SYNACKTIV

# So I became a node

## Exploiting bootstrap token in AKS

27/06/2024

TROOPERS

# Who are we

SYNACKTIV

**Paul Barbé**

Pentester/Redteamer

paul.barbe@synacktiv.com

**Kévin Schouteeten**

Pentester/Redteamer

kevin.schouteeten@synacktiv.com

@Scouty__

# Synacktiv

SYNACKTIV

- French offensive security company
- 170 security experts
- 4 departments :
  - Pentest/ Redteam
  - RE / VR
  - Development
  - IR
- Hexacon (Paris - october 2024)

# Overview

SYNACKTIV

- Kubernetes presentation and definition
- Kubernetes authentication, TLS, and nodes' joining process
- Exploitation of the enrollment process
  - Azure Kubernetes Cluster
  - Pod sharing the host's network namespace
- Demo
- Mitigations and conclusions

# Previous work

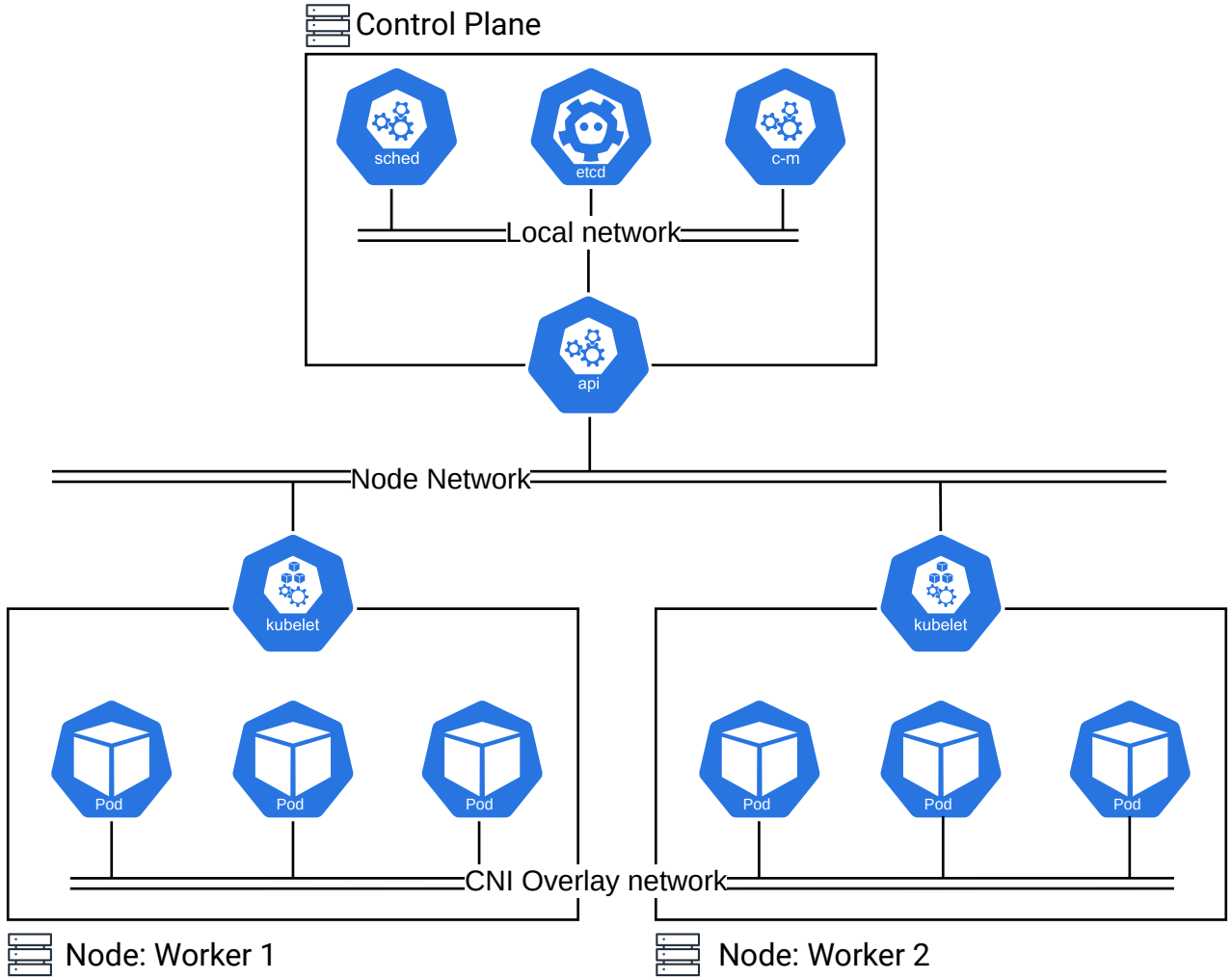
SYNACKTIV

- GKE enrollment exploit
  - Marc Wickenden
    - https://www.4armed.com/blog/hacking-kubelet-on-gke/ (2018)
    - https://github.com/4ARMED/kubeletmein
  - Jack Ganbold:
    - https://rhinosecuritylabs.com/cloud-security/kubelet-tls-bootstrap-privilege-escalation/ (2020)
- Azure WireServer
  - Paul Litvak:
    - https://intezer.com/blog/cloud-security/cve-2021-27075-microsoft-azure-vulnerability-allows-privilege-escalation-and-leak-of-data/ (2021)
  - Nick Wojciechowski, Dajne Win:
    - https://cybercx.com.au/blog/azure-ssrf-metadata/ (2023)

