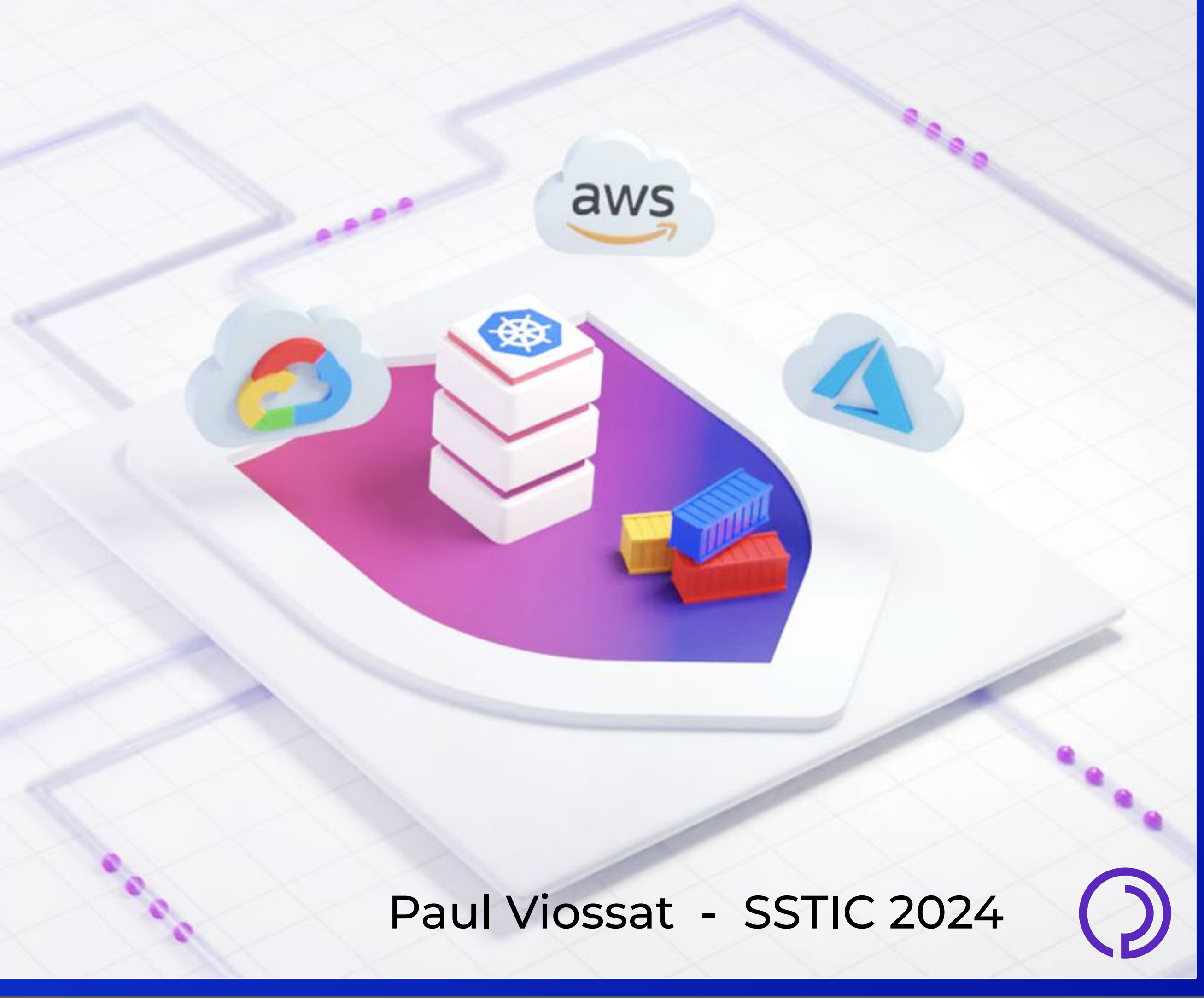
PADOK SECURITY

Getting ahead of the schedule:

manipulating the Kubernetes scheduler to perform lateral movement in a cluster



Paul Viossat - SSTIC 2024

Agenda

01

Introduction

02

Kubernetes scheduler

03

Attacking isolation with kubelet

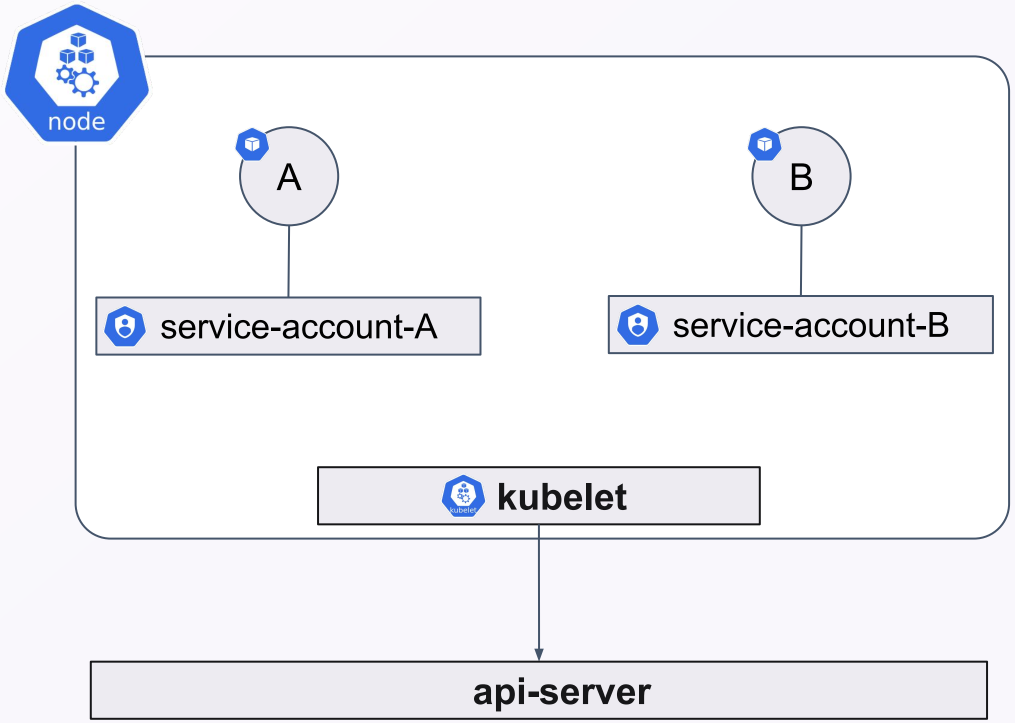
04

Attacking isolation with service account

05

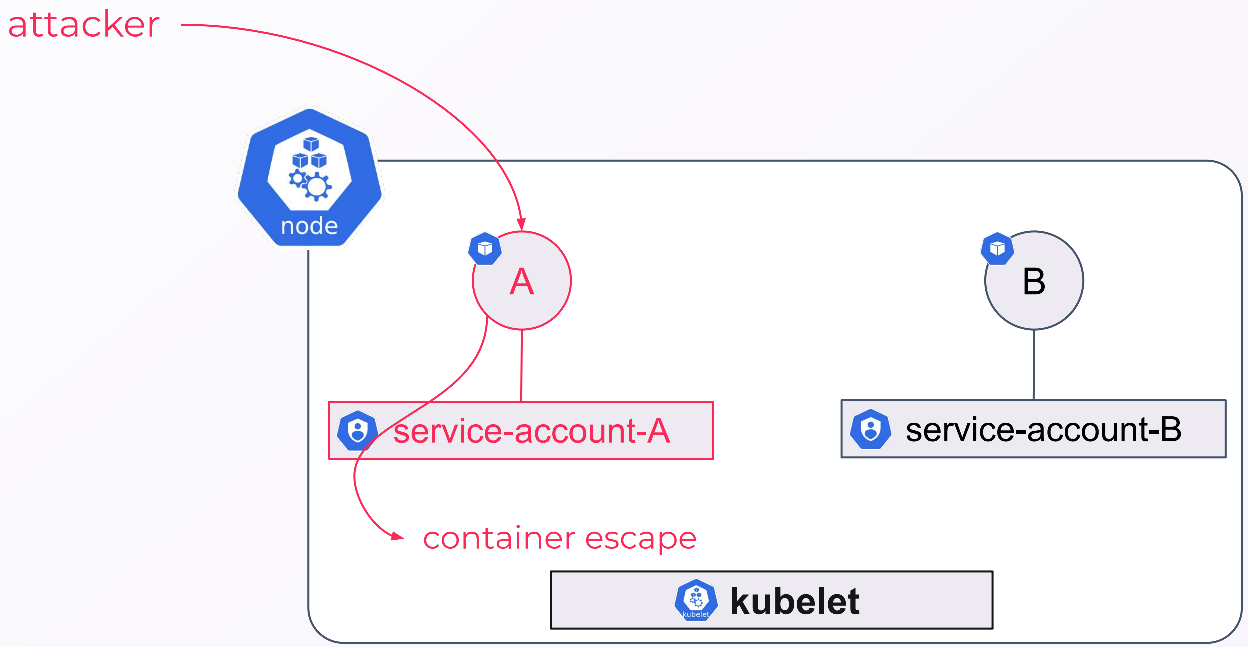
Conclusion

Considered threat



INTRODUCTION

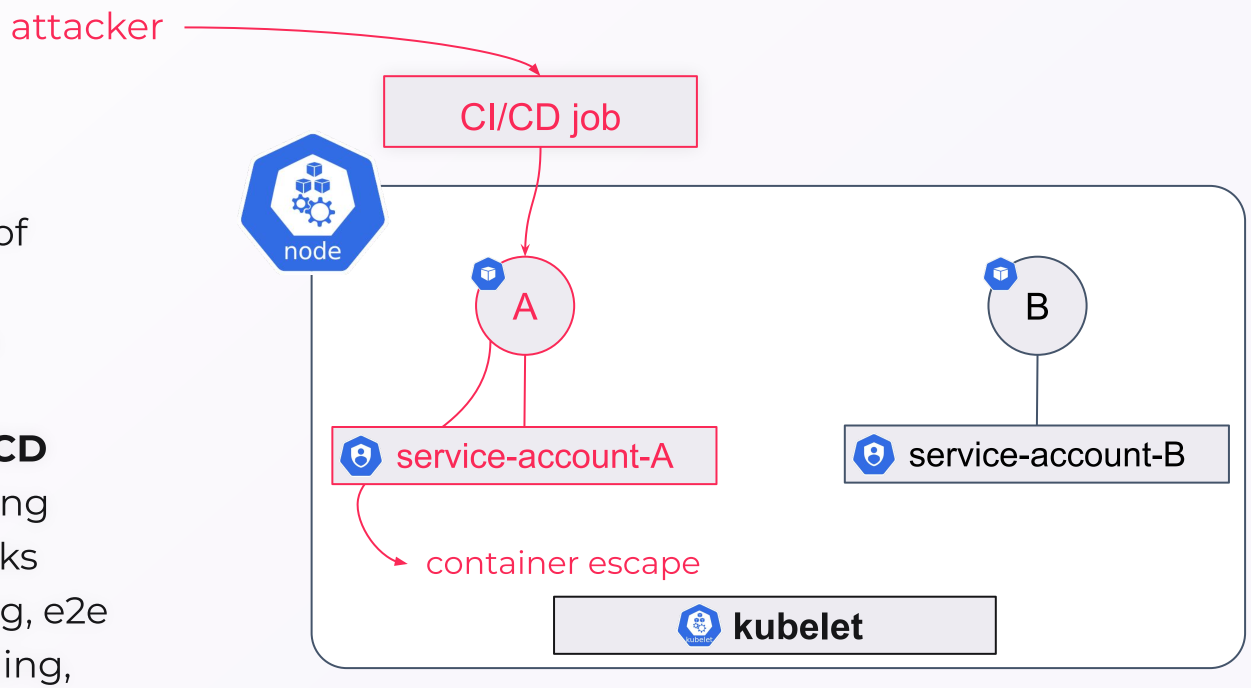
Considered threat



INTRODUCTION

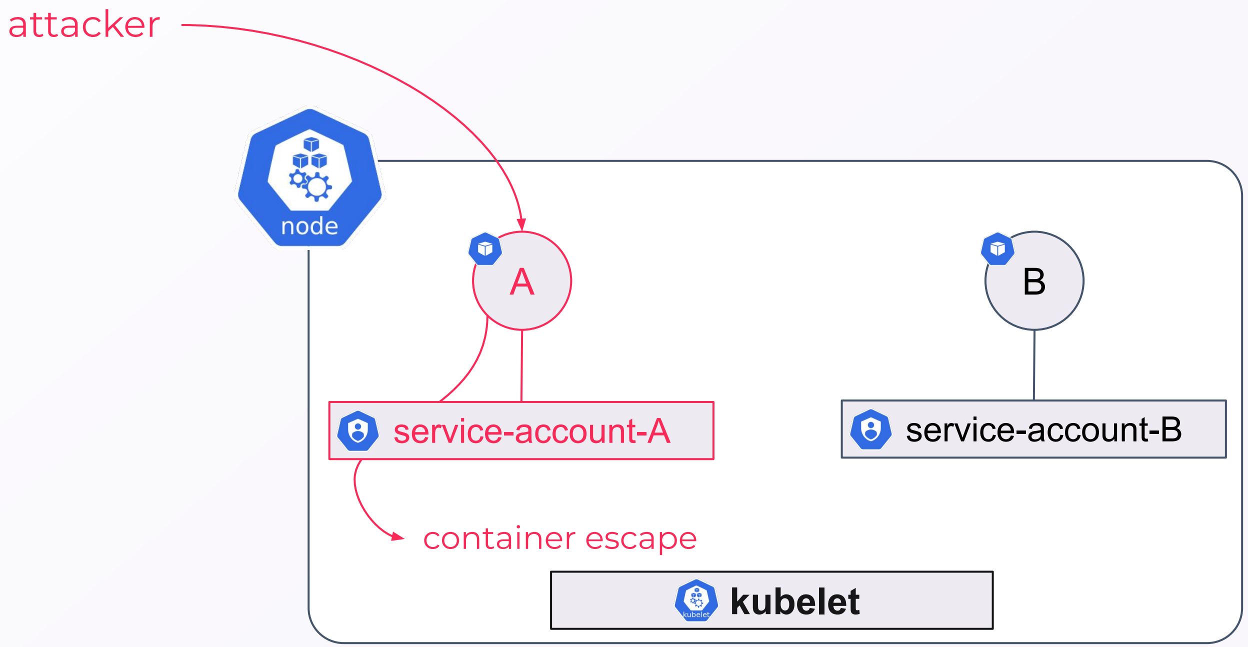
Considered threat

- Escape based on:
  - Configuration of pods
  - Kernel exploits
- Can be found in **CI/CD jobs** when performing DockerInDocker tasks (performance testing, e2e testing, image building, etc...



