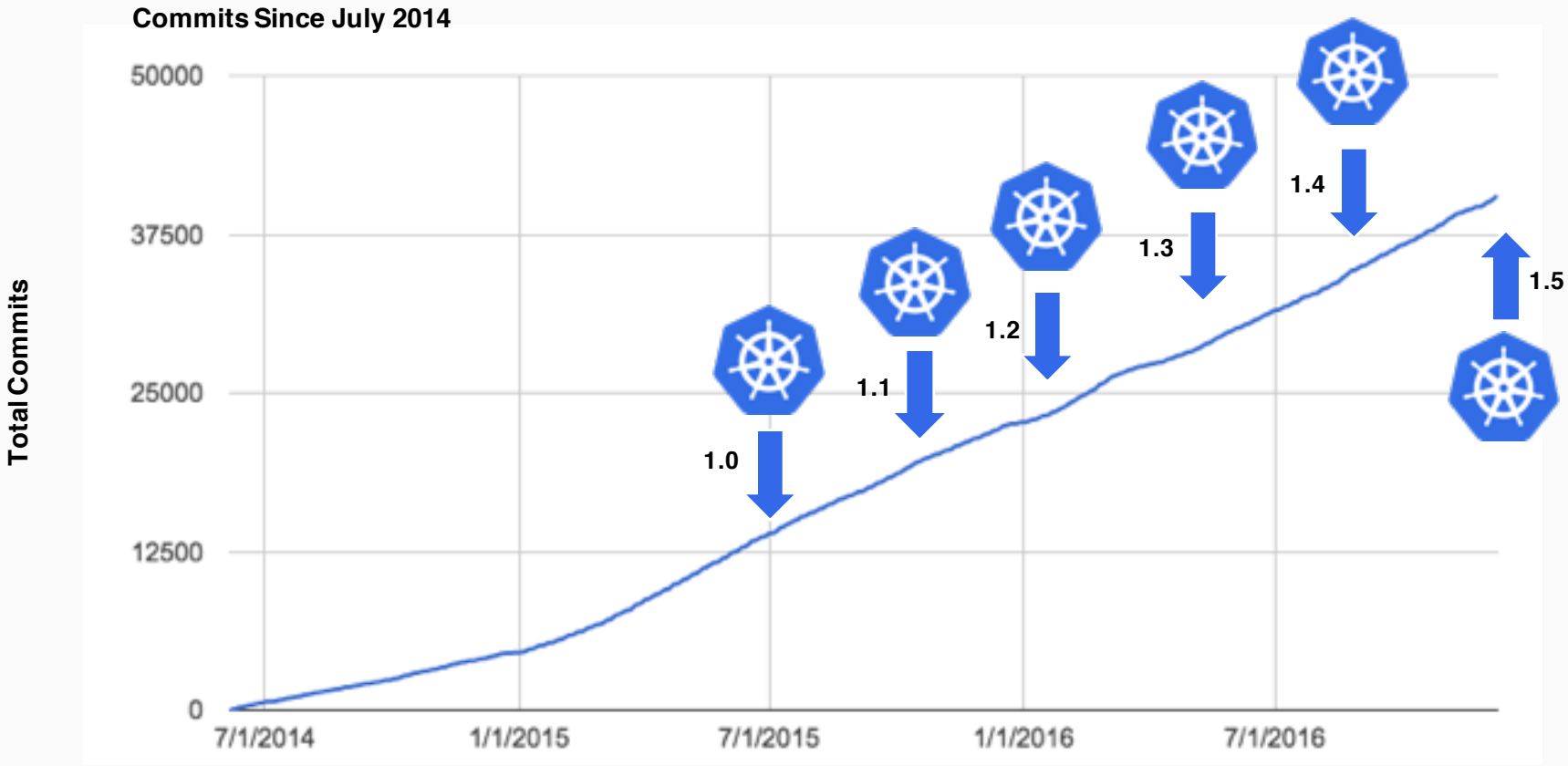
David Aronchick

Kubernetes 1.5 and Beyond

David Aronchick
Product Manager at Google
Container Engine & Kubernetes

Velocity


Commits Since July 2014
Total Commits
50000
37500
25000
12500
0
7/1/2014
1/1/2015
7/1/2015
1/1/2016
7/1/2016
1.0
1.1
1.2
1.3
1.4
1.5