# Kubernetes

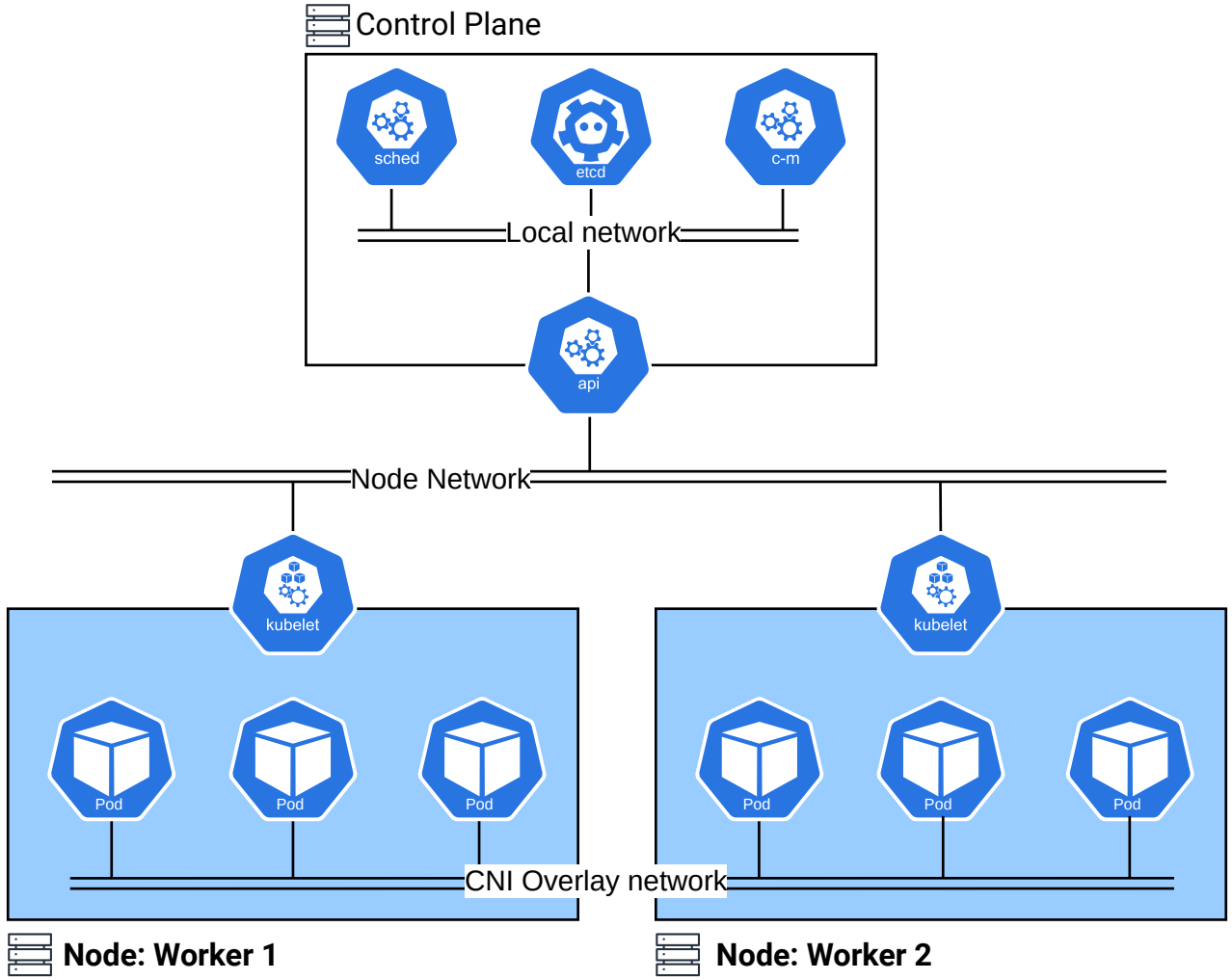
# Kubernetes

Cluster



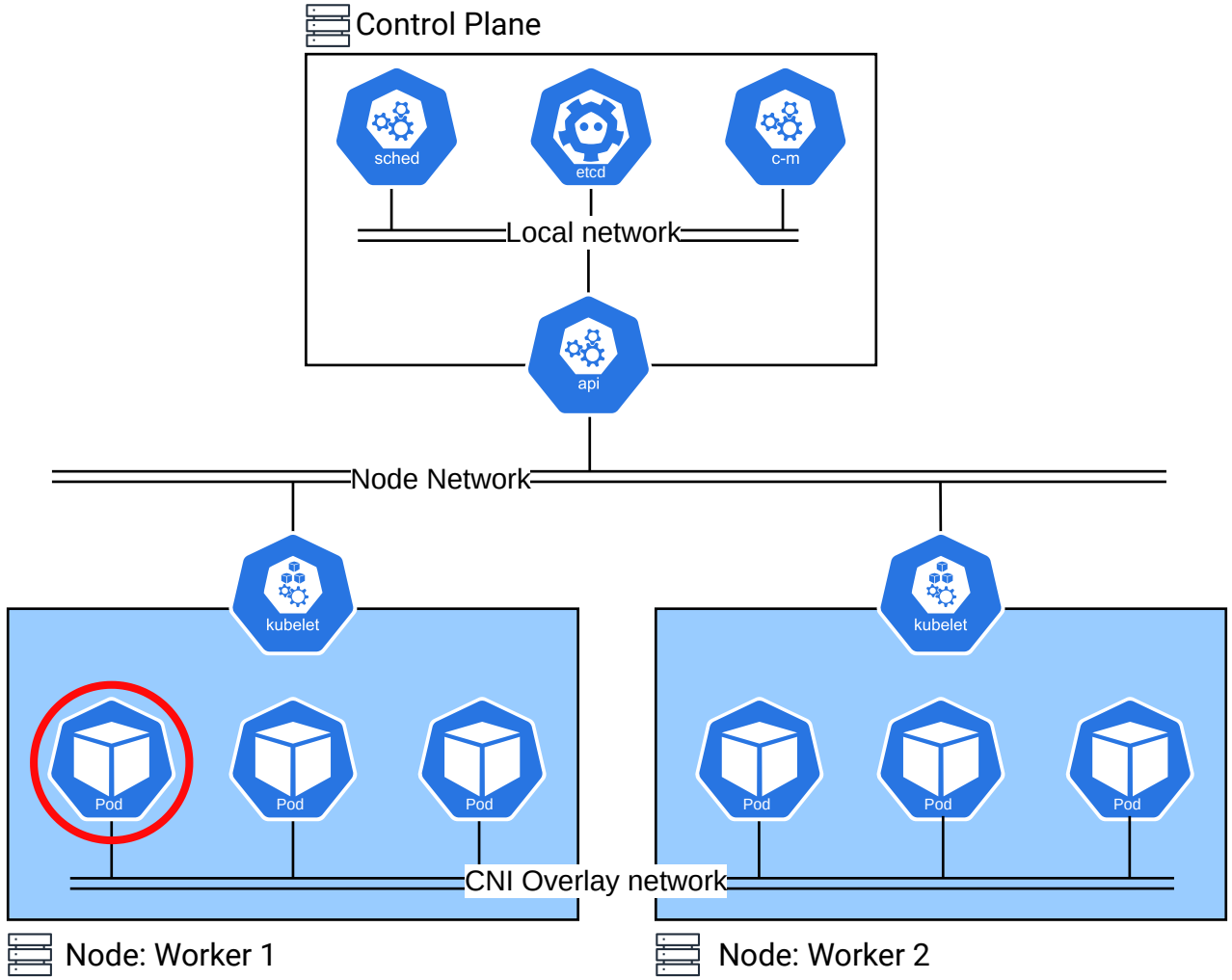
# Kubernetes

Node: Worker



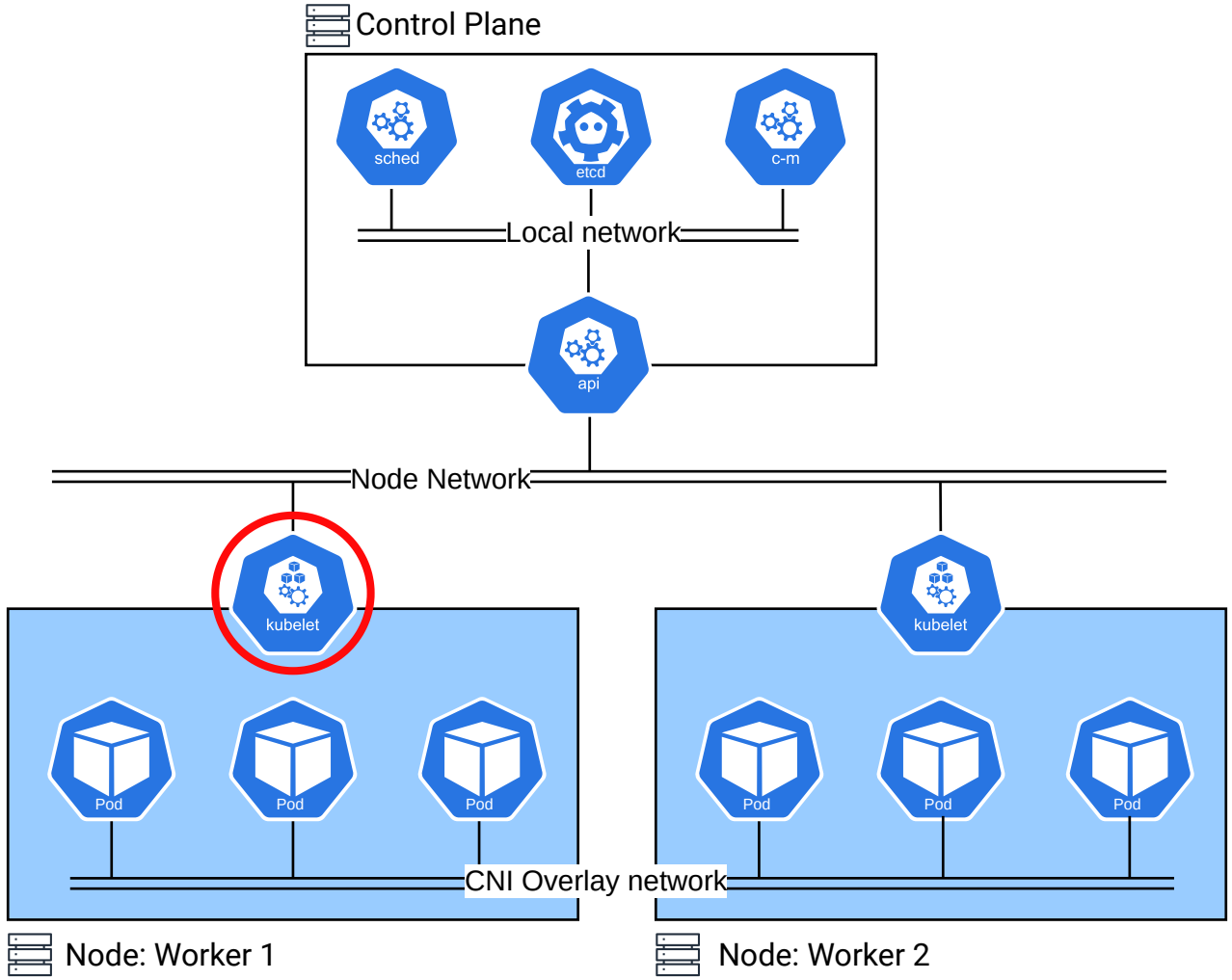
# Kubernetes

## Pod



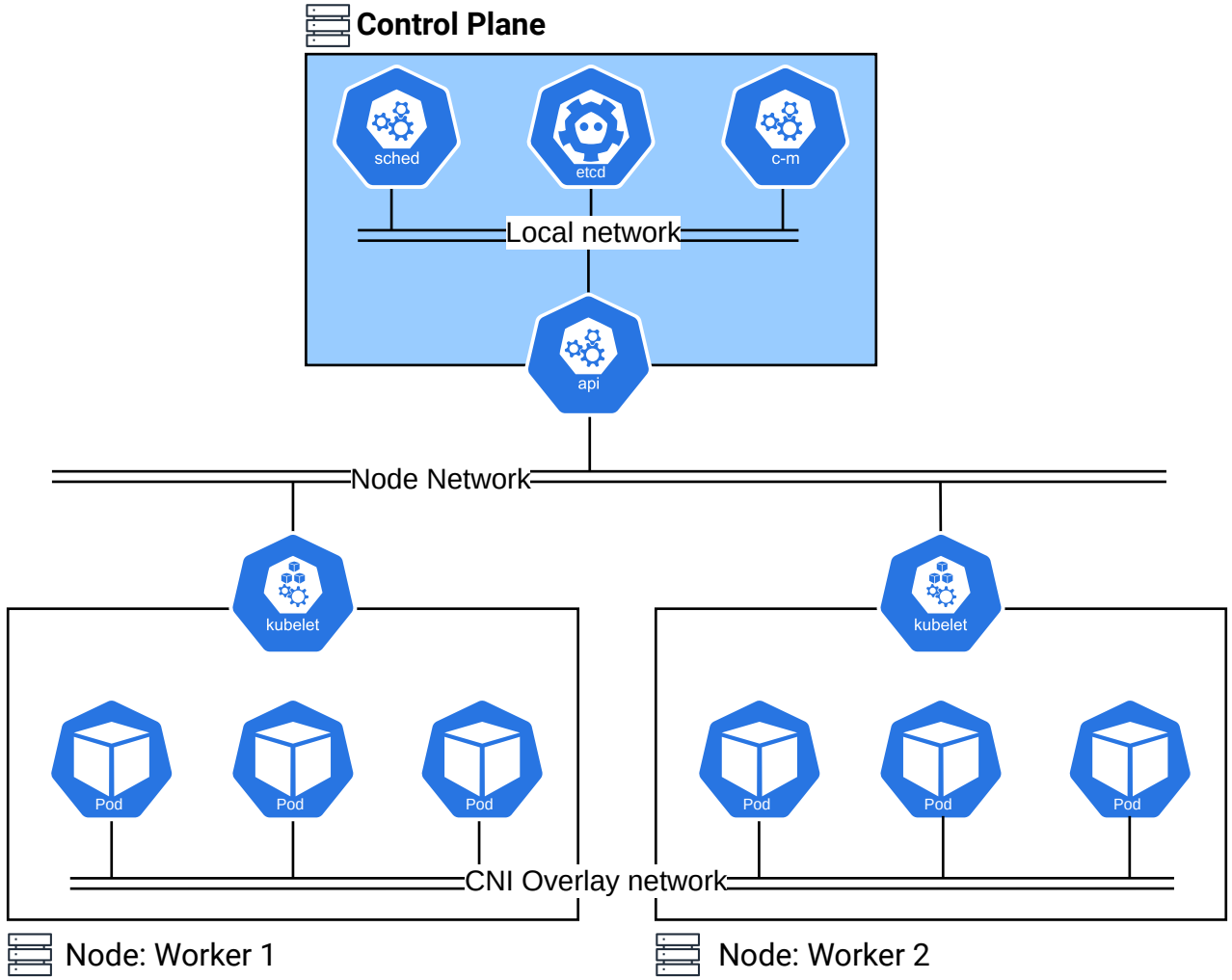
# Kubernetes

## kubelet



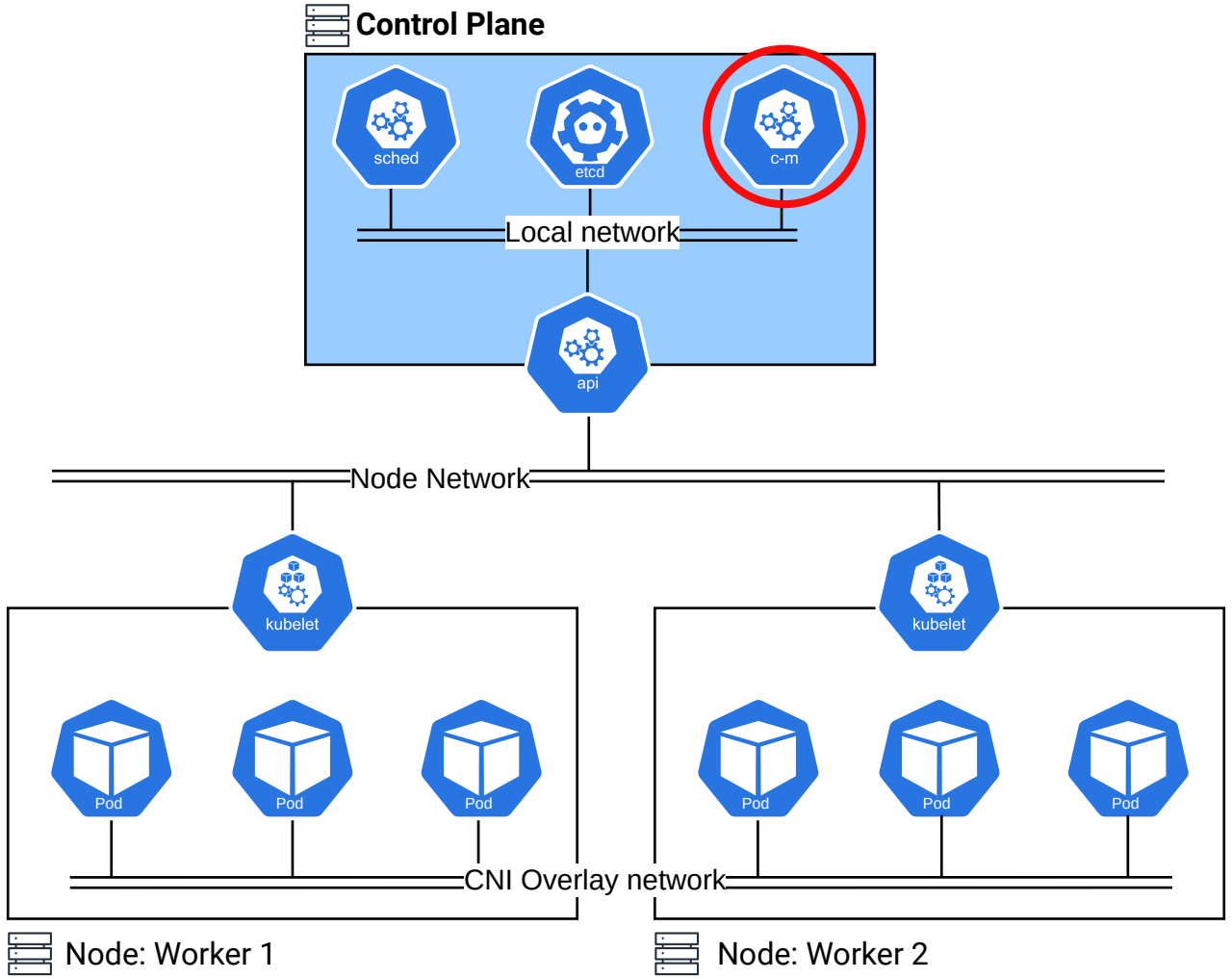
# Kubernetes

## Node: Control Plane



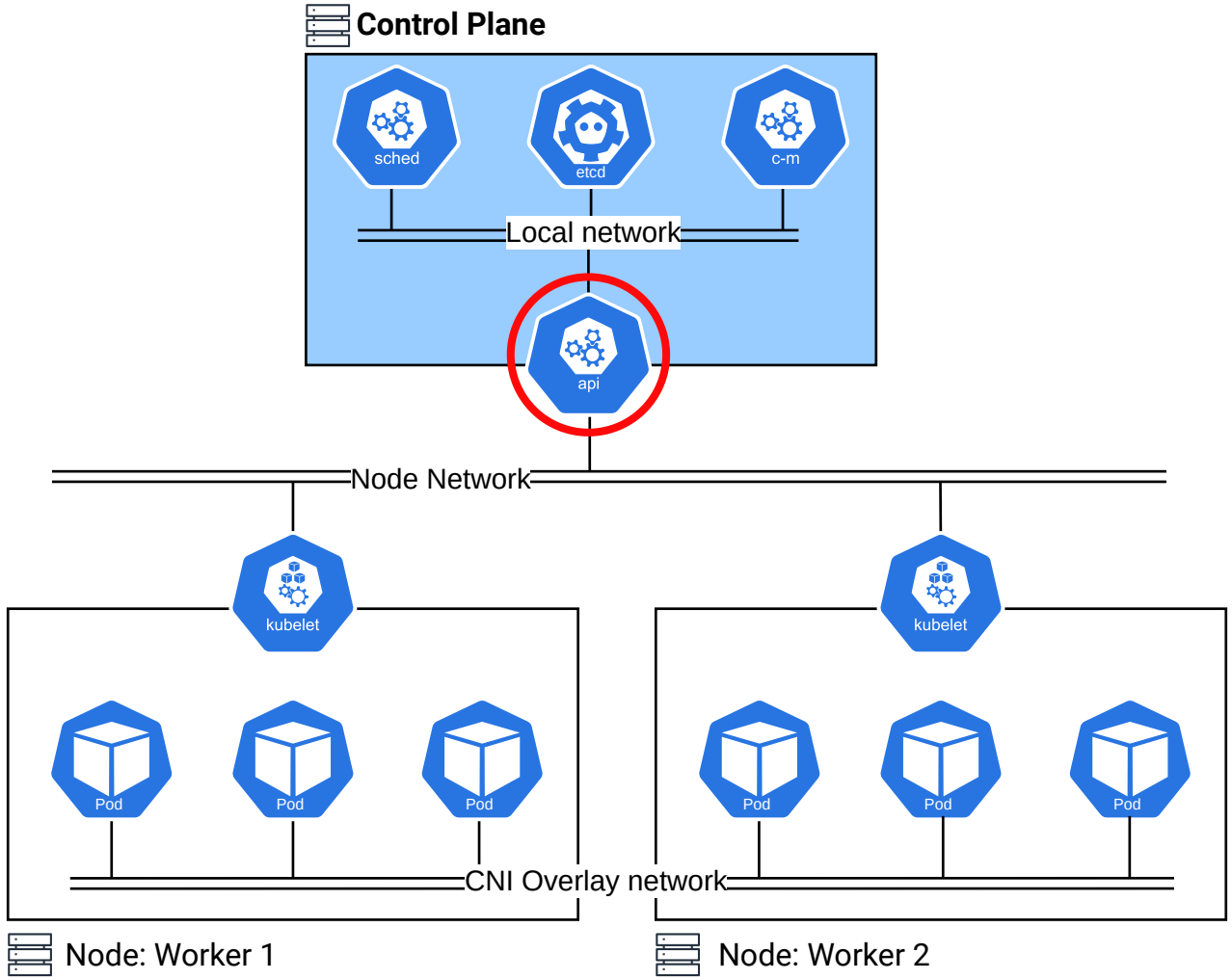
# Kubernetes

kube-controller-manager



# Kubernetes

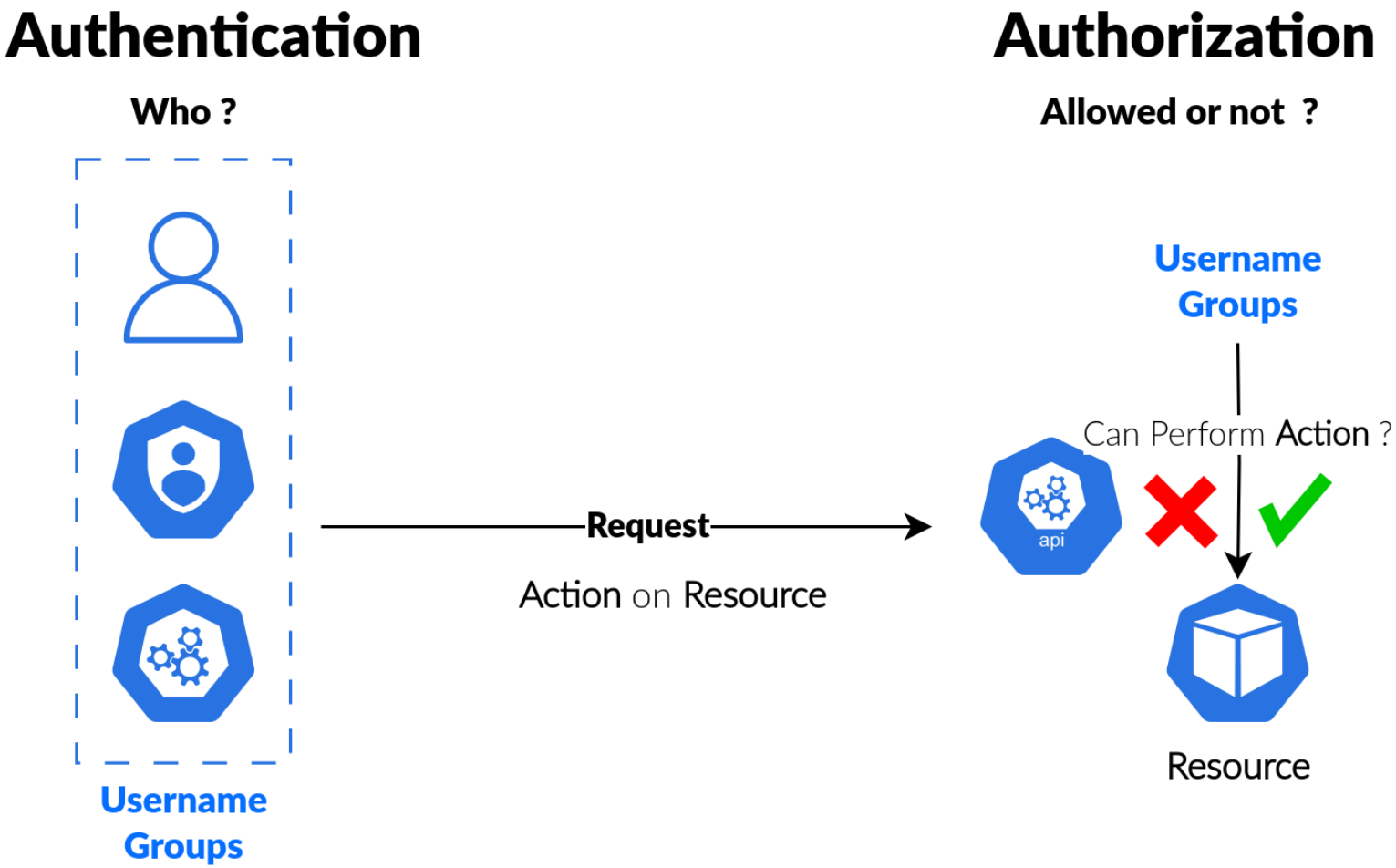
kube-apiserver



# Authentication & Authorization

# Authentication & Authorization

SYNACKTIV




TROOPERS

# Authentication & Authorization

SYNACKTIV

## Authentication

User

- X.509
- OID
- Proxy

Service account

- JWT

Infrastructure

- X.509

## Authorization

ABAC

RBAC

Webhook

Node Authorization Mode

# Authentication & Authorization

SYNACKTIV

## Authentication

User

- X.509
- OID
- Proxy

Service account

- JWT

Infrastructure

- X.509

## Authorization

ABAC

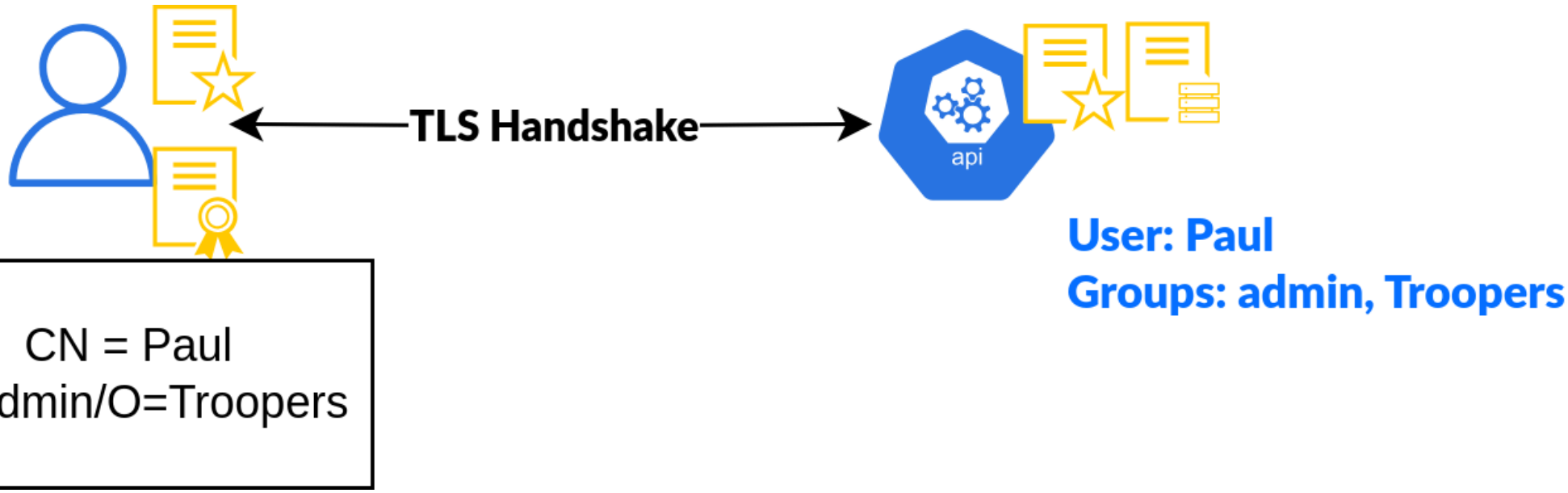
RBAC

Webhook

Node Authorization Mode

# X.509

User


TLS Handshake
api
CN = Paul
O=admin/O=Troopers
User: Paul
Groups: admin, Troopers

