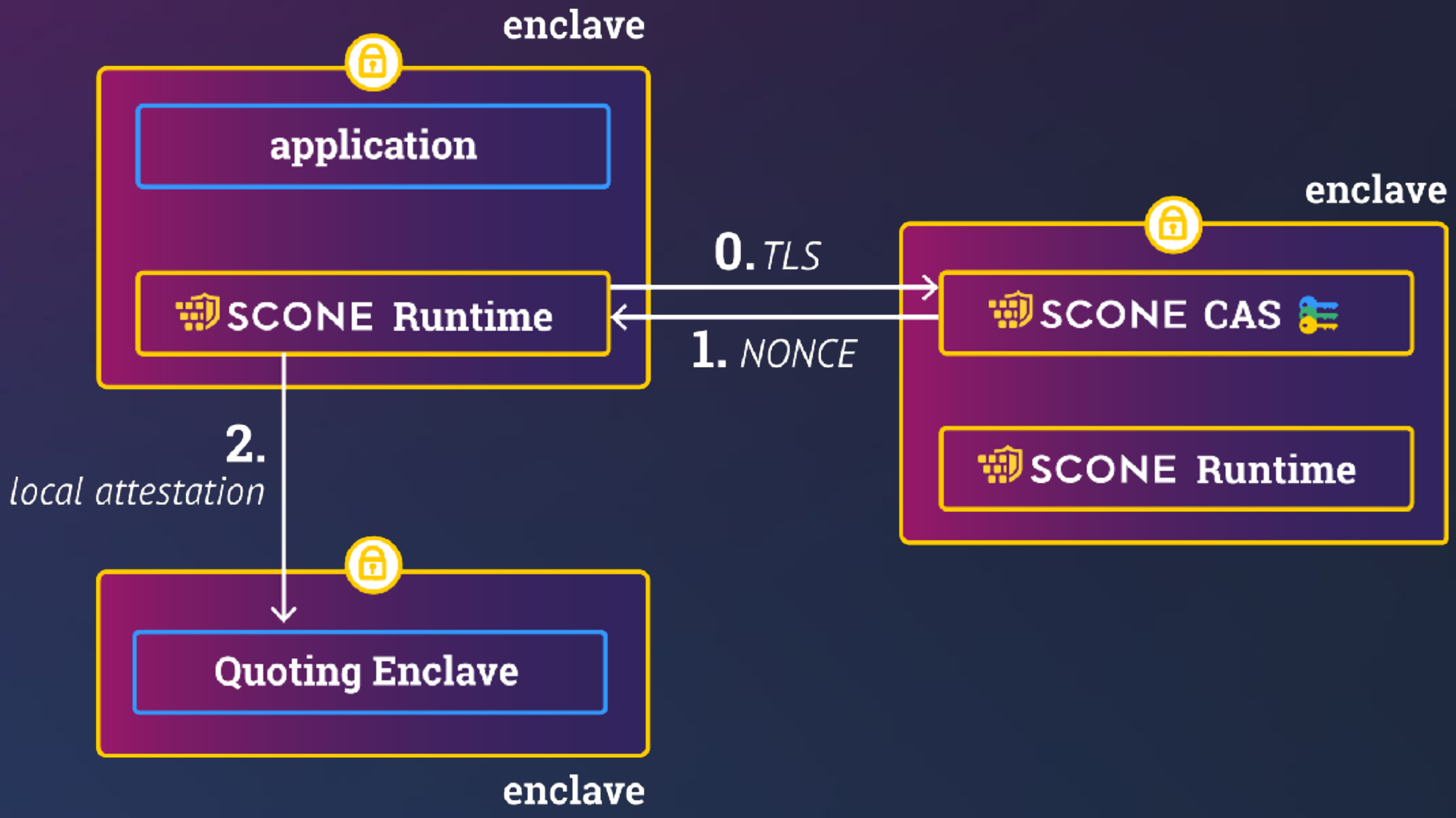
SCONE Attestation

- **SCONE**:
  - no need to change application
- **Attestation flow:**
  - transparently performed by SCONE runtime
  - application gets configuration
    - arguments
    - environment variables
    - configuration files