Adoption

~4k Commits in 1.5

+25% Unique Contributors

Top 0.01% of all Github Projects

3500+ External Projects Based on K8s

Companies Contributing

Companies Using

Give Everyone the Power to Run Agile, Reliable, Distributed Systems at Scale

Introducing Kubernetes 1.5

Kubernetes 1.5 Enterprise Highlights

Simple Setup (including multiple clusters!)

Sophisticated Scheduling

Network policy

Helm for application installation

Simplified Setup

Problem: Setting up a Kubernetes cluster is hard

Today:

Use kube-up.sh (and hope you don't have to customize)
Compile from HEAD and manually address security
Use a third-party tool (some of which are great!)

Simplified Setup

Solution: kubeadm!

Simplified Setup

Solution: kubeadm!

```
master.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
master.myco.com#  kubeadm init
```

Simplified Setup

Solution: kubeadm!

```
master.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
master.myco.com#  kubeadm init
Kubernetes master initialized successfully!
You can now join any number of nodes by running the following command:
kubeadm join --token 48b69e.b61e2d0dd5c 10.140.0.3
```

Simplified Setup

Solution: kubeadm!

```
master.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
master.myco.com#  kubeadm init
Kubernetes master initialized successfully!
You can now join any number of nodes by running the following command:
kubeadm join --token 48b69e.b61e2d0dd5c 10.140.0.3

node-01.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
node-01.myco.com#  kubeadm join --token 48b69e.b61e2d0dd5c 10.140.0.3
```

Simplified Setup

Solution: kubeadm!

```
master.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
master.myco.com#  kubeadm init
Kubernetes master initialized successfully!
You can now join any number of nodes by running the following command:
kubeadm join --token 48b69e.b61e2d0dd5c 10.140.0.3

node-01.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
node-01.myco.com#  kubeadm join --token 48b69e.b61e2d0dd5c 10.140.0.3
Node join complete.
```

Simplified Setup

Solution: kubeadm!

```
master.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
master.myco.com#  kubeadm init
Kubernetes master initialized successfully!
You can now join any number of nodes by running the following command:
kubeadm join --token 48b69e.b61e2d0dd5c 10.140.0.3

node-01.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
node-01.myco.com#  kubeadm join --token 48b69e.b61e2d0dd5c 10.140.0.3
Node join complete.

master.myco.com#  kubectl apply -f https://git.io/weave-kube
```

Simplified Setup

Solution: kubeadm!

```
master.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
master.myco.com#  kubeadm init
Kubernetes master initialized successfully!
You can now join any number of nodes by running the following command:
kubeadm join --token 48b69e.b61e2d0dd5c 10.140.0.3

node-01.myco.com#  apt-get install -y kubelet kubeadm kubectl kubernetes-cni
node-01.myco.com#  kubeadm join --token 48b69e.b61e2d0dd5c 10.140.0.3
Node join complete.

master.myco.com#  kubectl apply -f https://git.io/weave-kube
Network setup complete.
```

Simplified Setup: Federation Edition

Problem: Using *multiple*-clusters is hard

Today:

Clusters as multiple independent silos

Use Kubernetes federation from scratch

Simplified Setup: Federation Edition

Solution: kubefed!