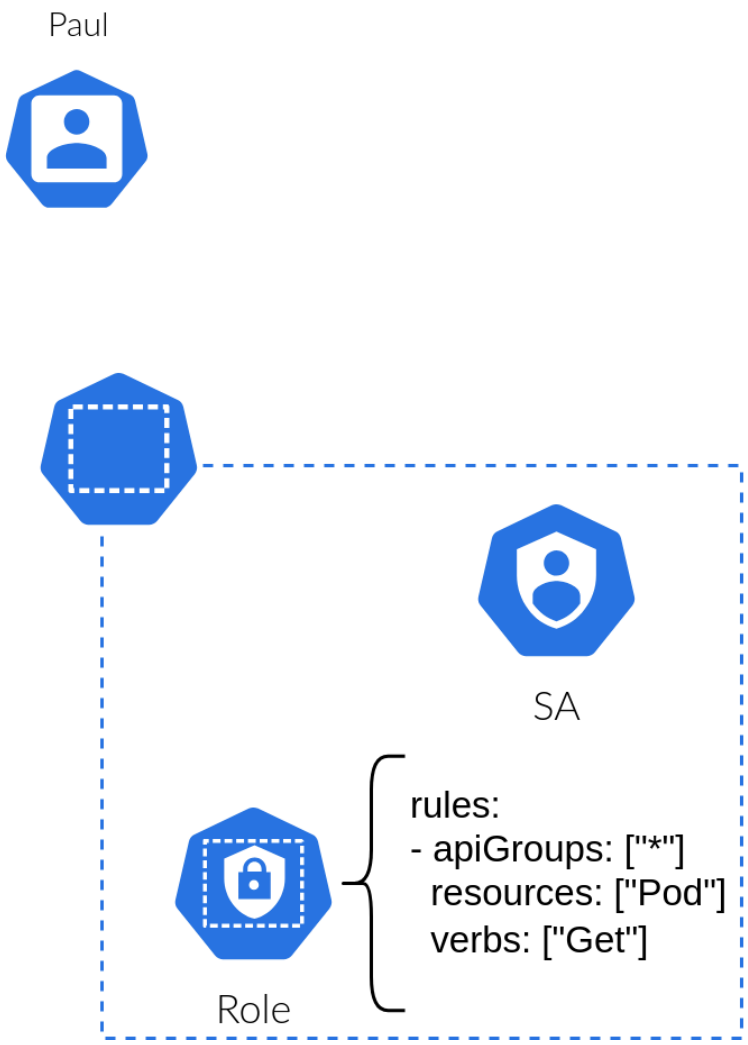
# X.509

Infra

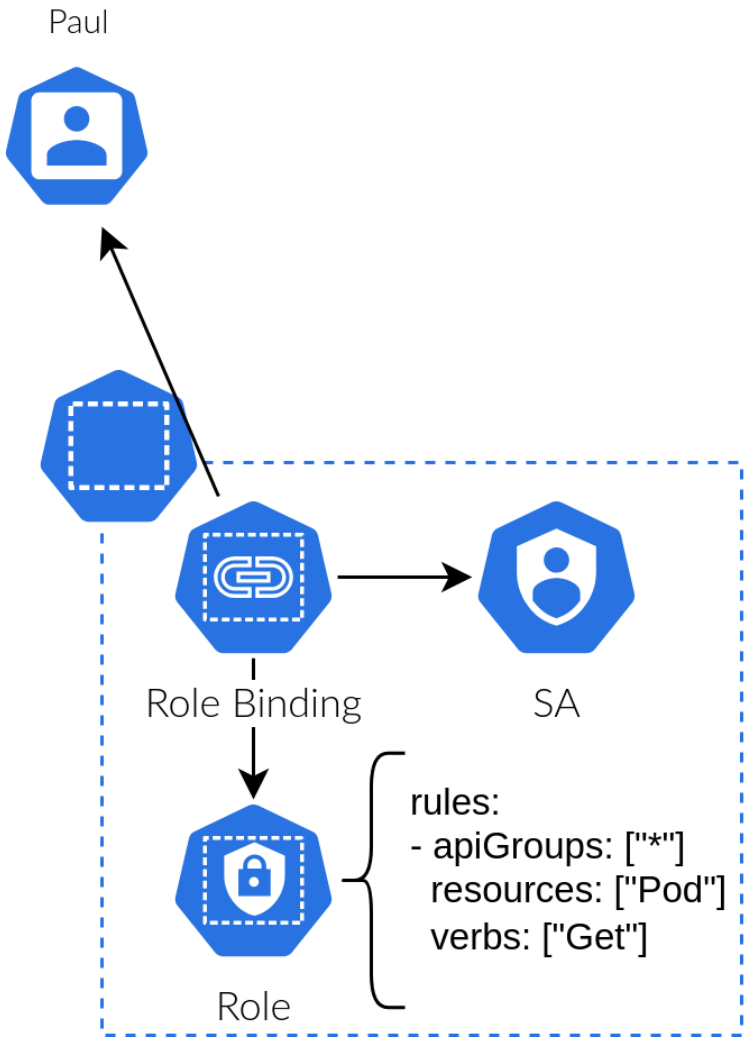

CN = system:node:<node name>
O=system:nodes

**User: system:node:<node name>**
**Groups: system:nodes**

# RBAC

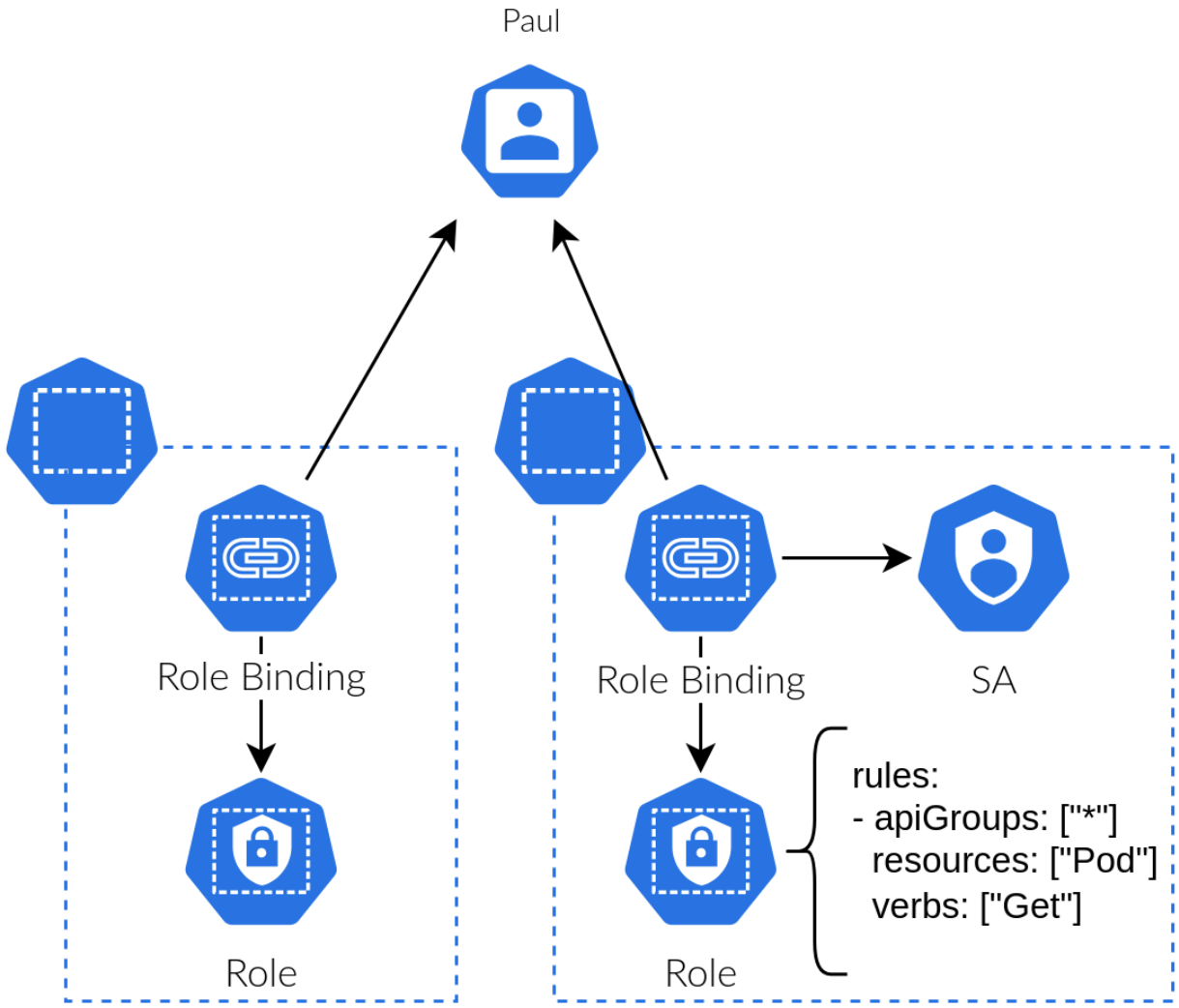


# RBAC


Paul
Role Binding
SA
rules:
- apiGroups: ["*"]
resources: ["Pod"]
verbs: ["Get"]
Role