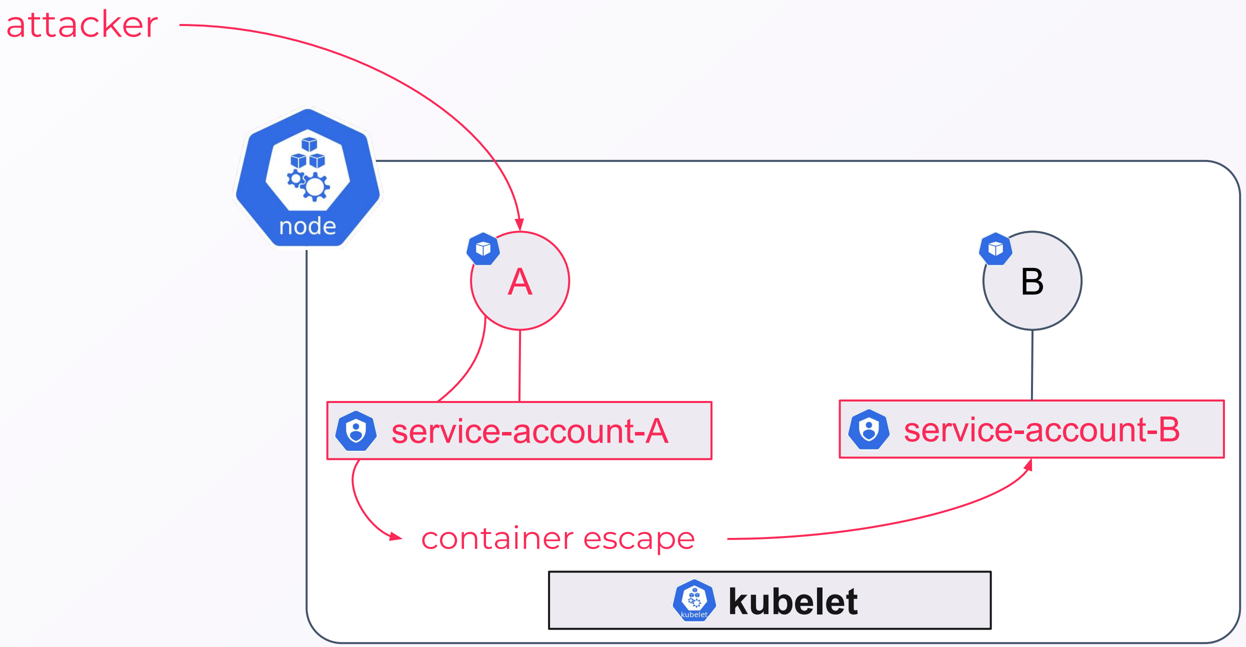
INTRODUCTION

Considered threat



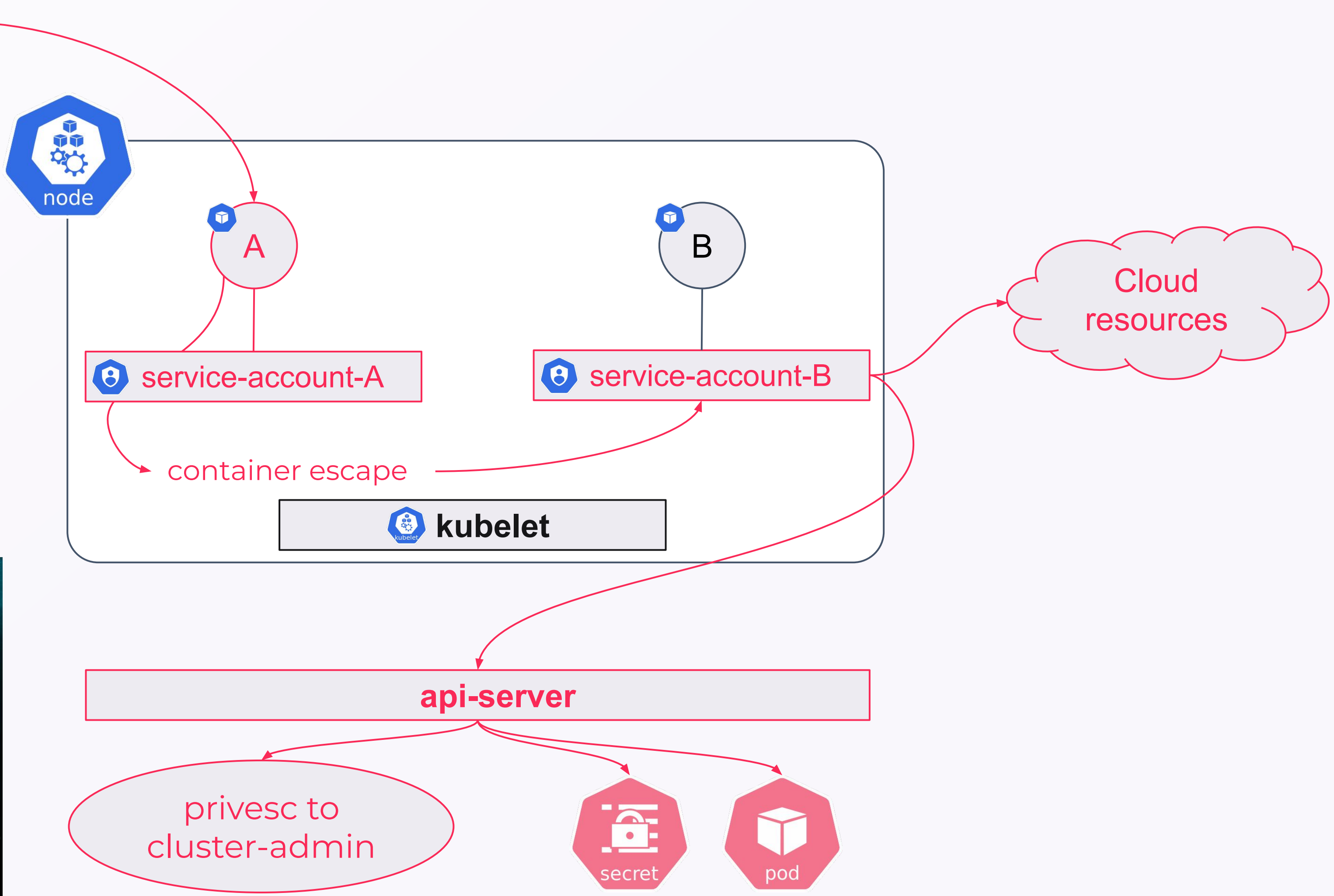
INTRODUCTION

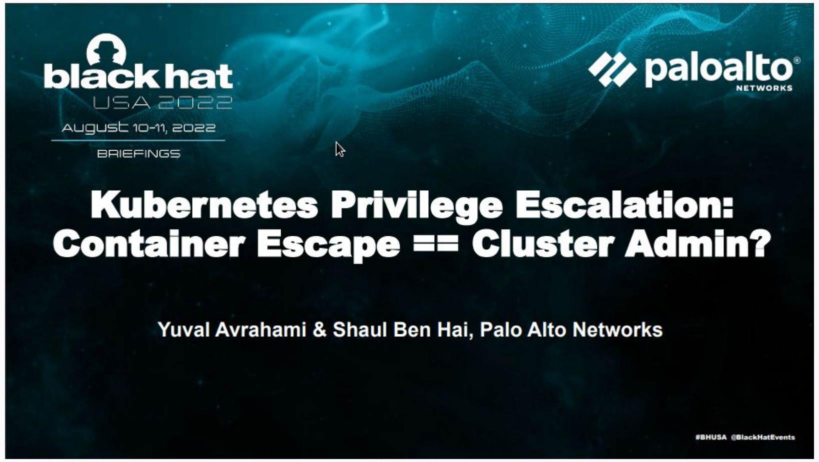
Considered threat



INTRODUCTION

Considered threat




black hat
USA 2022
August 10-11, 2022
BRIEFINGS

paloalto NETWORKS

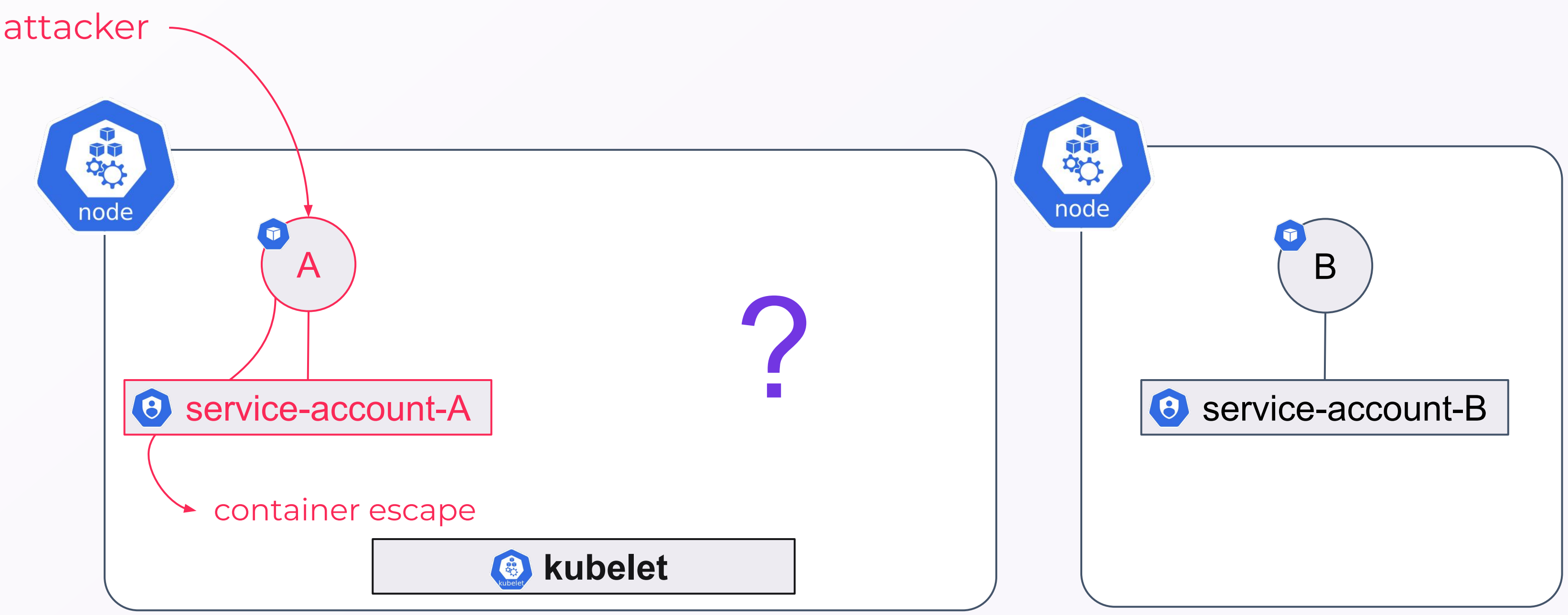
Kubernetes Privilege Escalation: Container Escape == Cluster Admin?

Yuval Avrahami & Shaul Ben Hai, Palo Alto Networks

#BHUSA @BlackHatEvents

INTRODUCTION

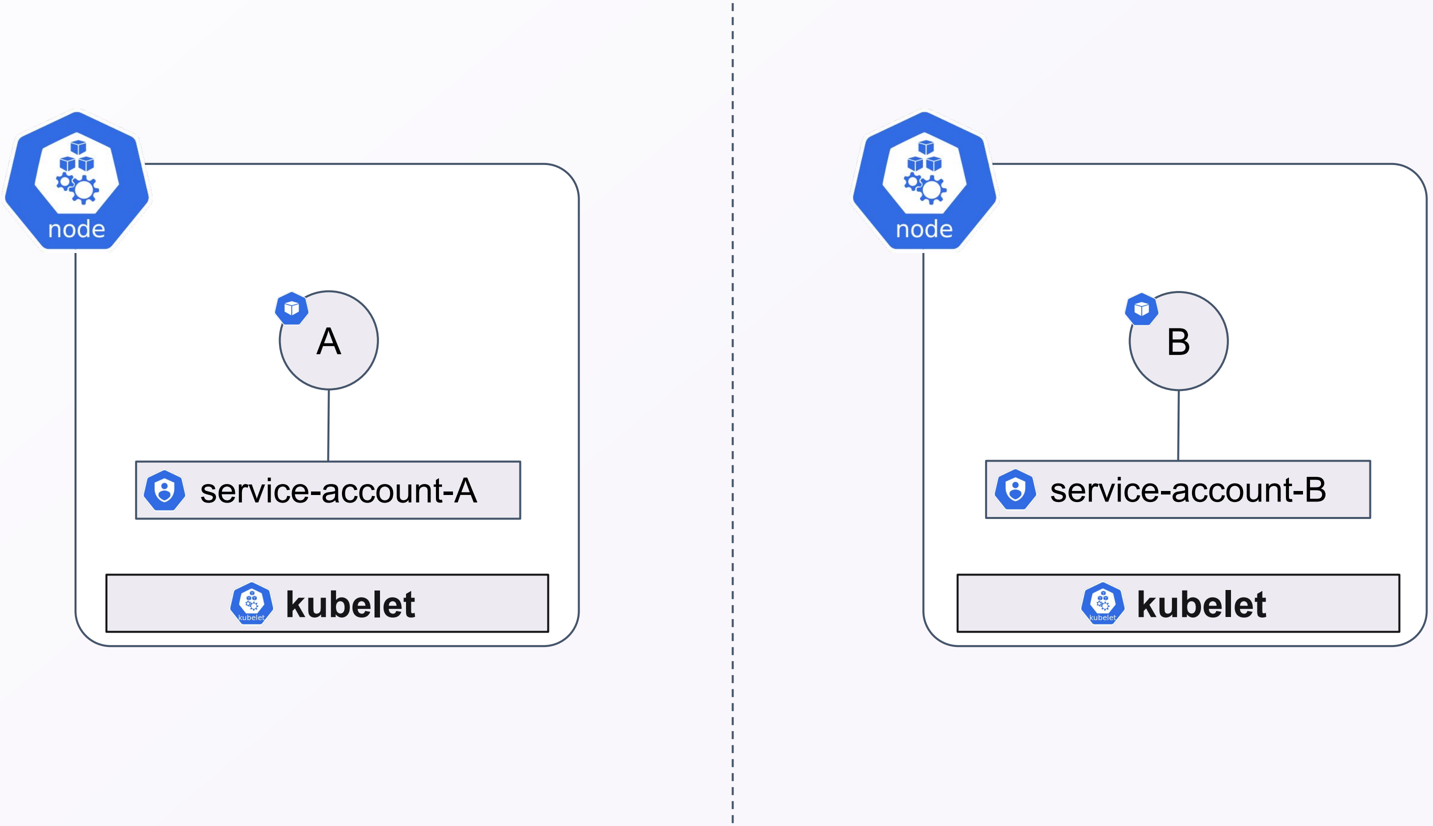
Considered threat



- As defender can I predict which service account can be compromised ?
- As attacker how can I can access other service account ?

INTRODUCTION

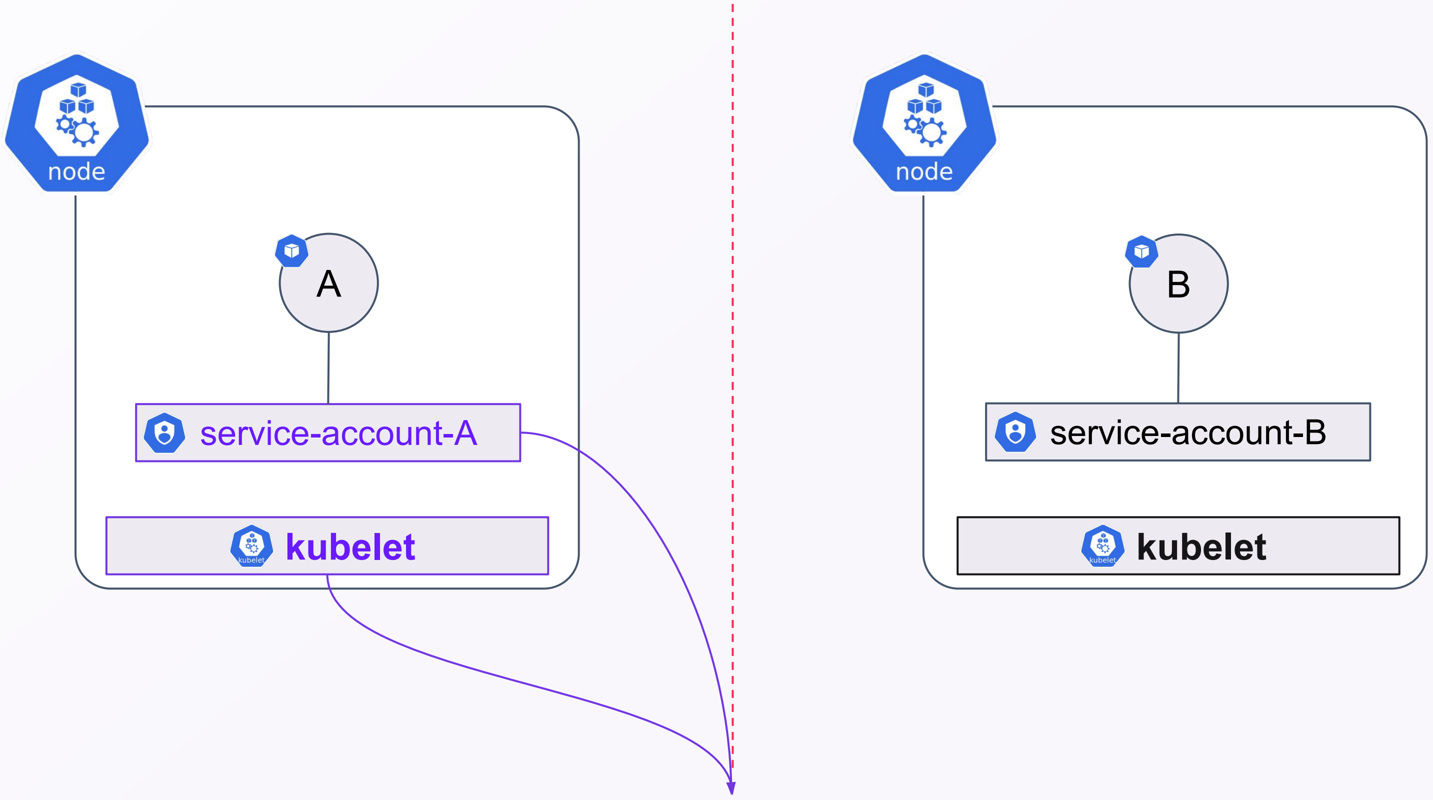
Considered threat



Workload node isolation

INTRODUCTION

Considered threat

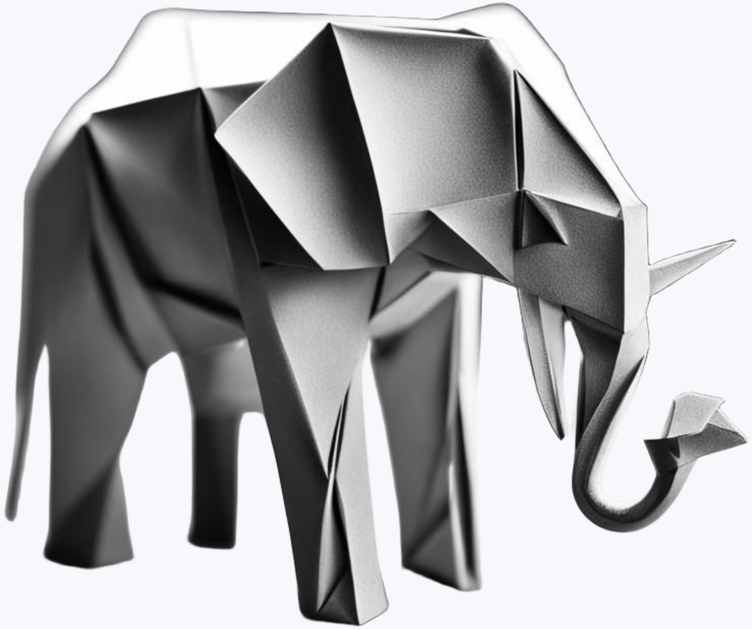


Workload node isolation

INTRODUCTION

The elephant in the room

- As defender can I **predict which service account can be compromised** ?
- As attacker how can I can **access other service accounts** ?
- How **robust** is workload node isolation ?



The Kubernetes Scheduler

Agenda

01

Introduction

02

Kubernetes scheduler

03

Attacking isolation with kubelet

04

Attacking isolation with service account

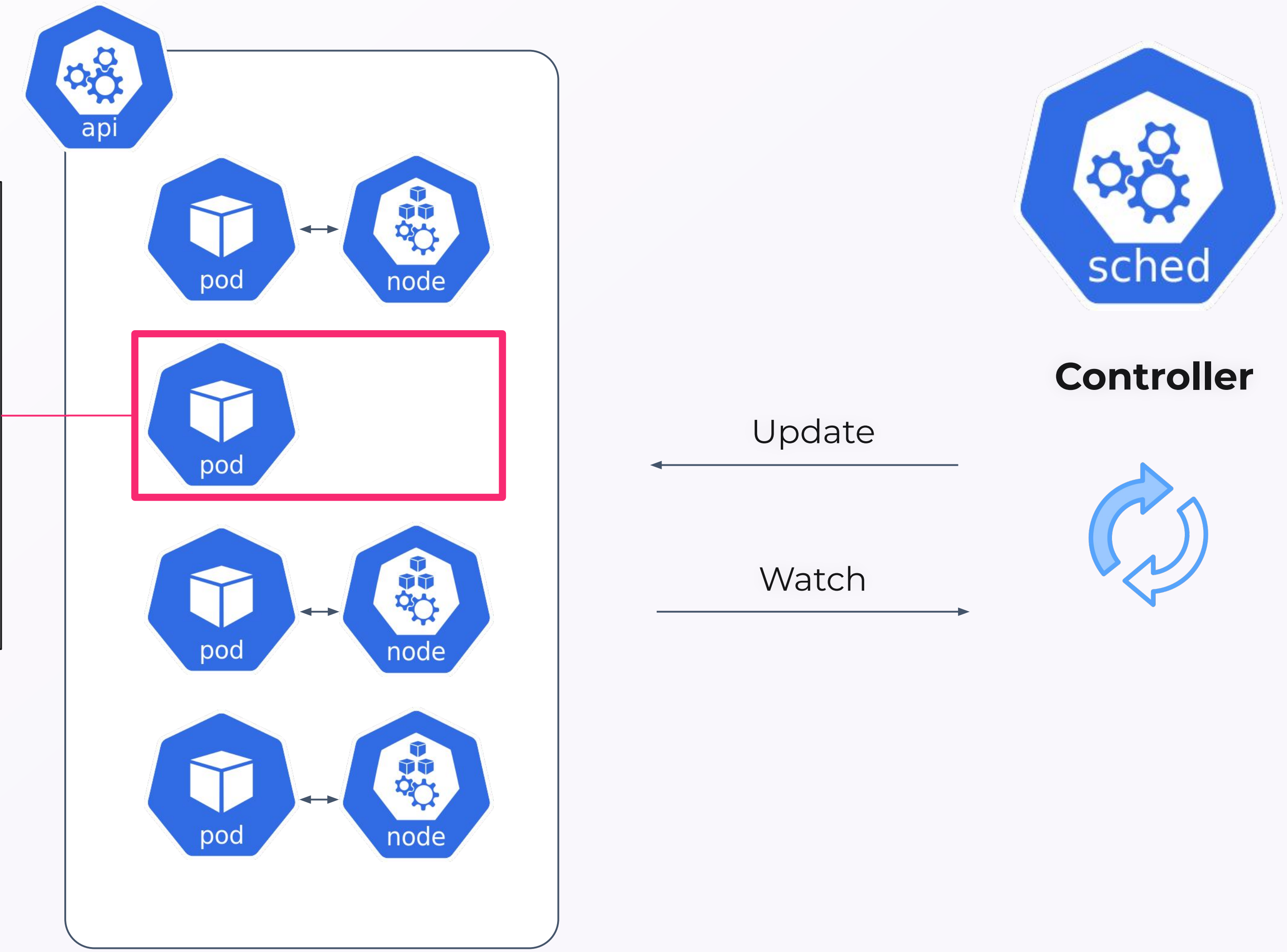
05

Conclusion

KUBERNETES SCHEDULER

Scheduler overview

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeName: null
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```