Simplified Setup: Federation Edition

Solution: kubefed!

```
dc1.example.com# kubefed init fellowship --host-cluster-context=rivendell --
dns-zone-name="example.com"
```

Simplified Setup: Federation Edition

Solution: kubefed!

```
dc1.example.com# kubefed init fellowship --host-cluster-context=rivendell --
dns-zone-name="example.com"
Federation “Rivendell” created.
```

Simplified Setup: Federation Edition

Solution: kubefed!

```
dc1.example.com# kubefed init fellowship --host-cluster-context=rivendell --
dns-zone-name="example.com"
Federation "Rivendell" created.

dc1.example.com# kubectl config use-context fellowship
```

Simplified Setup: Federation Edition

Solution: kubefed!

```
dc1.example.com# kubefed init fellowship --host-cluster-context=rivendell --
dns-zone-name="example.com"
Federation “Rivendell” created.

dc1.example.com# kubectl config use-context fellowship
switched to context "Fellowship”
```

Simplified Setup: Federation Edition

Solution: kubefed!

```
dc1.example.com# kubefed init fellowship --host-cluster-context=rivendell --
dns-zone-name="example.com"
Federation “Rivendell” created.

dc1.example.com# kubectl config use-context fellowship
switched to context "Fellowship”

dc1.example.com# kubefed join gondor --host-cluster-context=fellowship
```

Simplified Setup: Federation Edition

Solution: kubefed!

```
dc1.example.com# kubefed init fellowship --host-cluster-context=rivendell --
dns-zone-name="example.com"
Federation “Rivendell” created.

dc1.example.com# kubectl config use-context fellowship
switched to context "Fellowship”

dc1.example.com# kubefed join gondor --host-cluster-context=fellowship
Cluster “Gonder” joined to federation “Rivendell”.
```

Simplified Setup: Federation Edition

Solution: kubefed!

```
dc1.example.com# kubefed init fellowship --host-cluster-context=rivendell --
dns-zone-name="example.com"
Federation “Rivendell” created.

dc1.example.com# kubectl config use-context fellowship
switched to context "Fellowship”

dc1.example.com# kubefed join gondor --host-cluster-context=fellowship
Cluster “Gonder” joined to federation “Rivendell”.

dc1.example.com# kubectl create -f multi-cluster-deployment.yml
```

Simplified Setup: Federation Edition

Solution: kubefed!

```
dc1.example.com# kubefed init fellowship --host-cluster-context=rivendell --
dns-zone-name="example.com"
Federation “Rivendell” created.

dc1.example.com# kubectl config use-context fellowship
switched to context "Fellowship”

dc1.example.com# kubefed join gondor --host-cluster-context=fellowship
Cluster “Gonder” joined to federation “Rivendell”.

dc1.example.com# kubectl create -f multi-cluster-deployment.yml
deployment "multi-cluster-deployment" created
```

Sophisticated Scheduling

Problem: Deploying and managing workloads on large, heterogenous clusters is hard

Today:

Liberal use of labels (and keeping your team in sync)

Manual tooling

Didn't you use Kubernetes to avoid this?

