KubeCon | CloudNativeCon

North America 2024

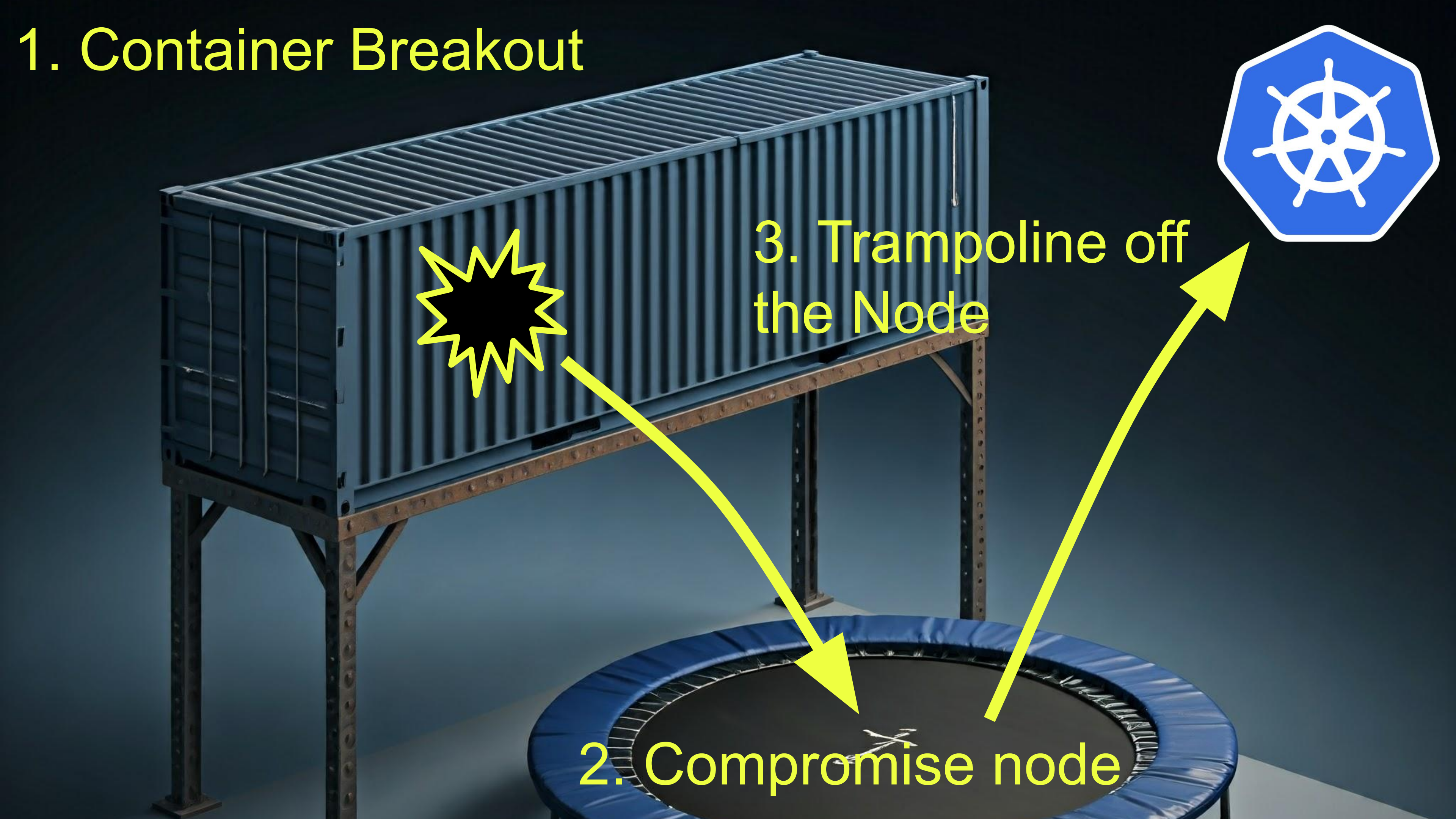
KubeCon | CloudNativeCon

North America 2024

Squashing Trampoline Pods

David Eads

Joe Betz

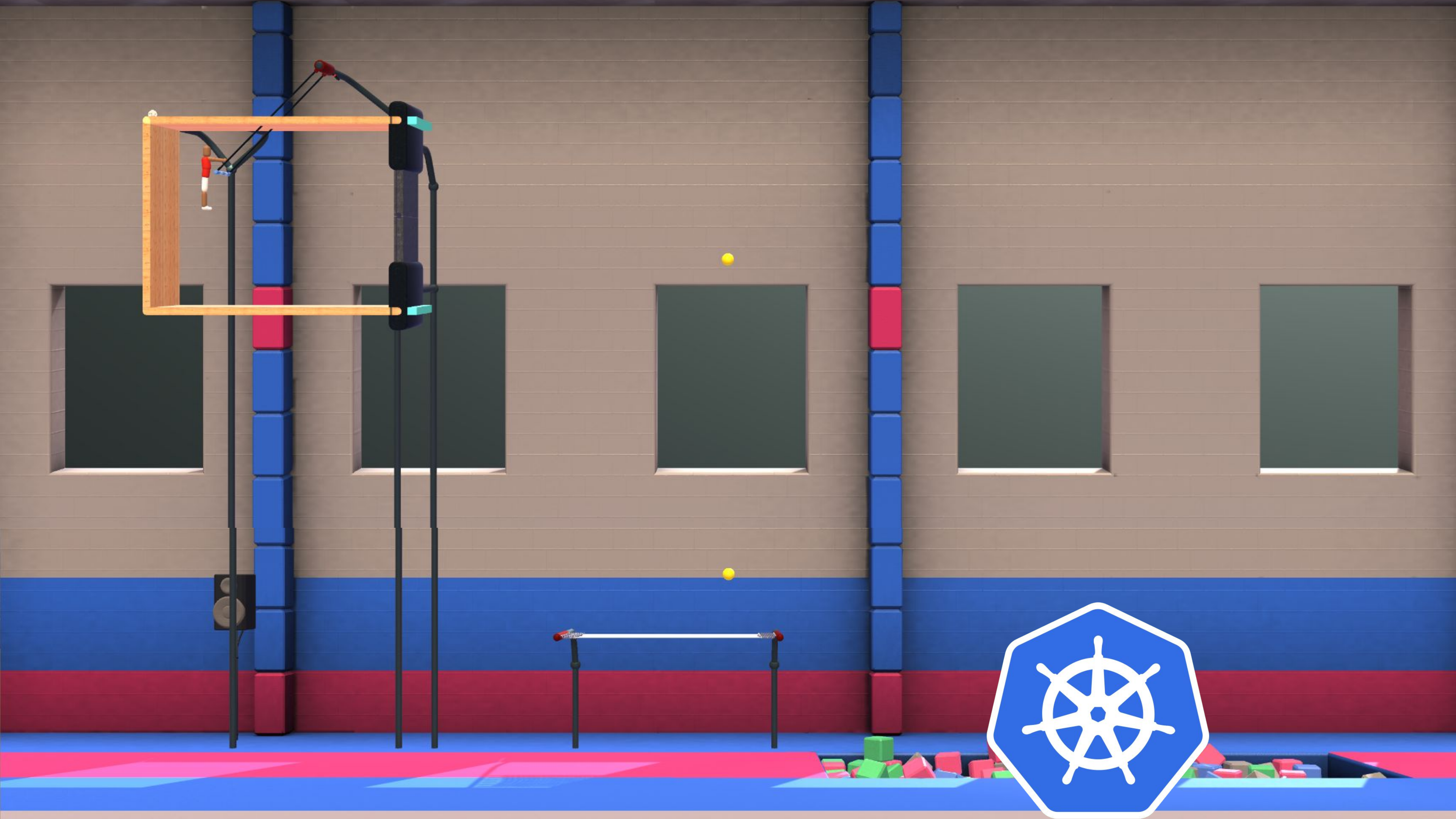


1. Container Breakout

2. Compromise node

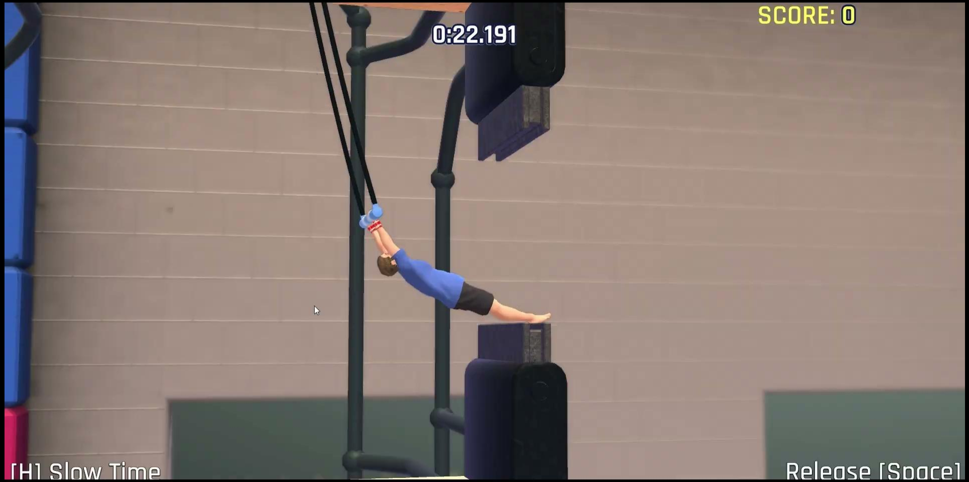