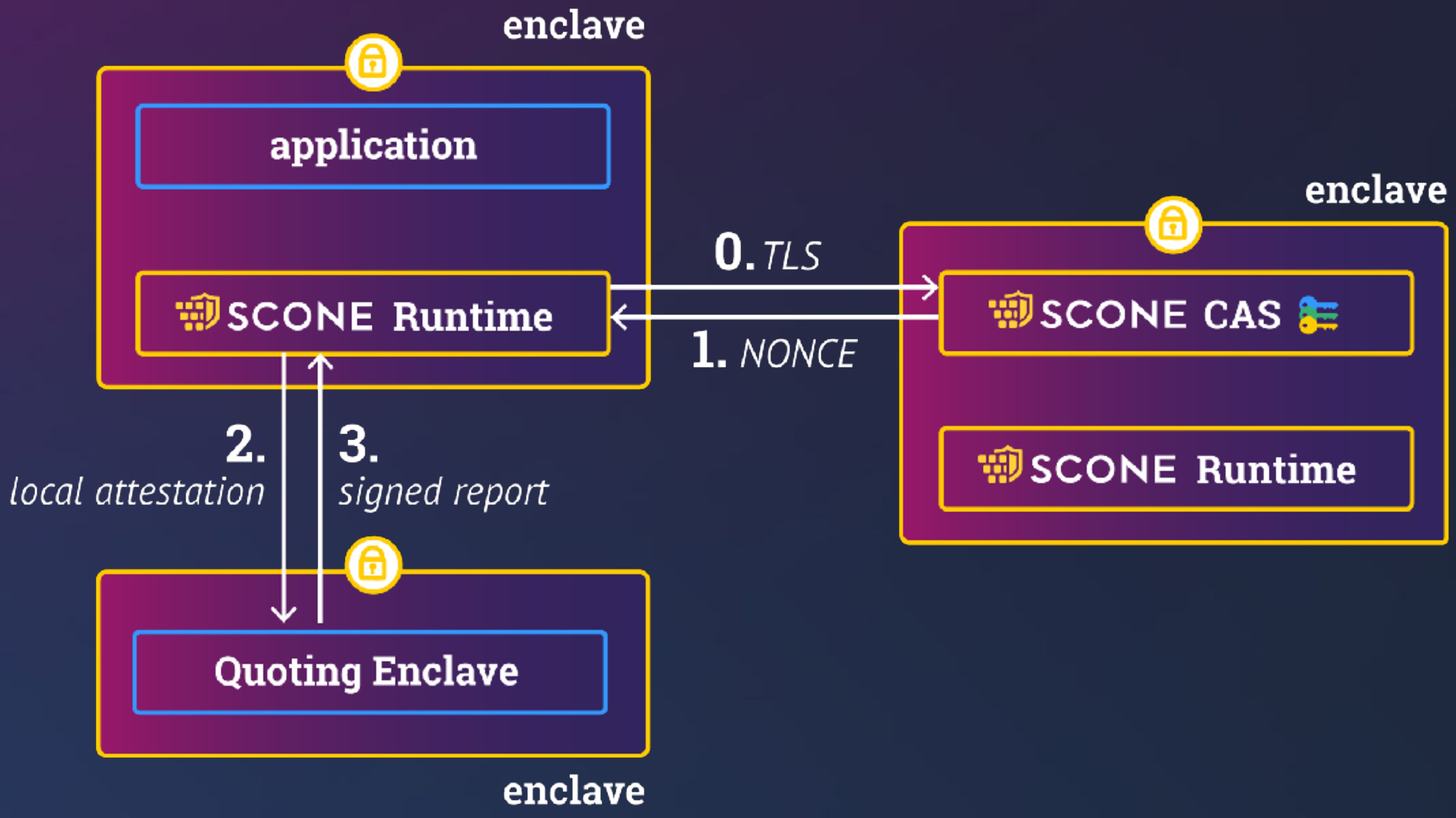
SCONE Attestation

- **SCONE**:
  - no need to change application
- **Attestation flow:**
  - transparently performed by SCONE runtime
  - application gets configuration
    - arguments
    - environment variables
    - configuration files