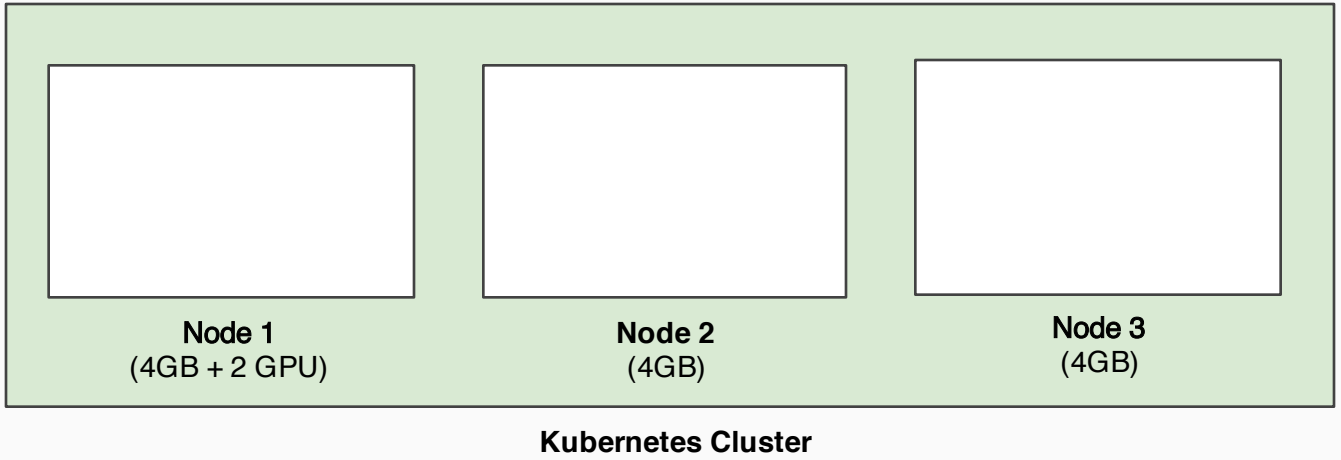
Sophisticated Scheduling

Solution: Sophisticated Scheduling!

Taints/tolerations
Forgiveness
Disruption budget

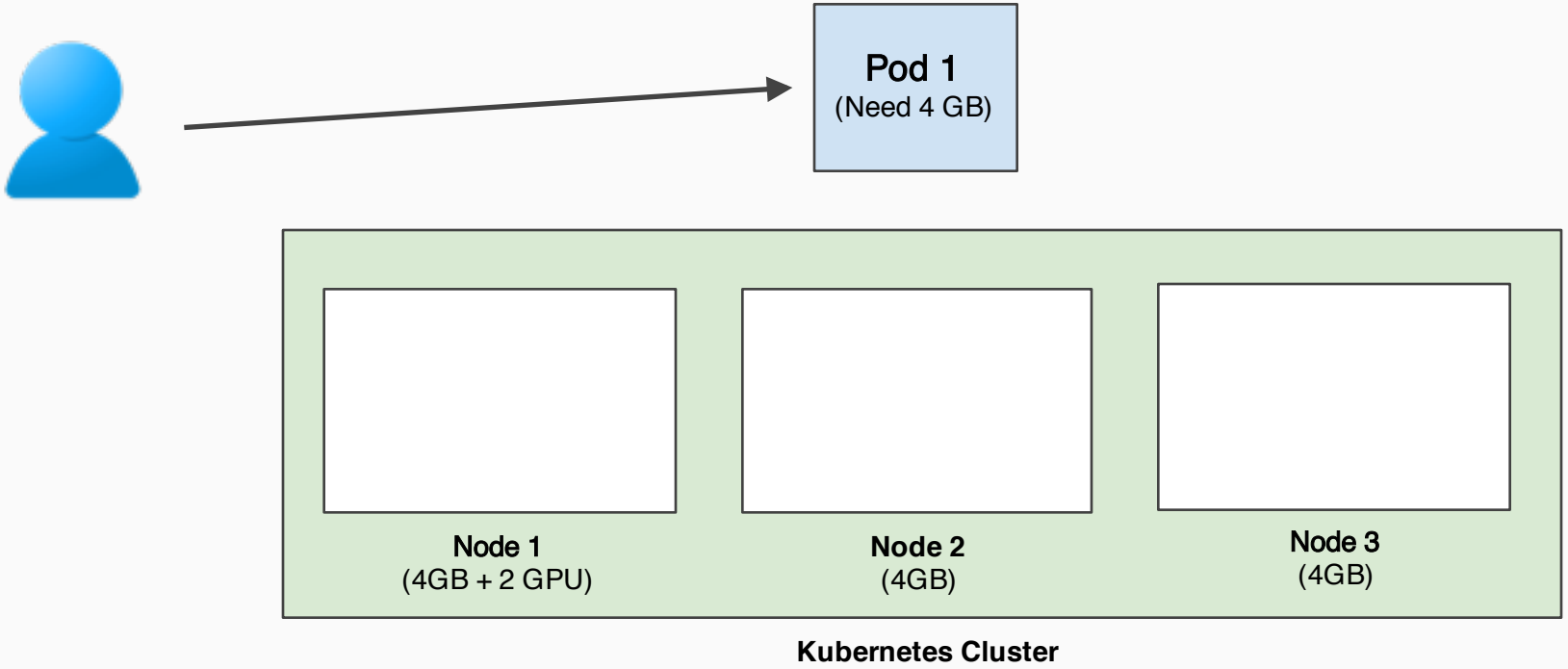
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