3. Trampoline off the Node

PRO
GYMNAST
SIMULATOR



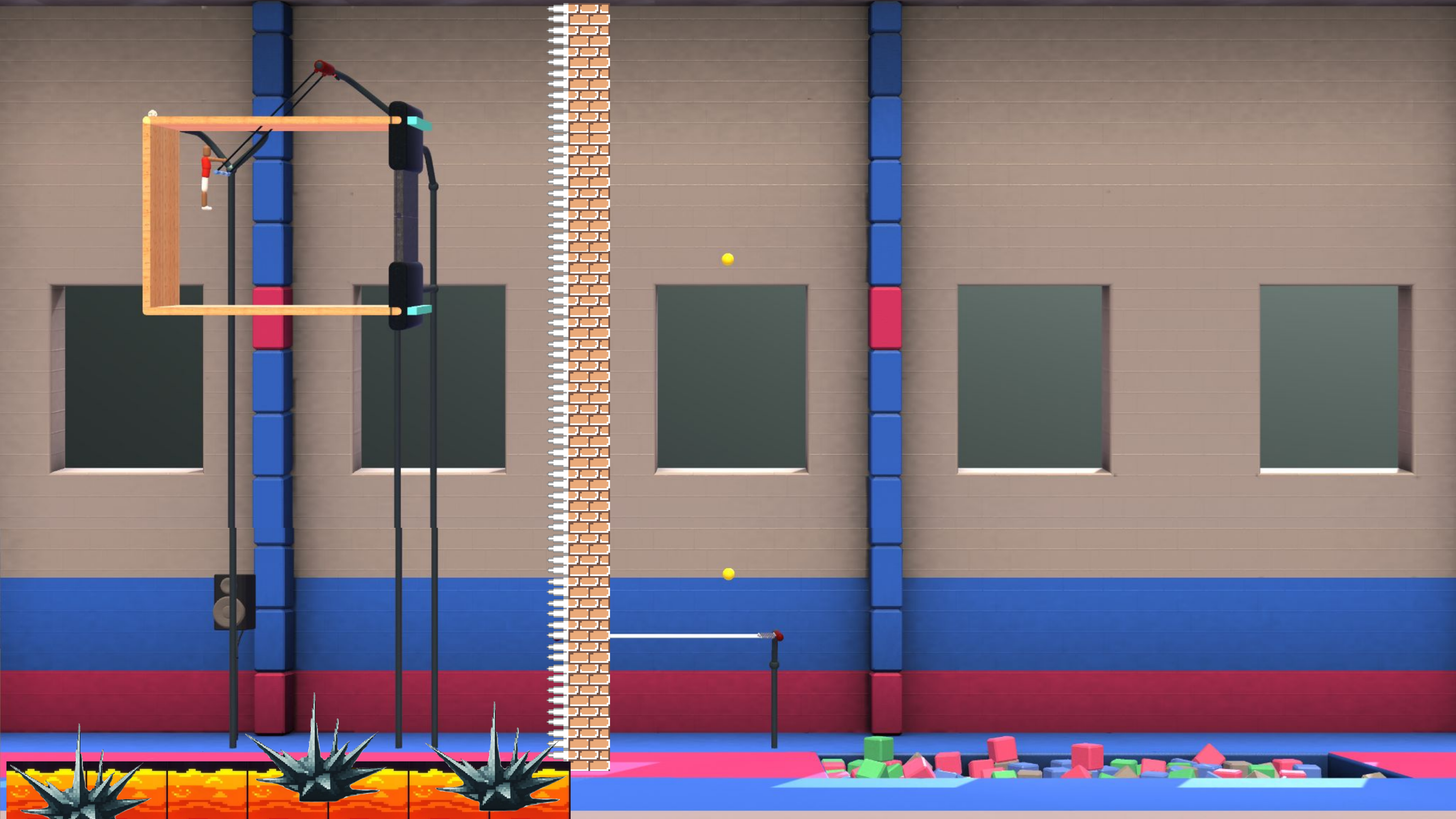
PRO GYMNAST SIMULATOR

PRO GYMNAST SIMULATOR



PRO GYMNAST SIMULATOR

"DaemonSets are GOATed."
 - Simone Biles



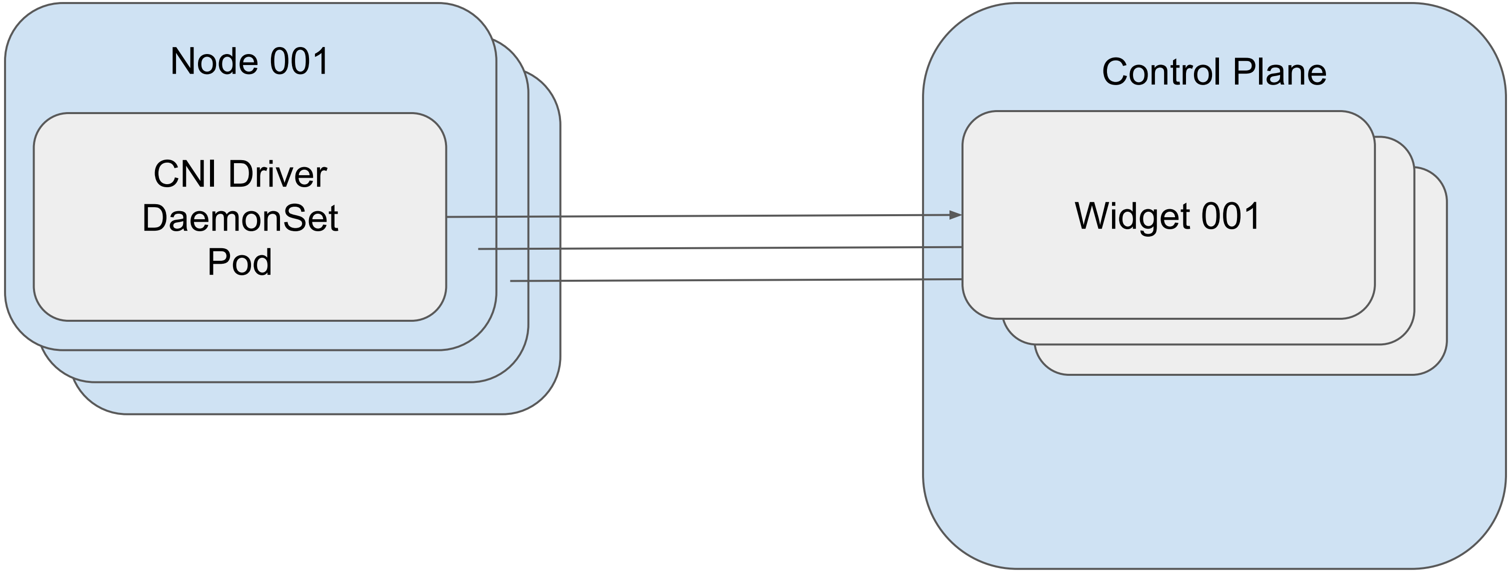
How it came together?