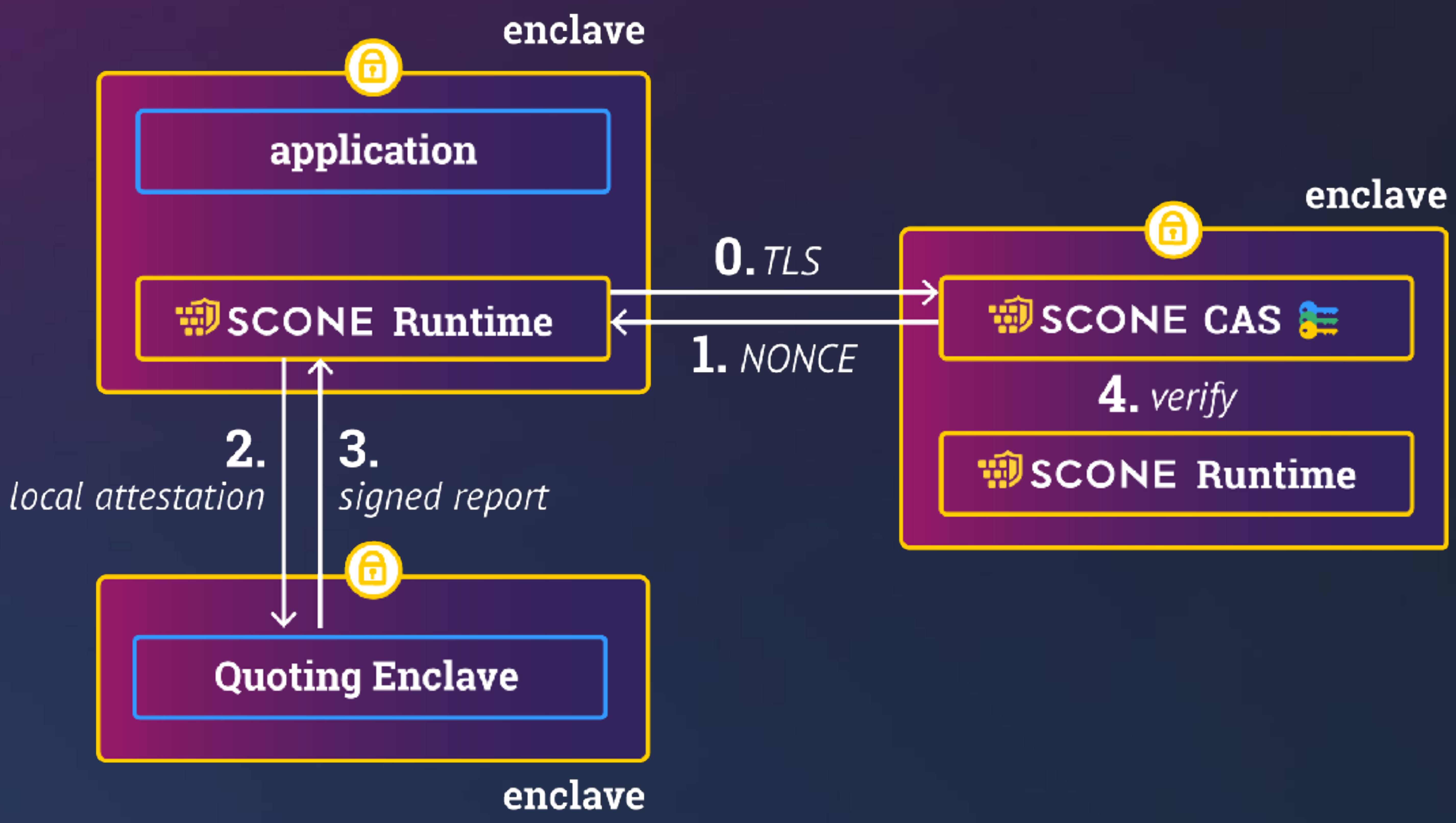
SCONE Attestation

- **SCONE**:
  - no need to change application
- **Attestation flow:**
  - transparently performed by SCONE runtime
  - application gets configuration
    - arguments
    - environment variables
    - configuration files