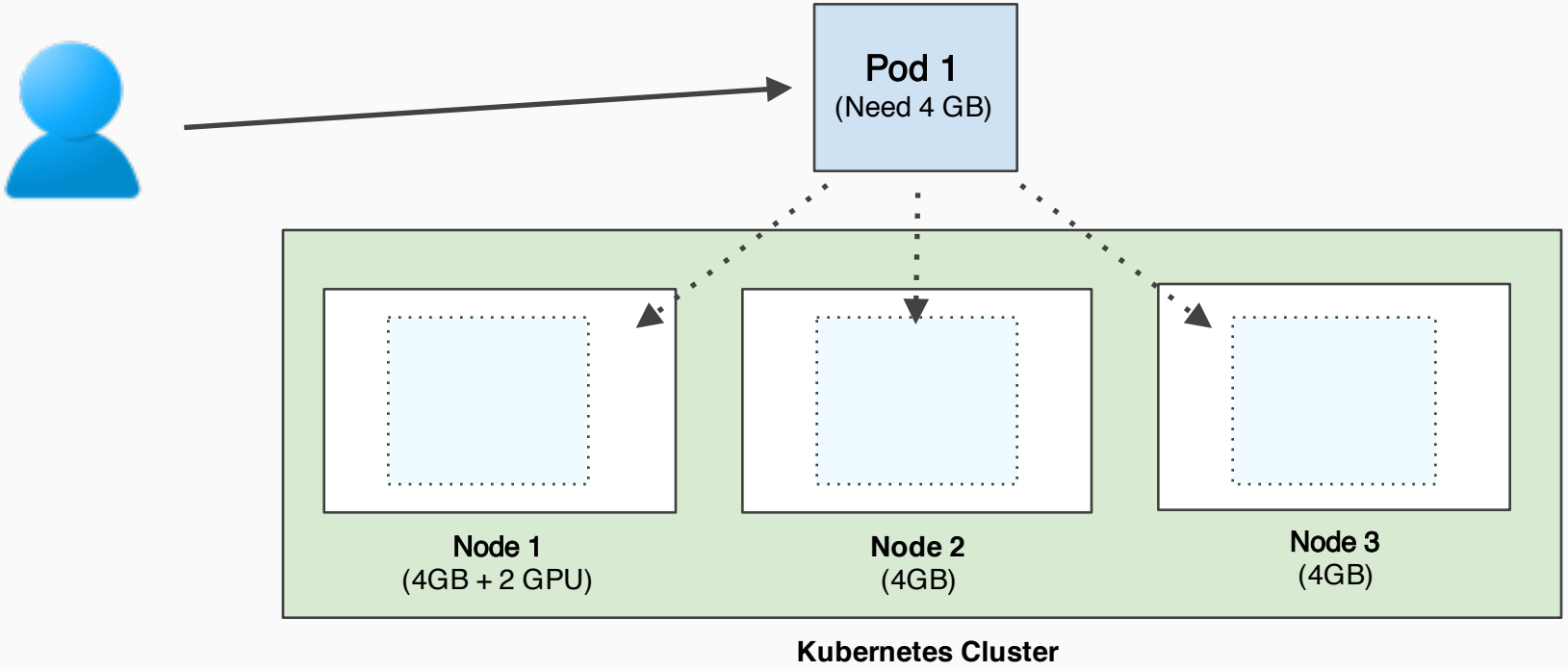
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