1. CRD Field Selectors - KEP-4358
2. Validation Admission Policy - KEP-3488
3. ServiceAccount Node Claims - KEP-4193
4. Field and Label Selector Authorization - KEP-4601
5. Structured Authorization Config - KEP-3331

Preventing Trampoline Writes

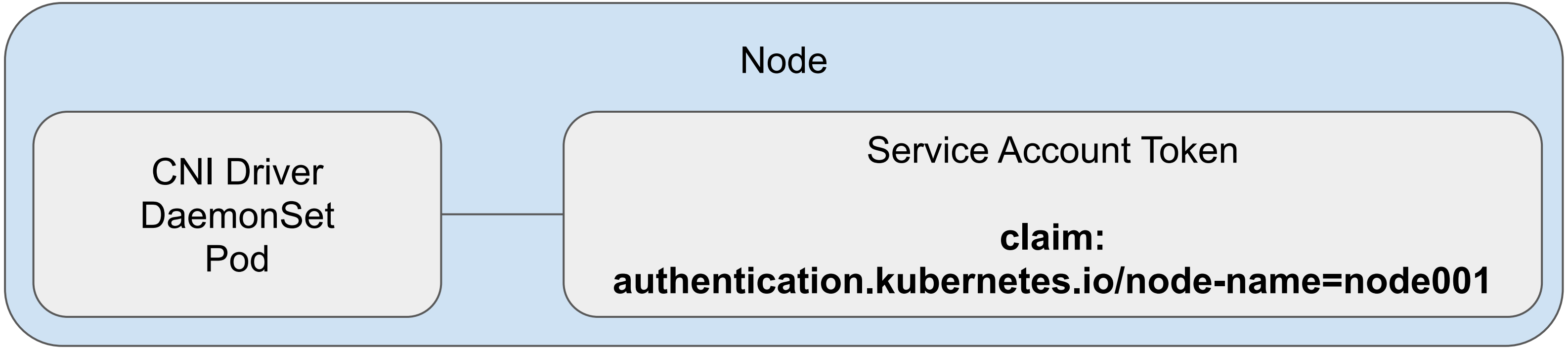
Preventing Trampoline Writes

Example: CNI Driver



Preventing Trampoline Writes

Service account node claims



Node

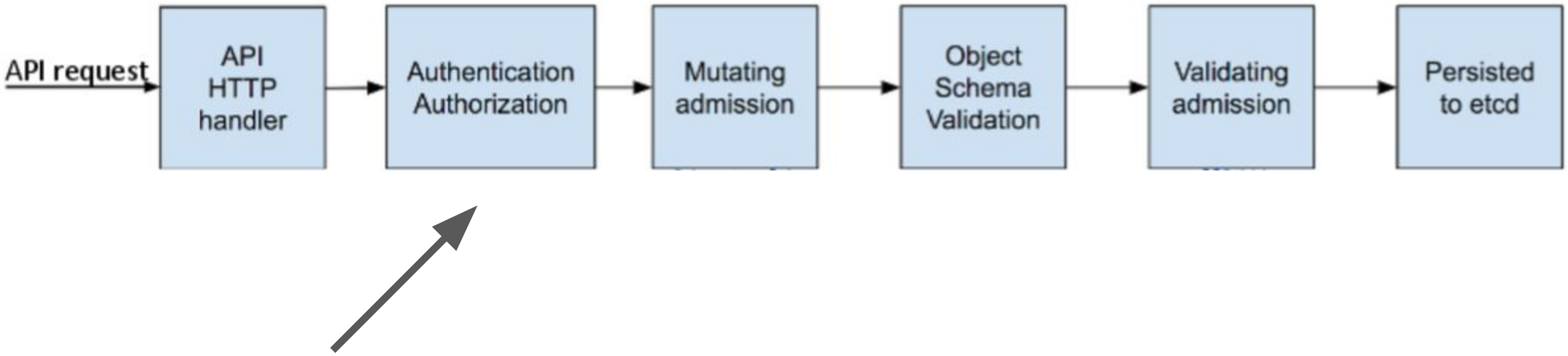
CNI Driver DaemonSet Pod

Service Account Token

claim: authentication.kubernetes.io/node-name=node001

Since Kubernetes 1.30 [beta]

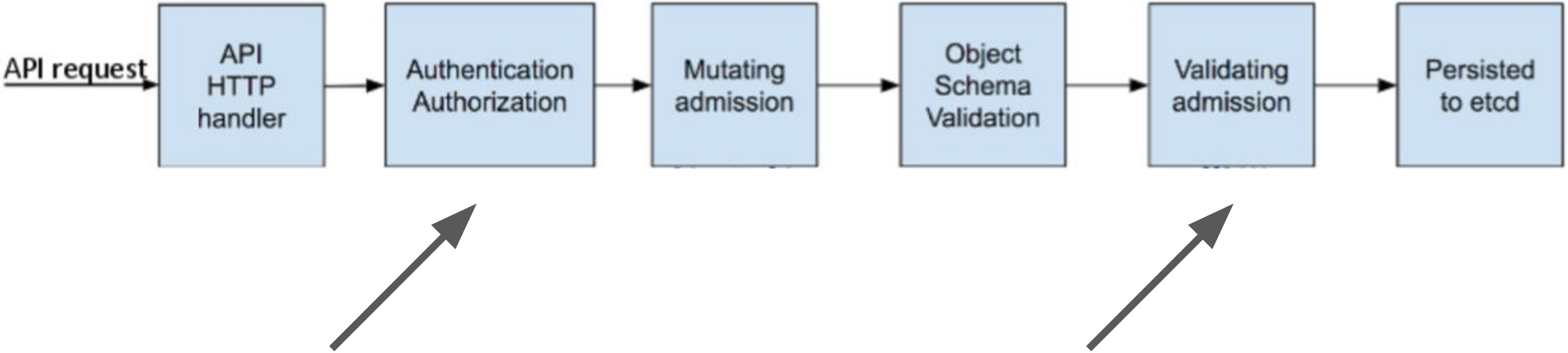
Preventing Trampoline Writes



Primary authorization

Allow pods to writes to all **Widgets**

Preventing Trampoline Writes



Primary authorization
Allow pods to writes to all **Widgets**

Secondary authorization
Only allow pod to write to **Widgets** with same name as node

Preventing Trampoline Writes

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "only-allow-name-matching-node-widget"
spec:
  matchConstraints:
    resourceRules:
      - apiGroups:   ["cnidriver.example.com"]
        apiVersions: ["v1"]
        operations:  ["CREATE", "UPDATE", "DELETE"]
        resources:   ["widgets"]
  ...
```

Available since Kubernetes v1.31 [GA]

Preventing Trampoline Writes

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
...
spec:
  ...
  matchConditions:
  - name: isRestrictedUser
    expression: >-
      request.userInfo.username == "system:serviceaccount:default:cni-sa"
```

Preventing Trampoline Writes

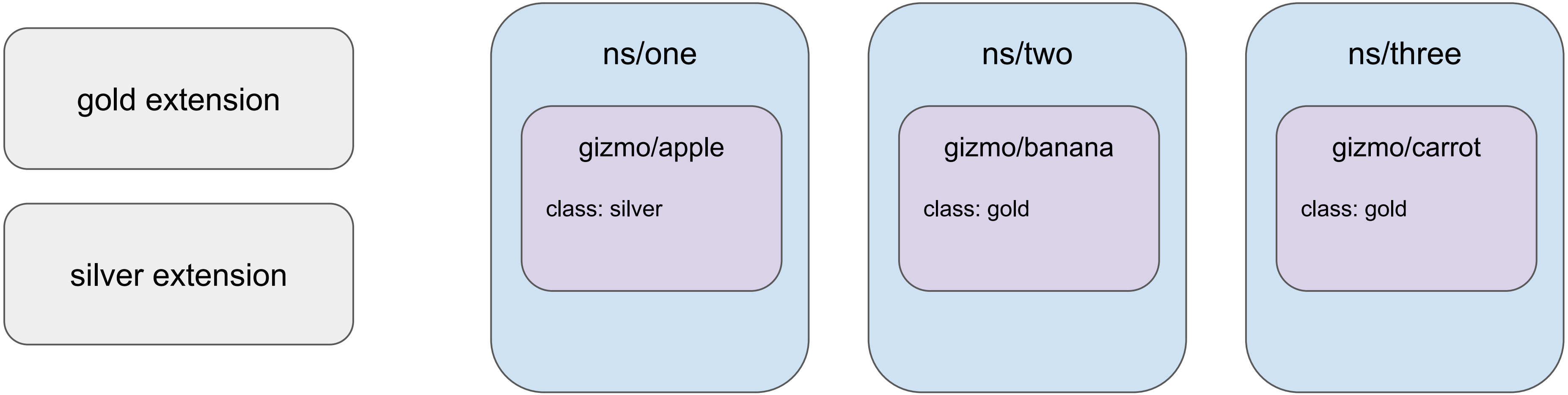
```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
...
spec:
  ...
  variables:
  - name: userNodeName
    expression: >-
      request.userInfo.extra[?'authentication.kubernetes.io/node-name'][0].orValue('')
  - name: objectNodeName
    expression: >-
      (request.operation == 'DELETE' ? oldObject : object).?metadata.name.orValue('')
```