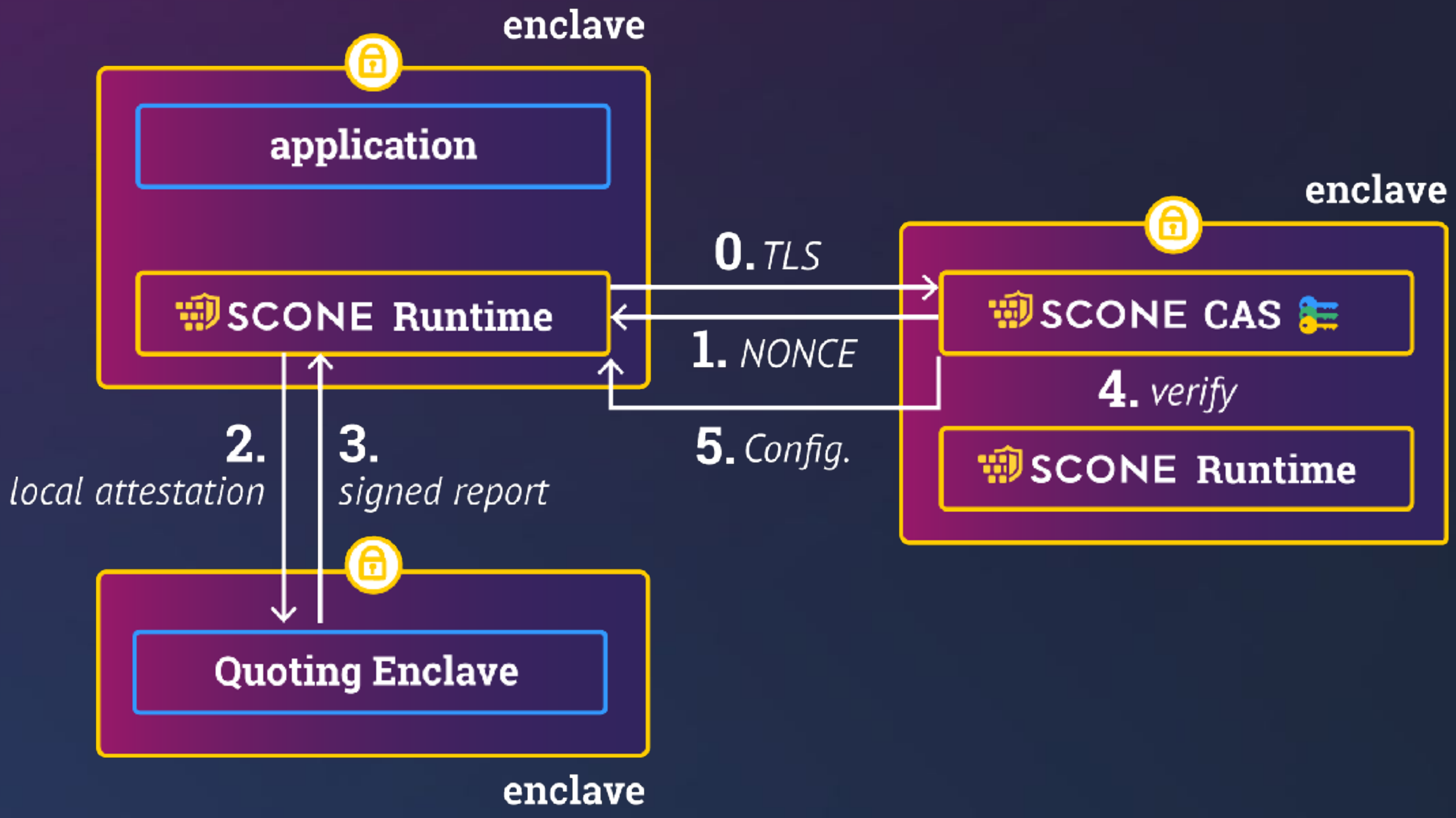
SCONE Attestation

- **SCONE**:
  - no need to change application
- **Attestation flow:**
  - transparently performed by SCONE runtime
  - application gets configuration
    - arguments
    - environment variables
    - configuration files