Controller

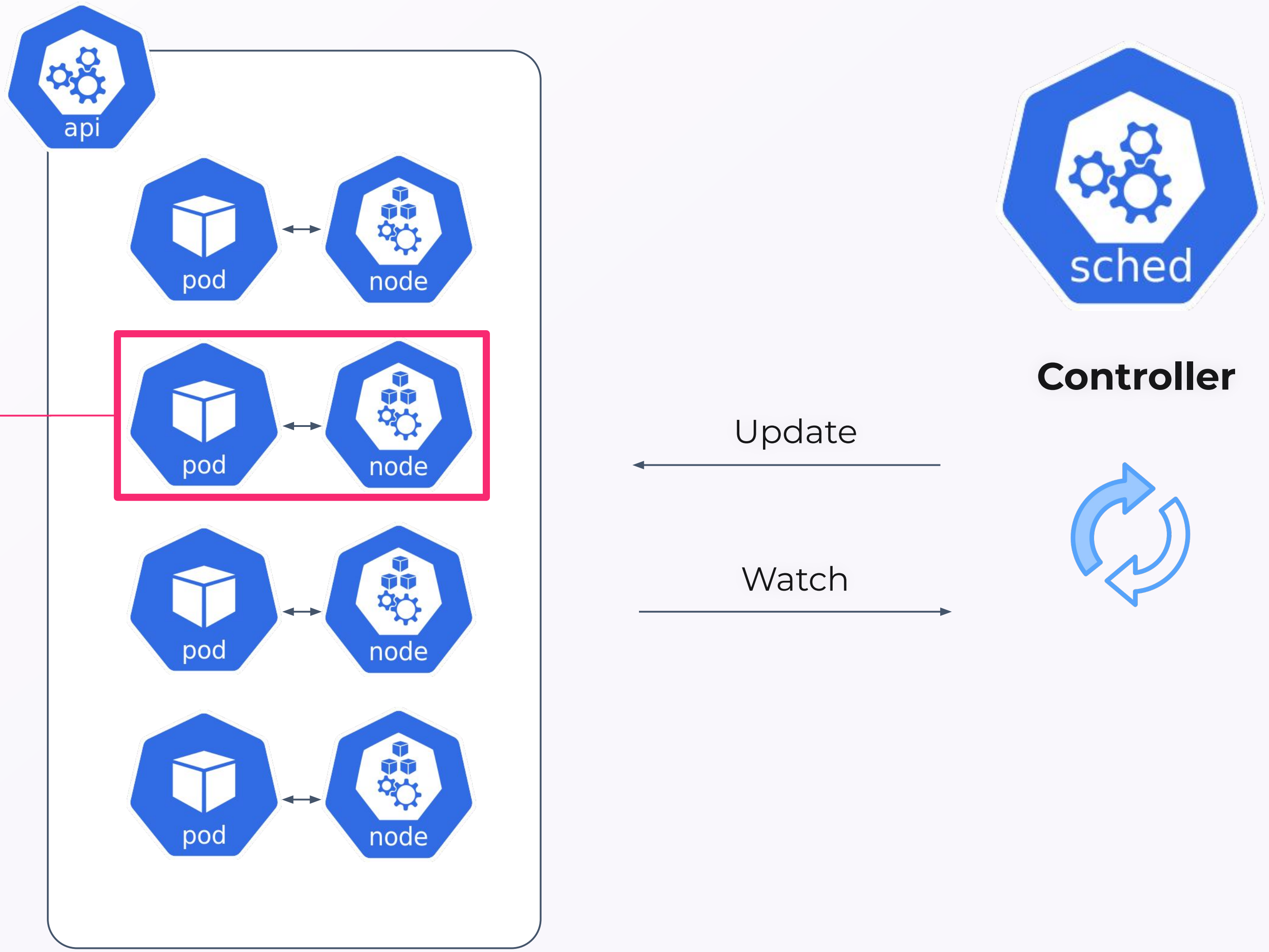
Update

Watch

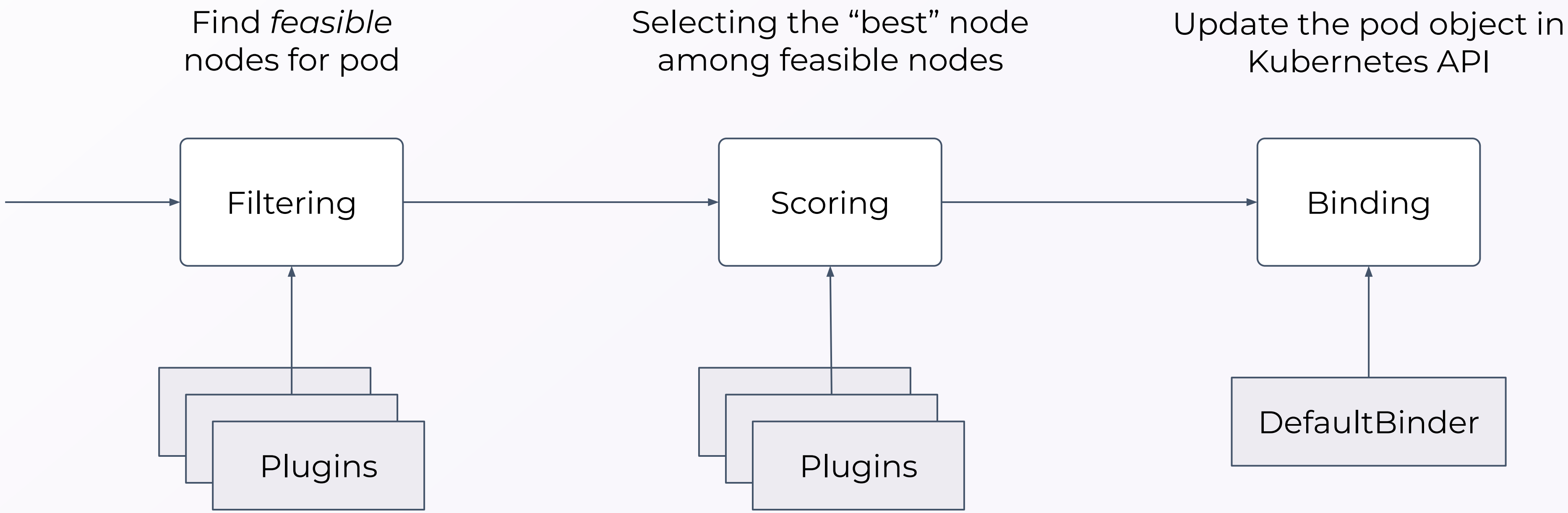
KUBERNETES SCHEDULER

Scheduler overview

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeName: my-node
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```