# RBAC

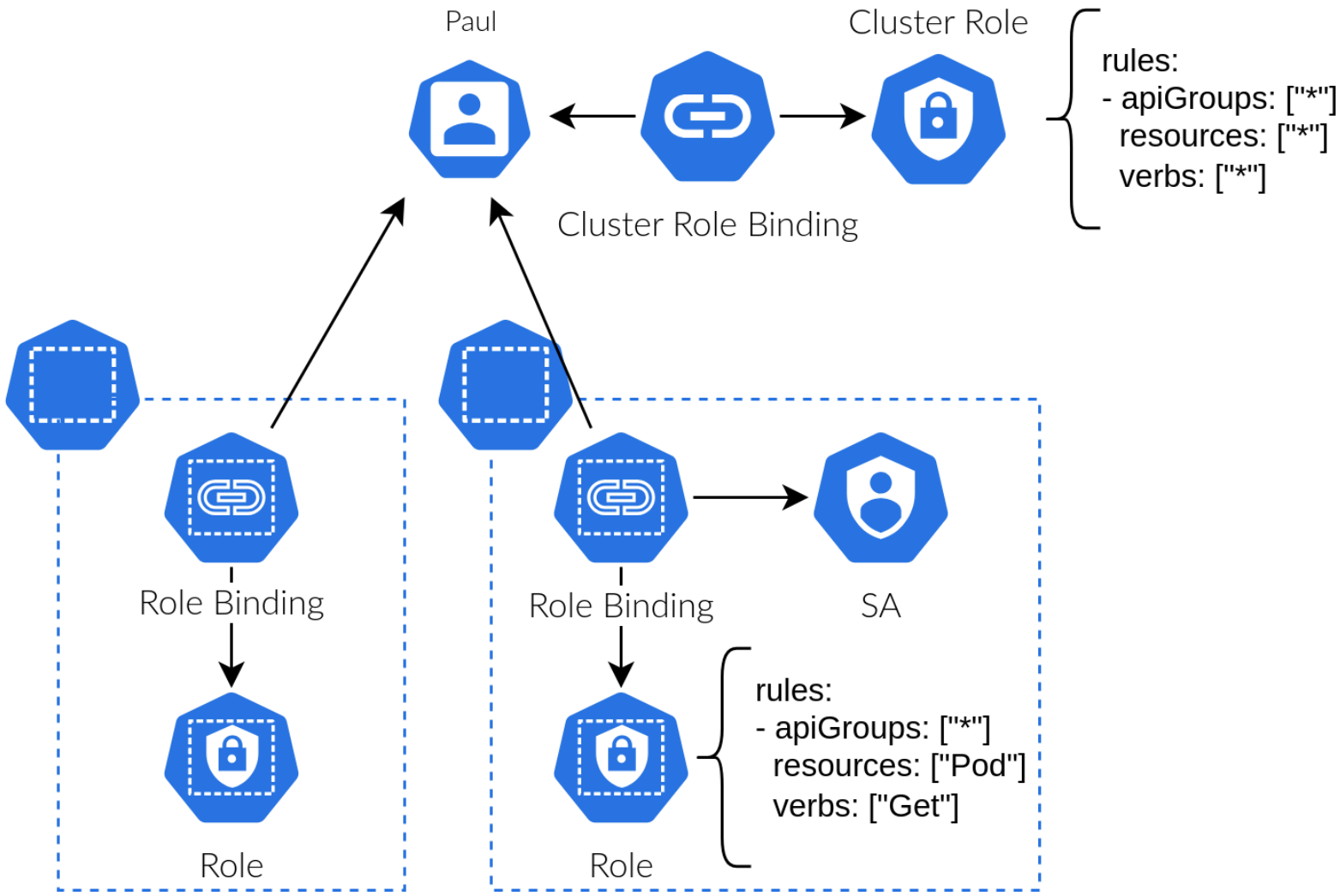
SYNACKTIV




TROOPERS

# RBAC

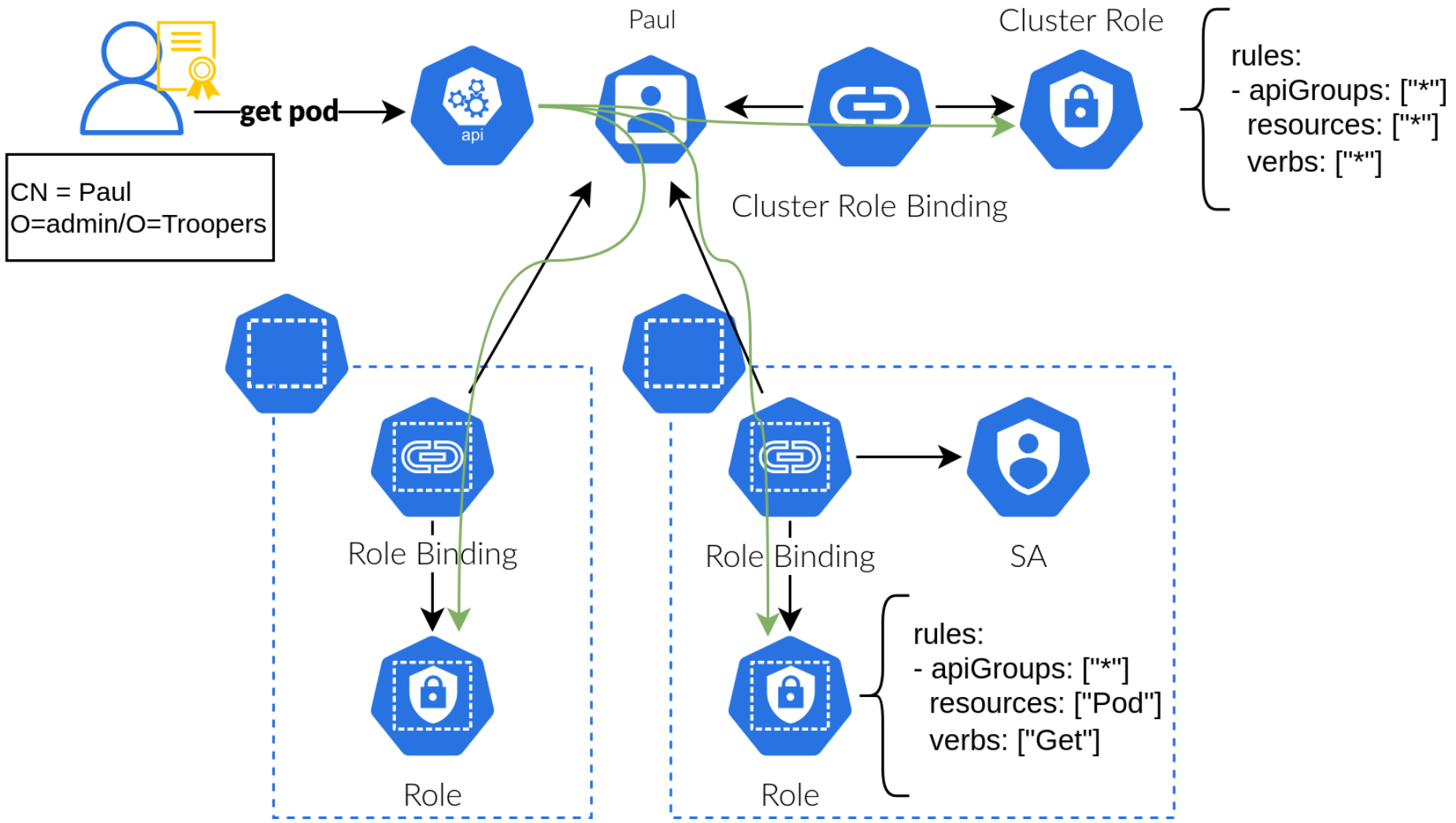
SYNACKTIV




TROOPERS

# RBAC

SYNACKTIV

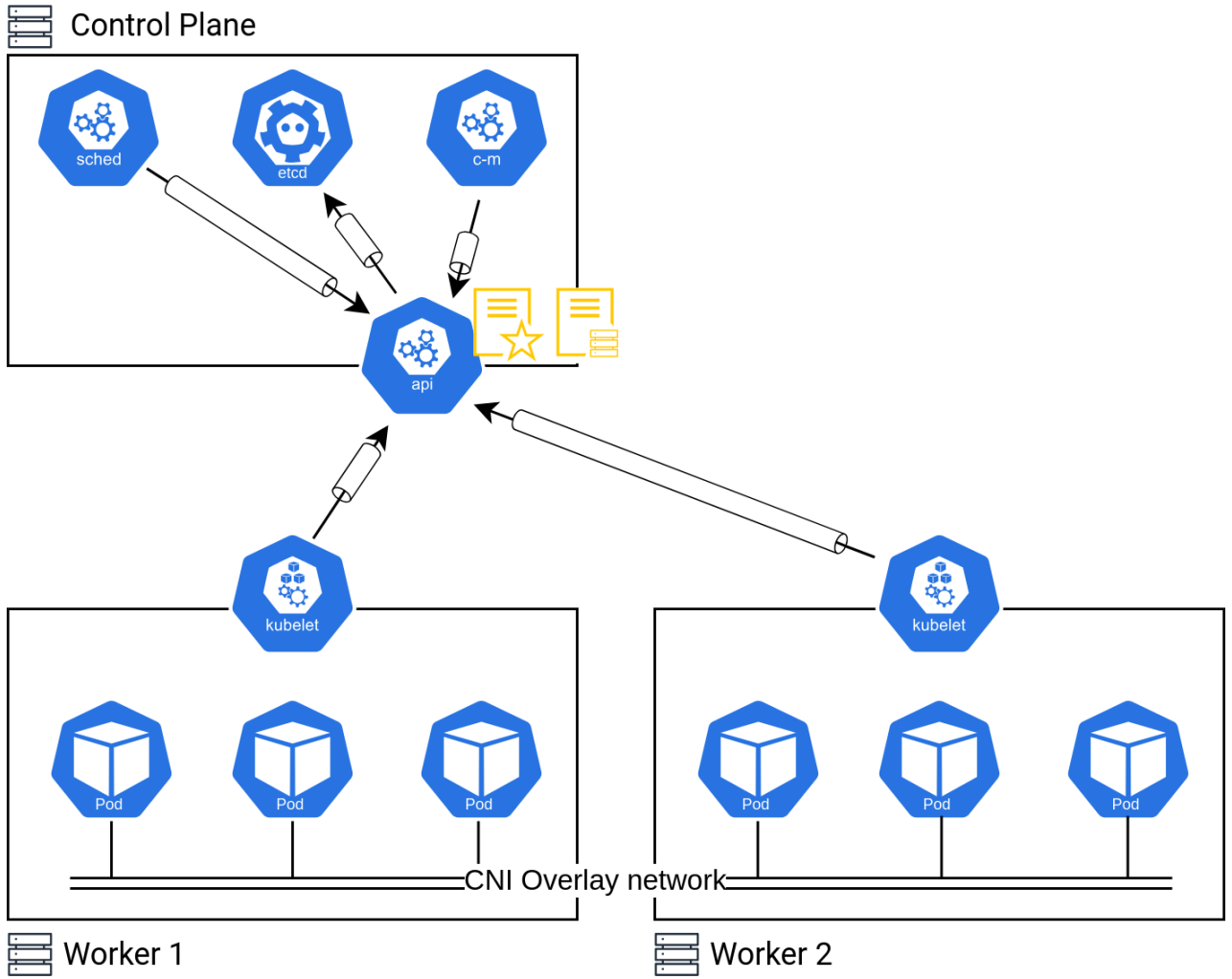



TROOPERS

# TLS Communication

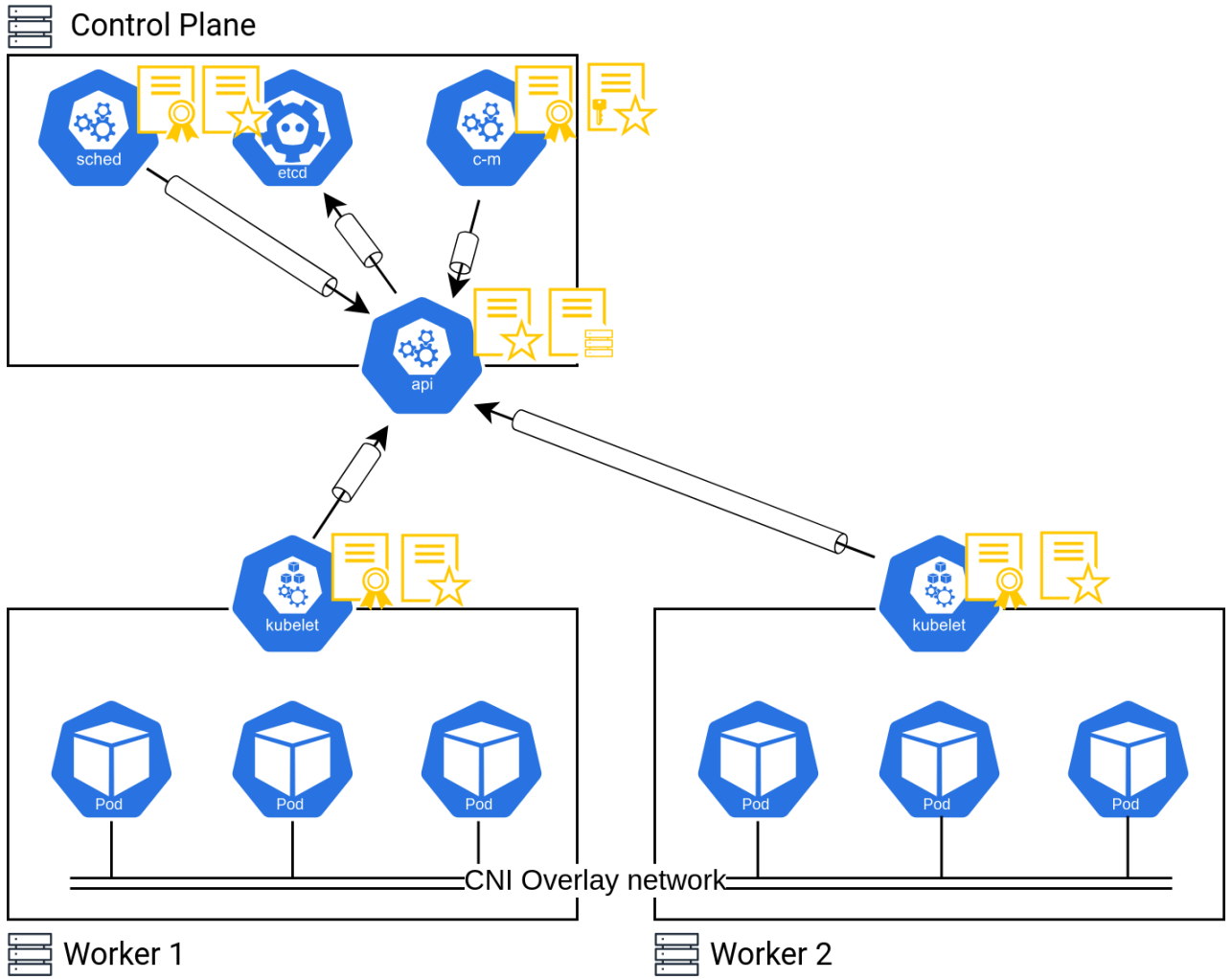
# TLS

## Components communication



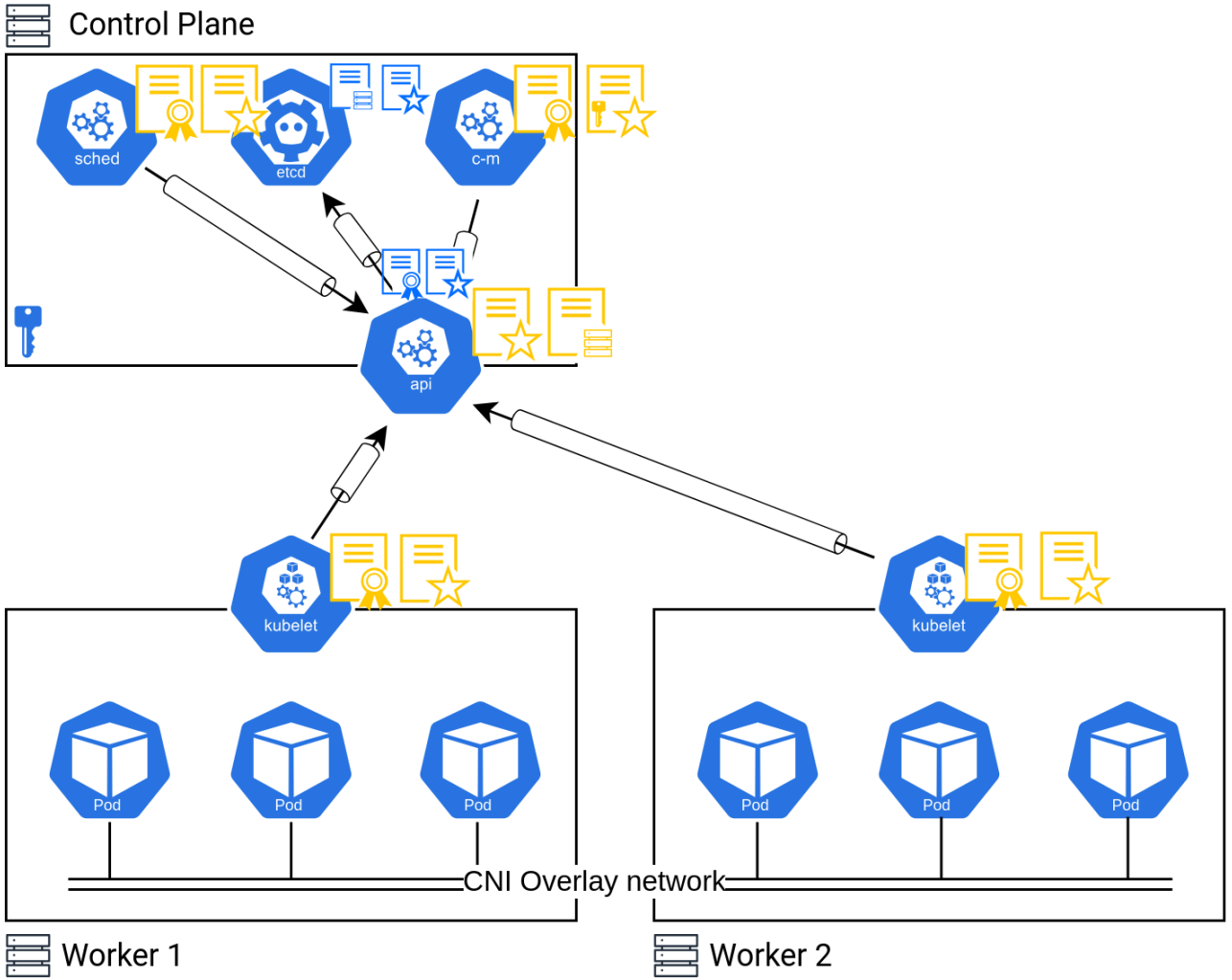
# TLS

## Client certs



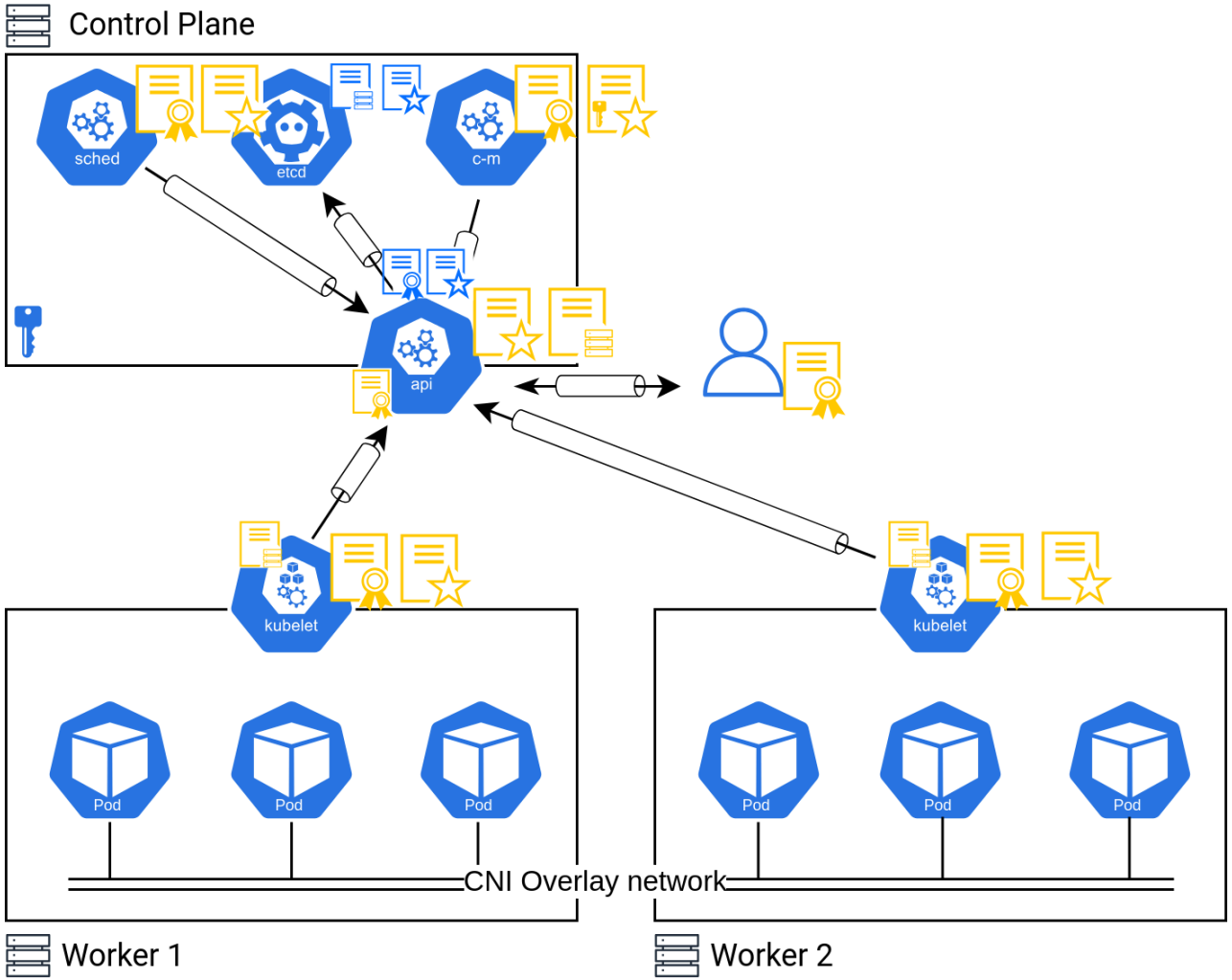
# TLS

ETCD certs



# TLS

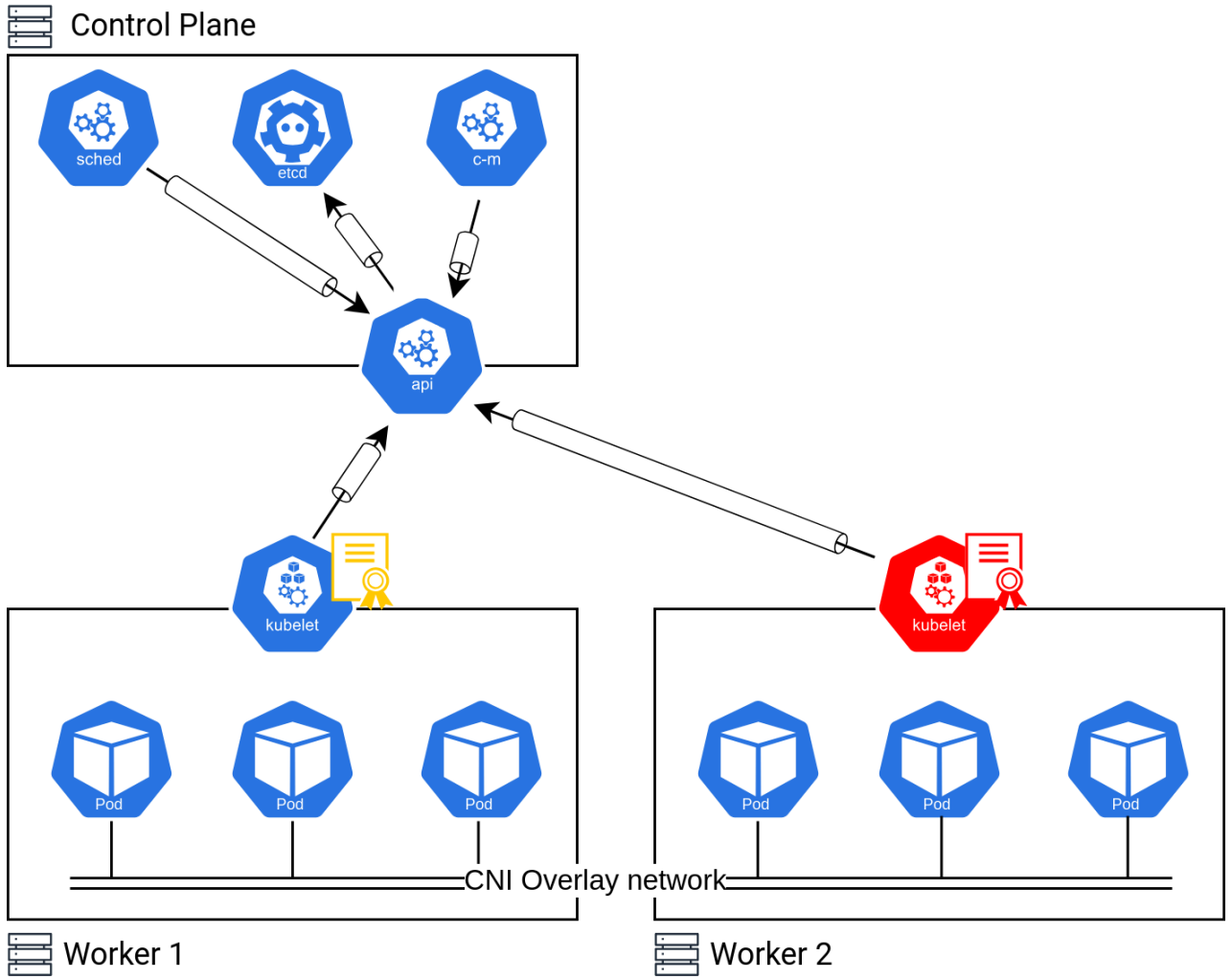
MORE CERTS !



# Node Authorization Mode

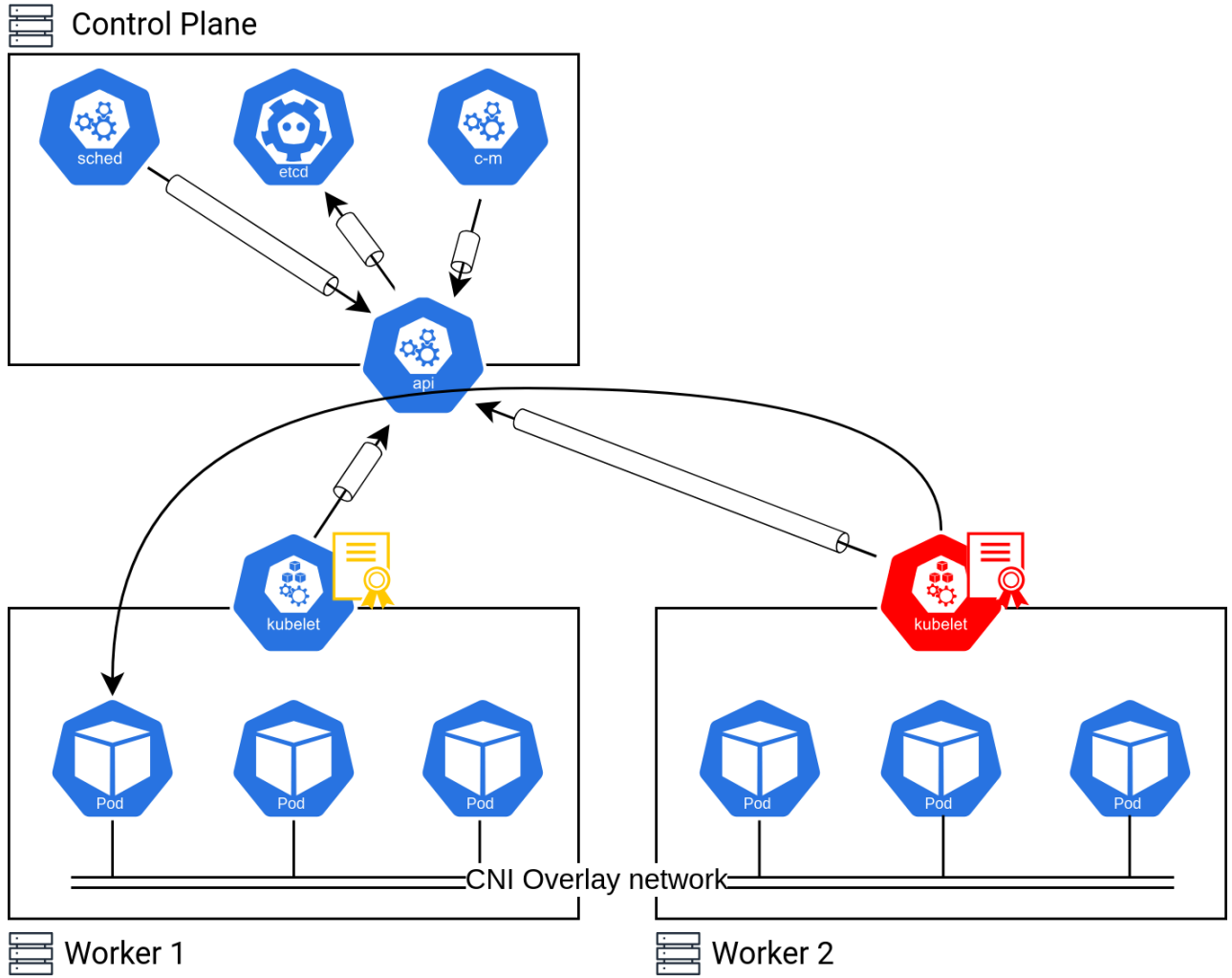
# Node Authorization Mode

Node compromise