Preventing Trampoline Writes

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
...
spec:
  ...
  validations:
  - expression: variables.userNodeName != ""
    message: >-
      no node association found for user, this user must run in a pod on a node and
ServiceAccountTokenPodNodeInfo must be enabled

  - expression: variables.userNodeName == variables.objectNodeName
    messageExpression: >-
      "this user running on node '"+variables.userNodeName+"' may not modify Widget '" +
variables.objectNodeName +
      "' because the name does not match the node name"
```

Preventing Trampoline Reads

Preventing trampoline reads



gold extension

silver extension

ns/one

gizmo/apple

class: silver

ns/two

gizmo/banana

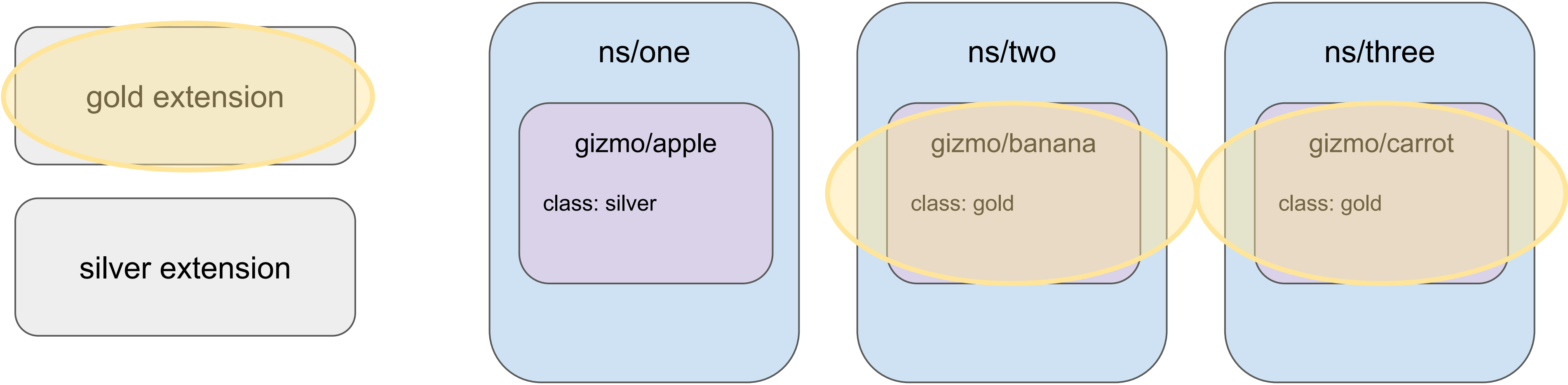
class: gold

ns/three

gizmo/carrot

class: gold

Preventing trampoline reads



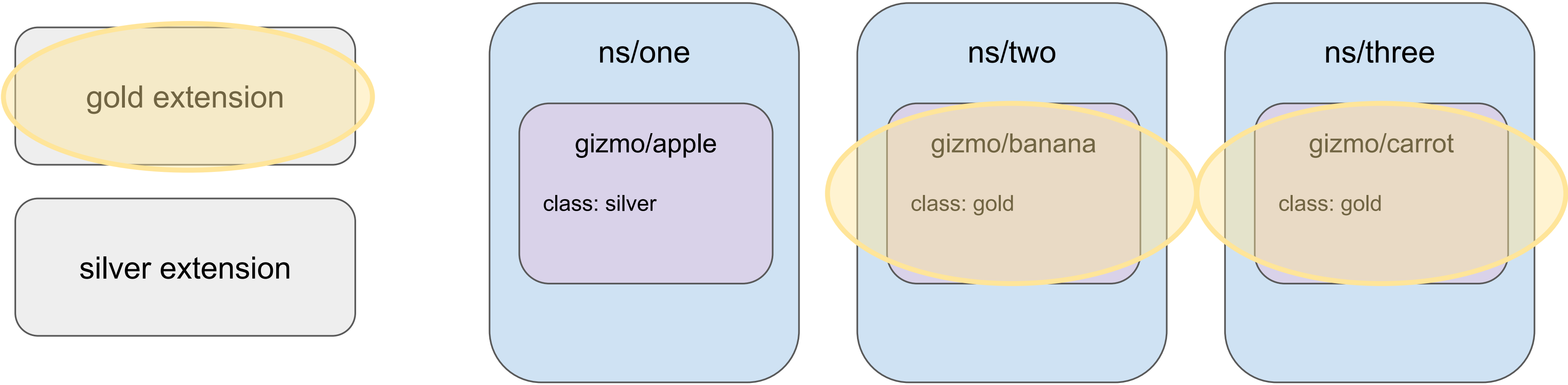
Defining Selectable Fields in CRDs

```
kind: CustomResourceDefinition
spec:
  versions:
  - name: v1
    selectableFields:
    - jsonPath: .class
```

Since Kubernetes 1.32 [stable]

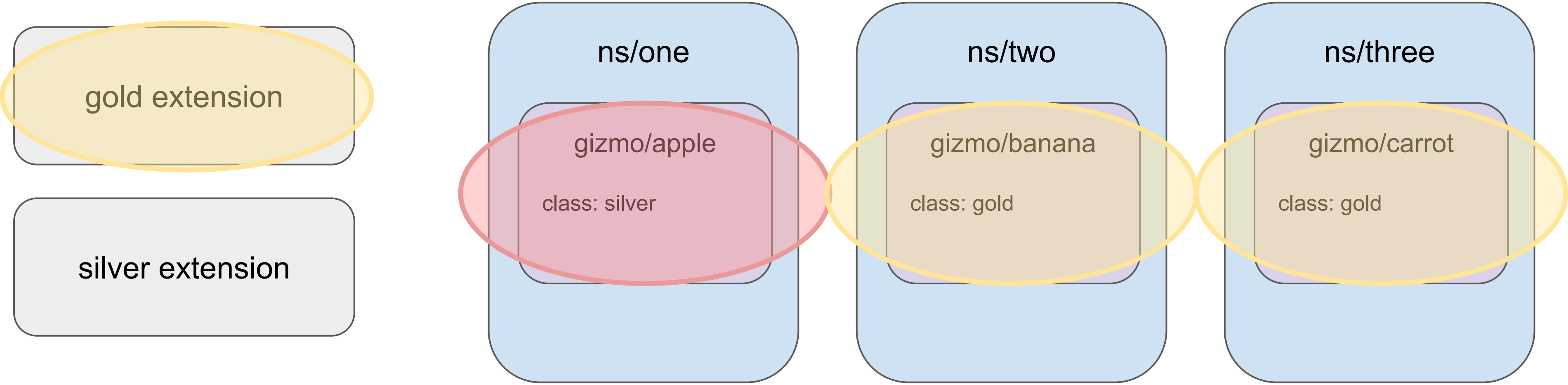
List CRD with Field Selectors

```
kubectl get --all-namespaces gizmos --field-selector=.class=gold
```



Our Listing Authorization problem

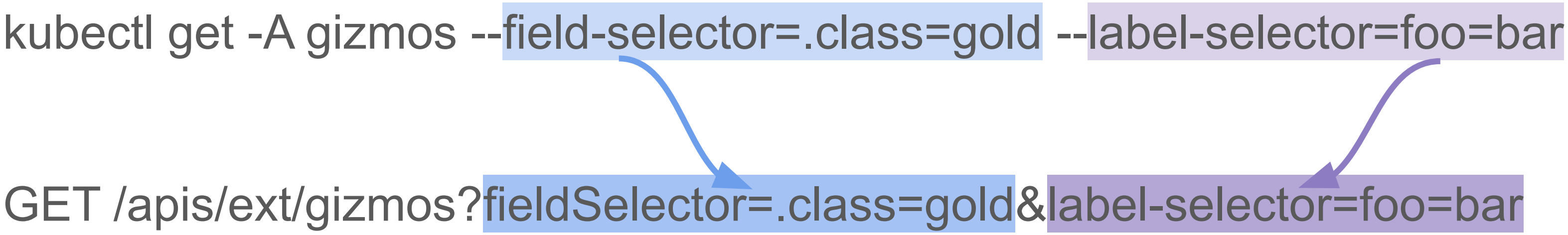
```
✗ kubectl get --all-namespaces gizmos
✗ kubectl get --all-namespaces gizmos --field-selector=.class=silver
```



The Journey of a Selector