Scheduler



Find *feasible* nodes for pod

Selecting the “best” node among feasible nodes

Update the pod object in Kubernetes API

Filtering

Scoring

Binding

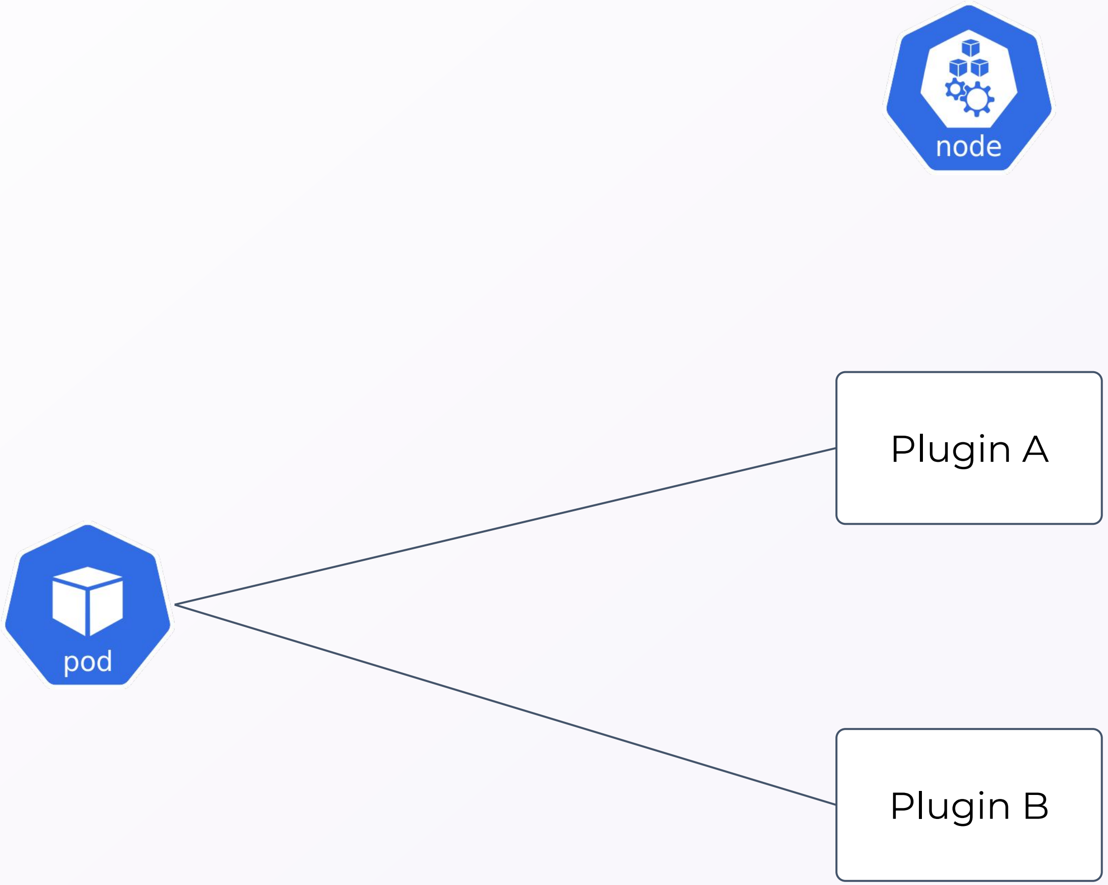
Plugins

Plugins

DefaultBinder

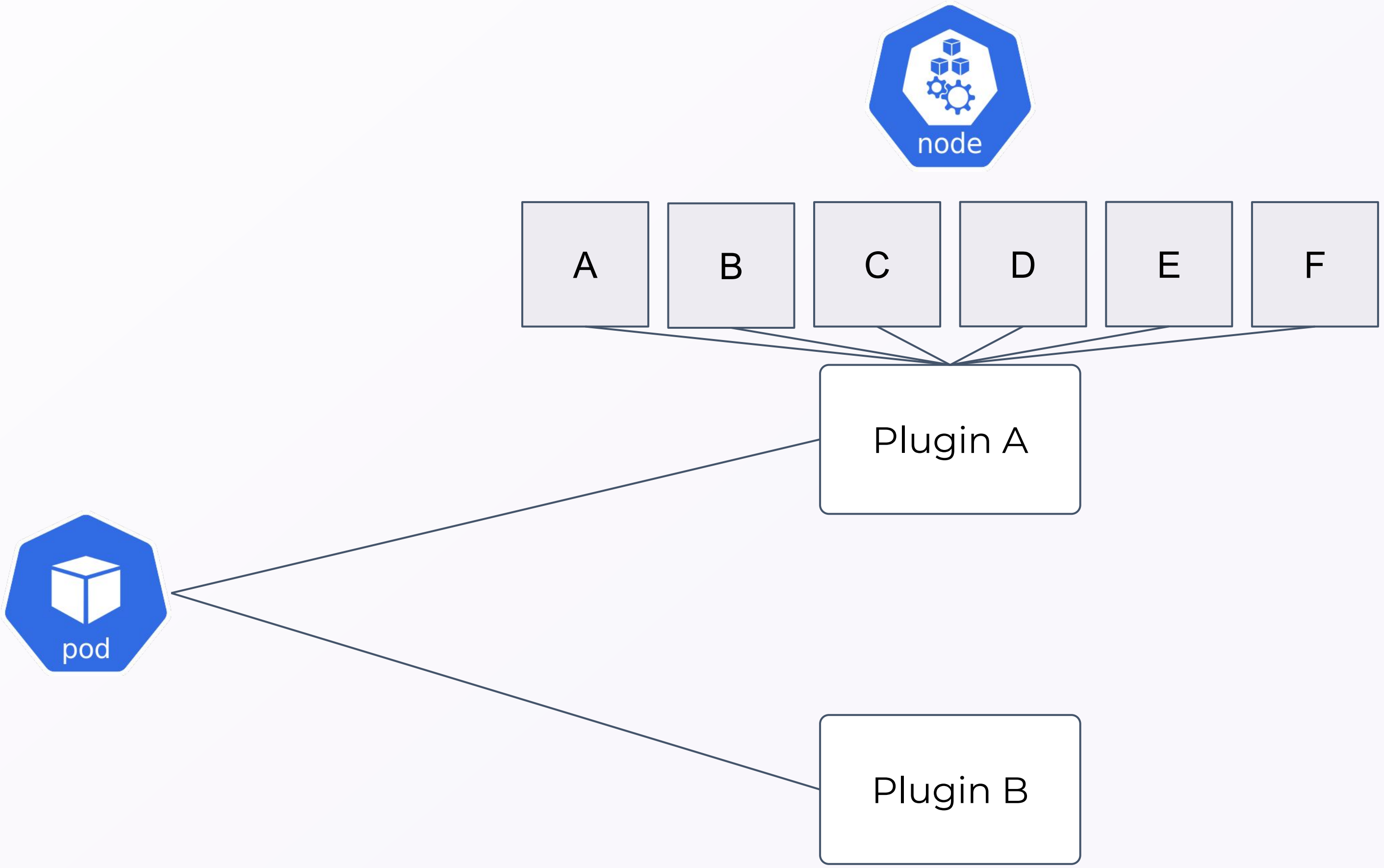
KUBERNETES SCHEDULER

Filtering



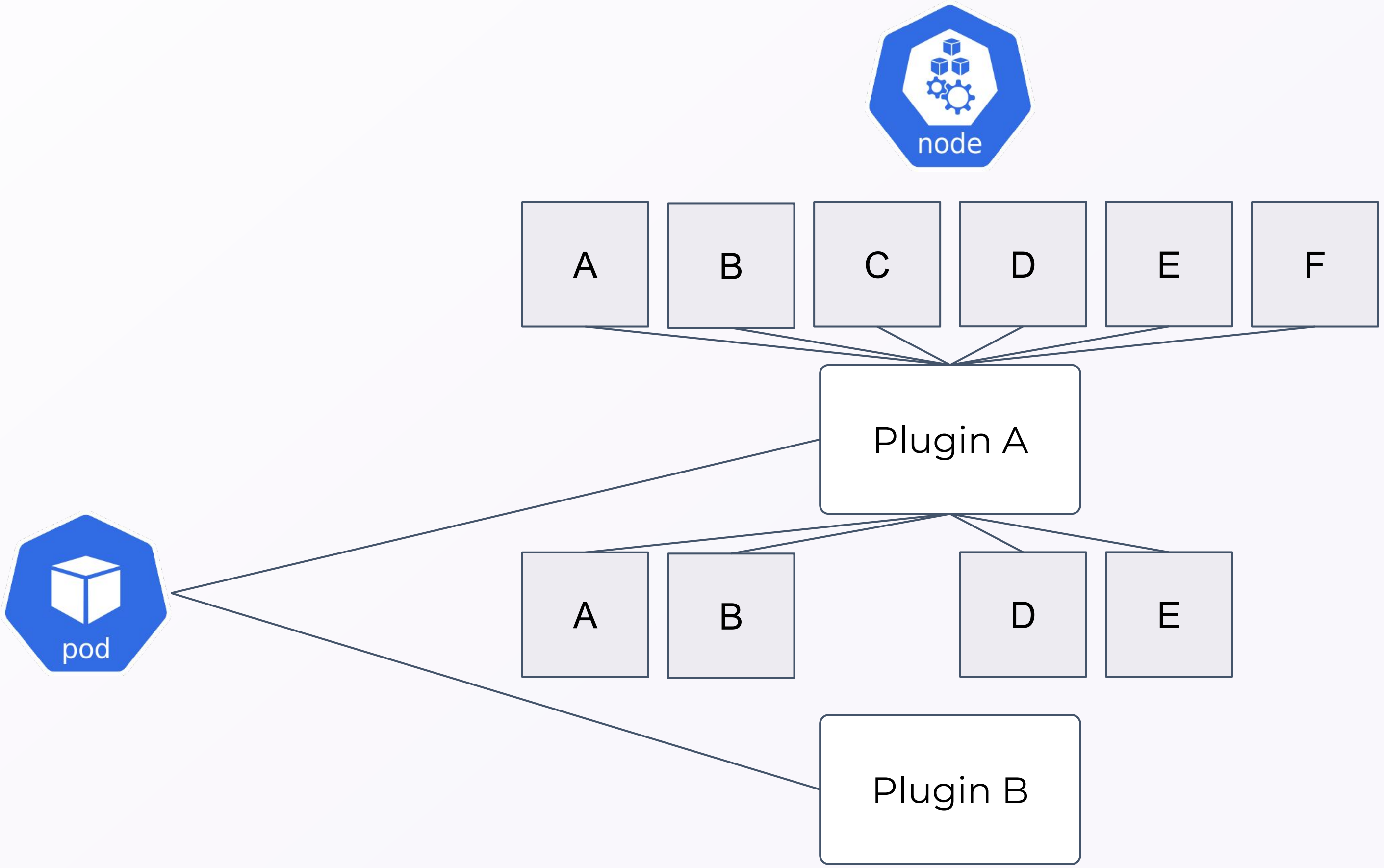
KUBERNETES SCHEDULER

Filtering



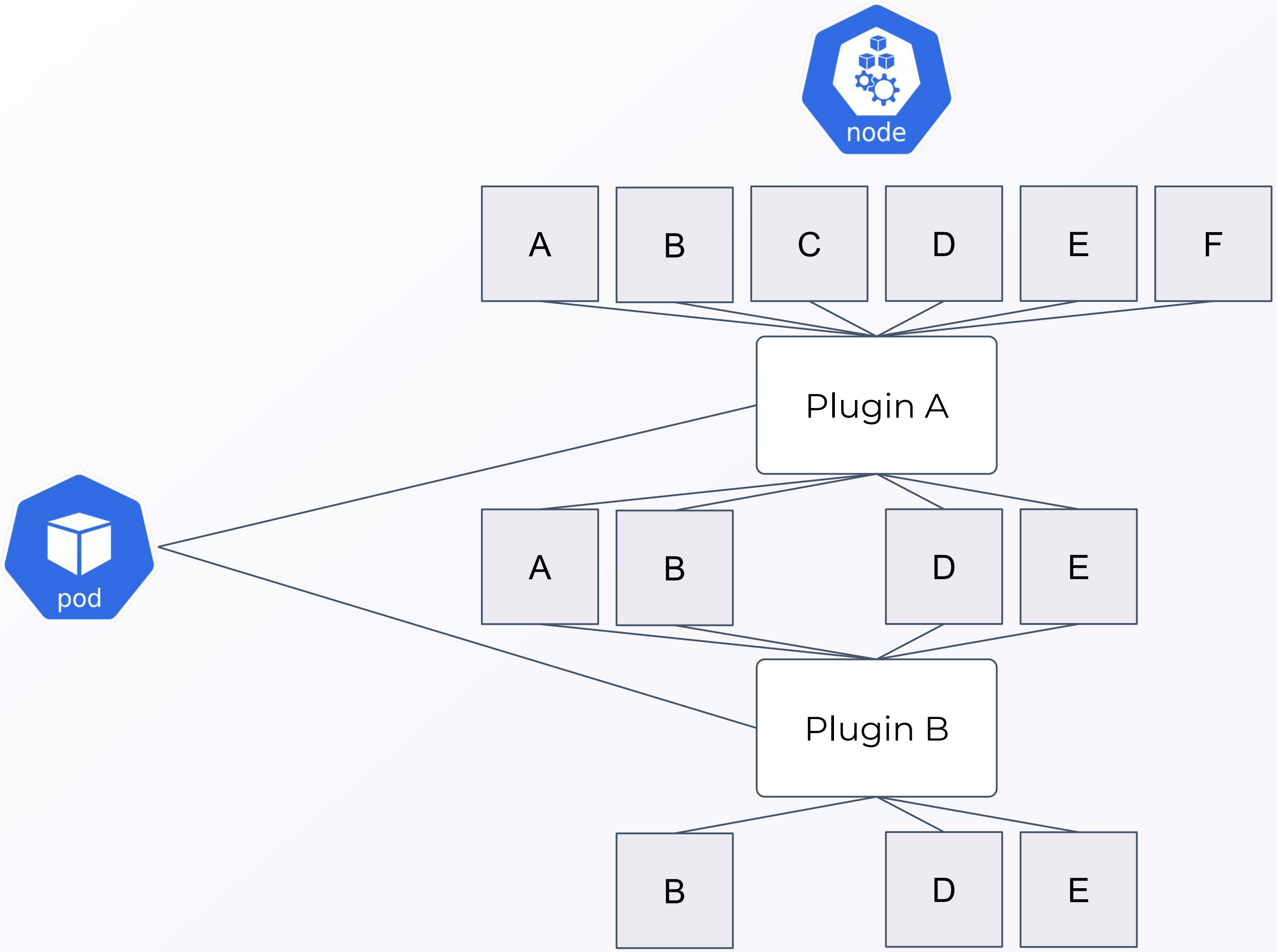
KUBERNETES SCHEDULER

Filtering



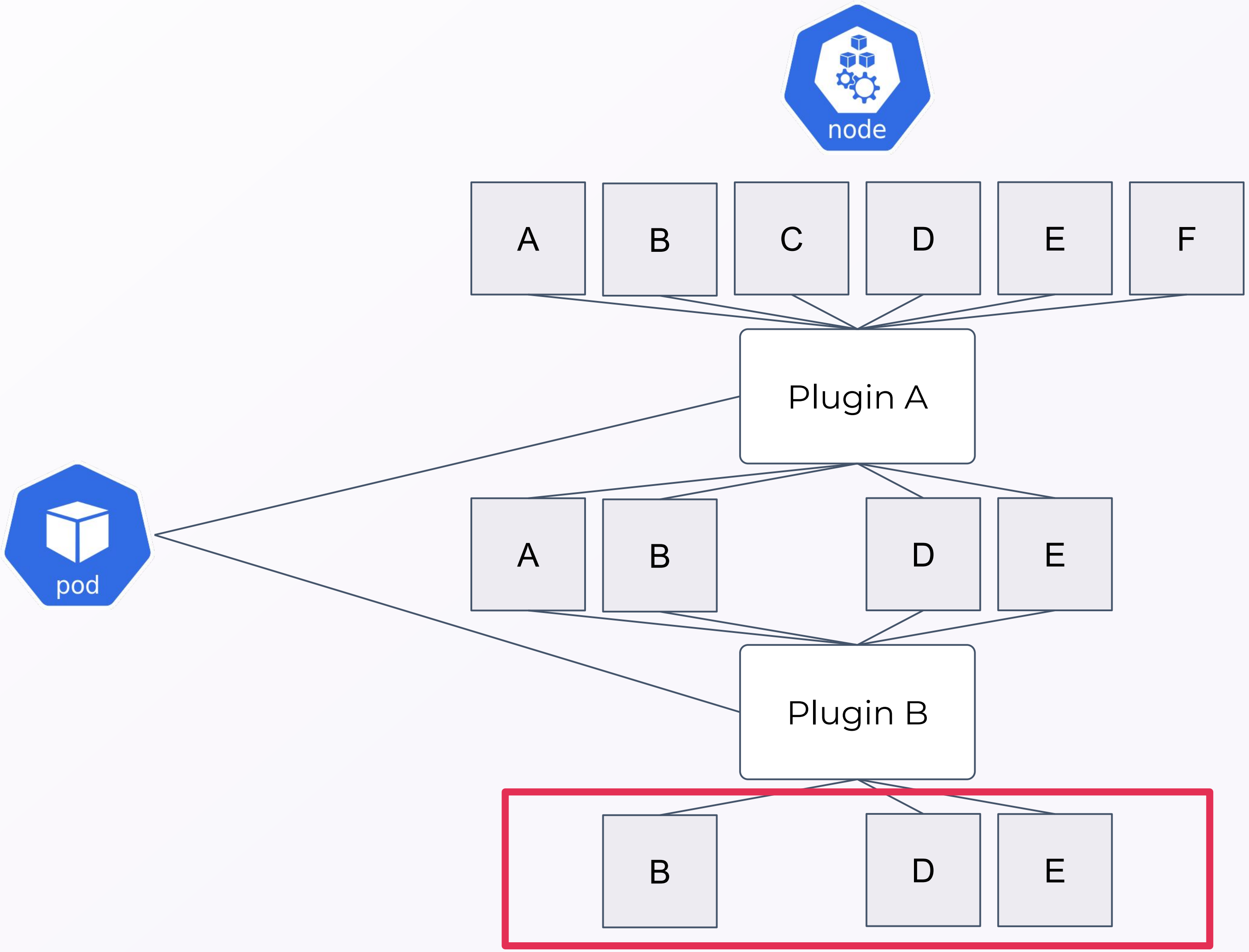
KUBERNETES SCHEDULER

Filtering



KUBERNETES SCHEDULER

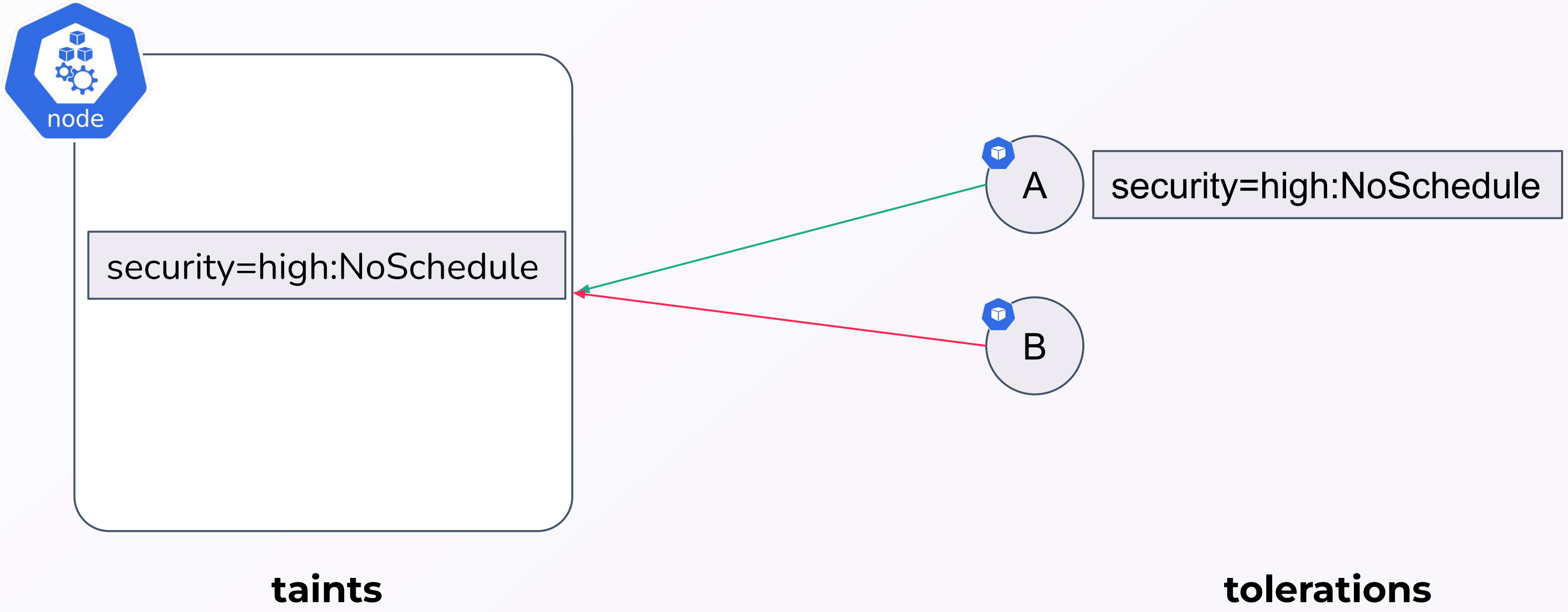
Filtering



feasible nodes for the pod

KUBERNETES SCHEDULER

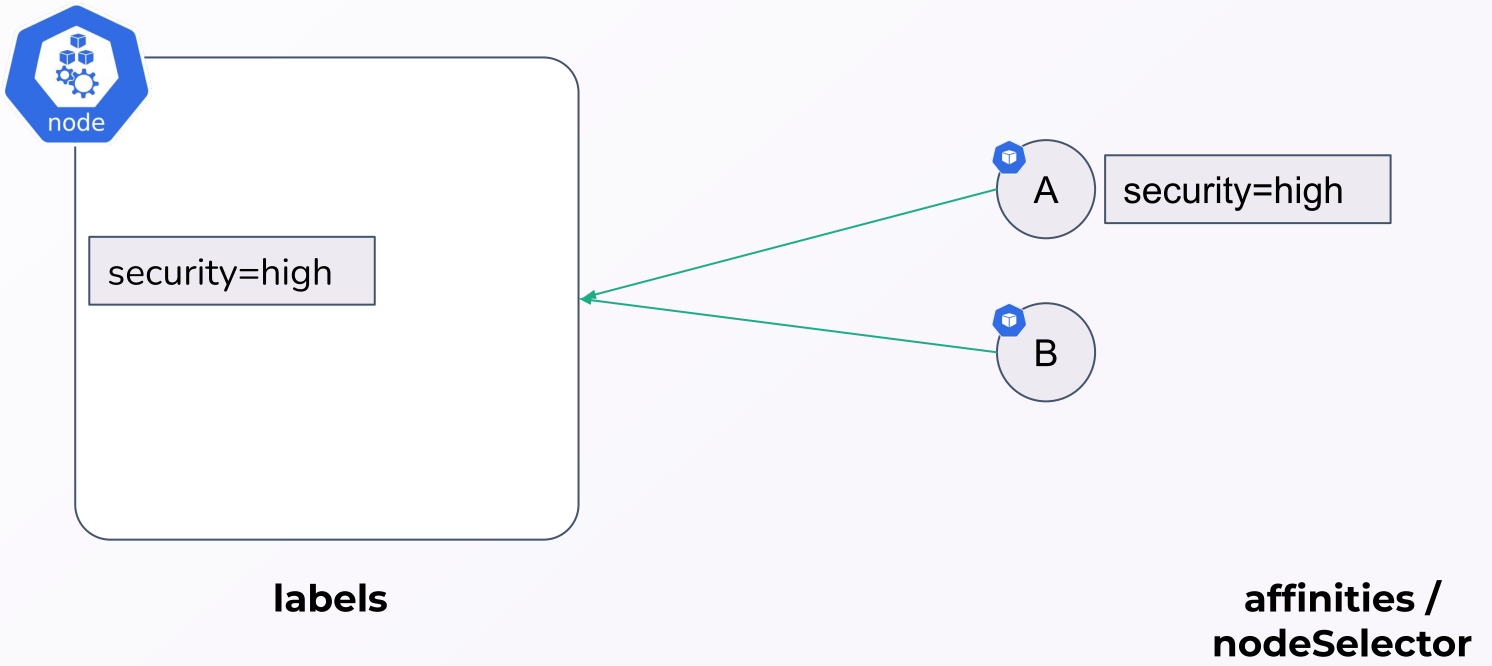
TaintTolerations



→ Nodes **repel** Pods

KUBERNETES SCHEDULER

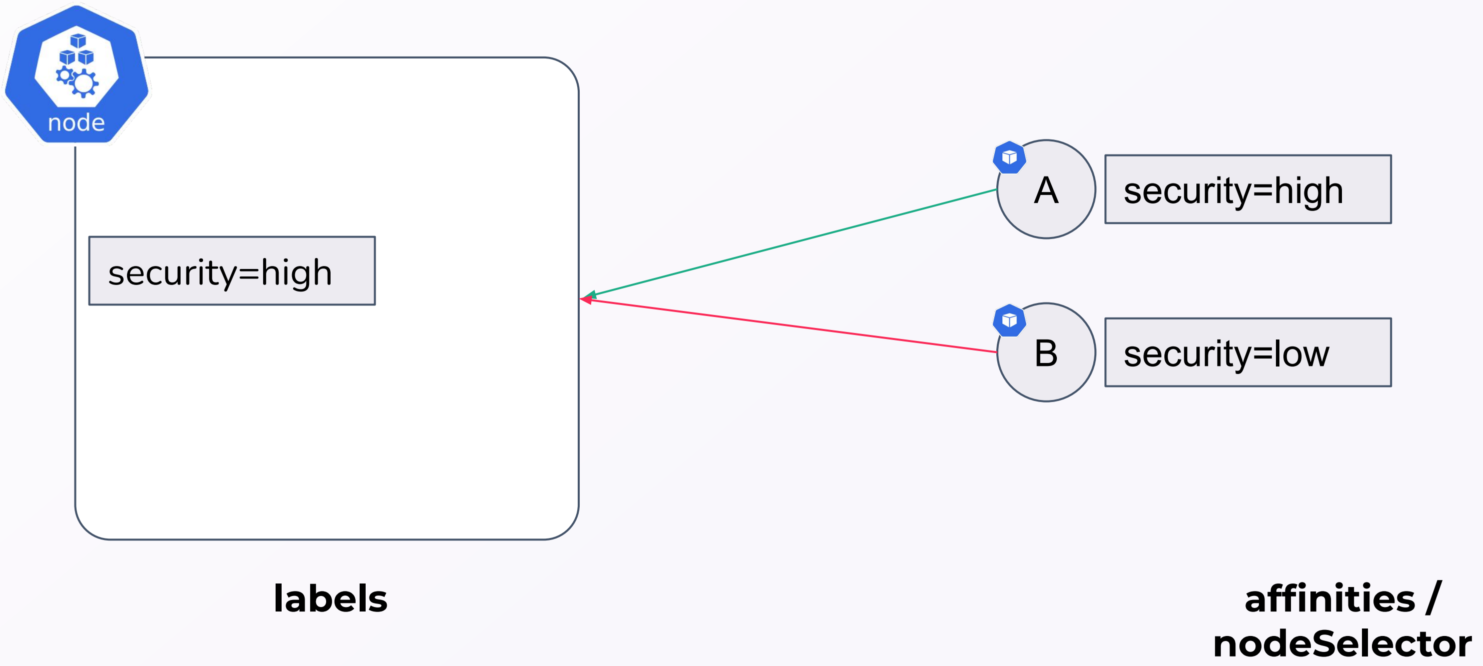
NodeAffinities



- Pods **chose** Nodes

KUBERNETES SCHEDULER

NodeAffinities

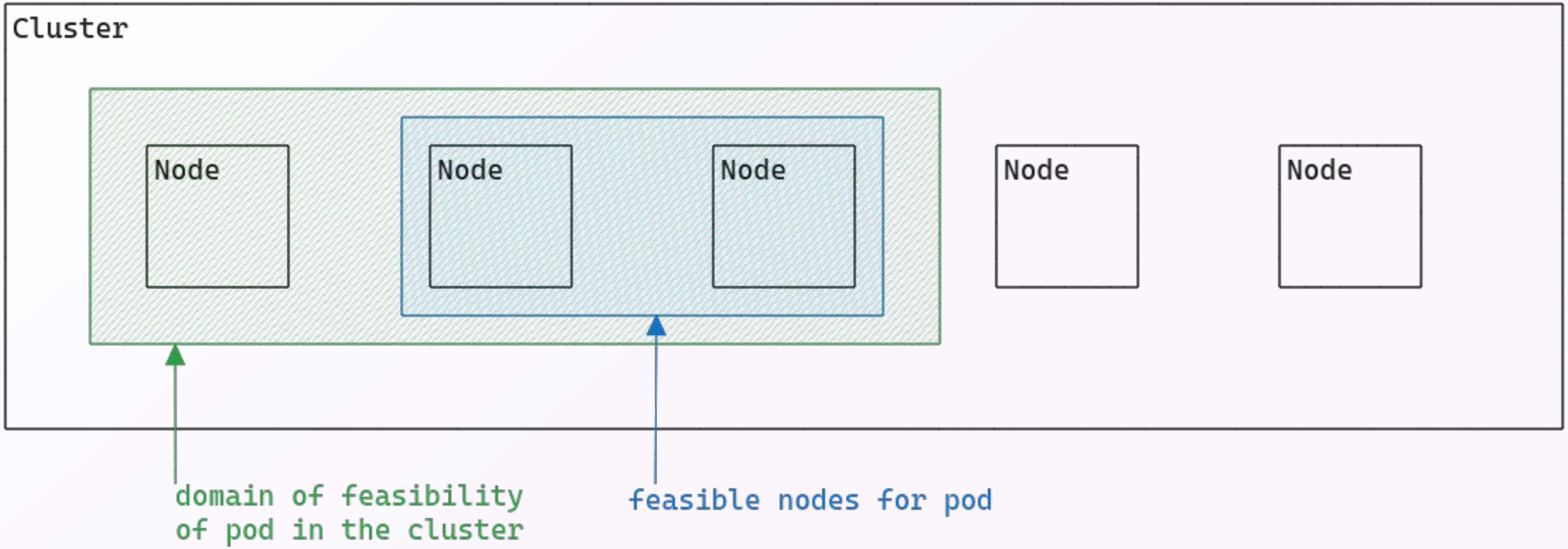


- Pods **chose** Nodes

KUBERNETES SCHEDULER

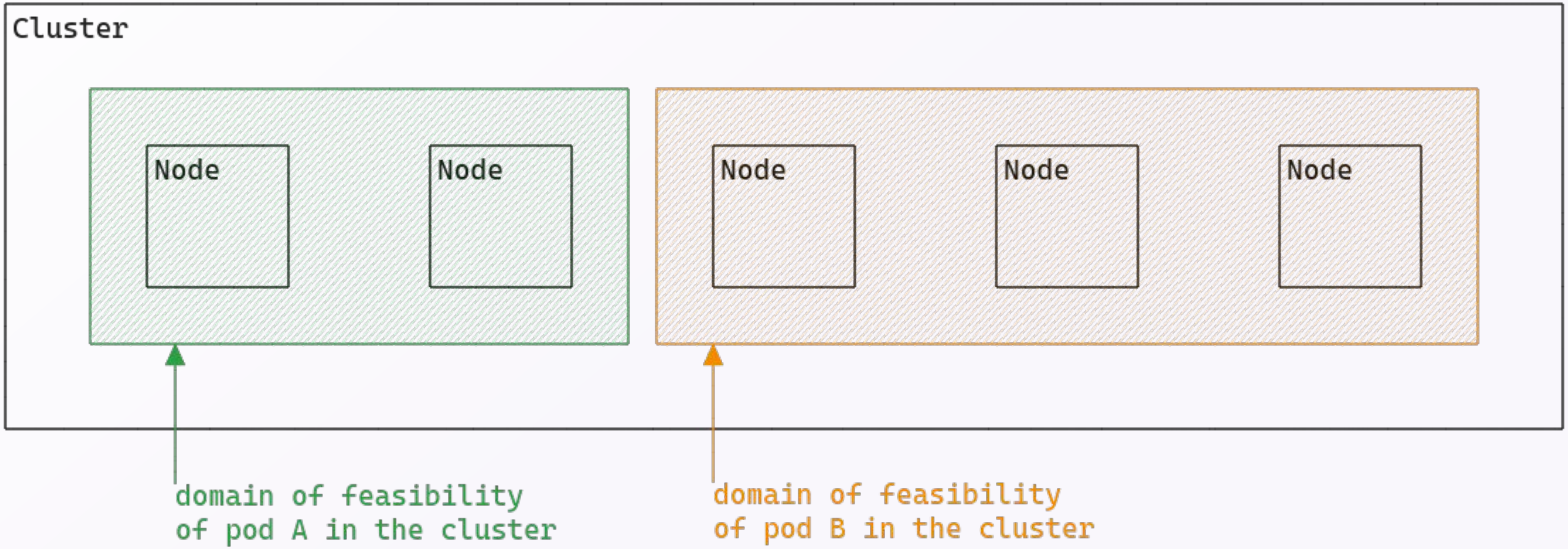
Domain of feasibility

- Domain of feasibility = results of plugins **NodeAffinites** + results of plugins **TaintToleration**