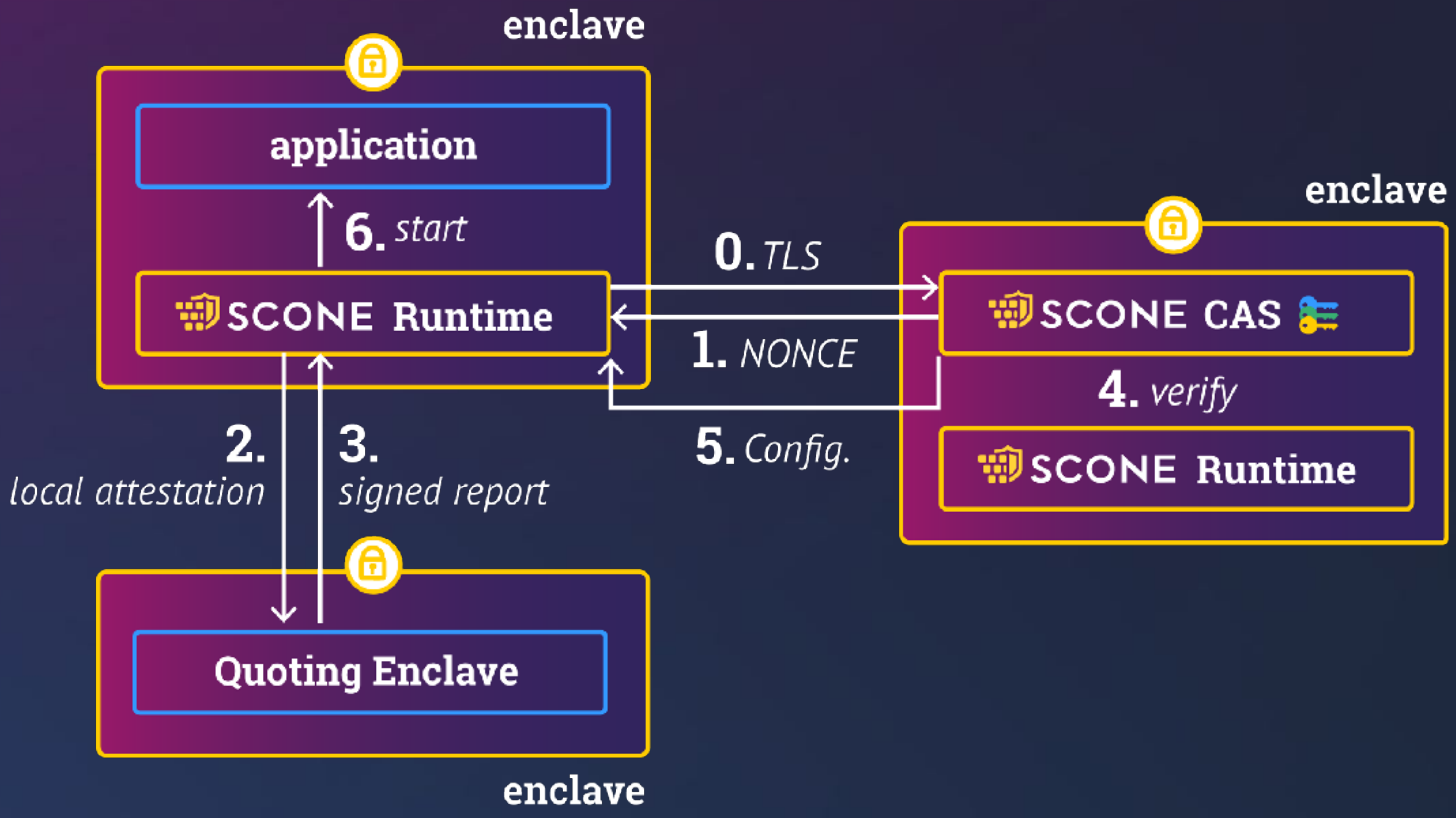
SCONE Attestation

- **SCONE**:
  - no need to change application
- **Attestation flow:**
  - transparently performed by SCONE runtime
  - application gets configuration
    - arguments
    - environment variables
    - configuration files