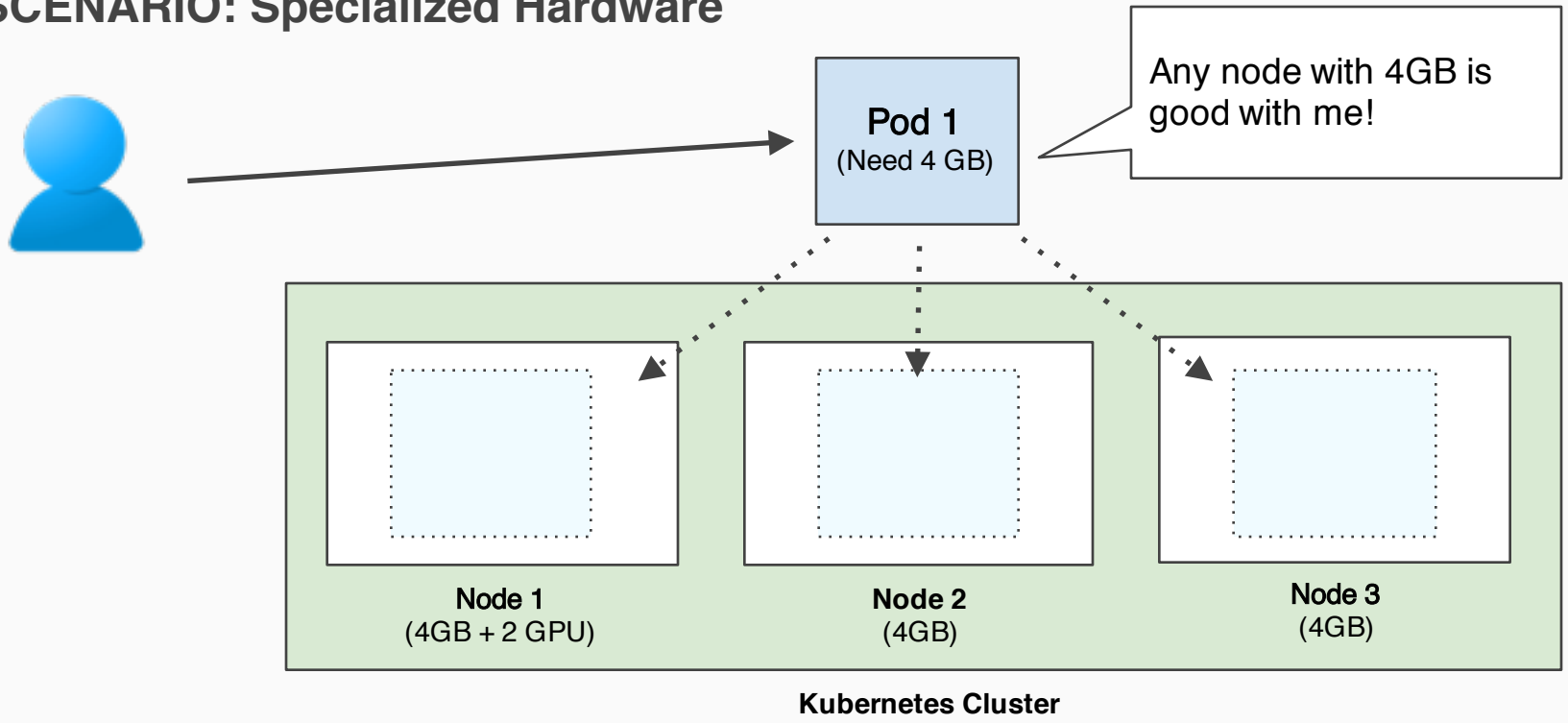
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