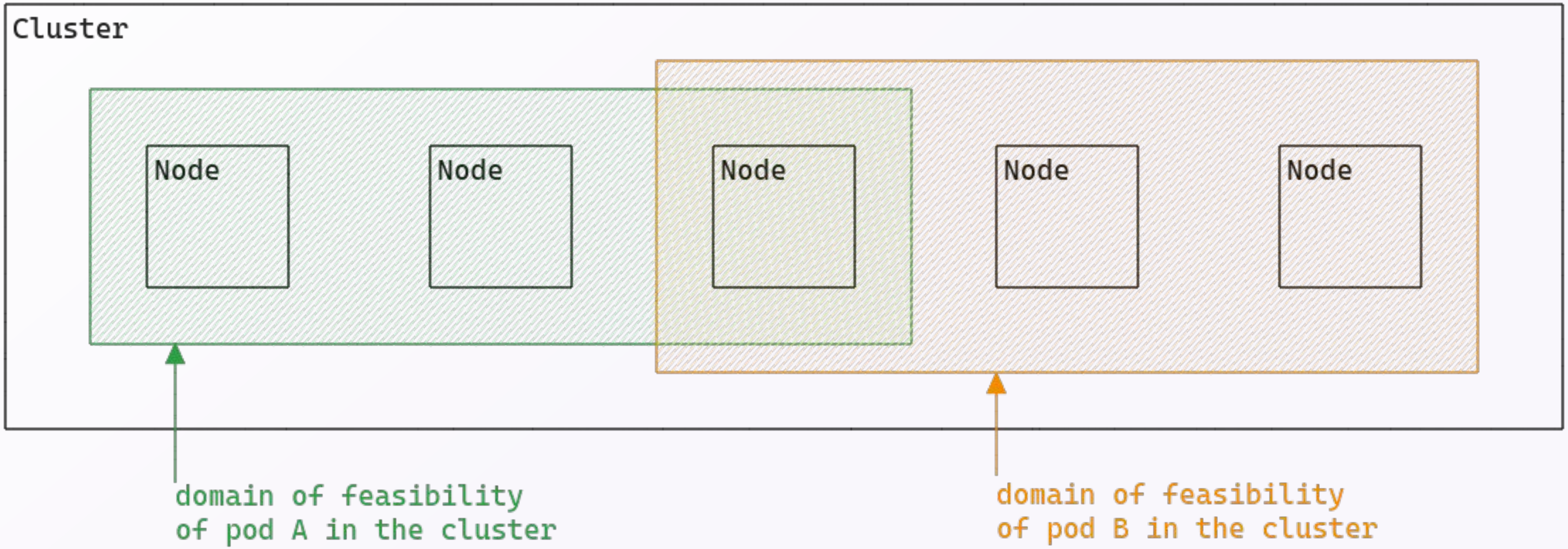
KUBERNETES SCHEDULER

Domain of feasibility: isolated pods

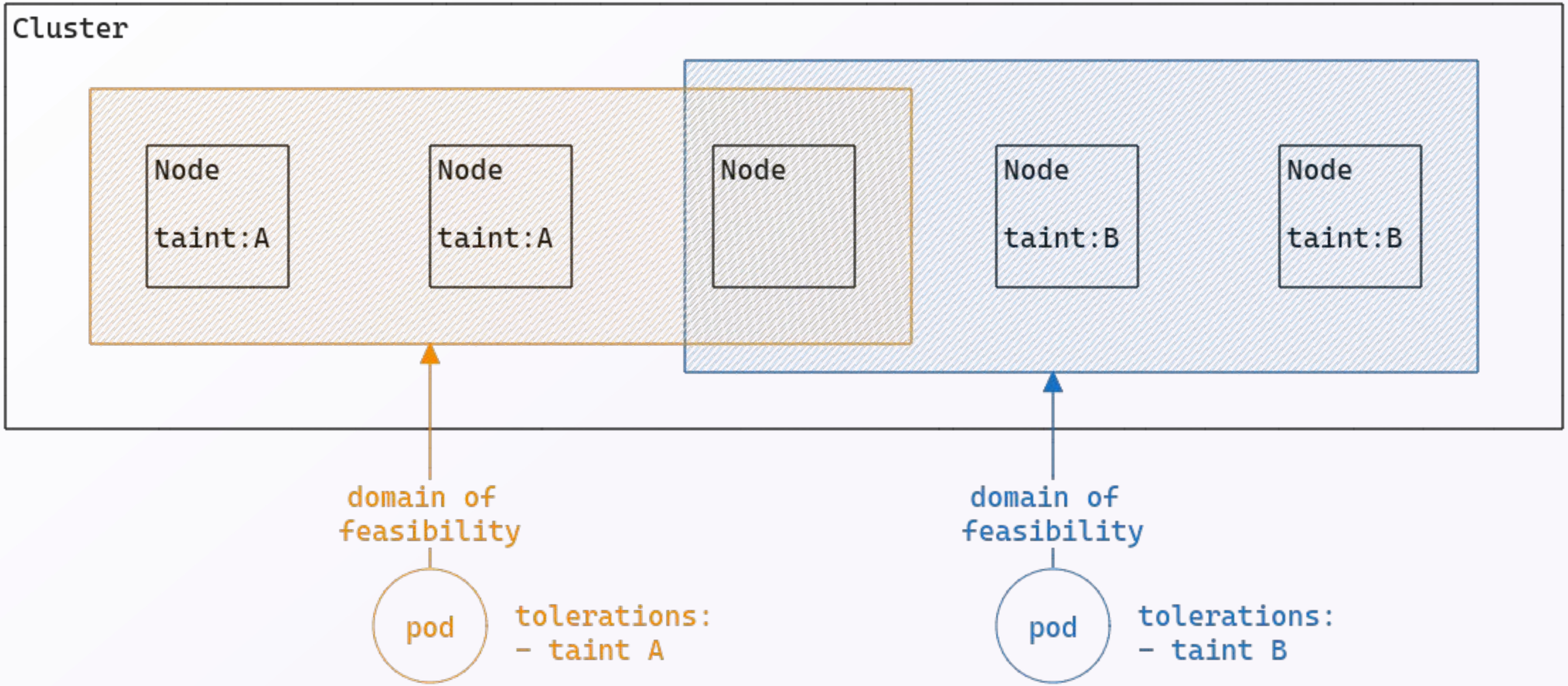


KUBERNETES SCHEDULER

Domain of feasibility: possibly non-isolated pods

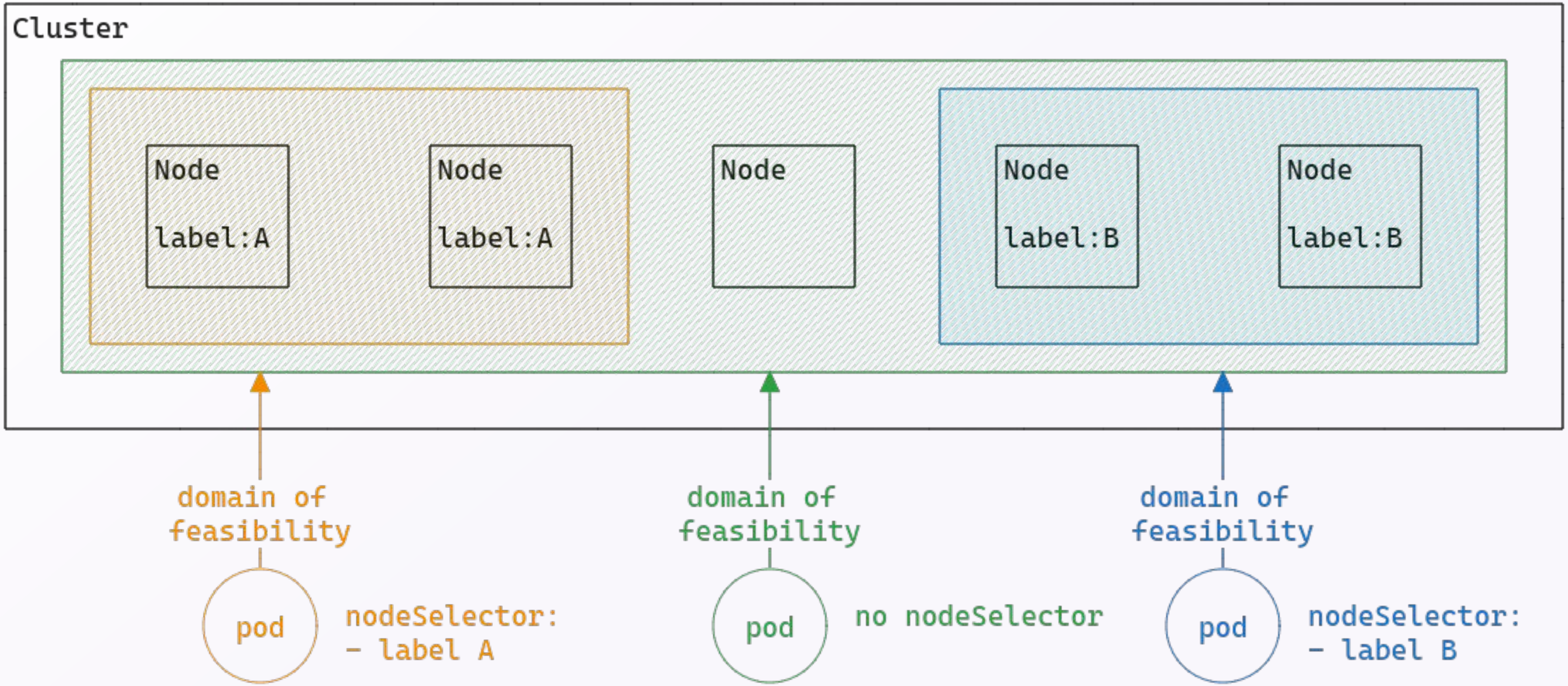


Isolation anti-patterns: only taints

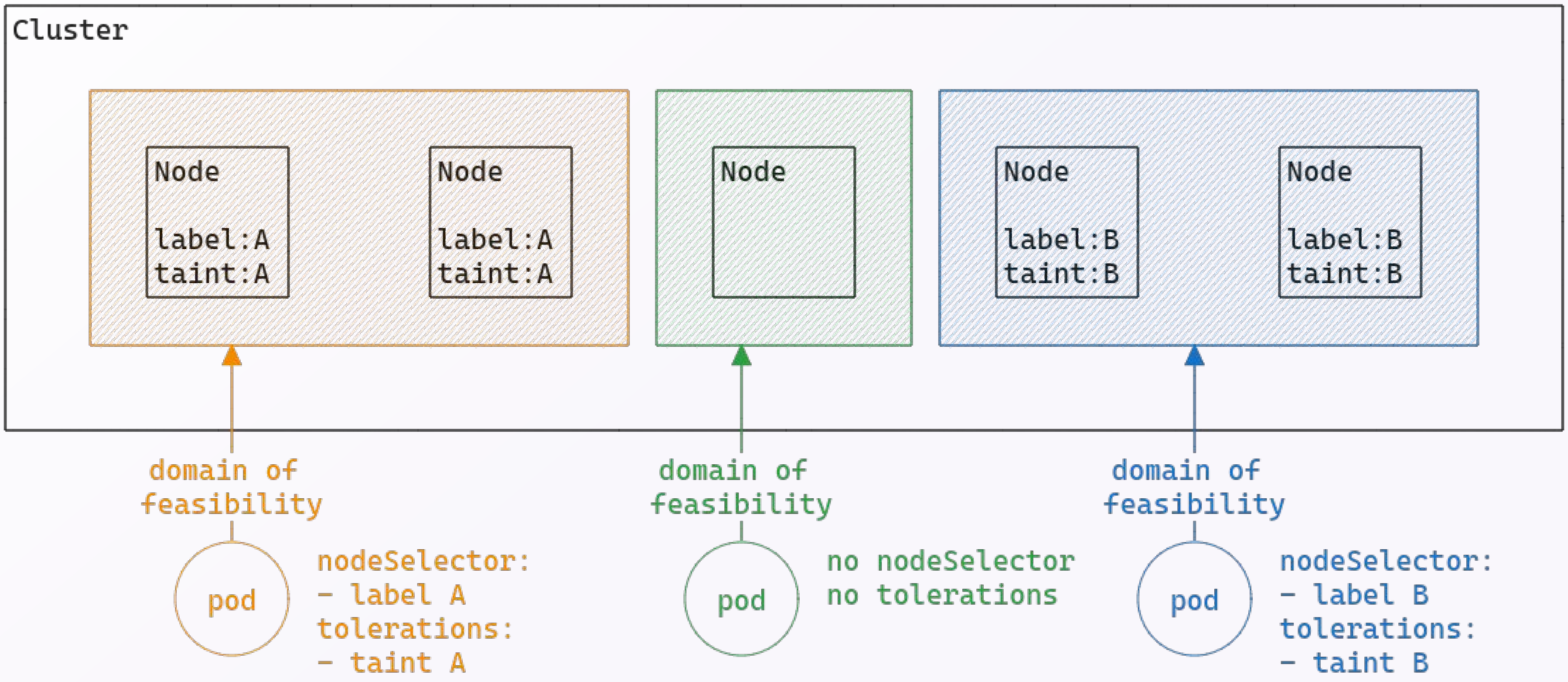


KUBERNETES SCHEDULER

Isolation anti-patterns: only affinities and node selector

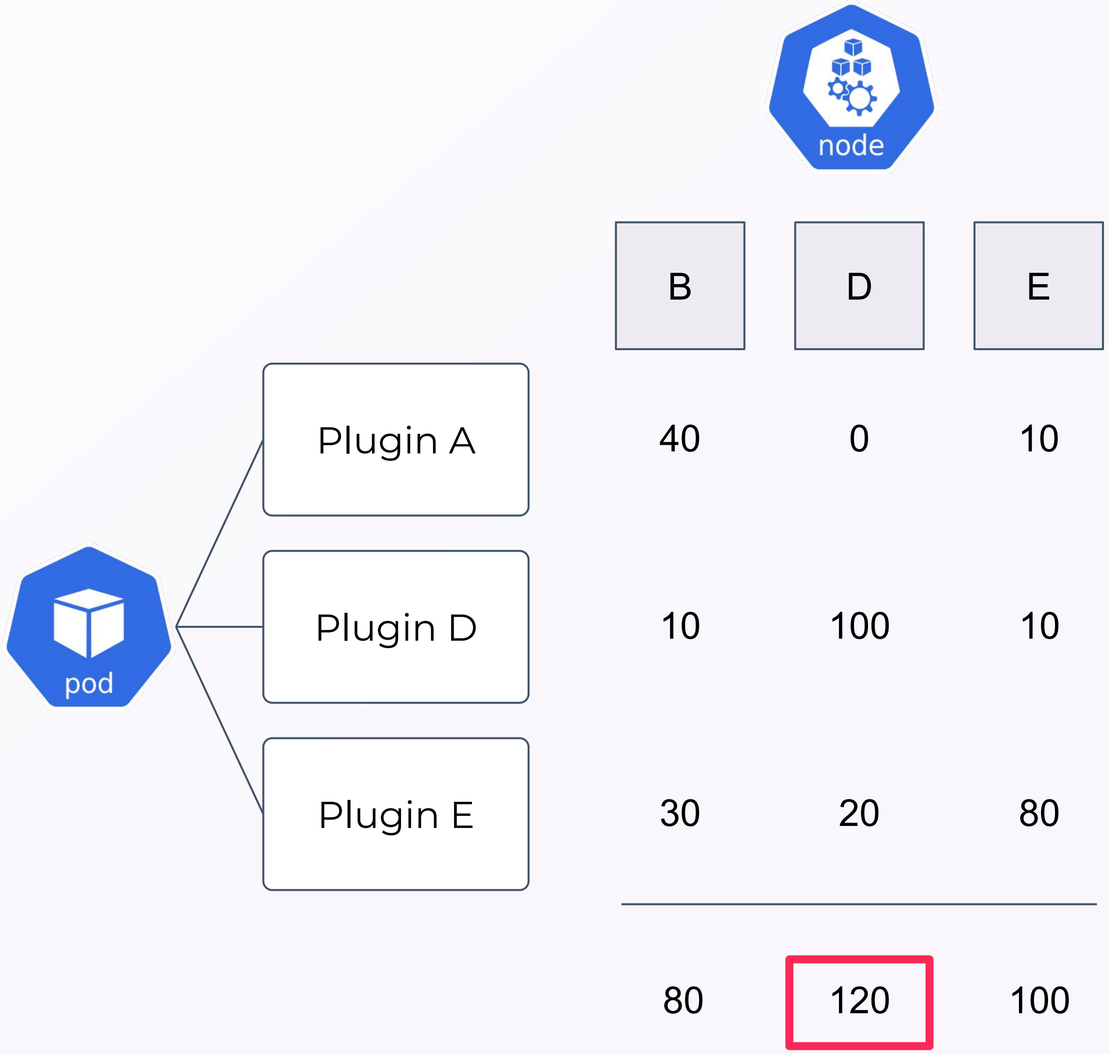


Isolation pattern: using both



KUBERNETES SCHEDULER

Scoring

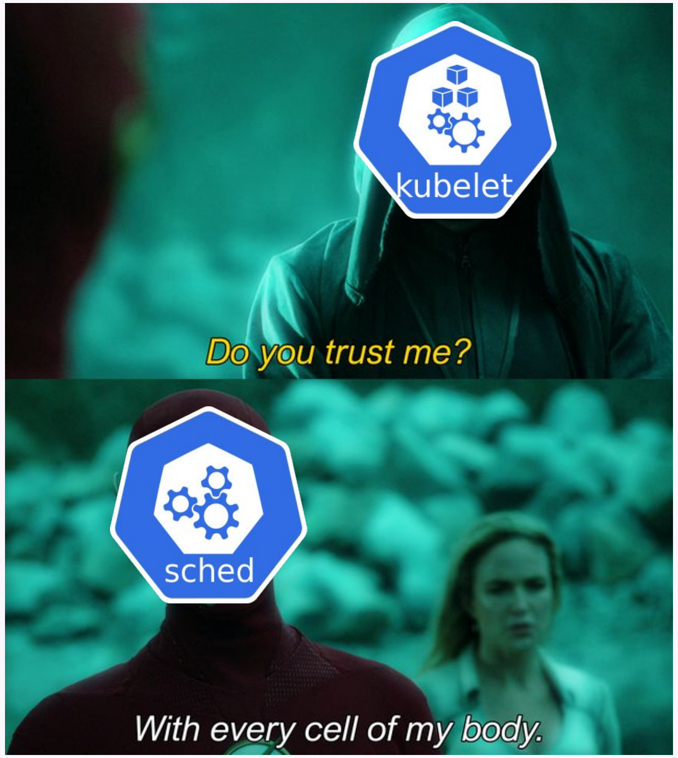


	B	D	E
Plugin A	40	0	10
Plugin D	10	100	10
Plugin E	30	20	80
	80	120	100

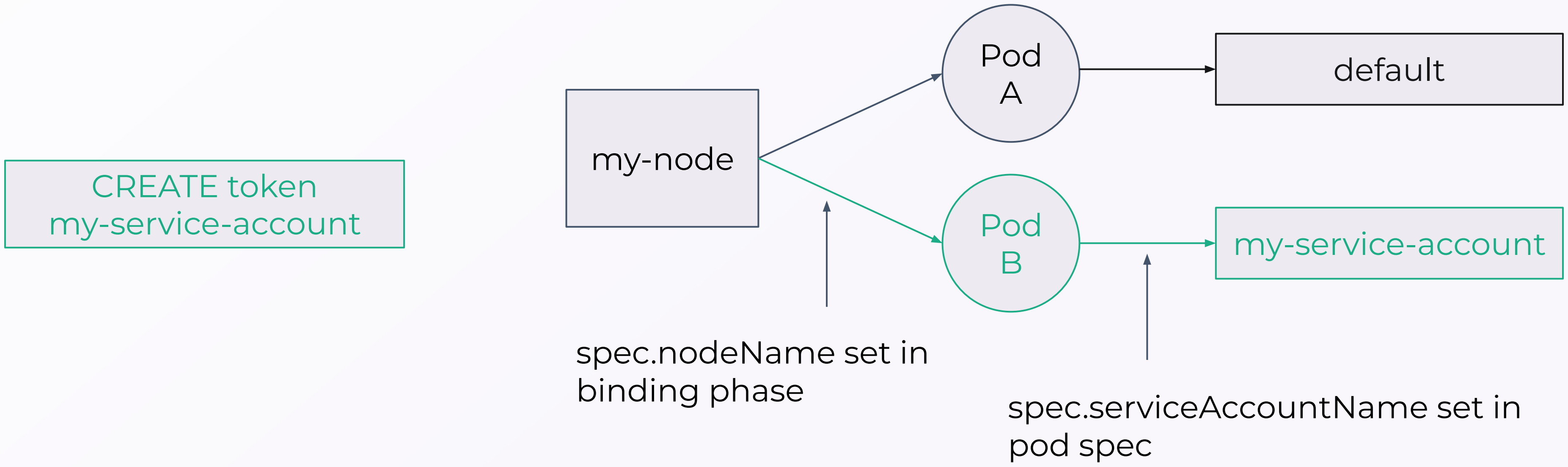
KUBERNETES SCHEDULER

Scoring

- Mainly based on **Node status**
- Node status is **fully editable** by the kubelet account
- We are **almost certain** to be able to **attract Pod** when the compromised node is in the **domain of feasibility**



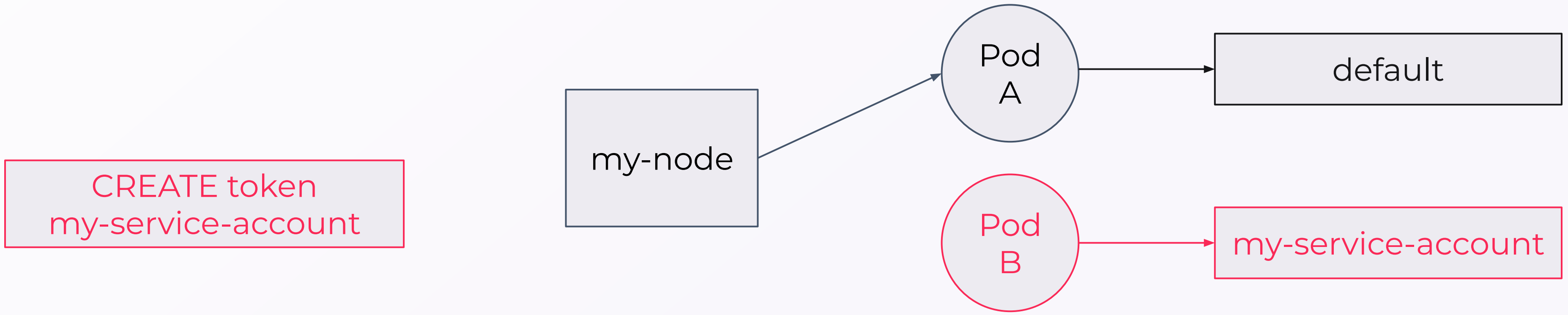
Binding: Node authorizer



spec.nodeName set in binding phase

spec.serviceAccountName set in pod spec

Binding: Node authorizer



Agenda

01

Introduction

02

Kubernetes scheduler

03

Attacking isolation with kubelet

04

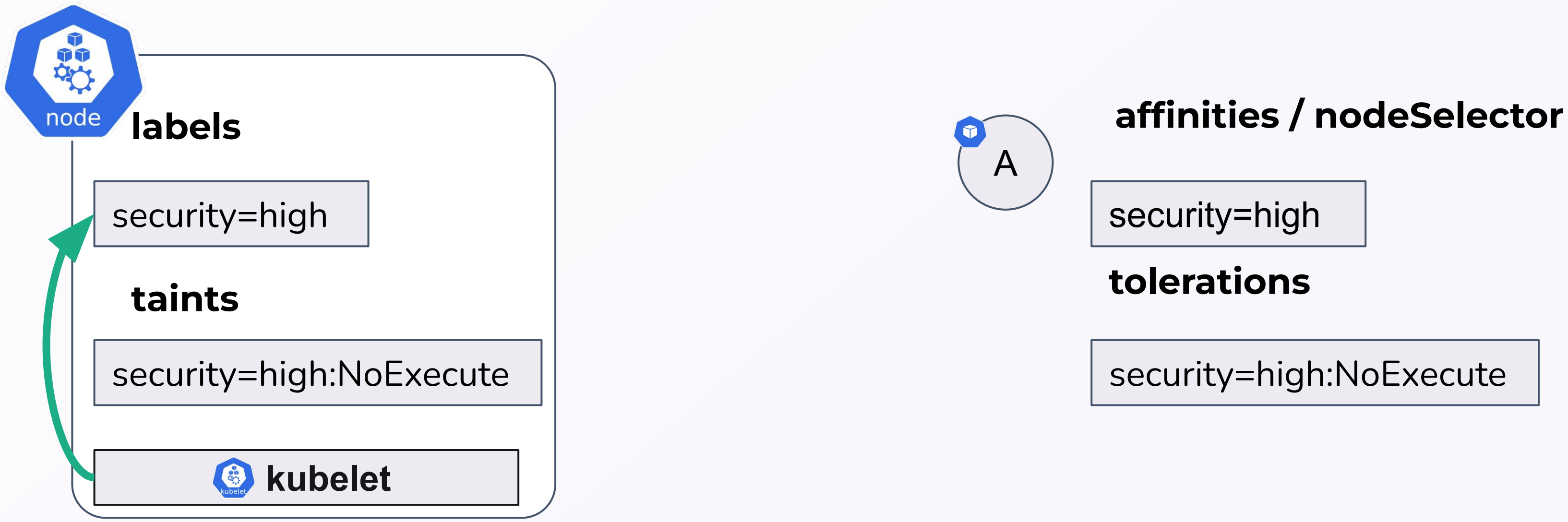
Attacking isolation with service account