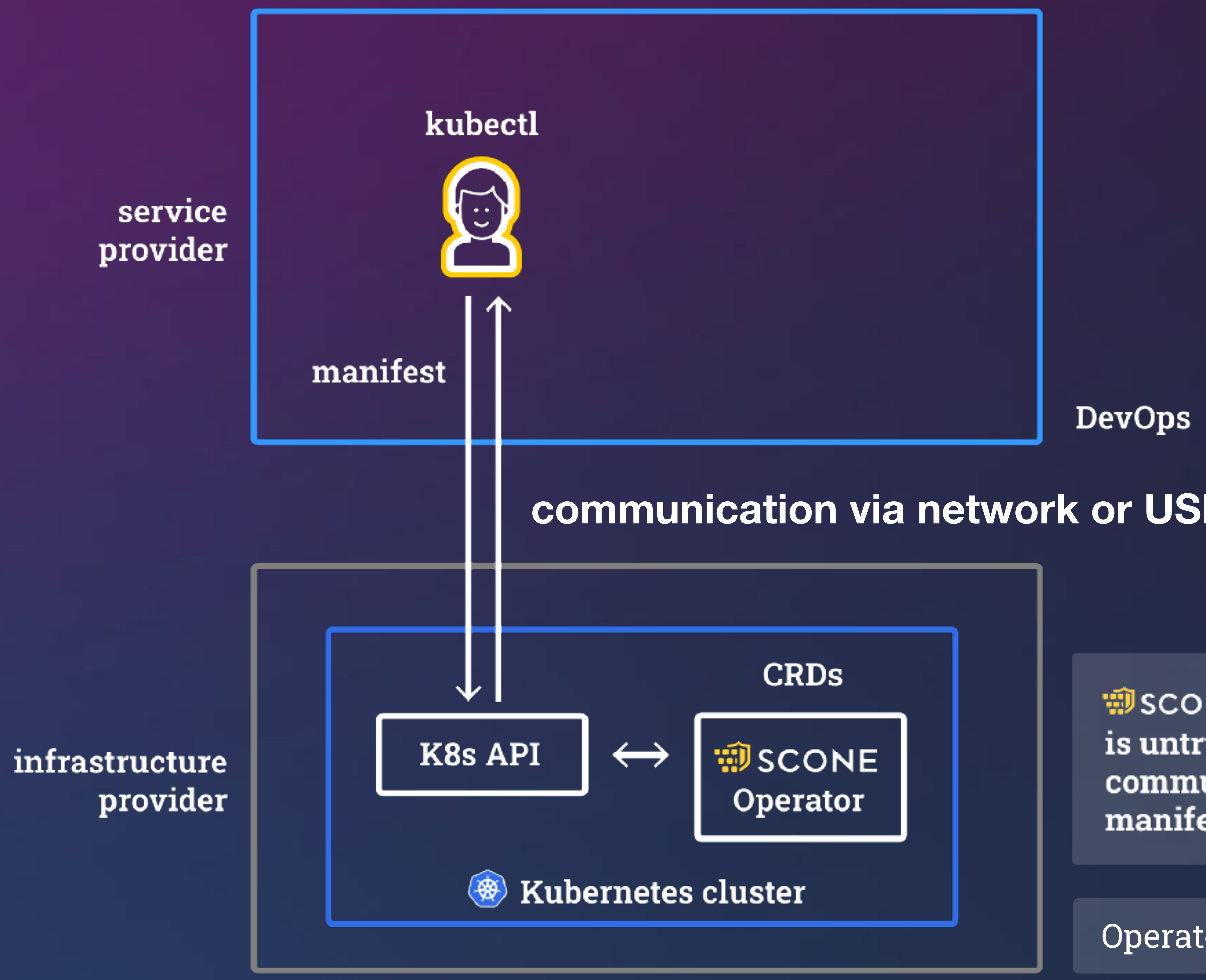
SCONE Attestation

- **SCONE**:
  - no need to change application
- **Attestation flow:**
  - transparently performed by SCONE runtime
  - application gets configuration
    - arguments
    - environment variables
    - configuration files



Air-Gapped Operation / Bootstrap

SCONE Operator



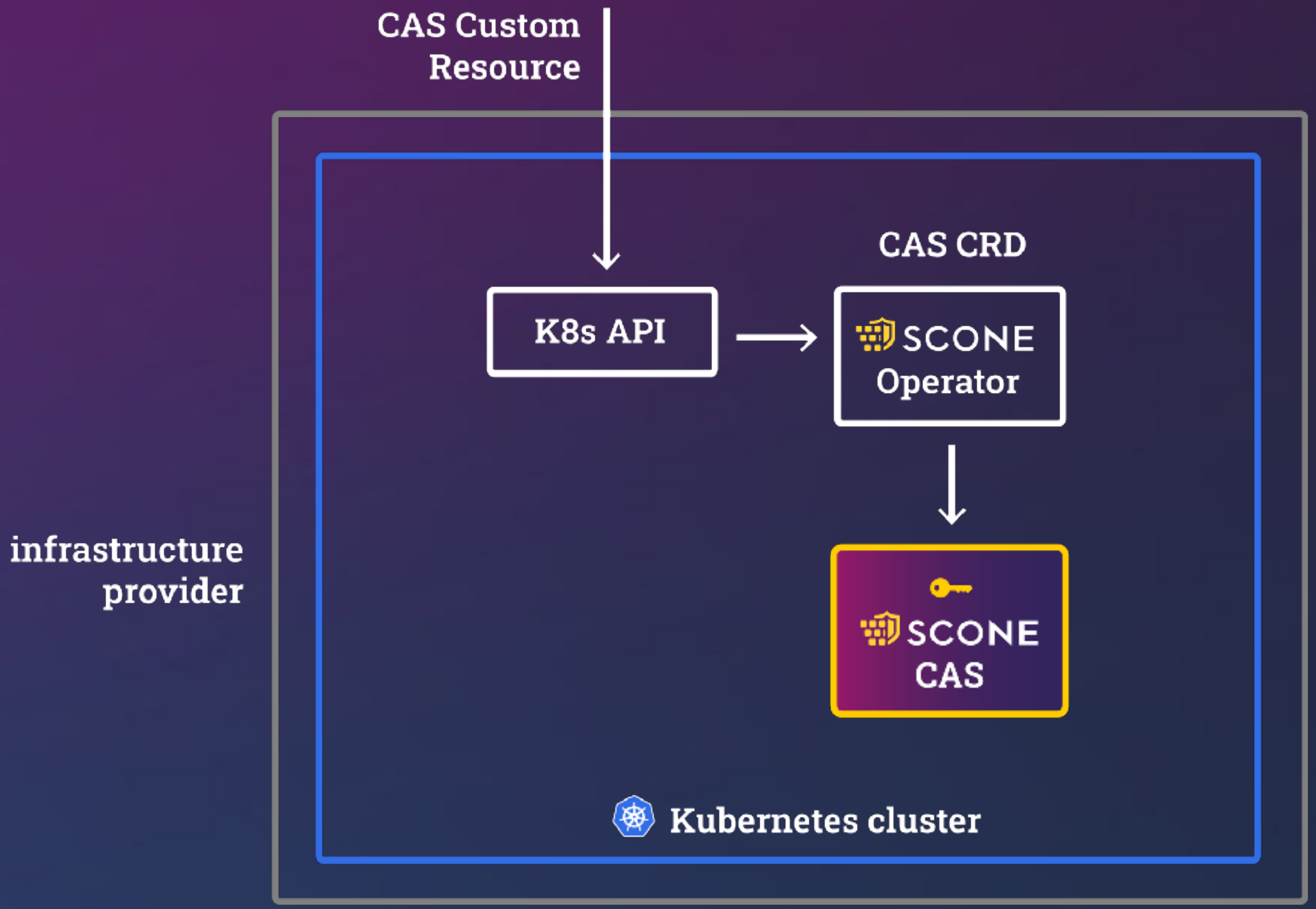
communication via network or USB sticks...