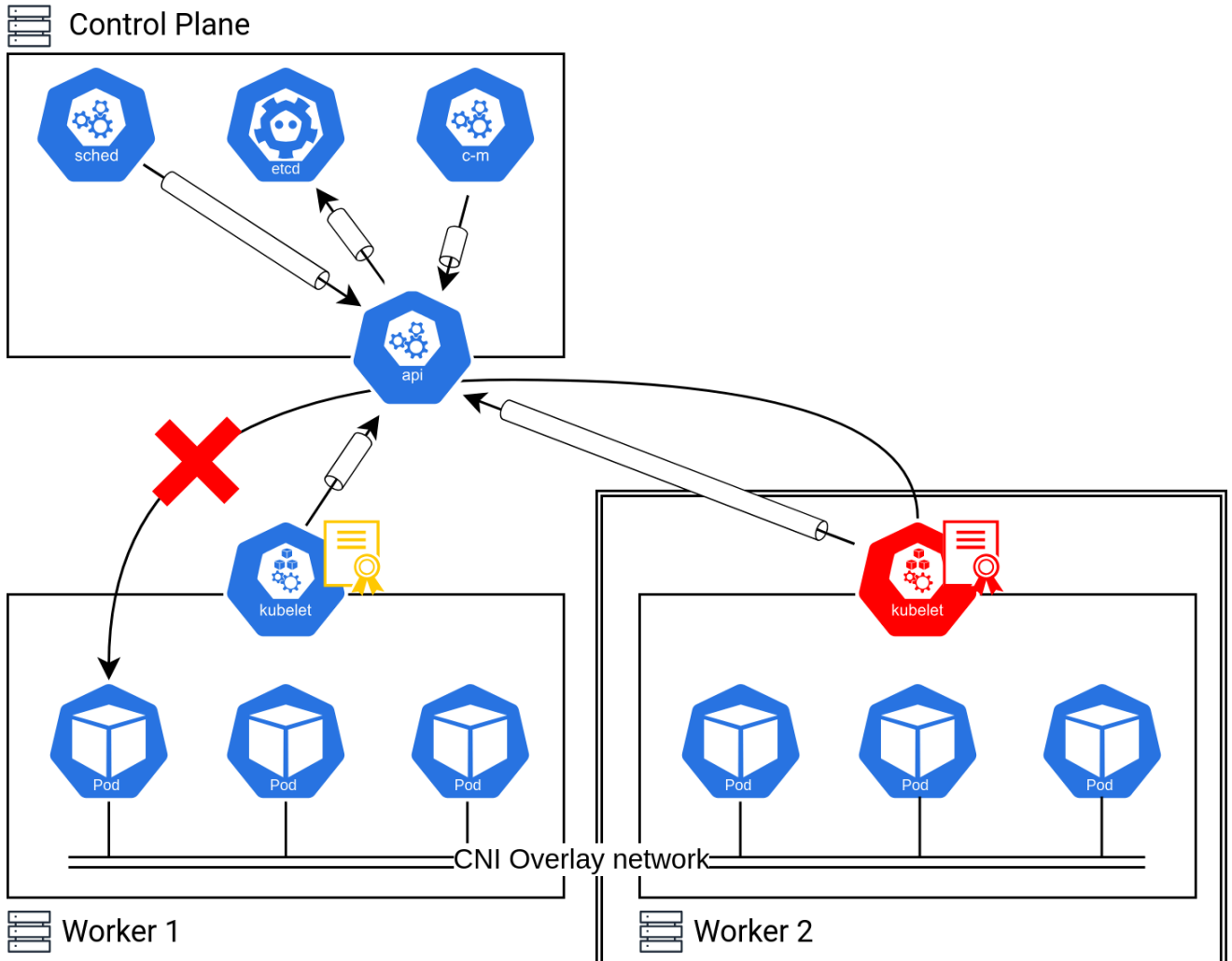
# Node Authorization Mode

Node compromise



# Node Authorization Mode

Node compromise



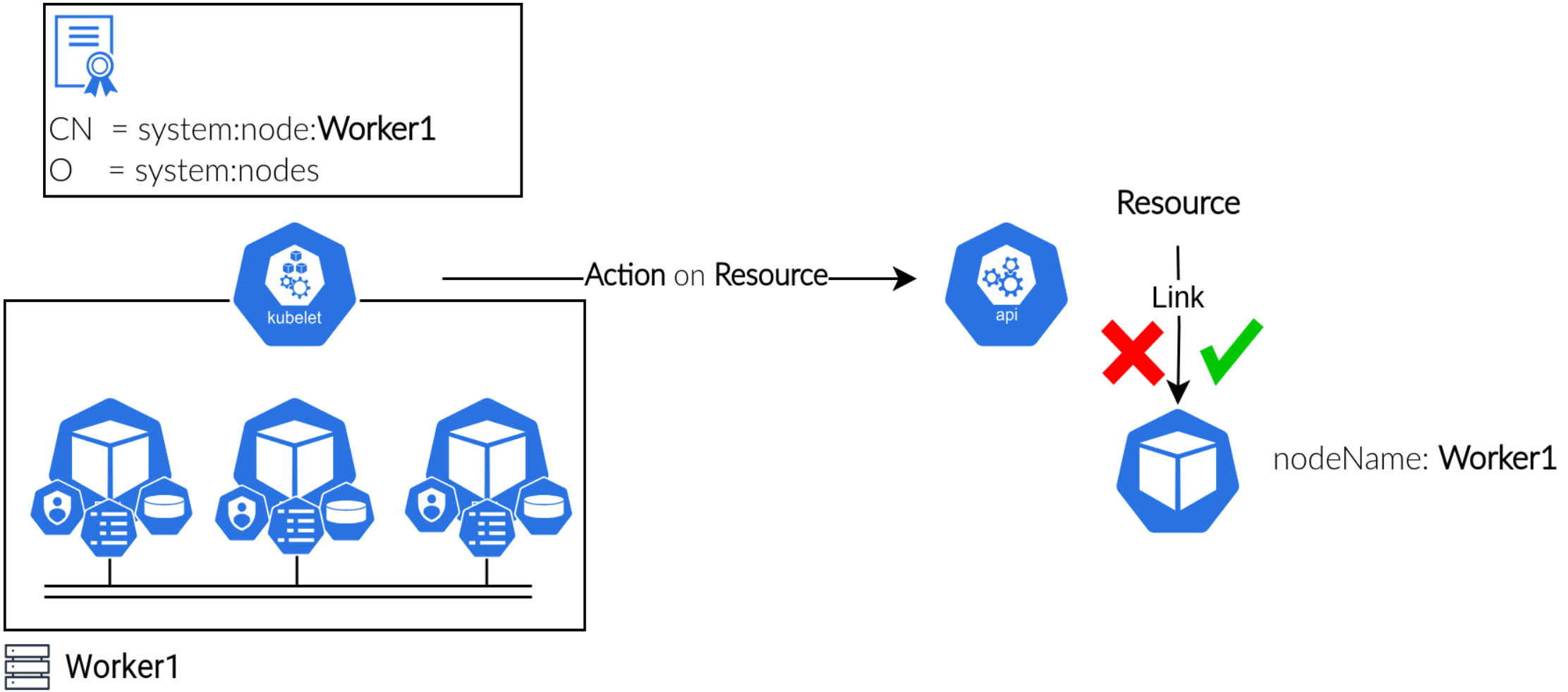
# Node Authorization Mode

SYNACKTIV

- Special authorization mode
  - User: `system:node:<node name>`
  - Group: `system:nodes`
- Prevent actions on resources not linked to the node
  - secrets, configmaps, persistent volumes, resource claims, persistent volume claims, volume attachments, service accounts, leases, csinodes

# Node Authorization Mode

SYNACKTIV



# Problems

SYNACKTIV

Complexity for certificates provisioning:

- Lots of certificates
- Different kind of certificates
- Strict conditions for the node Authorization mode
- Auto-scaling cluster

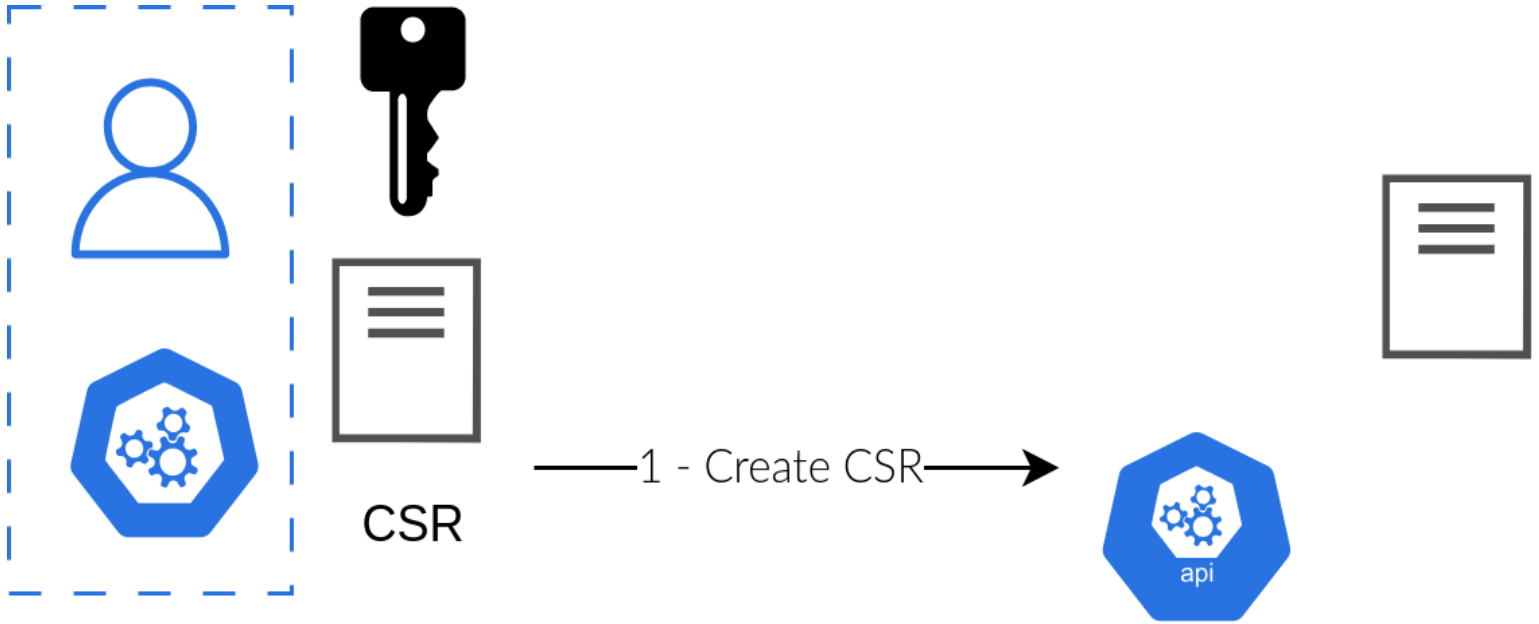
# Certificate Signing Request

# Certificate Signing Request

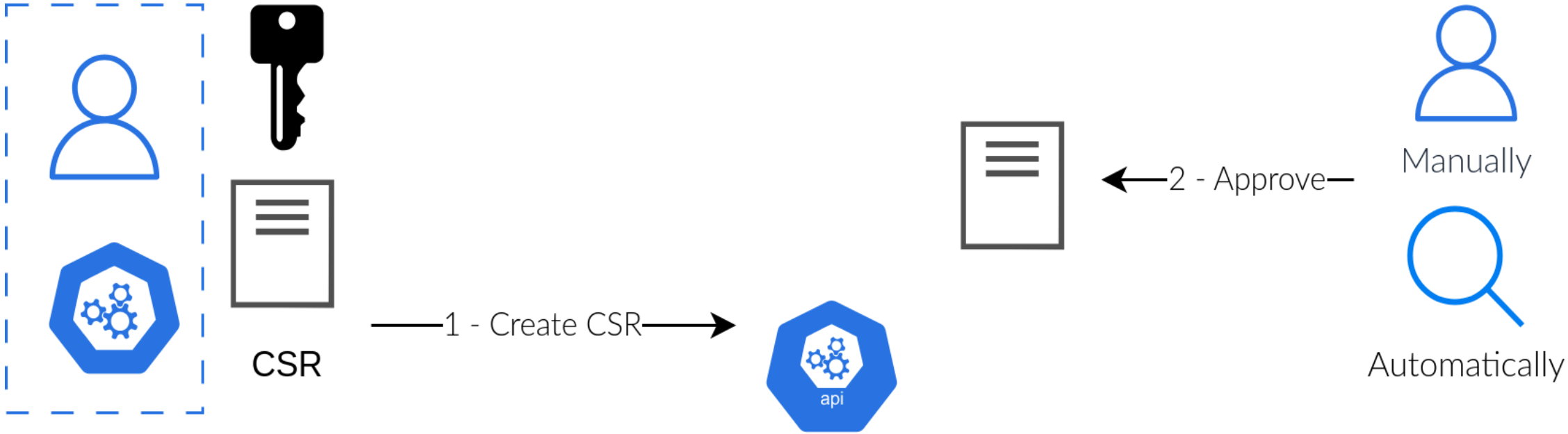
SYNACKTIV

- Builtin way to sign certificate
- Auto or manual approved
- Signer
  - `kubernetes.io/kube-apiserver-client-kubelet`

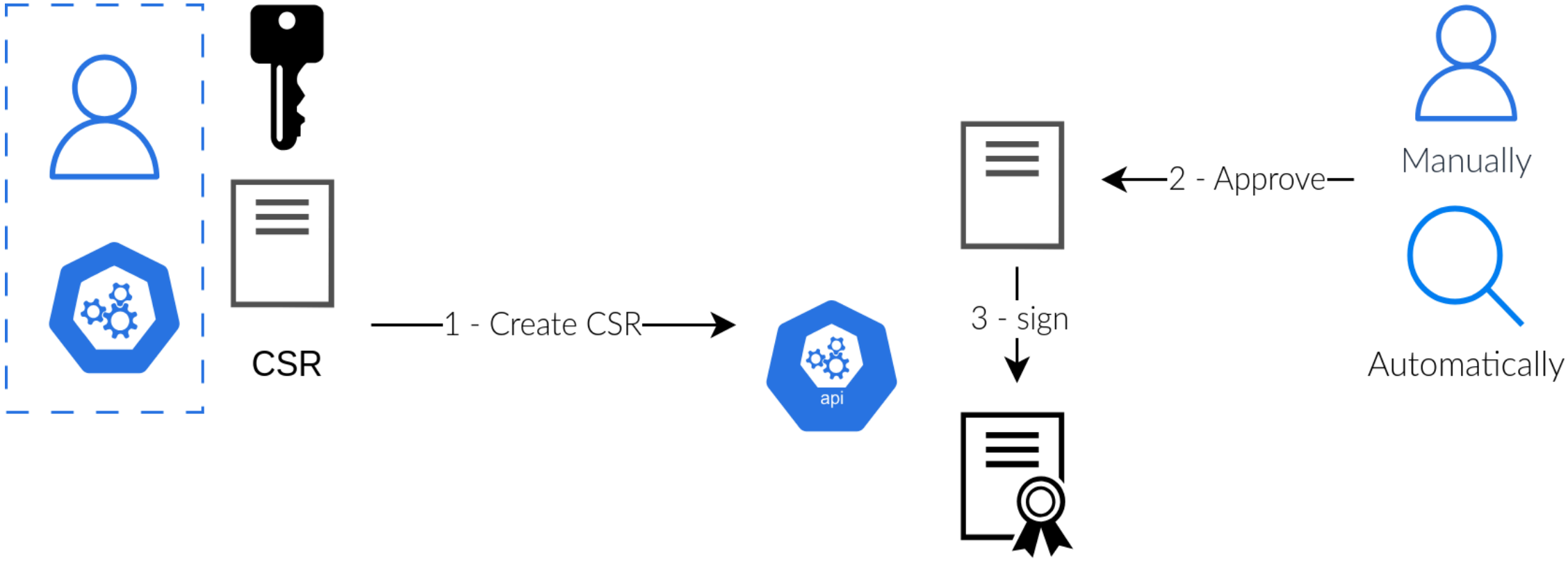
# CSR



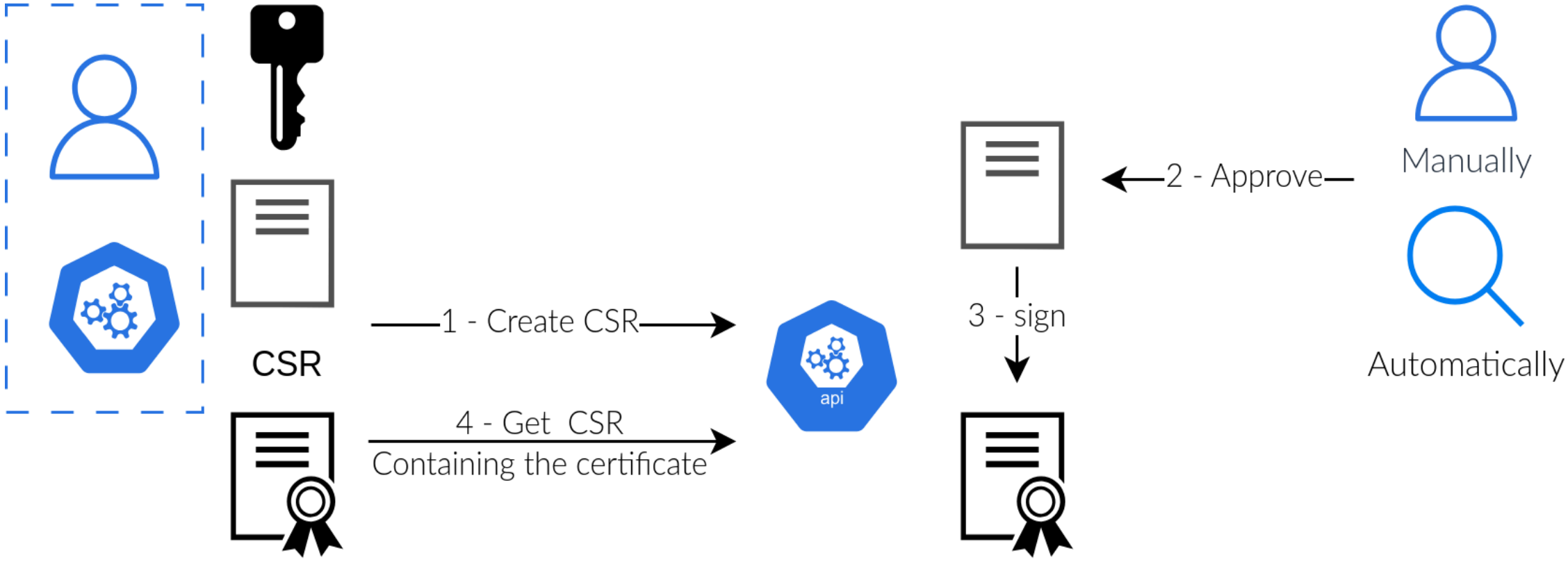
# CSR



# CSR


CSR
1 - Create CSR
api
3 - sign
2 - Approve
Manually
Automatically

# CSR


CSR
1 - Create CSR
api
4 - Get CSR
Containing the certificate
3 - sign
2 - Approve
Manually
Automatically

# What is missing ?

SYNACKTIV

- Privileges to create and retrieve CSR
  - `create`, `get`, `list` and `watch` on `certificatesigningrequests`
- Auto approval need a specific permission
  - `create certificatesigningrequests.certificates.k8s.io/nodeclient`

→ Bootstrap token

# Bootstrap tokens

# Bootstrap token

SYNACKTIV

- `Secret` in the cluster
- Give needed privileges for CSR
- Auto approval for kubelet client certificate
- Authenticate the API server
  - `configmap` containing the CA signed with the `token-secret`

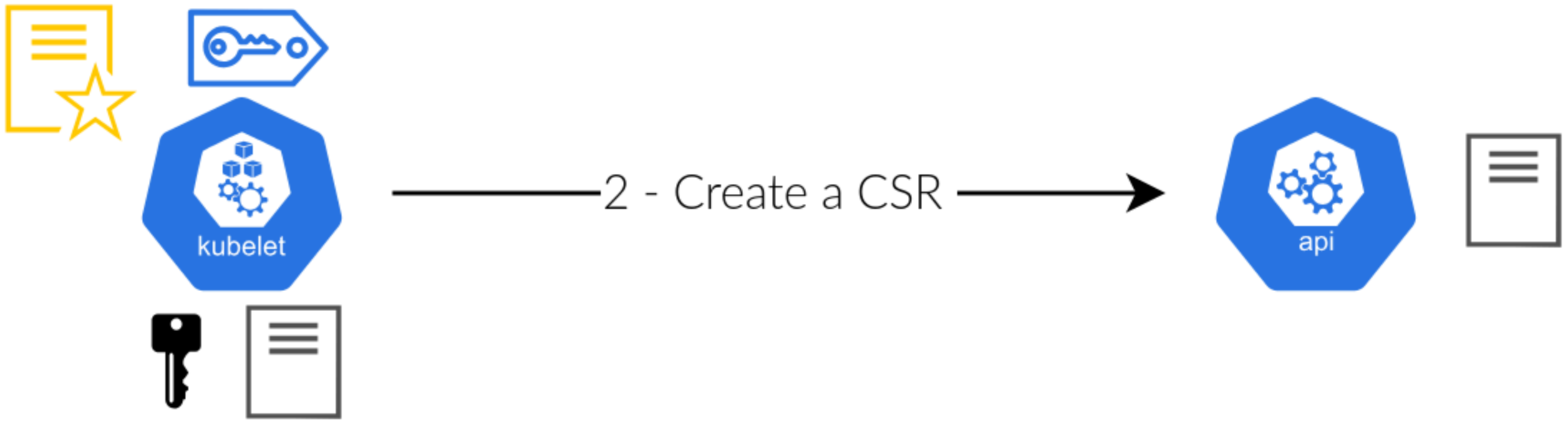
# Secret in the cluster

SYNACKTIV

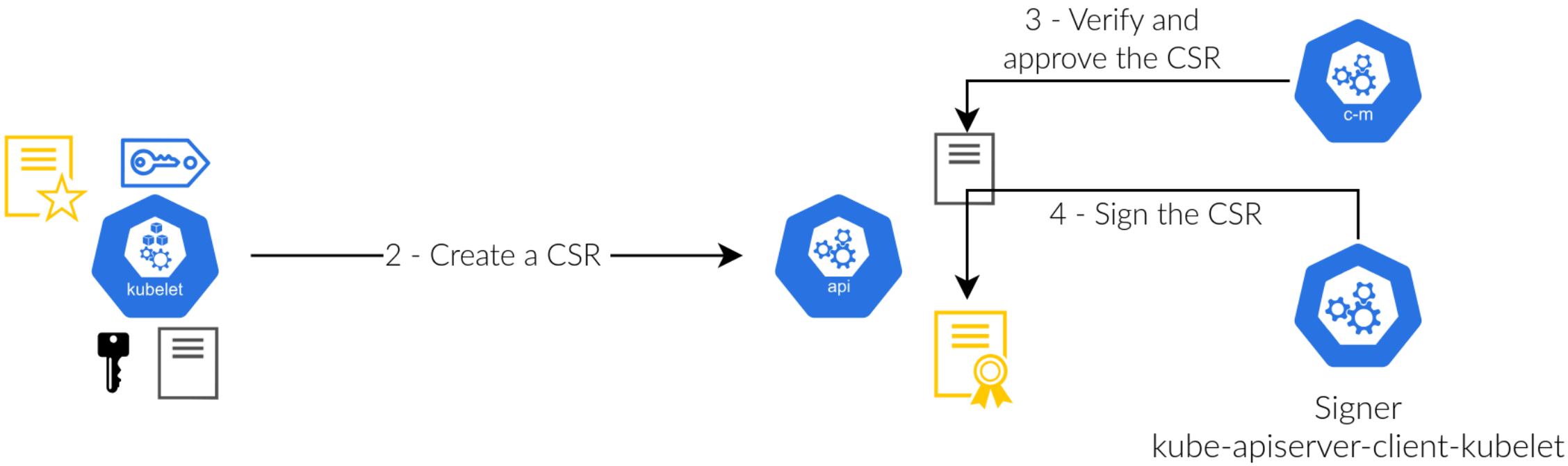
```yaml
apiVersion: v1
kind: Secret
metadata:
  # Name MUST be of form "bootstrap-token-<token id>"
  name: bootstrap-token-07401b
  namespace: kube-system
# Type MUST be 'bootstrap.kubernetes.io/token'
type: bootstrap.kubernetes.io/token
stringData:
  # Human readable description. Optional.
  description: "The default bootstrap token generated by 'kubeadm init'."
  # Token ID and secret. Required.
  token-id: 07401b
  token-secret: f395accd246ae52d
  # Expiration. Optional.
  expiration: 2017-03-10T03:22:11Z
  # Allowed usages.
  usage-bootstrap-authentication: "true"
  usage-bootstrap-signing: "true"
  # Extra groups to authenticate the token as. Must start with "system:bootstrappers:"
  auth-extra-groups: system:bootstrappers:worker,system:bootstrappers:ingress
```