05

Conclusion

ATTACKING ISOLATION WITH KUBELET

Modify domain of feasibility using the kubelet account



node

labels

security=high

taints

security=high:NoExecute

kubelet

A

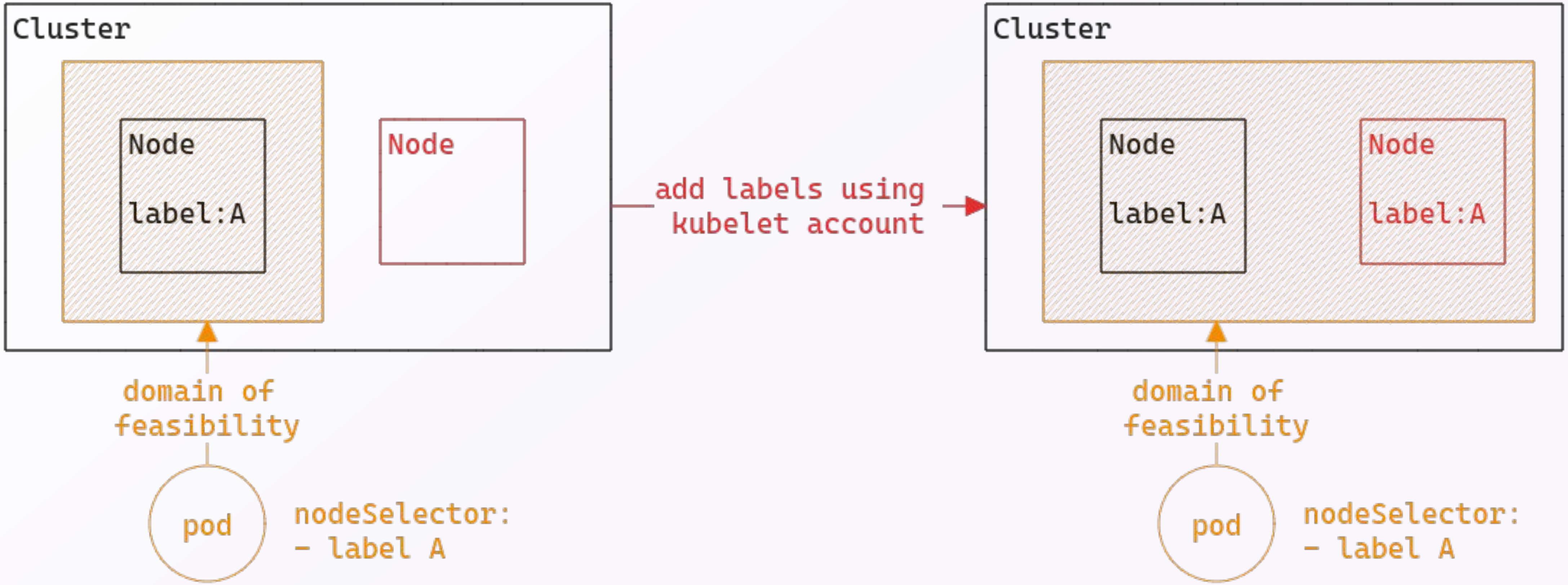
affinities / nodeSelector

security=high

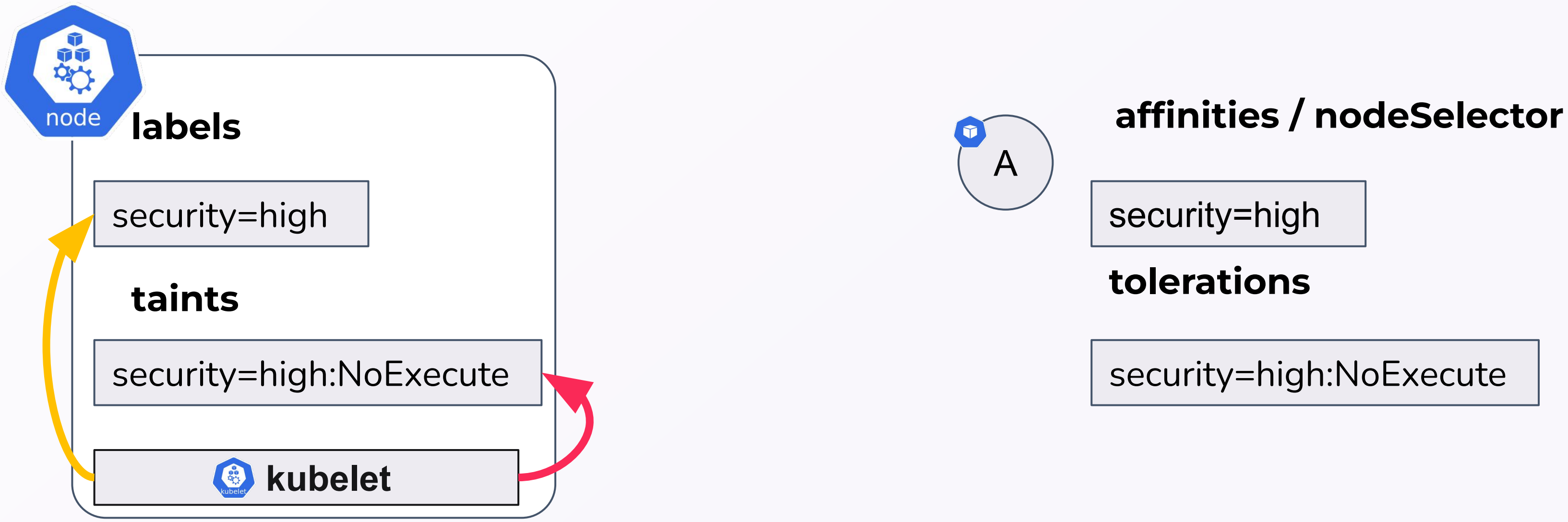
tolerations

security=high:NoExecute

Modify domain of feasibility using the kubelet account



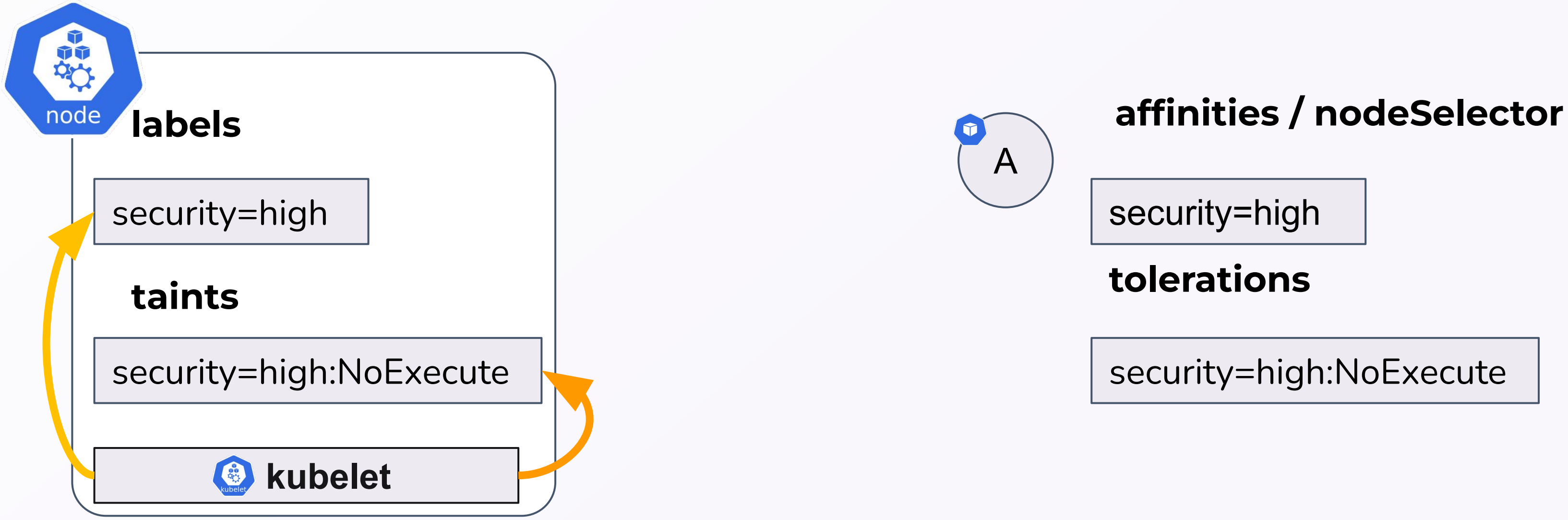
Modify domain of feasibility using the kubelet account



- **NodeRestriction** admission plugin limits the editable labels
- It defines a **blacklist** that contains, among others:
  - **node-restriction.kubernetes.io/**
- It prevents also modifying the taints

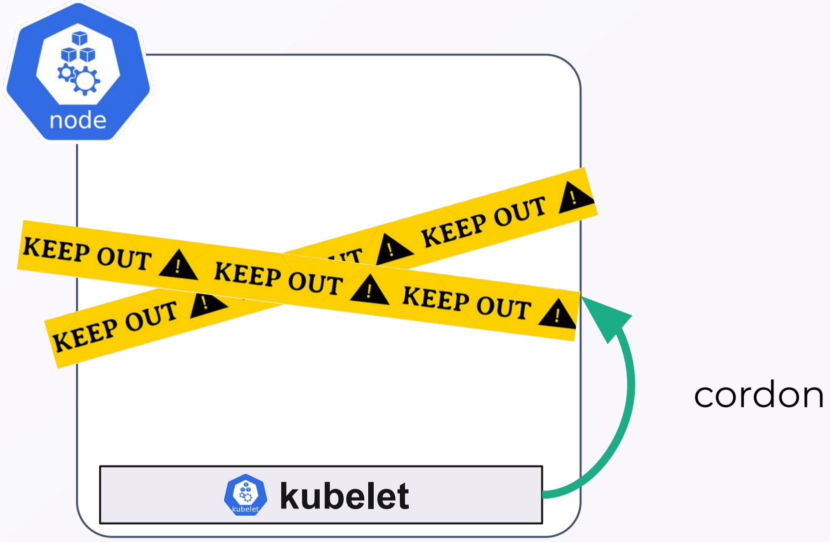
ATTACKING ISOLATION WITH KUBELET

Modify domain of feasibility using the kubelet account



Modify domain of feasibility using the kubelet account

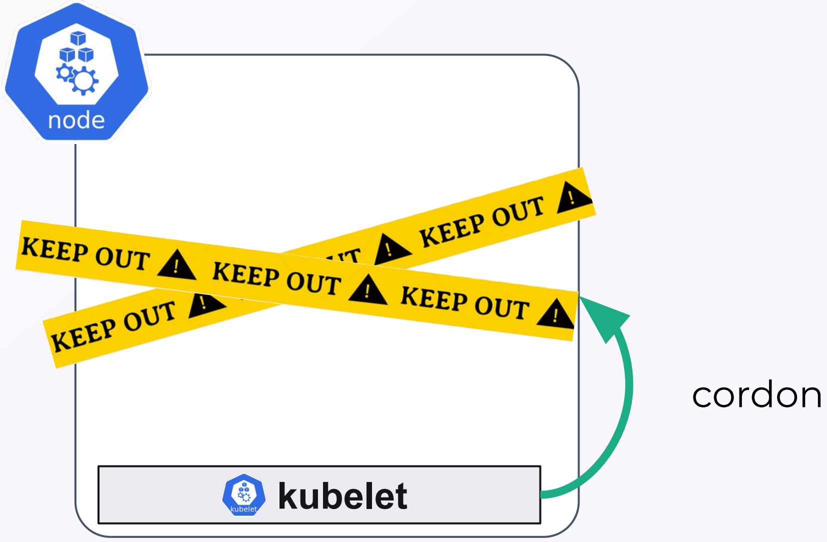
```yaml
apiVersion: v1
kind: Node
spec:
  unschedulable: true
```



ATTACKING ISOLATION WITH KUBELET

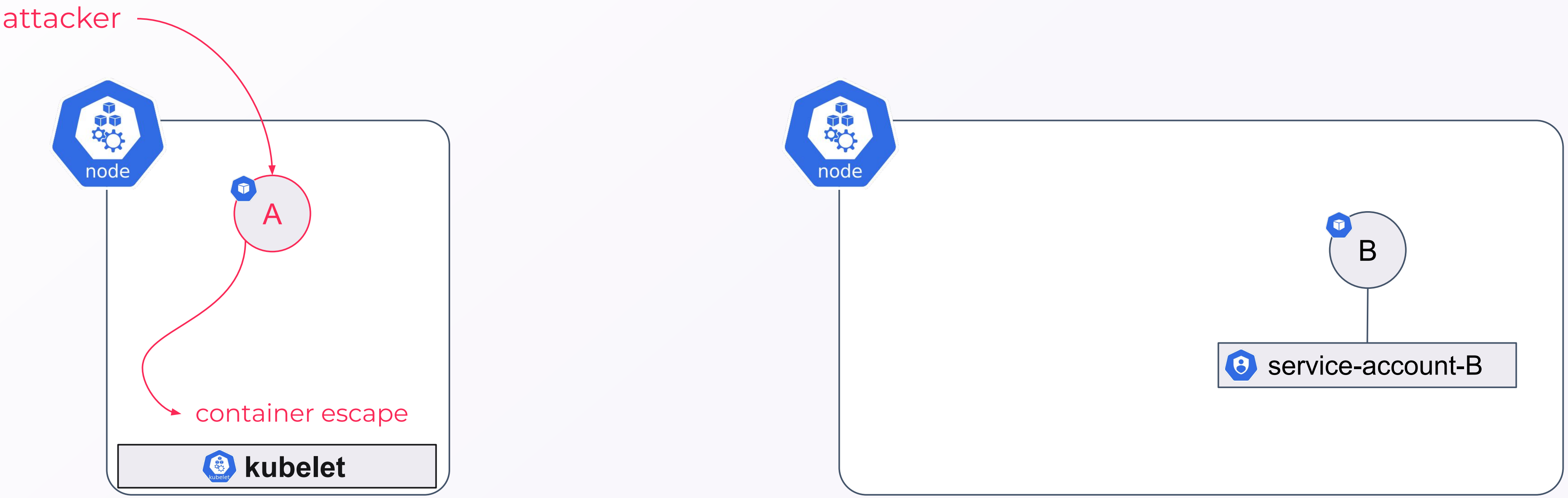
Modify domain of feasibility using the kubelet account

```
apiVersion: v1
kind: Node
spec:
  unschedulable: true
  taints:
    - effect: NoSchedule
      key:node.kubernetes.io/unschedulable
```



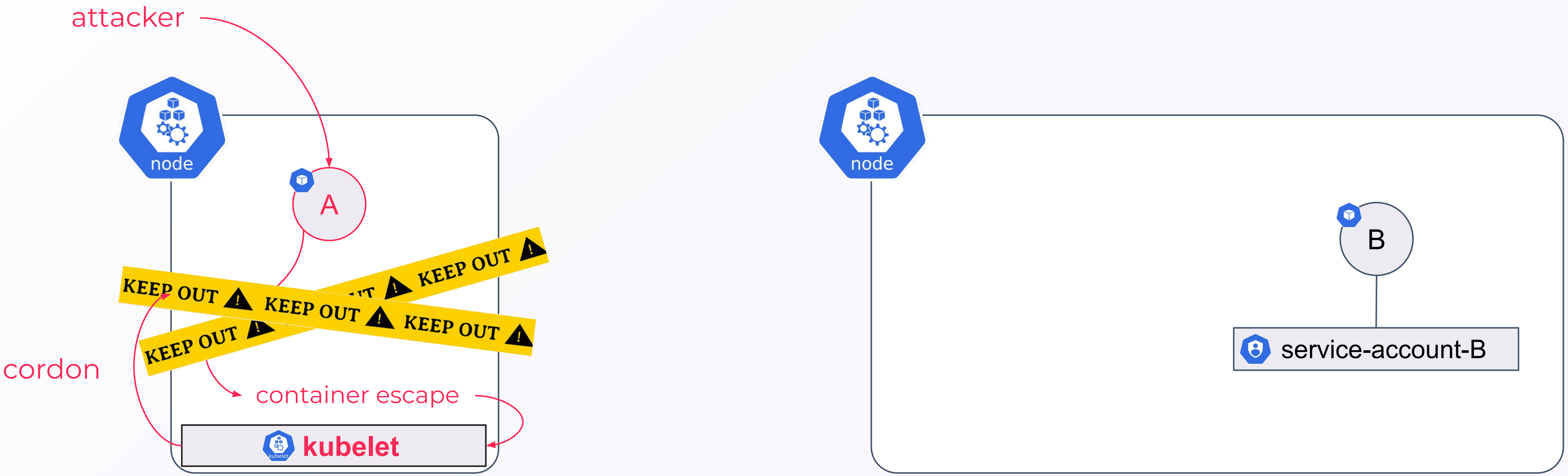
ATTACKING ISOLATION WITH KUBELET

Modify domain of feasibility using the kubelet account



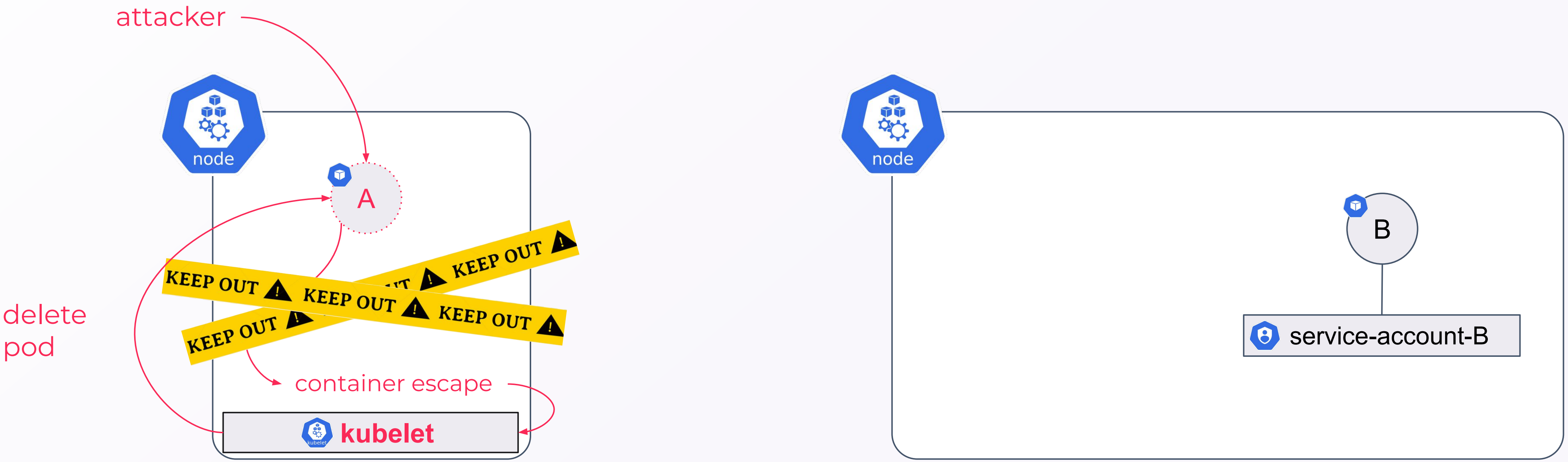
ATTACKING ISOLATION WITH KUBELET

Modify domain of feasibility using the kubelet account



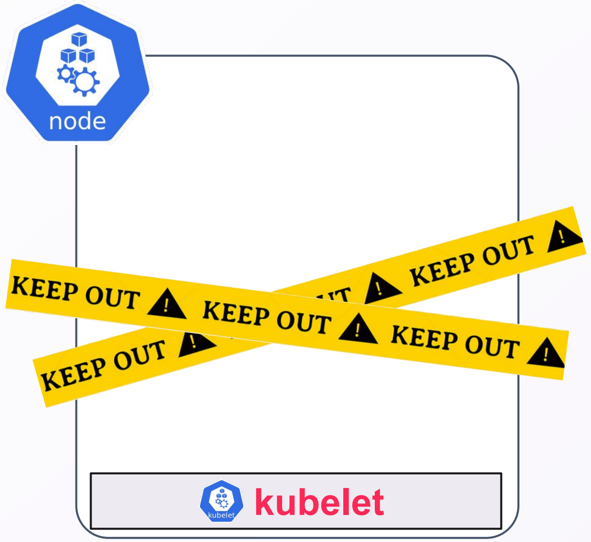
ATTACKING ISOLATION WITH KUBELET

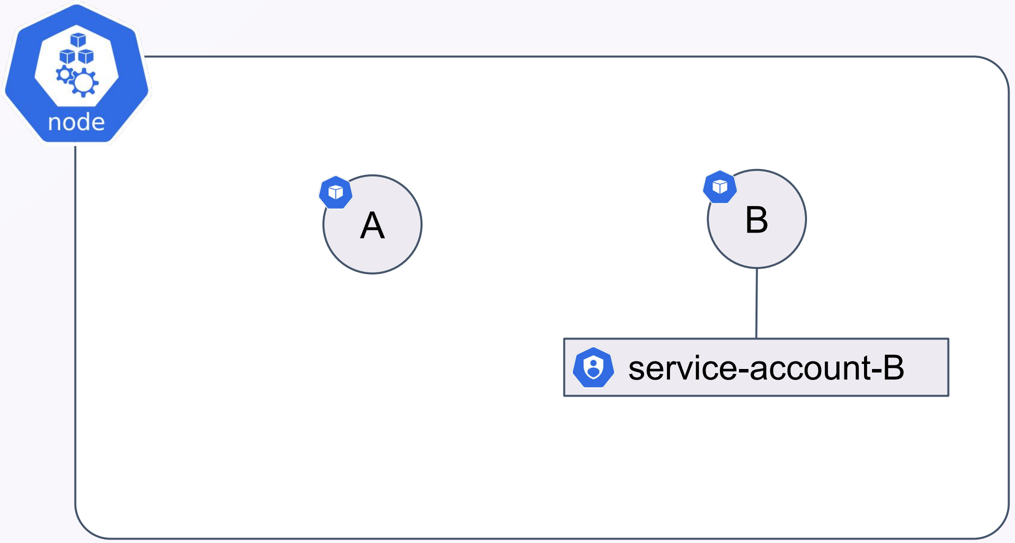
Modify domain of feasibility using the kubelet account