SCONE Operator is untrusted - enables communication via manifests!

Operator is installed via a helm chart

https://sconedocs.github.io

SCONE CAS: Policy Engine in TEE

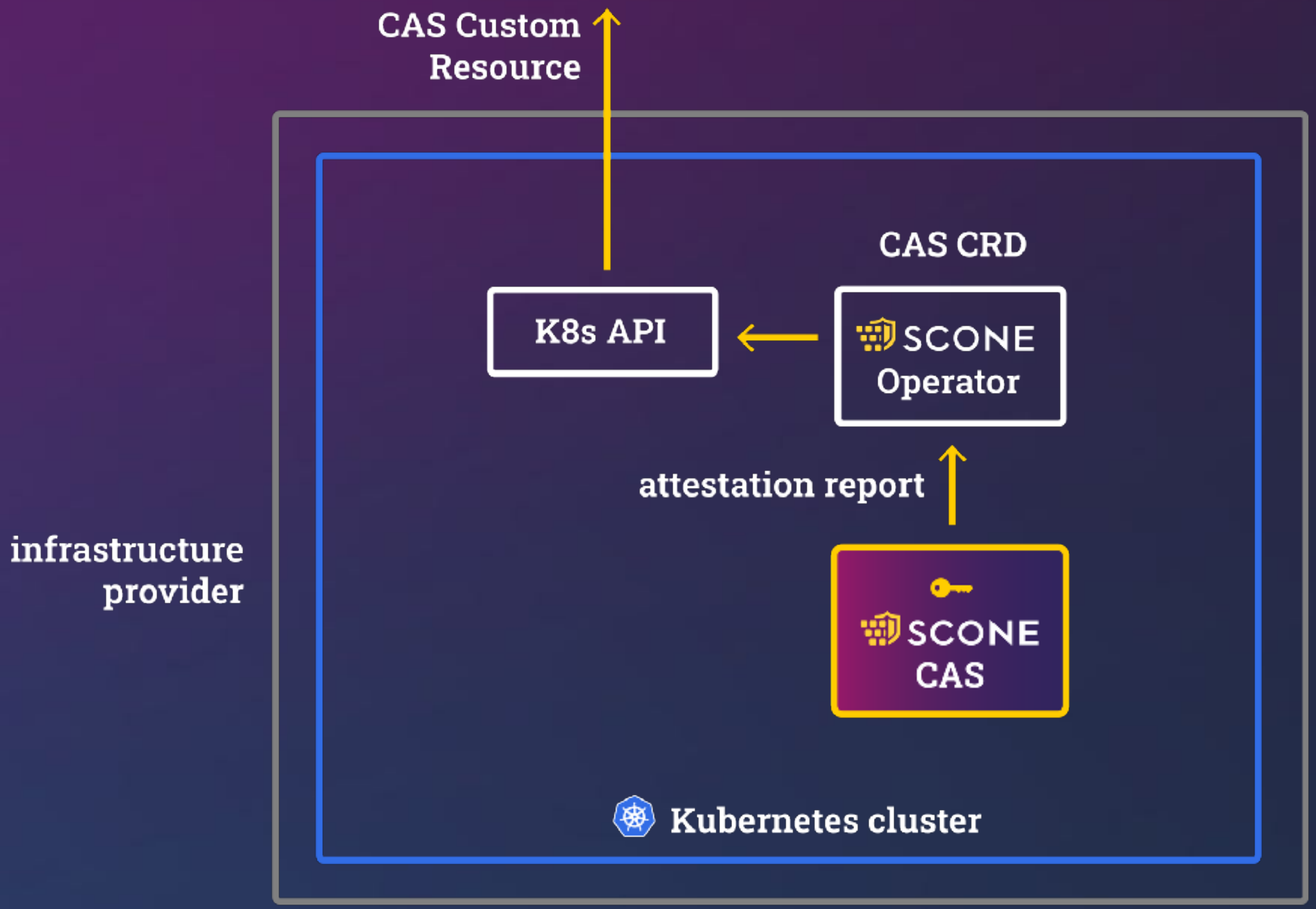


We install a CAS as a Custom Resource

SCONE CAS: Configuration & Attestation Service

https://sconedocs.github.io

Attesting SCONE CAS

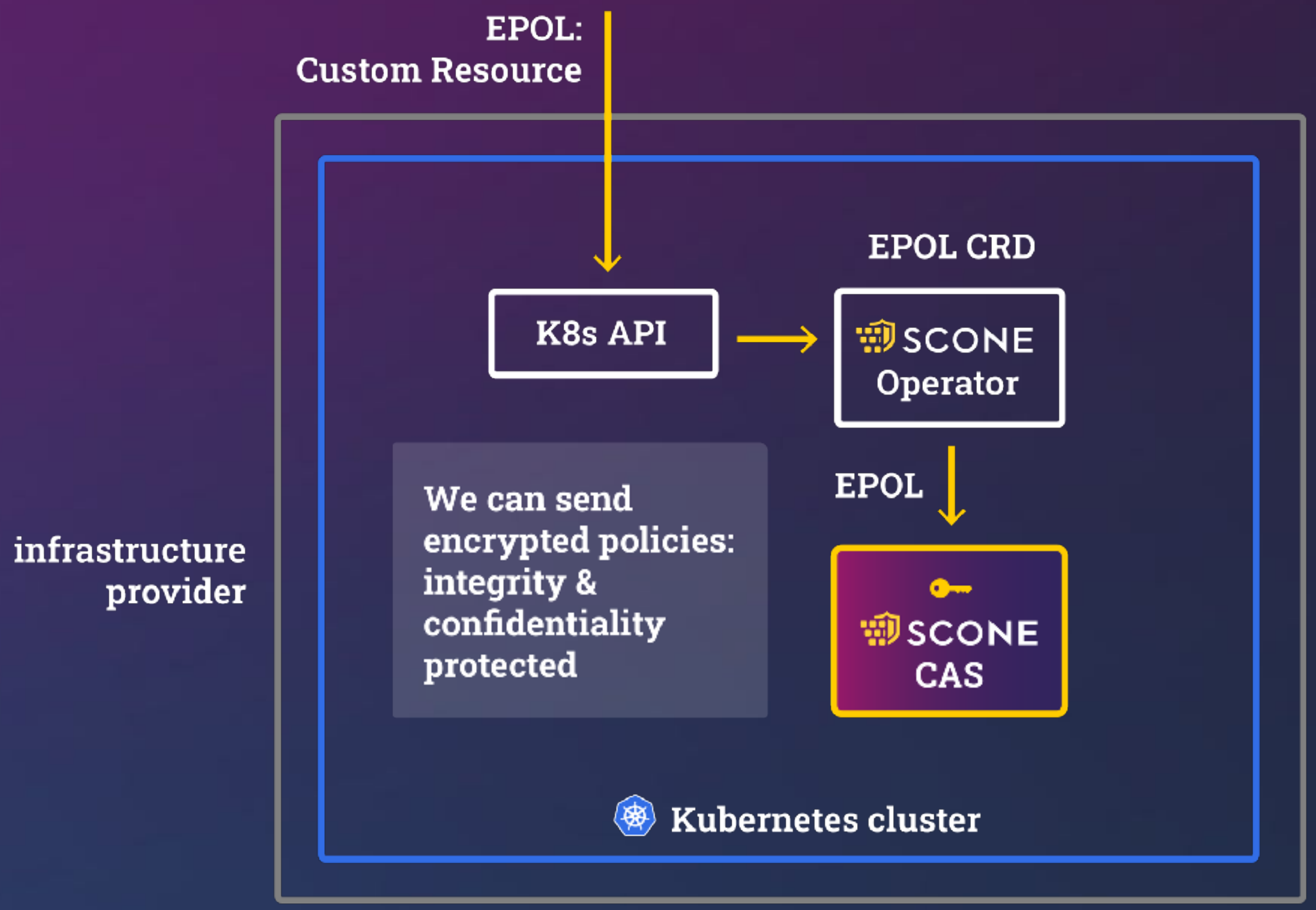


We can verify the attestation report

We can learn that the CAS is trustworthy and its public encryption key via the kubeAPI (using offline attestation)