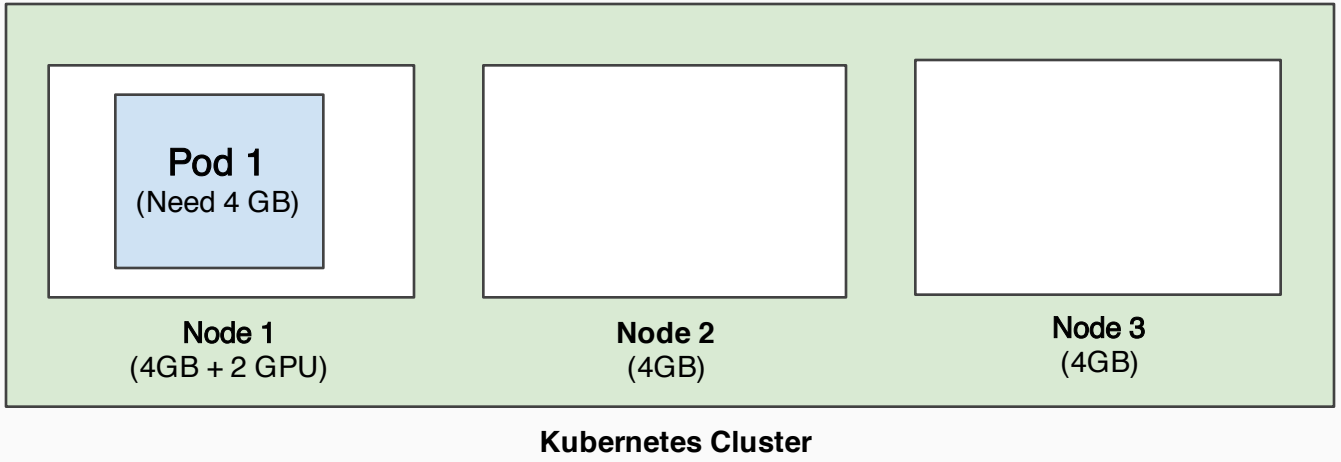
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



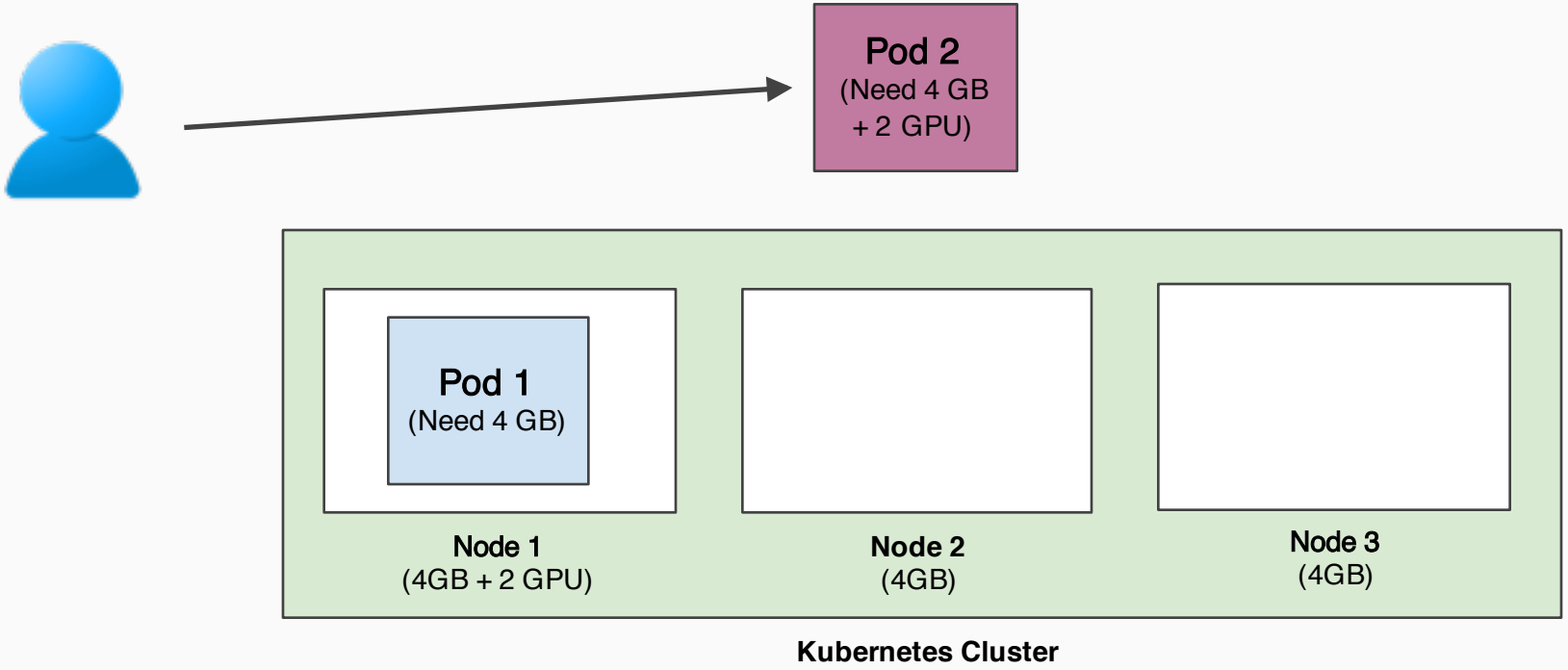
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