# Bootstrap Flow

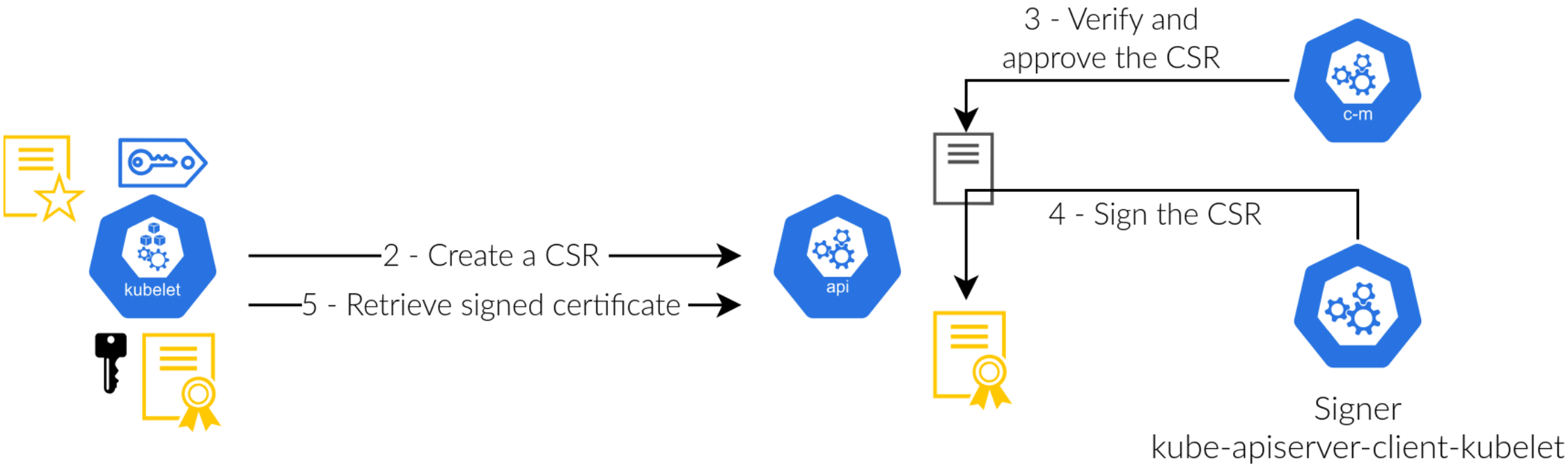
# Bootstrap Flow



# Bootstrap Flow



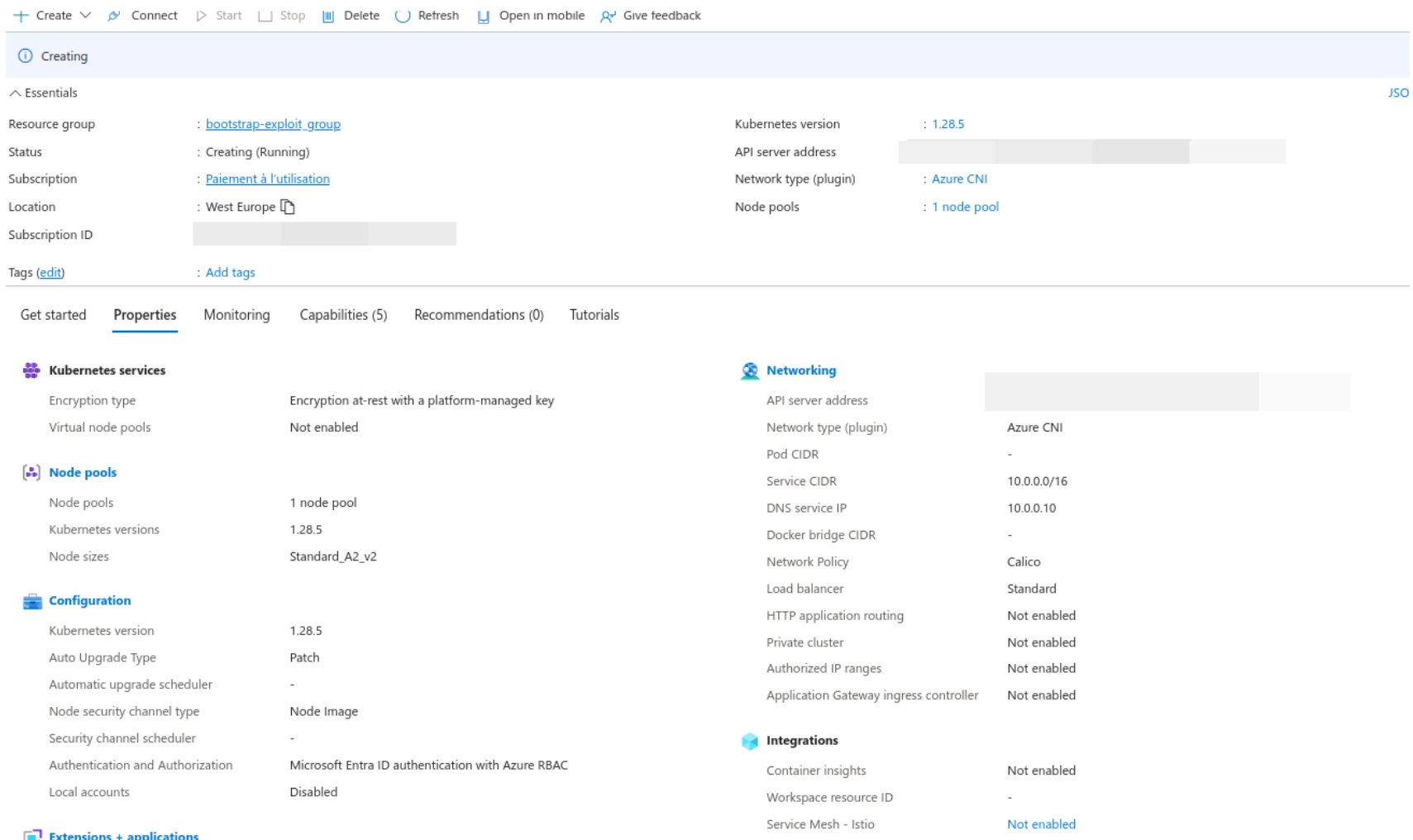
# Bootstrap Flow



# Lab setup

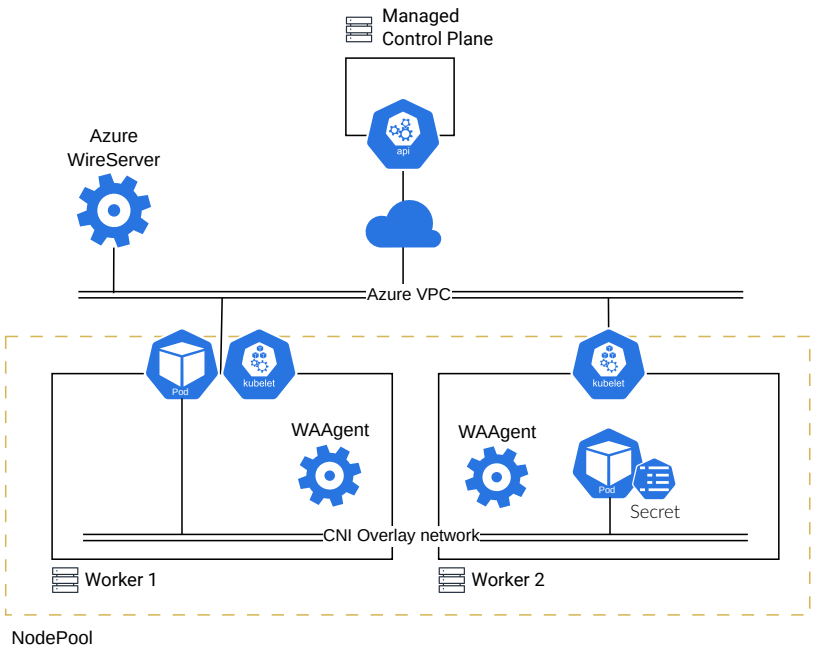
# Lab setup

## Cluster



# Lab setup

## Cluster




- Running nodes

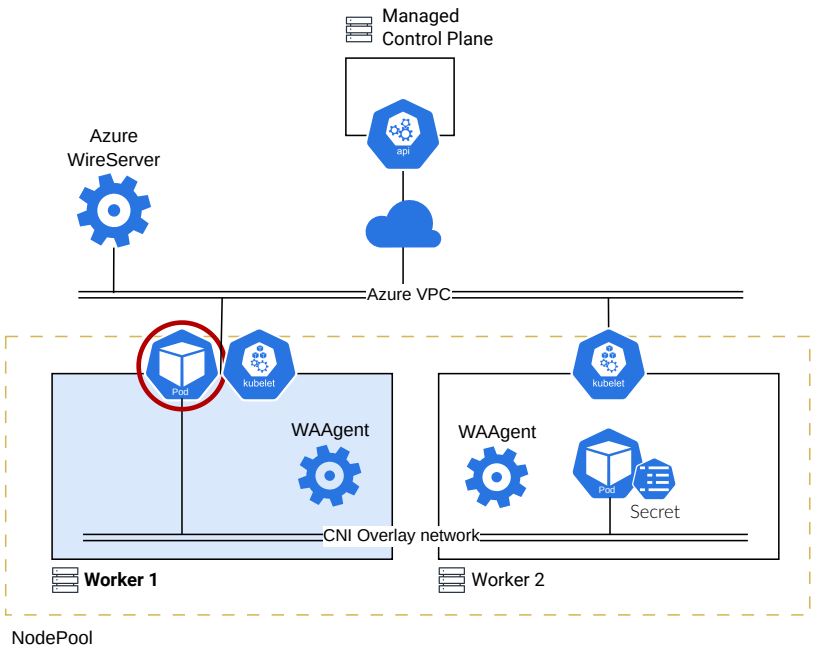
```
$ kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-agentpool-21994783-vmss000000   Ready    agent   27m   v1.28.9
aks-agentpool-21994783-vmss000001   Ready    agent   27m   v1.28.9
```

- Running pods

```
$ kubectl get pod -o wide -n default
NAME              READY   STATUS      NODE
pod-hostnetwork   1/1     Running      aks-agentpool-21994783-vmss000000
pod-target        1/1     Running      aks-agentpool-21994783-vmss000001
```

# Lab setup

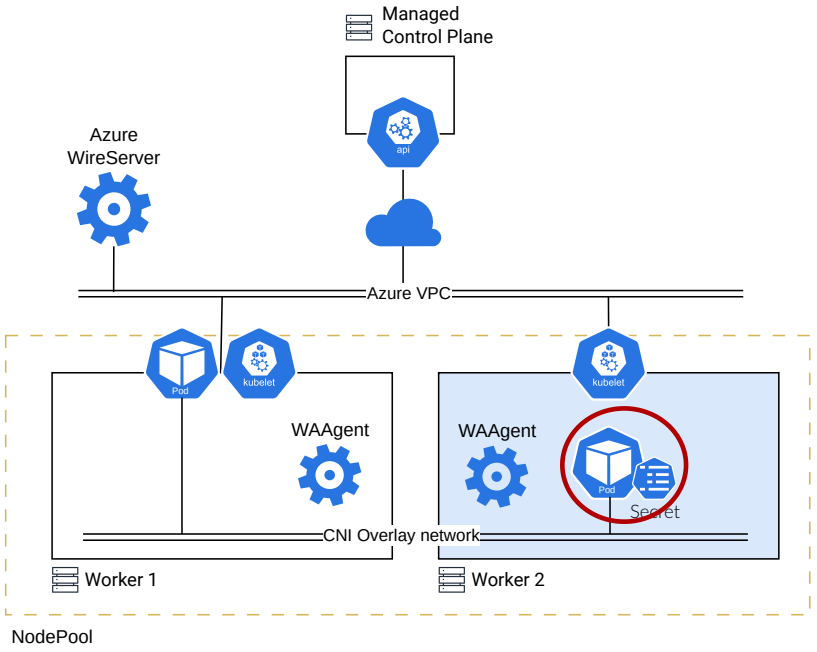
## Worker 1




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-hostnetwork
spec:
  nodeName: aks-agentpool-21994783-vmss000000
  containers:
  - name: nginx
    image: nginx
  hostNetwork: true
```

# Lab setup

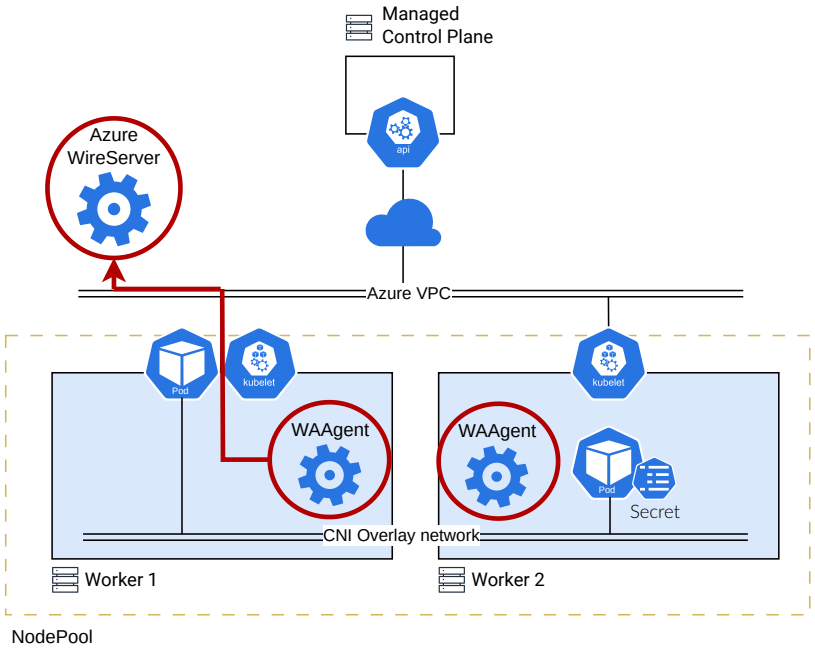
## Worker 2




```
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
data:
   secret: TXlTdXBlclNlY3JldAo=
---
apiVersion: v1
kind: Pod
metadata:
  name: pod-target
spec:
  nodeName: aks-agentpool-21994783-vmss000001
  volumes:
  - name: secret
    secret:
      secretName: my-secret
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - name: secret
      mountPath: /secret
```

# Lab setup

## WireServer & WAAgent

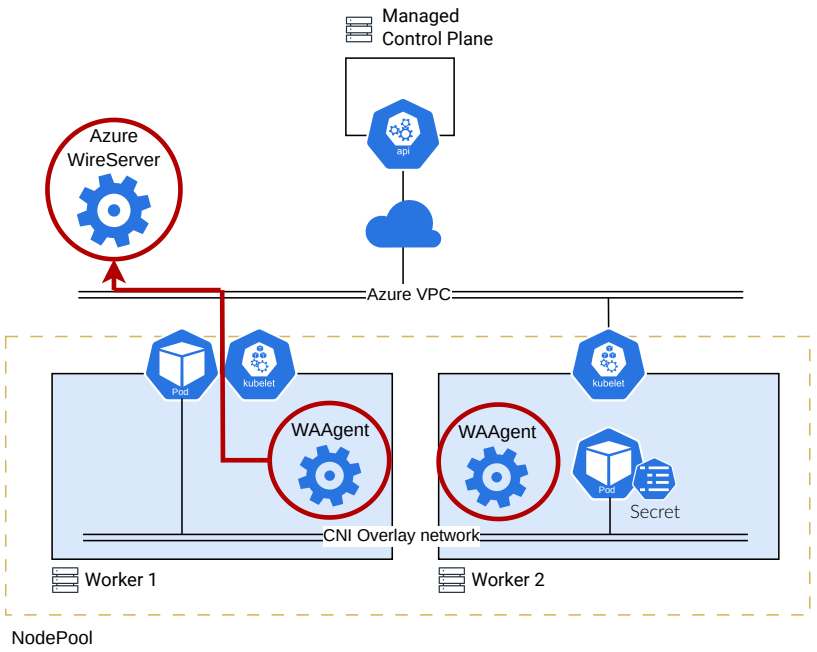



- http://168.63.129.16:32526/vmSettings