```
kubectl get -A gizmos --field-selector=.class=gold --label-selector=foo=bar
```

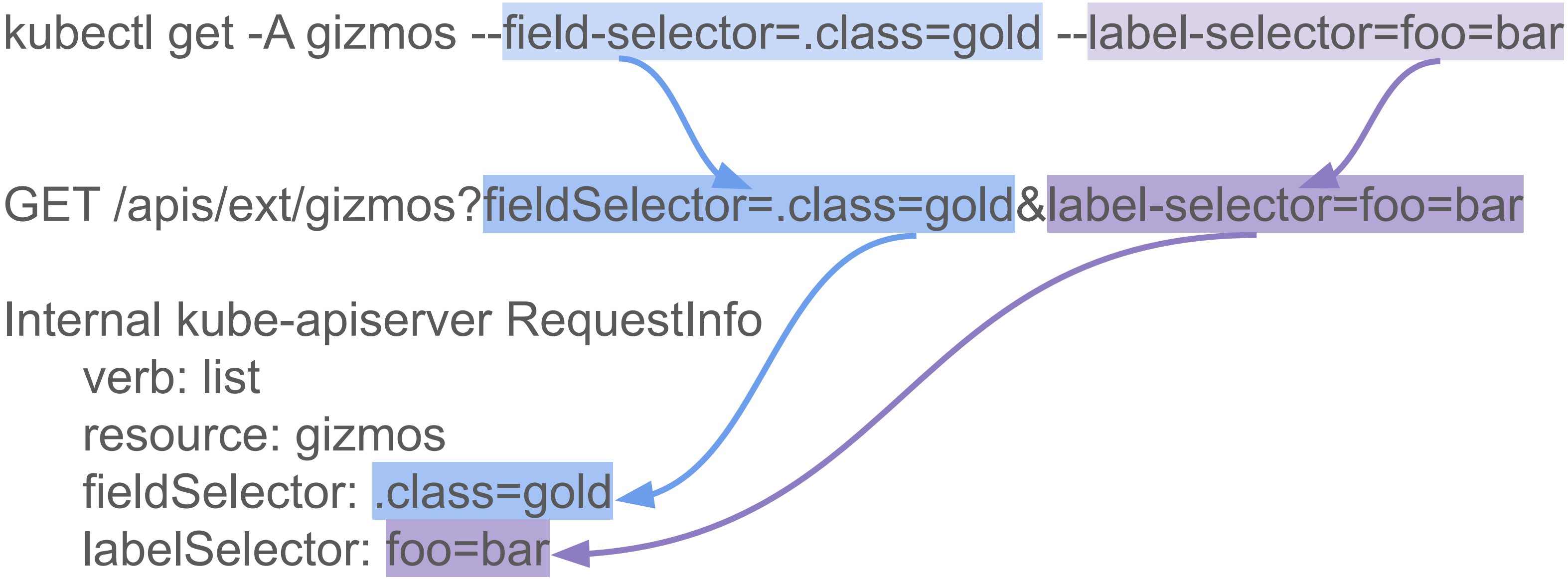
The Journey of a Selector



```
kubectl get -A gizmos --field-selector=.class=gold --label-selector=foo=bar
```

```
GET /apis/ext/gizmos?fieldSelector=.class=gold&label-selector=foo=bar
```

The Journey of a Selector



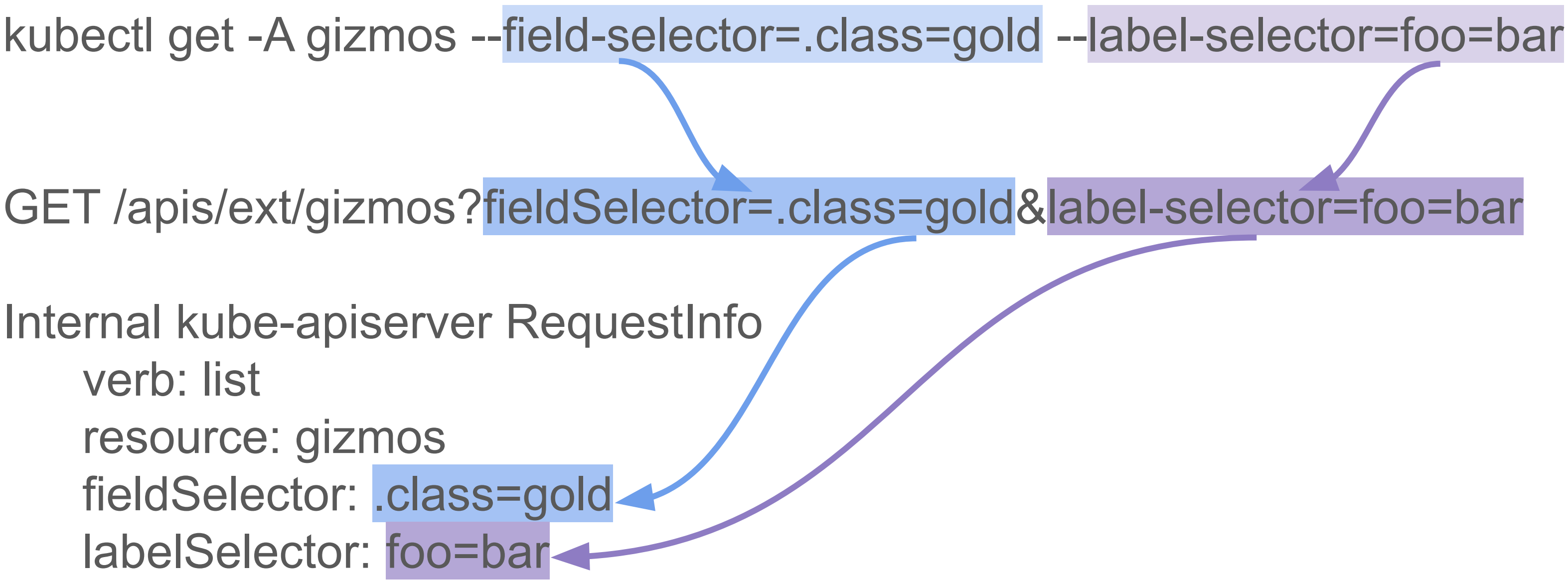
```
kubectl get -A gizmos --field-selector=.class=gold --label-selector=foo=bar
```

```
GET /apis/ext/gizmos?fieldSelector=.class=gold&label-selector=foo=bar
```

Internal kube-apiserver RequestInfo

```
    verb: list
    resource: gizmos
    fieldSelector: .class=gold
    labelSelector: foo=bar
```

The Journey of a Selector



```
kubectl get -A gizmos --field-selector=.class=gold --label-selector=foo=bar
```