ATTACKING ISOLATION WITH KUBELET

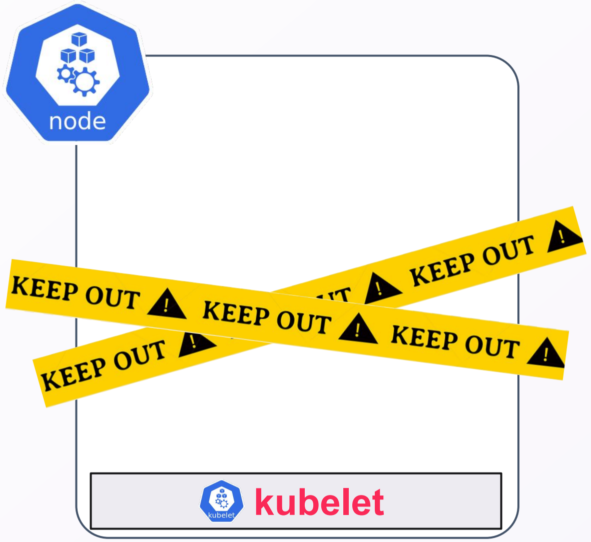
Modify domain of feasibility using the kubelet account

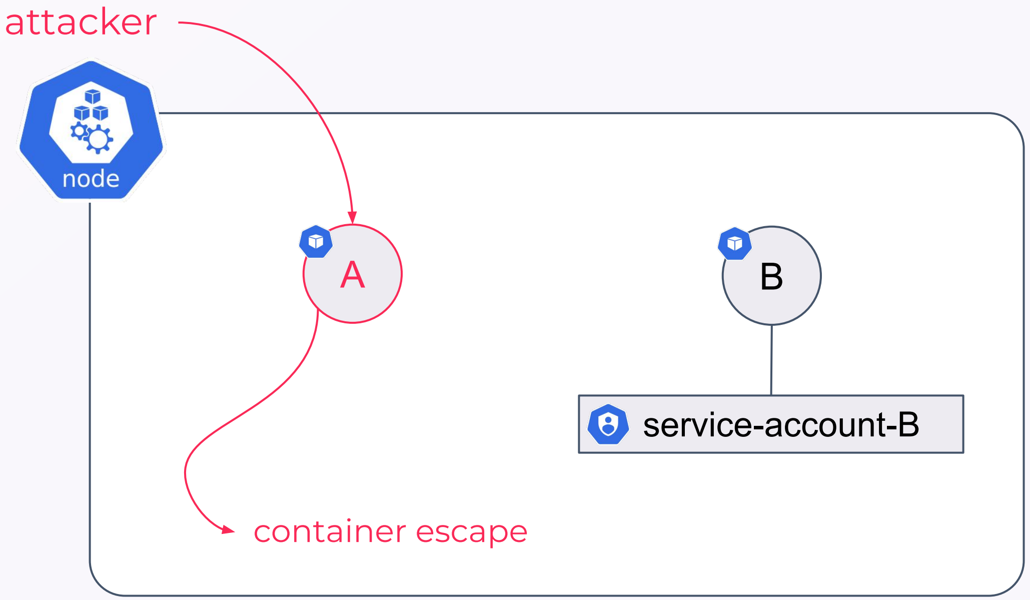




ATTACKING ISOLATION WITH KUBELET

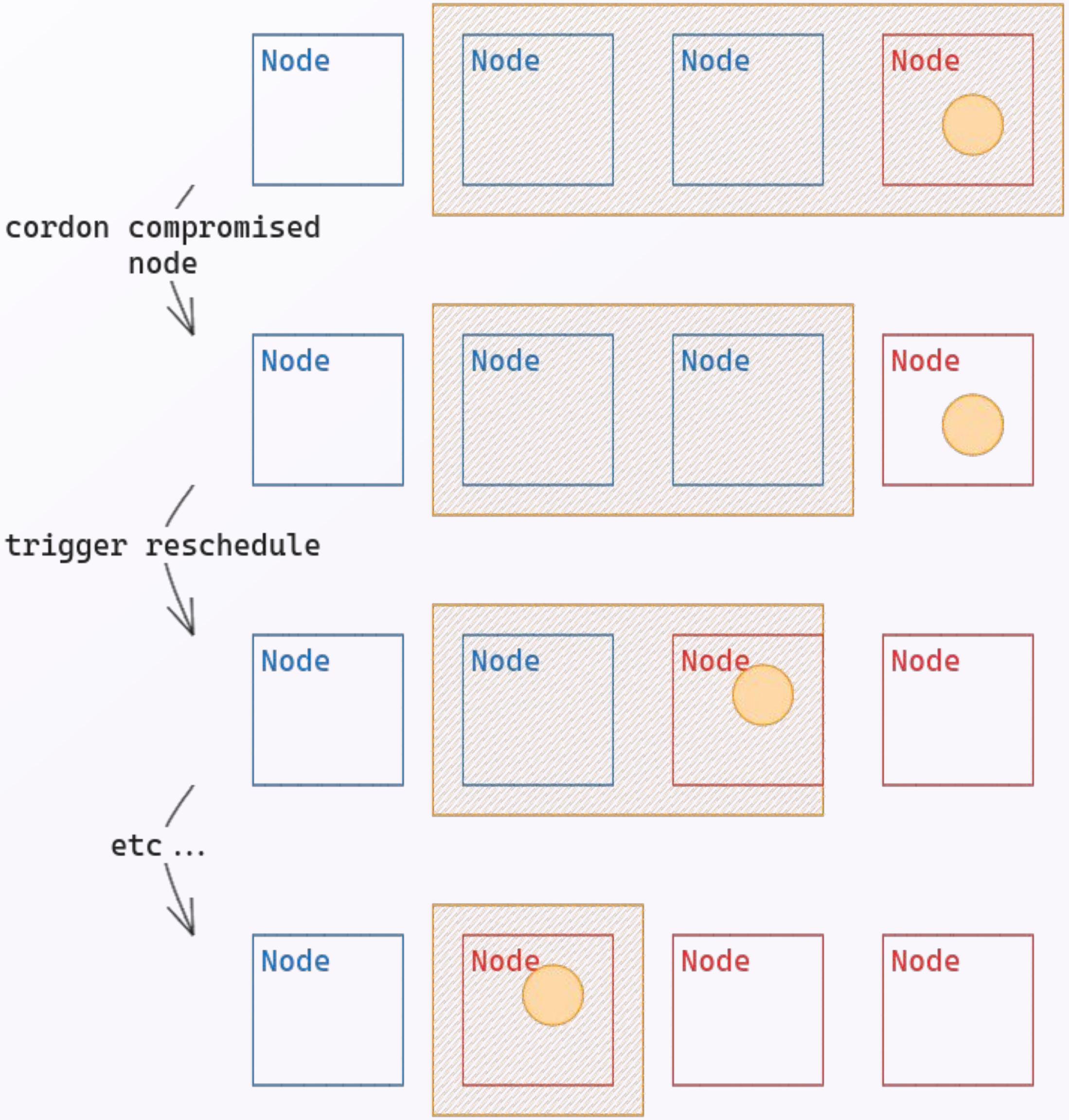
Modify domain of feasibility using the kubelet account





ATTACKING ISOLATION WITH KUBELET

Modify domain of feasibility using the kubelet account



ATTACKING ISOLATION WITH KUBELET

Wrap up

To implement node isolation:

- Use **node affinities + taints and tolerations**
- Use **NodeRestriction** admission plugin (enabled by default on AWS, GCP and Azure)
- Use protected labels with prefix: **node-restriction.kubernetes.io/**

In case of compromise:

- The domain of feasibility of the **vulnerable “entry” pod** can be compromised only using kubelet **privileges**
- If **no isolation** all nodes can be compromised

Agenda

01

Introduction

02

Kubernetes scheduler

03

Attacking isolation with kubelet

04

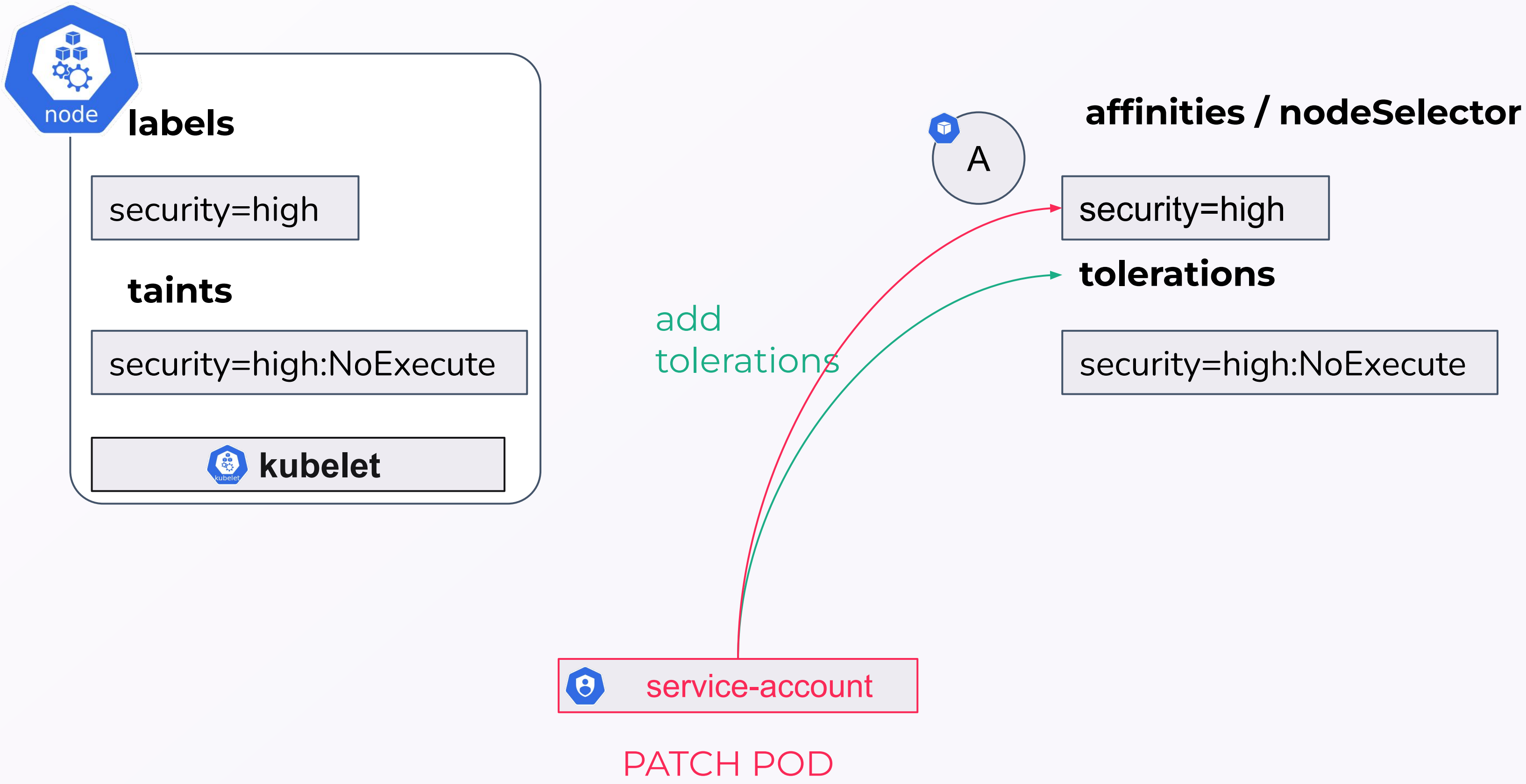
Attacking isolation with service account