```
$ curl http://168.63.129.16:32526/vmSettings
[...]
{
  [...]
  "extensionGoalStates": [
    {
      [...]
      "settings": [
        {
          "protectedSettingsCertThumbprint": "331AA8624[...],
          "protectedSettings": "MIJkLAYJKoZI[...]",
          "publicSettings": "{}"
        }
      ]
    }
    [...]
```

# Lab setup

## WireServer & WAAgent

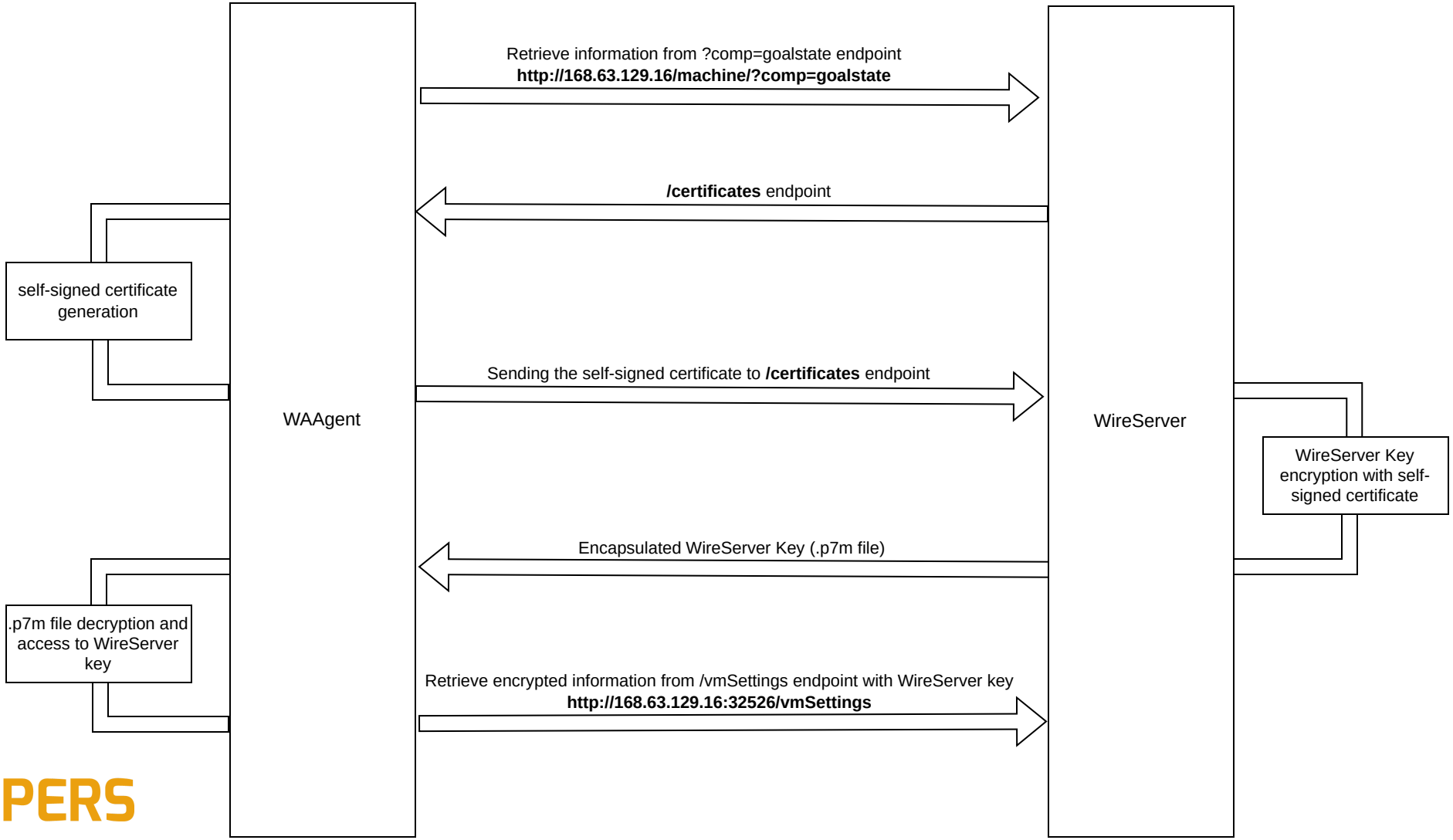



- http://168.63.129.16/machine/?comp=goalstate

```
$ curl 'http://168.63.129.16/machine/?comp=goalstate' -H 'x-ms-version: 2015-04-05'
<?xml version="1.0" encoding="utf-8"?>
[...]
  <RoleInstance>
    <State>Started</State>
     <Configuration>
      [...]
      <Certificates>http://168.63.129.16:80/machine/e98c9d15-6215[...]</Certificates>
     </Configuration>
   </RoleInstance>
  </RoleInstanceList>
 </Container>
</GoalState>
```

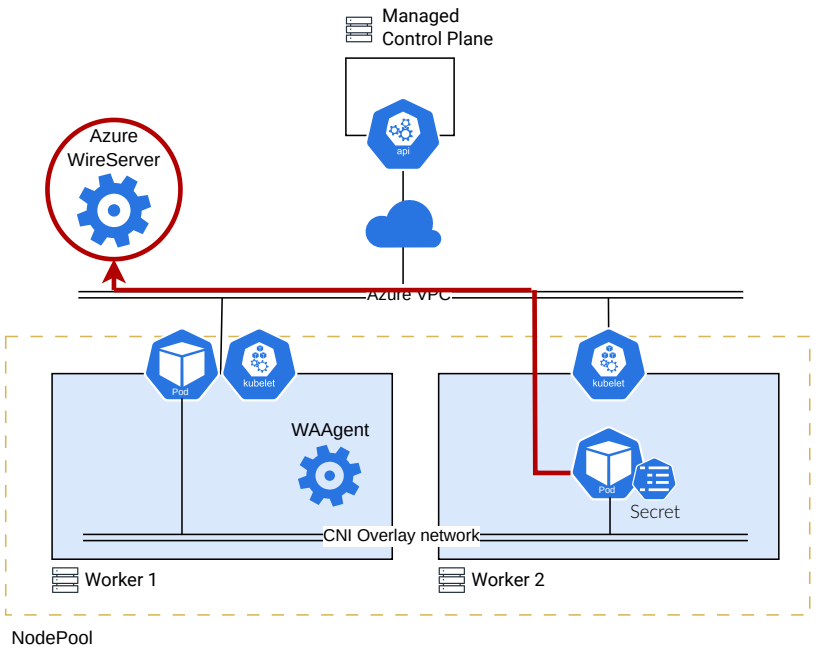
# Lab setup

## WireServer Key retrieval workflow



# Lab setup

## HostNetwork namespace

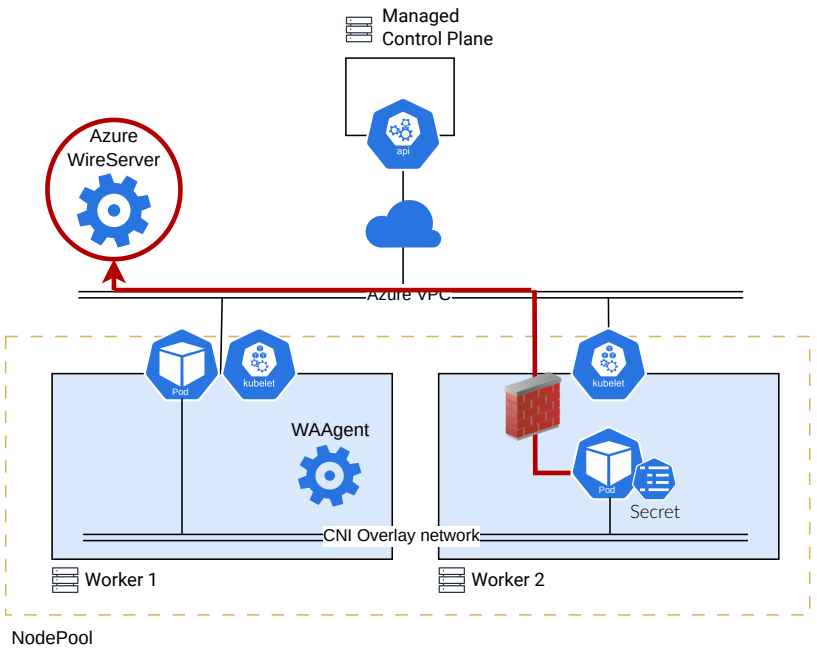



- HostNetwork namespace not shared

```
root@pod-target:/# curl 'http://168.63.129.16/machine/?comp=goalstate' \
                        -H 'x-ms-version: 2015-04-05'
curl: (28) Failed to connect to 168.63.129.16 port 80 after 129840 ms:
                        Couldn't connect to server
```

# Lab setup

## HostNetwork namespace




- Firewall rule on the host

```
root@aks-pool-xx-vmss000000:/# iptables -vnL
[...]
Chain FORWARD (policy ACCEPT 862 packets, 97768 bytes)
 pkts bytes target     source       destination
[...]
    0     0 DROP       0.0.0.0/0    168.63.129.16        tcp dpt:80
```

# Demo

# Demo

Presentation of the attack

SYNACKTIV

## Leak of the bootstrap token from within *pod-hostnetwork*

1. Retrieving the WireServer decryption key (using the Cybercx' script)
2. Decrypting the bootstrap token

## Compromise of kubelets' identities

3. Using the bootstrap token to sign a kubelet certificate for a non-existing node
4. Using this certificate to list other nodes and pods
5. Signing a certificate for the node where `target-pod` is deployed
6. Retrieving the secret inside the pod

# Demo 1

# Demo

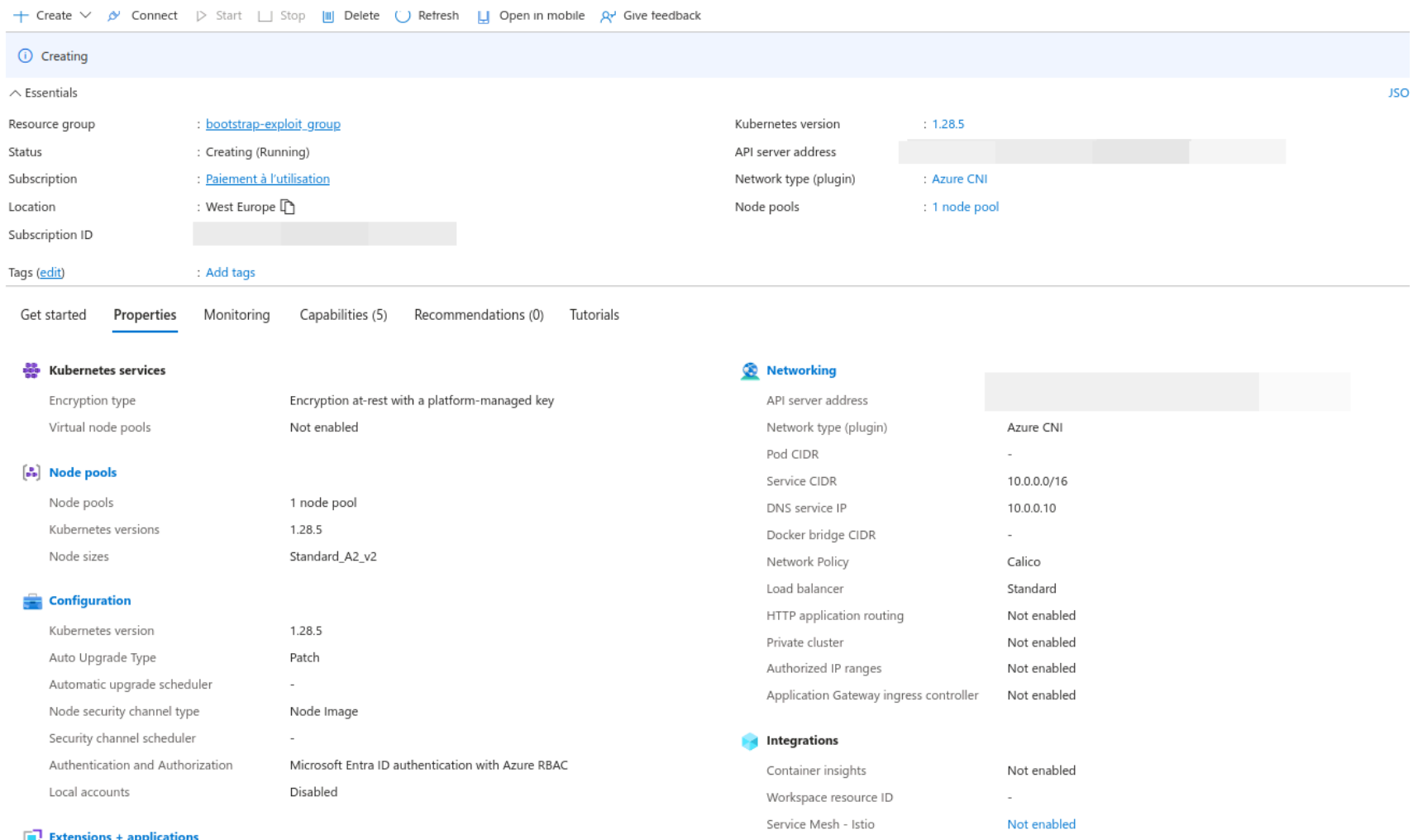
Kubelets compromise


## Sign certificates for any kubelets

- Access any secrets / configmaps mounted in pods
  - Often enough in real life cluster
- Access any service accounts used in pods
  - Create new JWT → Privilege escalation

# Demo

## Privilege escalation out-of-the-box



# Demo

Presentation of the privilege escalation

SYNACKTIV

### Enumeration of out-of-the-box resources

1. Enumerating pods
2. Inspecting the `tigera-operator` service account

### Exploiting the bootstrap token

3. Using the non-existing node certificate to find the `tigera-operator` pod
4. Signing a certificate for this node
5. Retrieving a JWT for the `tigera-operator` service account

### Privilege escalation

6. Creating a new *ClusterRoleBinding* to gain cluster admin privileges

# Demo 2

# Response and Mitigation

# Timeline

SYNACKTIV

| Date | Event |
| --- | --- |
| 2023-06-22 | First reporting of the vulnerability to Microsoft |
| 2023-08-09 | Case close due to miscommunication about the Host Network namespace requirement |
| 2023-08-09 | Opening a new case regarding the issue |
| 2023-08-27 | Follow-up email to Microsoft |
| 2023-10-07 | Follow-up to Microsoft |
| 2023-11-13 | Microsoft's response and case closing. |

# Microsoft response

SYNACKTIV

```
Thank you for your submission. We determined your finding is valid but does
not meet our bar for immediate servicing because even though the issue can
bypass a security boundary, it only compromises it at a cluster level.
However, we've marked your finding for future review as an opportunity to
improve our products. I do not have a timeline for this review. As no further
action is required at this time, I am closing this case.
```

They are working on a fix → https://github.com/Azure/aks-secure-tls-bootstrap

# Mitigation proposal

# Mitigation proposal

SYNACKTIV

- **No current mitigation**:
  - Do not use pods sharing the host's network namespace.
- **Bootstrap token**:
  - Long-lived token in AKS.
  - Manually deleting the bootstrap token is ineffective.
  - Token will be automatically recreated: bound to the nodepool.

  → Recreate the node pool
- **Monitor CSR**

# Conclusion

# Conclusion

SYNACKTIV

- AKS is not less secure than other (GKE, EKS, Digital Ocean)
  - Learn the cloud magic to defend yourself
- We will add the attack to kubletmein
  - https://github.com/4ARMED/kubeletmein - Marc Wickenden
- https://www.synacktiv.com/en/publications/so-i-became-a-node-exploiting-bootstrap-tokens-in-azure-kubernetes-service

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com