```
GET /apis/ext/gizmos?fieldSelector=.class=gold&label-selector=foo=bar
```

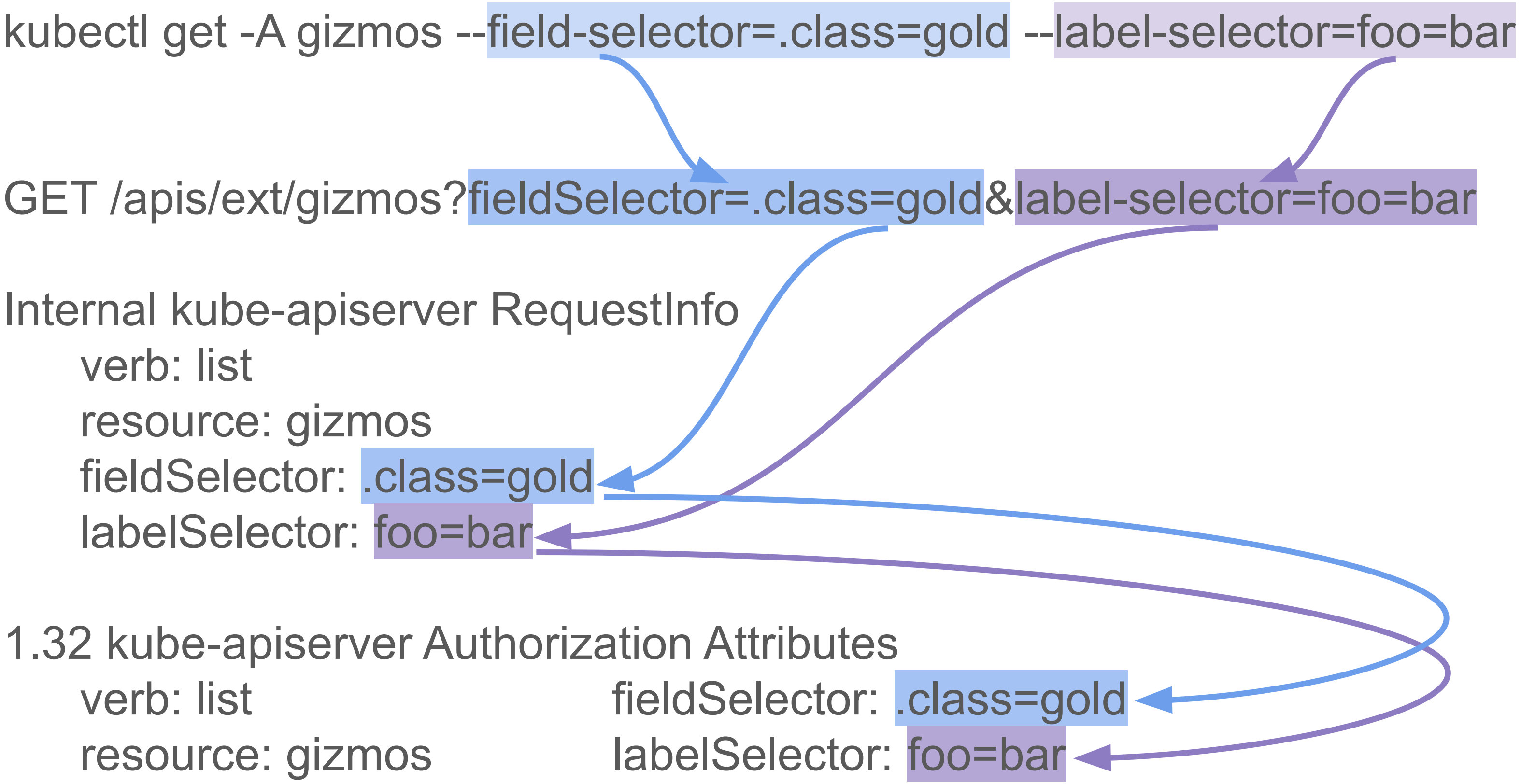
Internal kube-apiserver RequestInfo

```
verb: list
resource: gizmos
fieldSelector: .class=gold
labelSelector: foo=bar
```

1.31 kube-apiserver Authorization Attributes