05

Conclusion

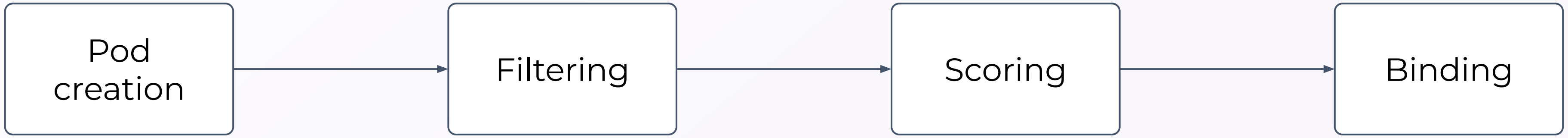
ATTACKING ISOLATION WITH SERVICE ACCOUNT

Filtering



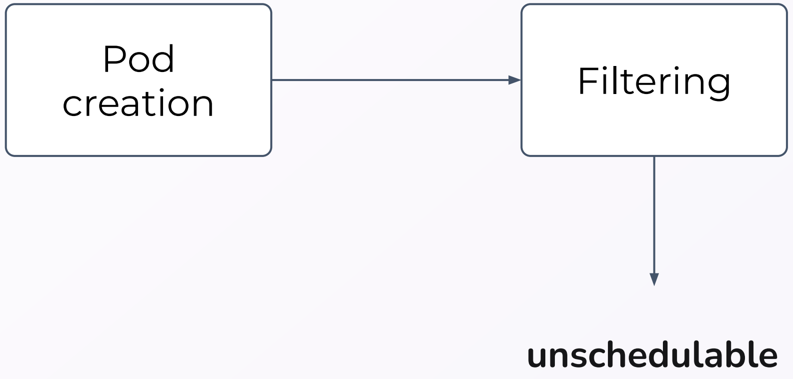
ATTACKING ISOLATION WITH SERVICE ACCOUNT

Scheduler



ATTACKING ISOLATION WITH SERVICE ACCOUNT

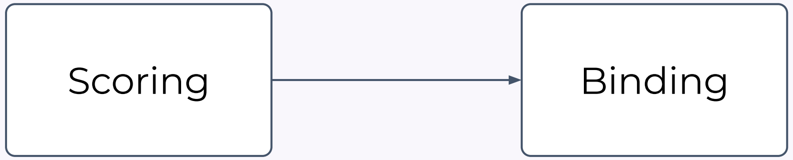
Scheduler



Pod creation

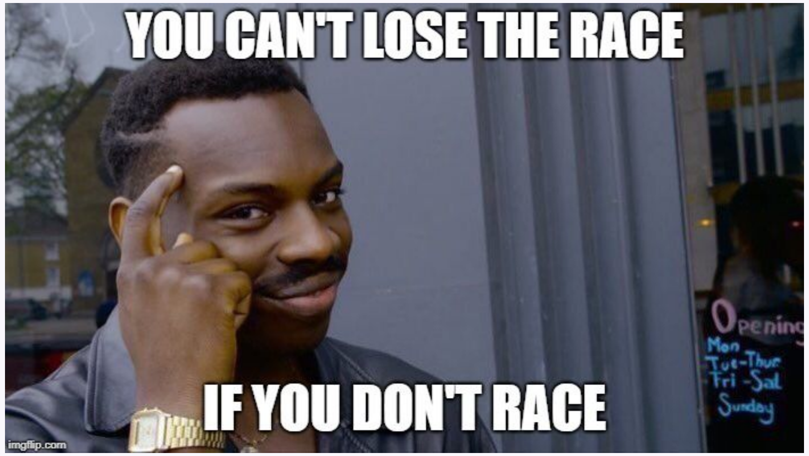
Filtering

unschedulable



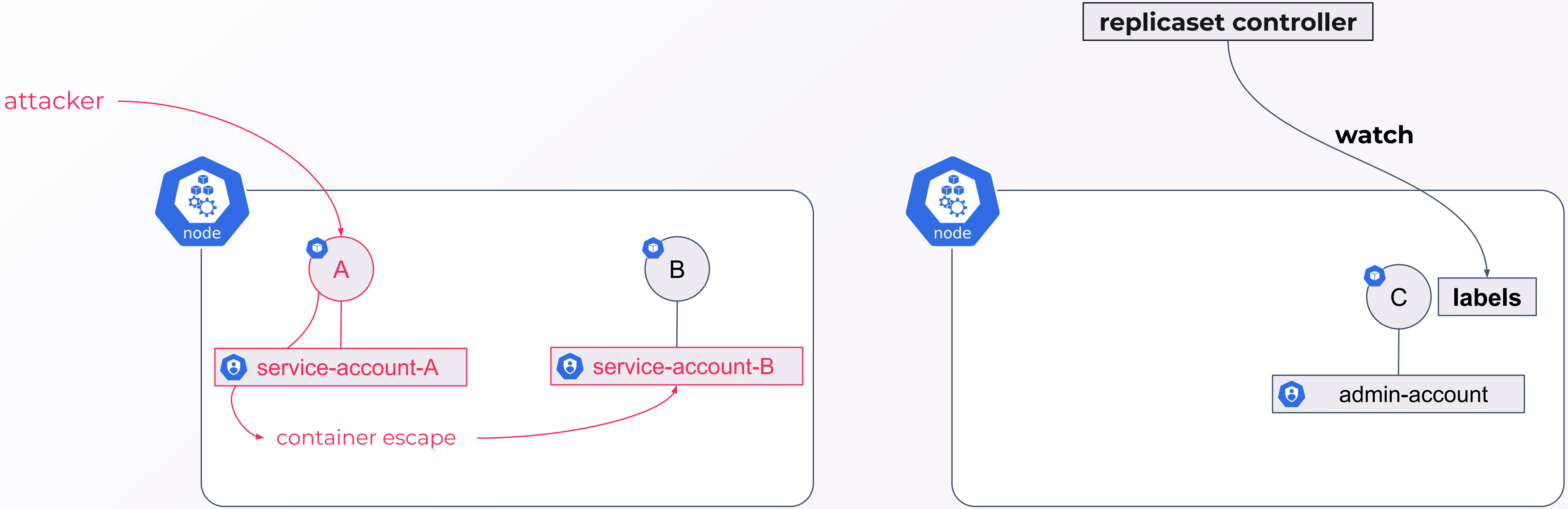
Scoring

Binding



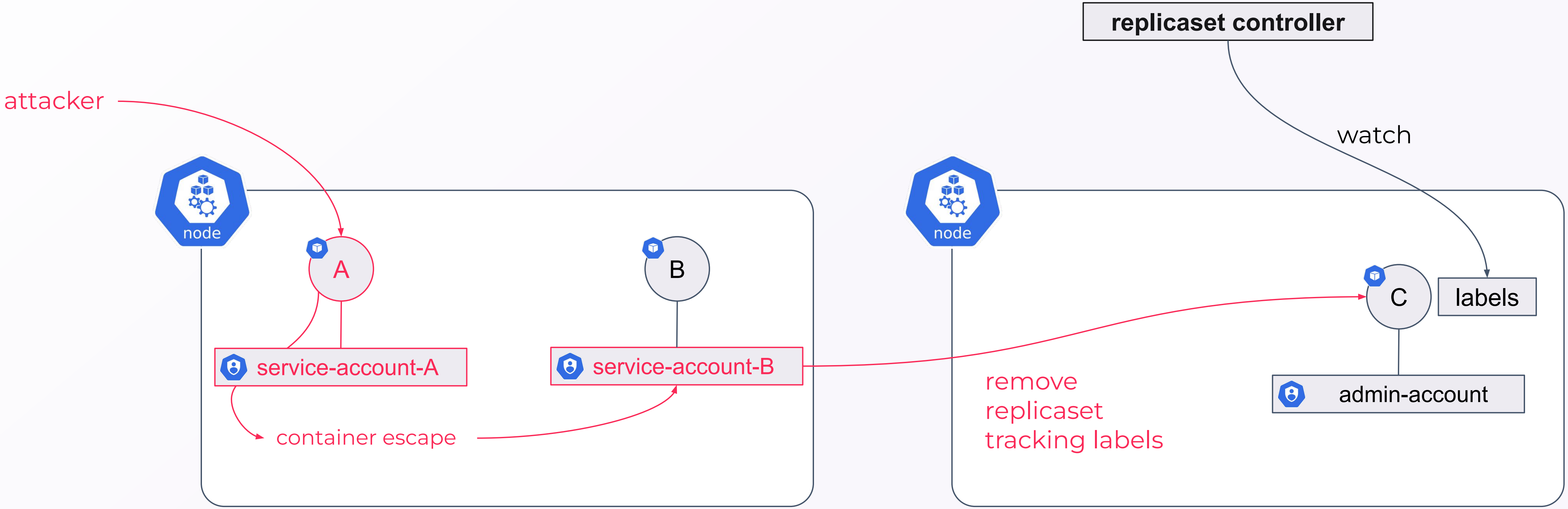
ATTACKING ISOLATION WITH SERVICE ACCOUNT

Using patch pod right



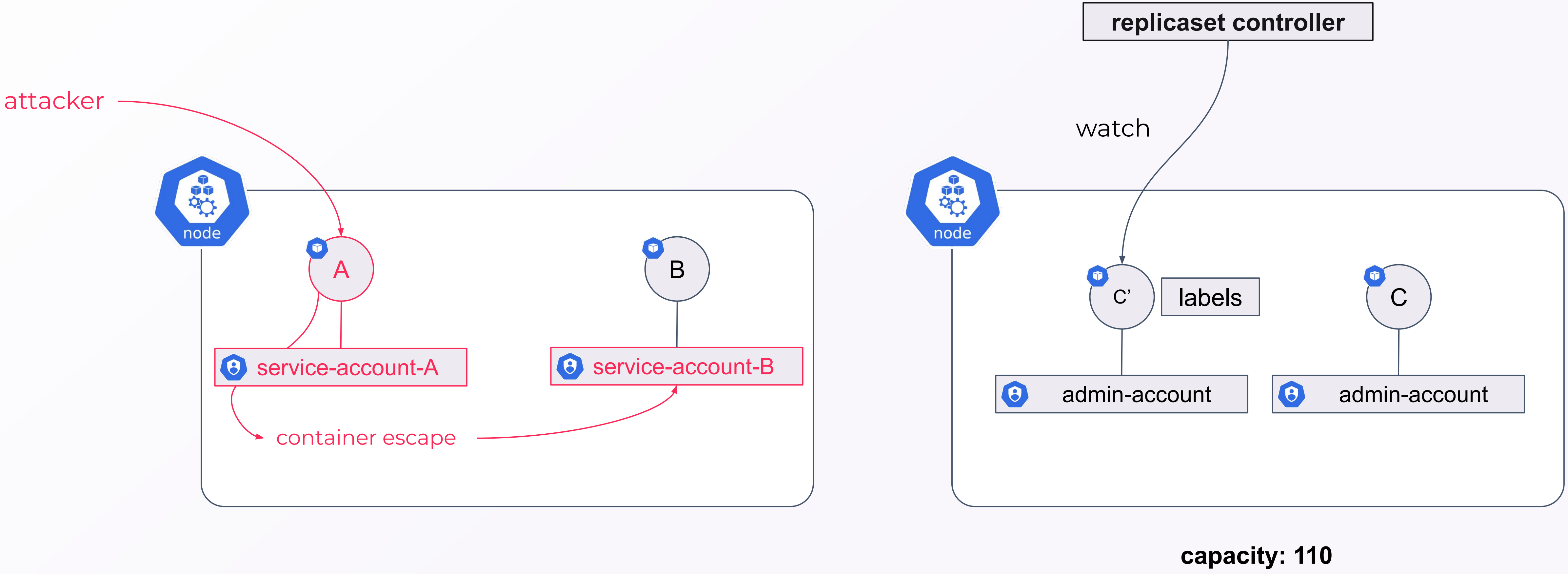
ATTACKING ISOLATION WITH SERVICE ACCOUNT

Using patch pod right



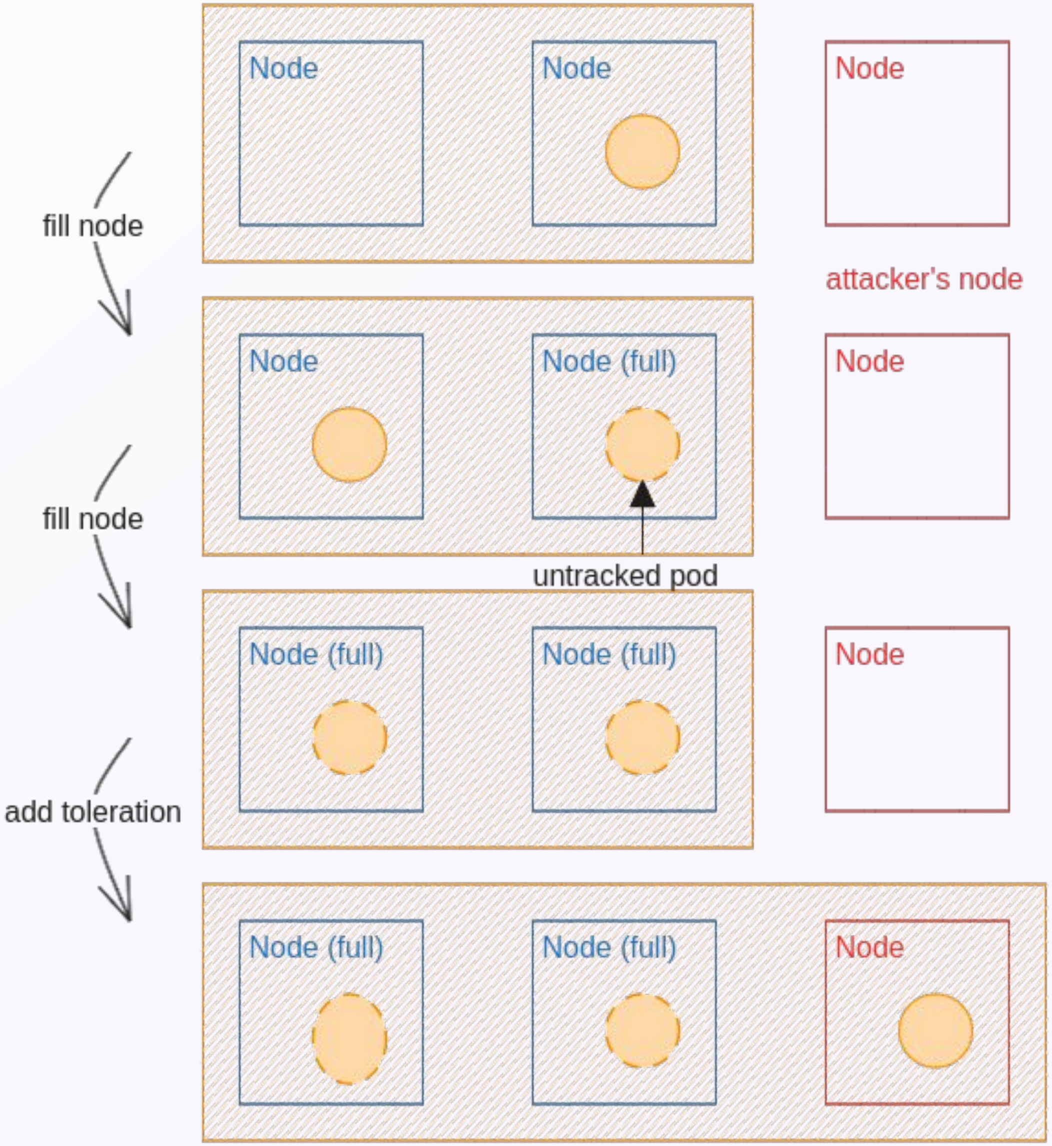
ATTACKING ISOLATION WITH SERVICE ACCOUNT

Using patch pod right



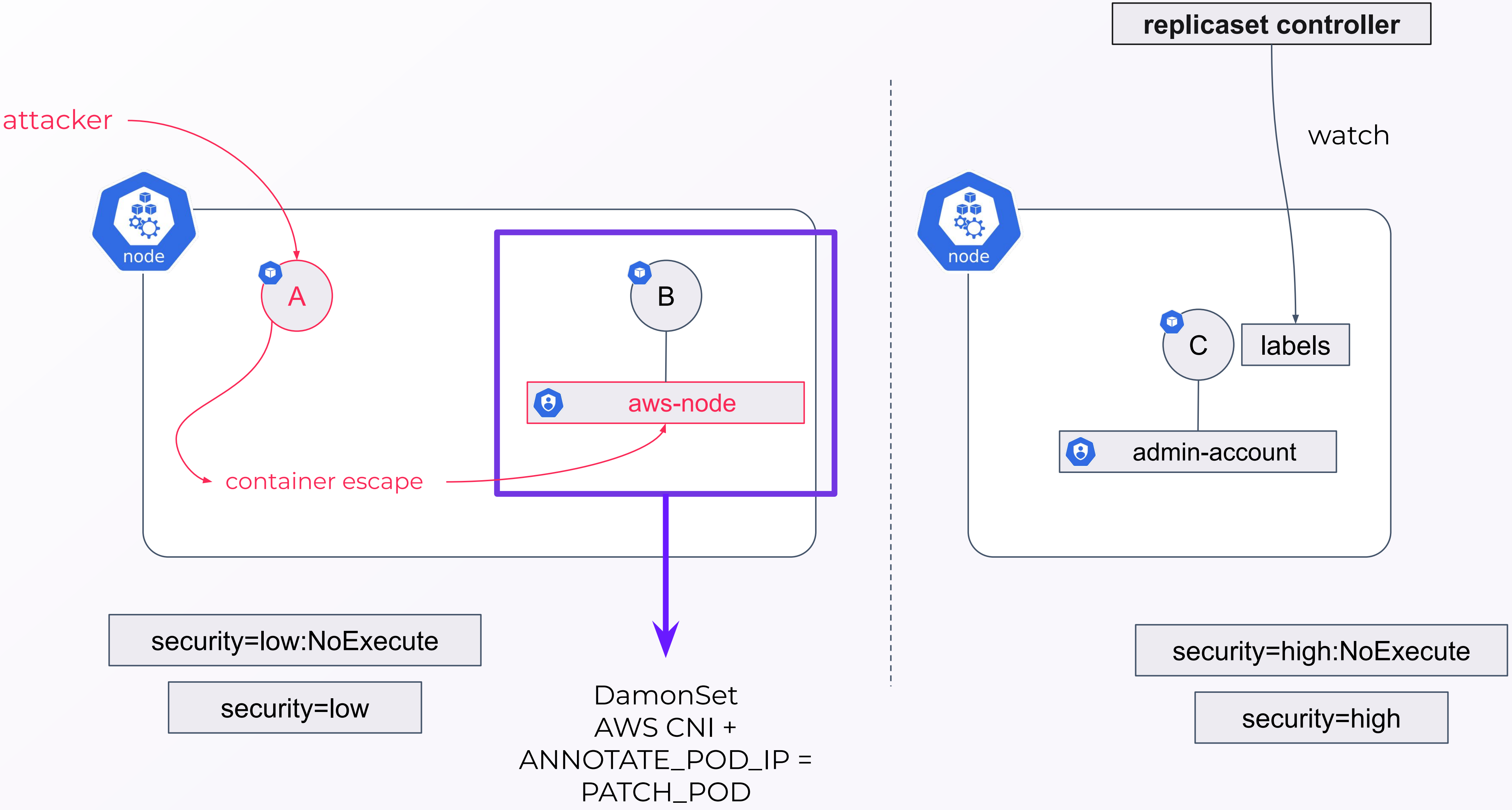
- When reaching max capacity **NodeResourcesFit** plugin filters out nodes !

Using patch pod right



ATTACKING ISOLATION WITH SERVICE ACCOUNT

DEMO



attacker

node

A

container escape

B

aws-node

security=low:NoExecute

security=low

DamonSet
AWS CNI +
ANNOTATE_POD_IP =
PATCH_POD

replicaset controller

watch

node

C

labels

admin-account

security=high:NoExecute

security=high

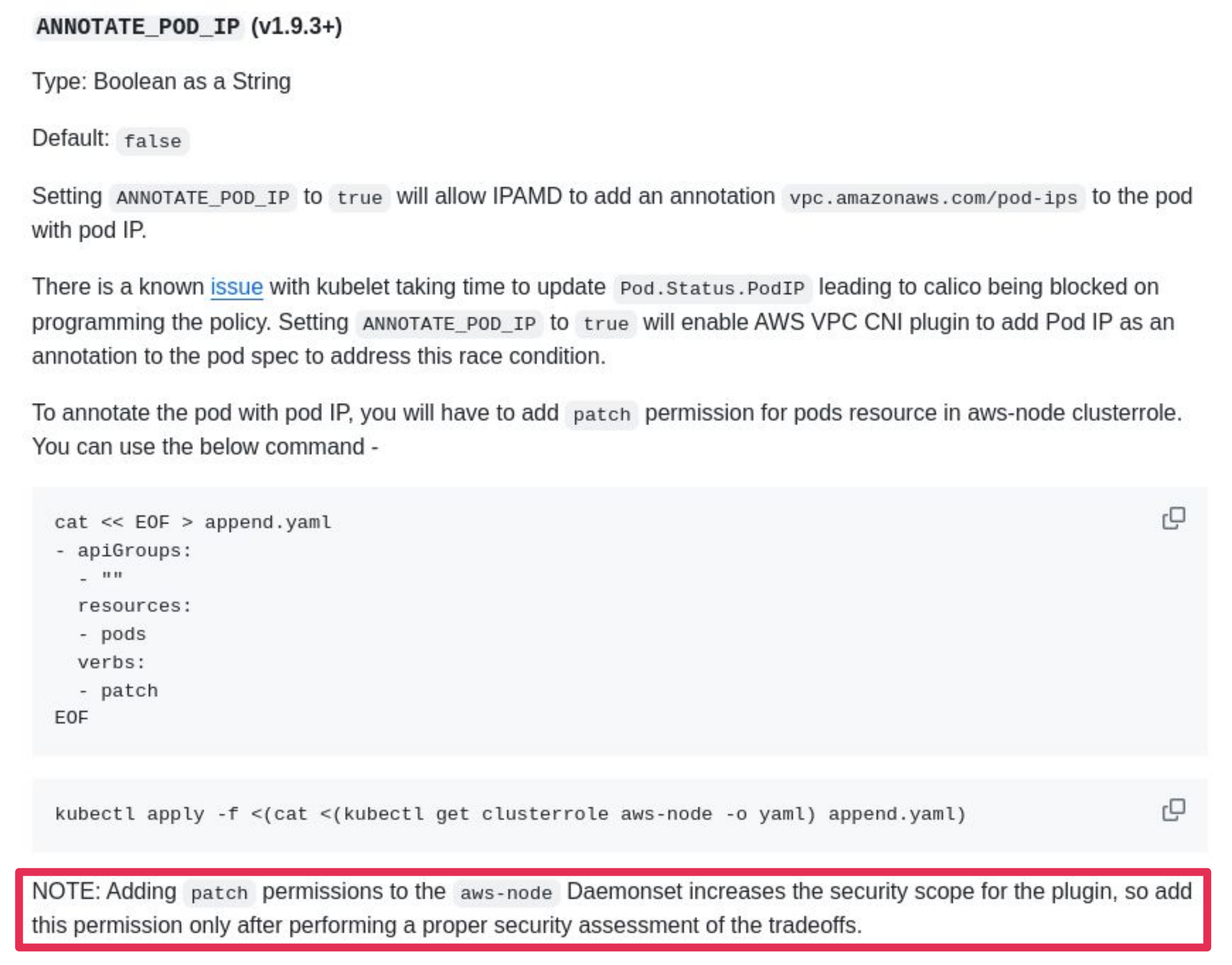
ATTACKING ISOLATION WITH SERVICE ACCOUNT

DEMO

ATTACKING ISOLATION WITH SERVICE ACCOUNT

Some thoughts on node isolation on managed Kubernetes

- Cloud providers do not seems to consider node isolation flaws as security issues
- It may be better to create multiple clusters



`ANNOTATE_POD_IP` (v1.9.3+)

Type: Boolean as a String

Default: `false`

Setting `ANNOTATE_POD_IP` to `true` will allow IPAMD to add an annotation `vpc.amazonaws.com/pod-ips` to the pod with pod IP.

There is a known issue with kubelet taking time to update `Pod.Status.PodIP` leading to calico being blocked on programming the policy. Setting `ANNOTATE_POD_IP` to `true` will enable AWS VPC CNI plugin to add Pod IP as an annotation to the pod spec to address this race condition.

To annotate the pod with pod IP, you will have to add `patch` permission for pods resource in aws-node clusterrole. You can use the below command -

```
cat << EOF > append.yaml
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - patch
EOF
```

```
kubectl apply -f <(cat <(kubectl get clusterrole aws-node -o yaml) append.yaml)
```

NOTE: Adding `patch` permissions to the `aws-node` Daemonset increases the security scope for the plugin, so add this permission only after performing a proper security assessment of the tradeoffs.

Agenda

01

Introduction

02

Kubernetes scheduler

03

Attacking isolation with kubelet

04

Attacking isolation with service account

05

Conclusion

CONCLUSION

Wrap up

To implement node isolation:

- Use **node affinities + taints and tolerations**
- Use **NodeRestriction** admission plugin (enabled by default on AWS, GCP and Azure)
- Use protected labels with prefix: **node-restriction.kubernetes.io/**

In case of compromise:

- The domain of feasibility of the **vulnerable "entry" pod** can be compromised without **privileges**
- If **no isolation** all nodes can be compromised
- **PATCH Pod** permission on a Pod is enough to move a pod around if not using protected labels

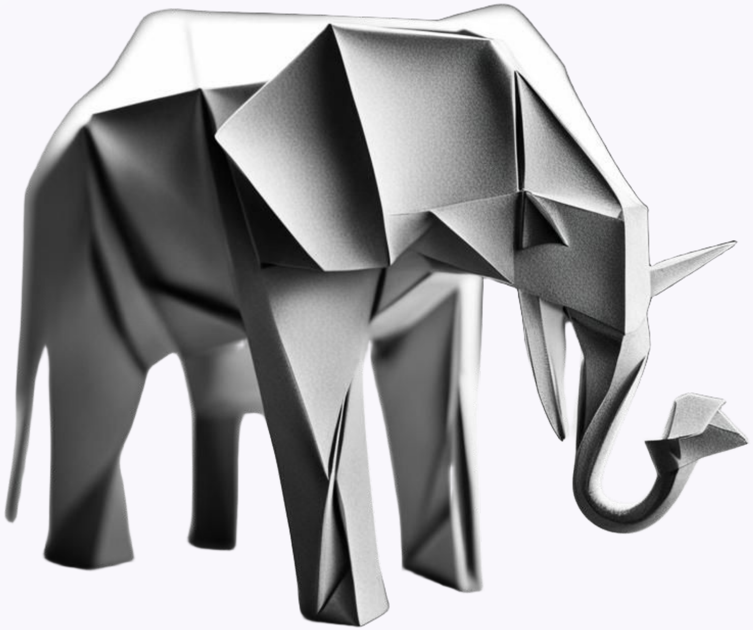
CONCLUSION

Ready for next elephant in the room ?

- How nodes are created and deleted in cloud managed environment ?

PADOK
SECURITY

https://security.padok.fr/blog



Cloud controller manager

Thank you !

PΛDOK