https://sconedocs.github.io

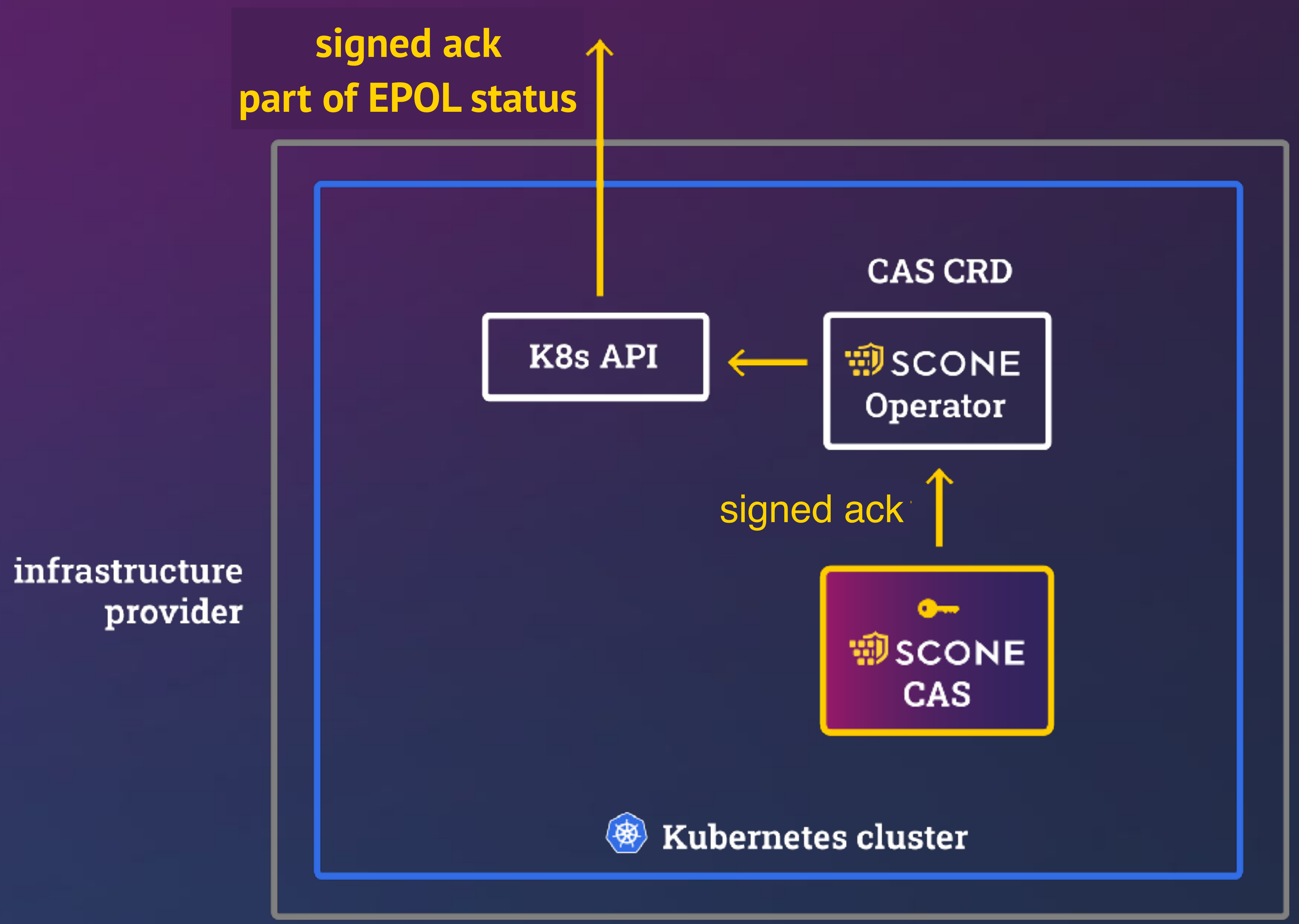
Encrypted Policies (EPOL)



We can send encrypted policies: integrity & confidentiality protected

https://sconedocs.github.io

Signed Acks

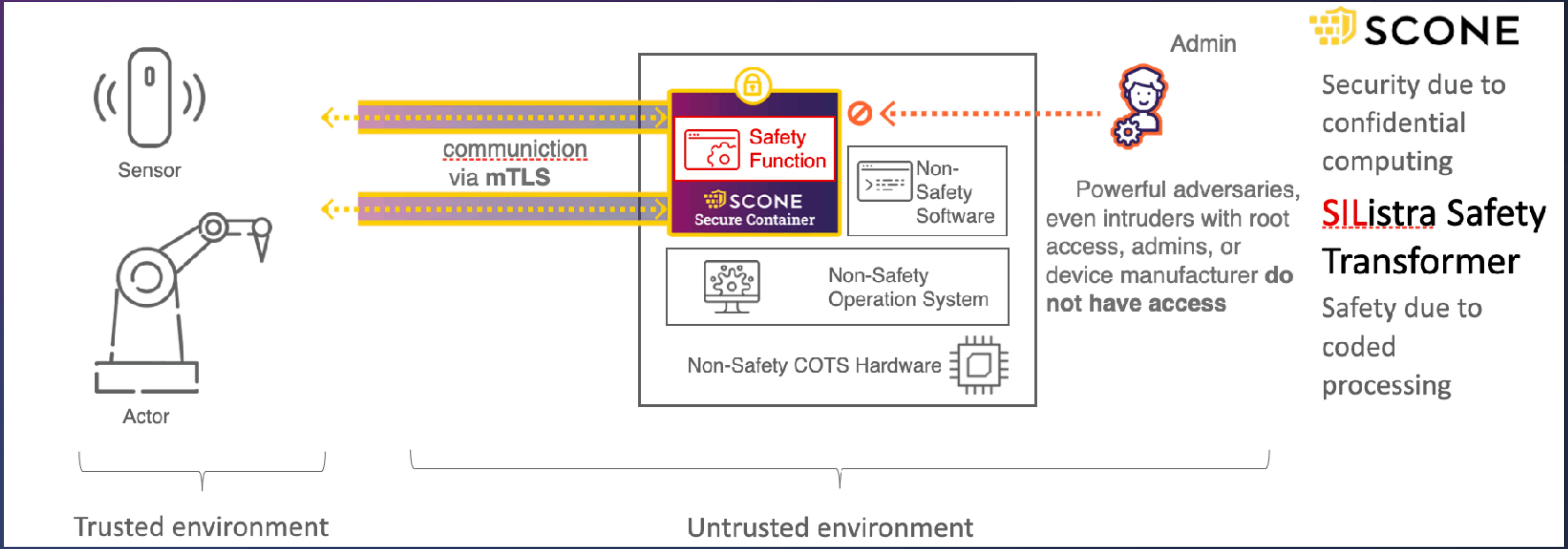


signed ack
part of EPOL status

Next Steps...

Confidential Computing & Encoded Processing

- Approach: Encoded Processing inside enclaves (i.e., attested programs inside of encrypted memory region)


Sensor
Actor
communiction
via mTLS
Safety Function
SCONE
Secure Container
Non-Safety Software
Non-Safety Operation System
Non-Safety COTS Hardware
Admin
Powerful adversaries, even intruders with root access, admins, or device manufacturer do not have access
SCONE
Security due to confidential computing
SIListra Safety Transformer
Safety due to coded processing
Trusted environment
Untrusted environment

Questions?

Prof. Dr. Christof Fetzer