```
verb: list
resource: gizmos
```

The Journey of a Selector



kubectl get -A gizmos --field-selector=.class=gold --label-selector=foo=bar

GET /apis/ext/gizmos?fieldSelector=.class=gold&label-selector=foo=bar

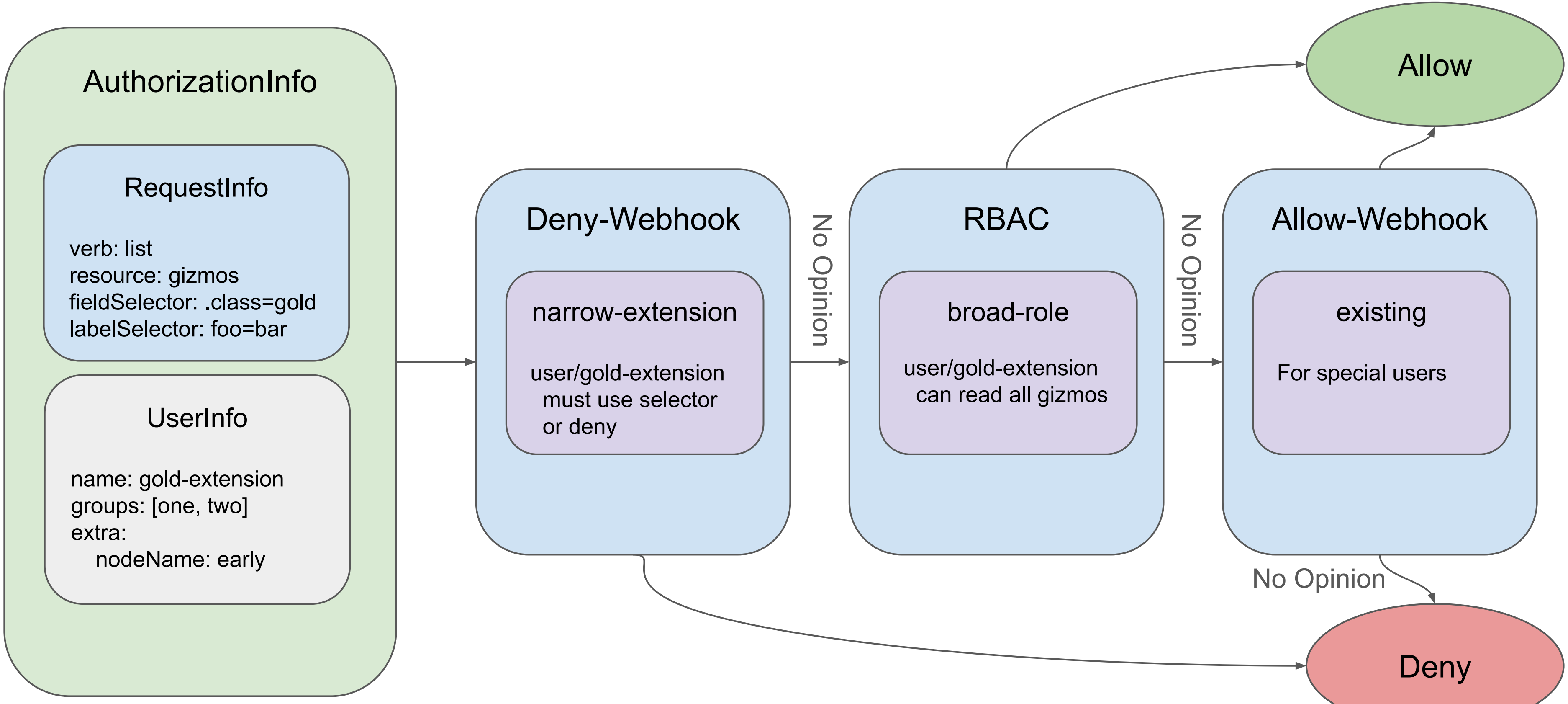
Internal kube-apiserver RequestInfo

verb: list
resource: gizmos
fieldSelector: .class=gold
labelSelector: foo=bar

1.32 kube-apiserver Authorization Attributes

verb: list
resource: gizmos
fieldSelector: .class=gold
labelSelector: foo=bar

Can I co-exist with RBAC?



AuthorizationInfo

RequestInfo

verb: list
resource: gizmos
fieldSelector: .class=gold
labelSelector: foo=bar

UserInfo

name: gold-extension
groups: [one, two]
extra:
nodeName: early

Deny-Webhook

narrow-extension

user/gold-extension must use selector or deny

RBAC

broad-role

user/gold-extension can read all gizmos

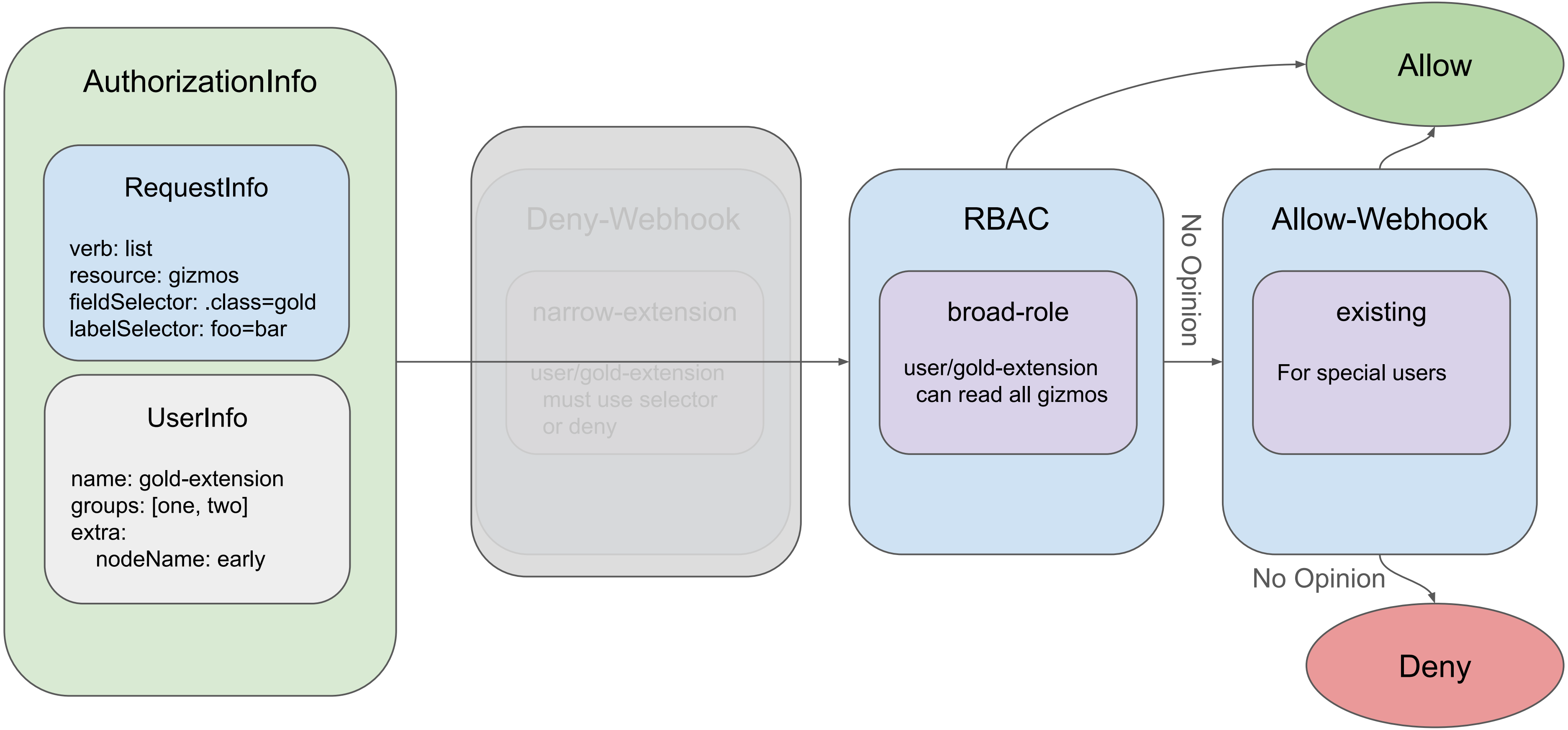
Allow-Webhook

existing

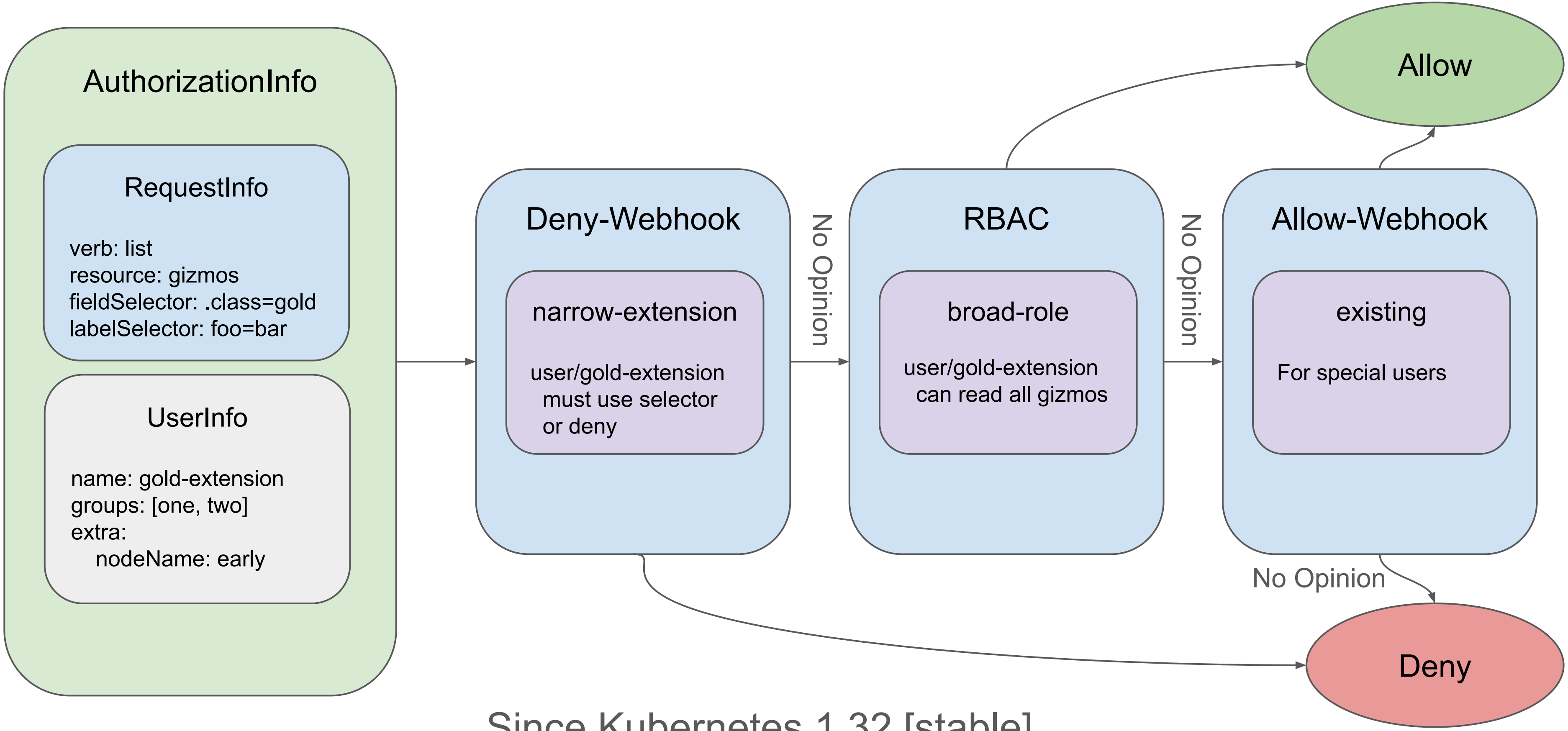
For special users

Since Kubernetes 1.32 [beta]

Can I co-exist with RBAC?



Can I co-exist with RBAC?



AuthorizationInfo

RequestInfo

verb: list
resource: gizmos
fieldSelector: .class=gold
labelSelector: foo=bar

UserInfo

name: gold-extension
groups: [one, two]
extra:
    nodeName: early

Deny-Webhook

narrow-extension

user/gold-extension must use selector or deny

RBAC

broad-role

user/gold-extension can read all gizmos

Allow-Webhook

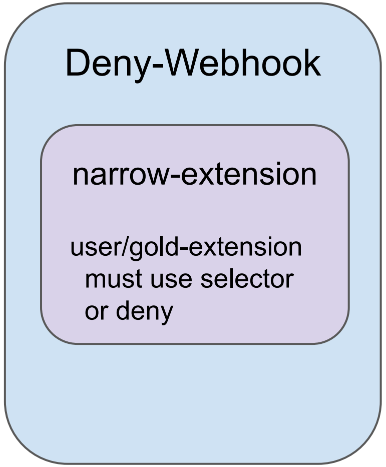
existing

For special users

Since Kubernetes 1.32 [stable]

Deny-Webhook

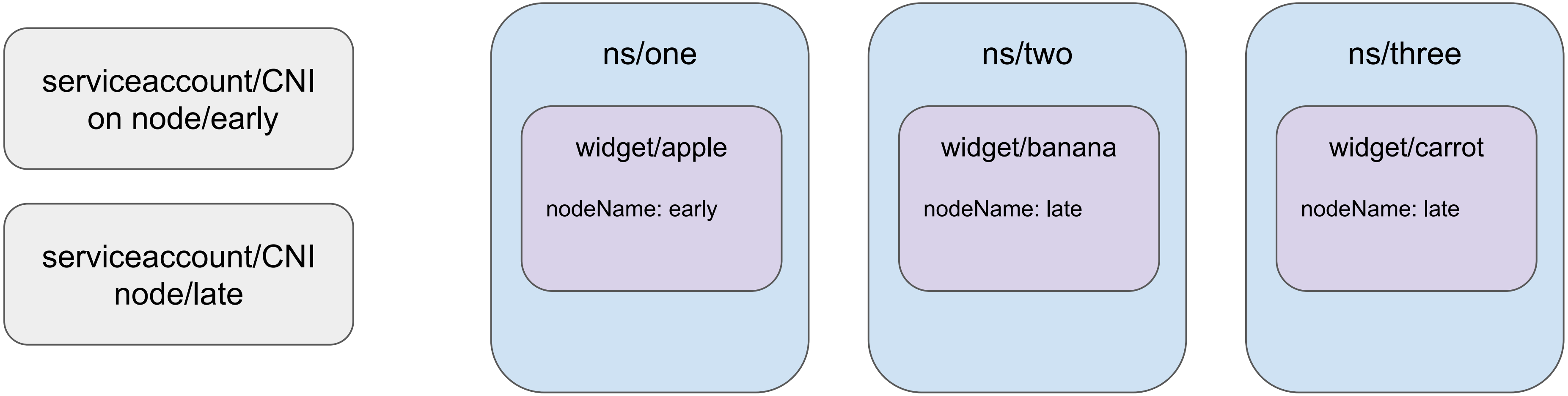
```
kind: AuthorizationConfiguration
authorizers:
  - type: Webhook
    name: webhook
    webhook:
      connectionInfo: …

      matchConditions:  # ALL expressions must evaluate to true
      # only send resource requests to the webhook
      - expression: has(request.resourceAttributes)
      # only intercept requests for gizmos
      - expression: request.resourceAttributes.resource == 'gizmos'
      # only restrict gold-extension
      - expression: "request.user == gold-extension"
```


Deny-Webhook

narrow-extension

user/gold-extension must use selector or deny

Preventing trampoline reads



serviceaccount/CNI on node/early

serviceaccount/CNI node/late

ns/one

widget/apple

nodeName: early

ns/two

widget/banana

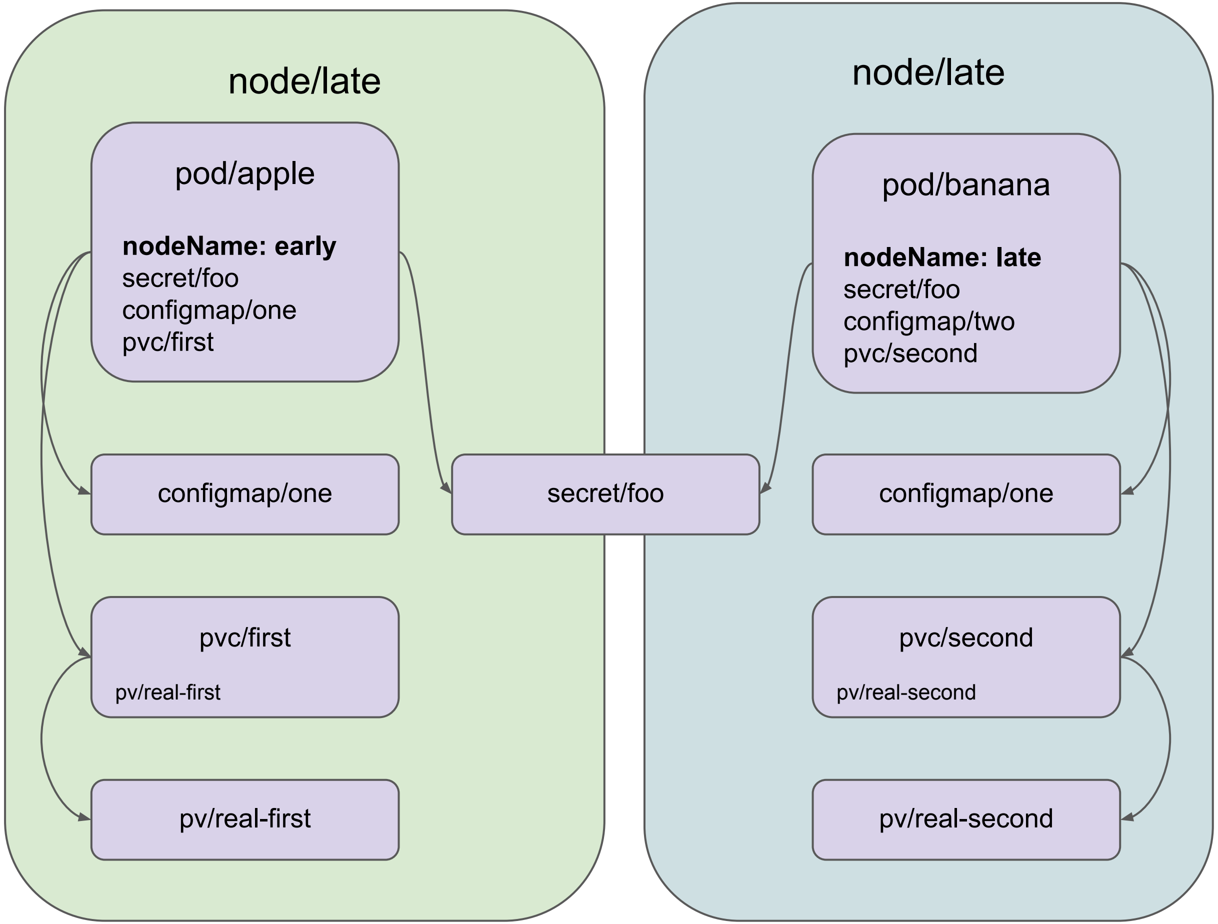
nodeName: late

ns/three

widget/carrot

nodeName: late

Wait, how safe is the kubelet?



node/late

pod/apple

nodeName: early
secret/foo
configmap/one
pvc/first

configmap/one

pvc/first

pv/real-first

pv/real-first

secret/foo

node/late

pod/banana

nodeName: late
secret/foo
configmap/two
pvc/second

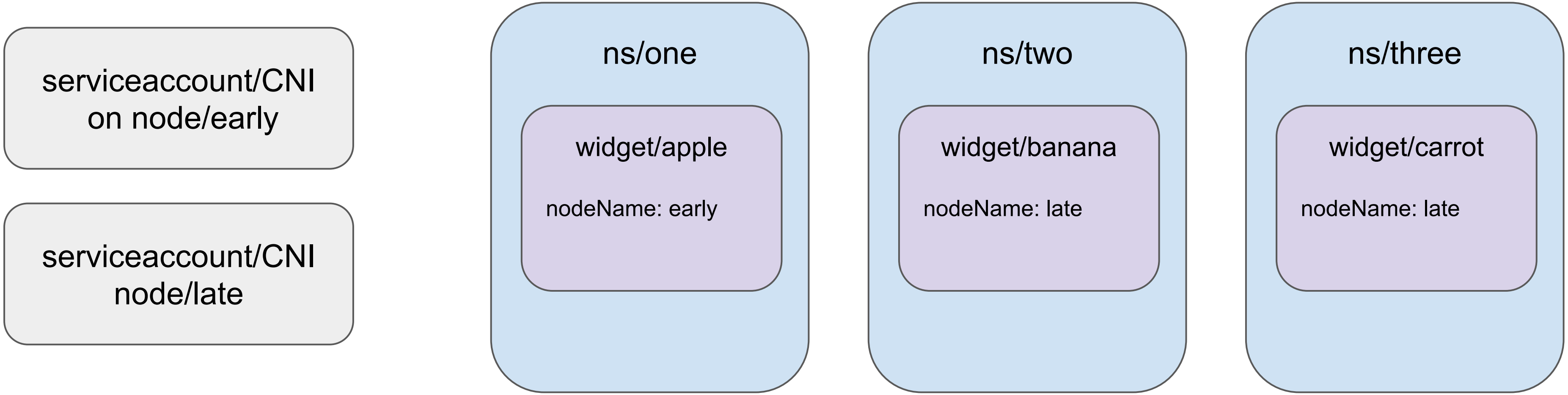
configmap/one

pvc/second

pv/real-second

pv/real-second

Preventing trampoline reads



serviceaccount/CNI on node/early

serviceaccount/CNI node/late

ns/one

widget/apple

nodeName: early

ns/two

widget/banana

nodeName: late

ns/three

widget/carrot

nodeName: late

Future

1. sig-auth's deep dive on Friday. RBAC++ with Jordan, Mo, and Rita
2. What if we made the apiserver honor field and label selectors on all verbs?
   a. This is an idea we've been considering with sig-auth to decide if it makes writing the implementation of an API easier enough to be valuable for the semantic changes.
3. How can we close the broad authorization with narrowing admission?
   a. Could we add a new concept of conditional authorization?
4. Detecting your own trampoline pods: RBAC Police