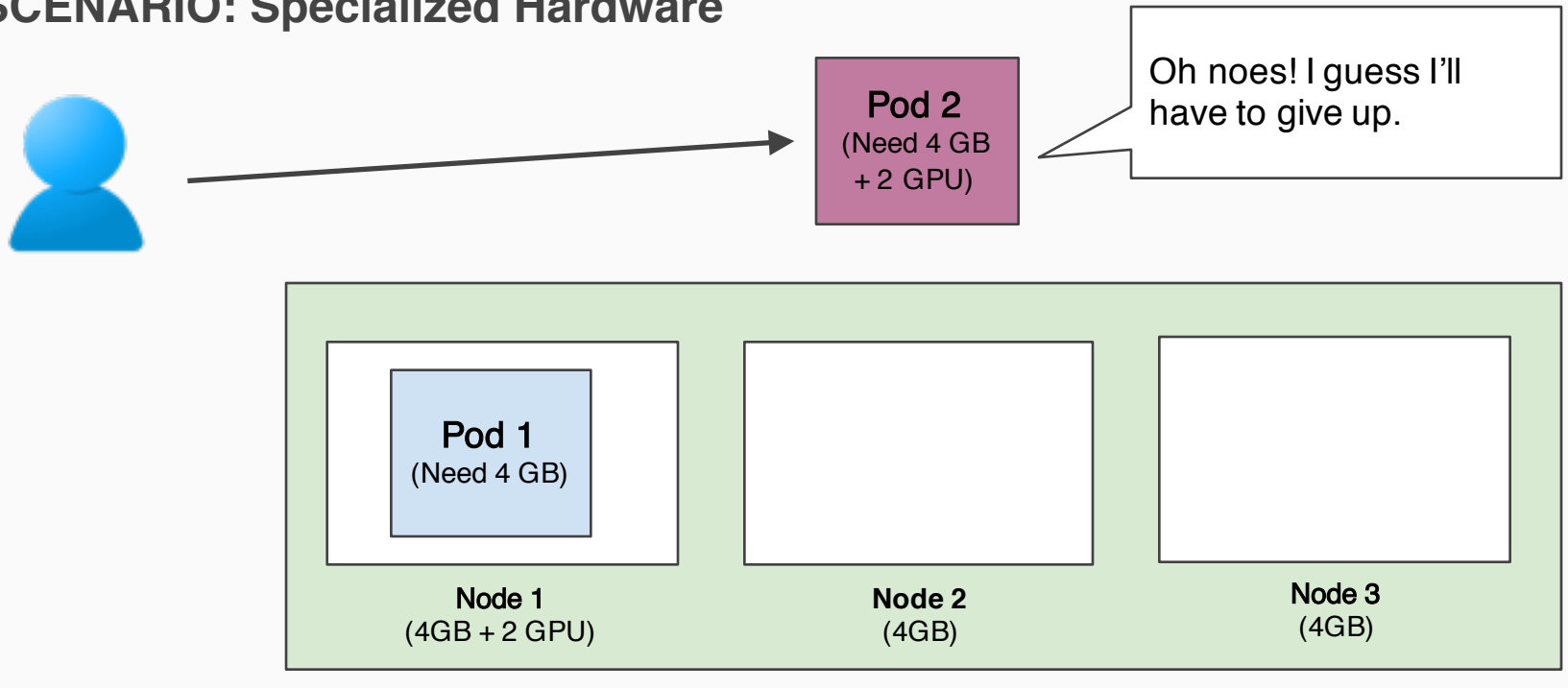
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



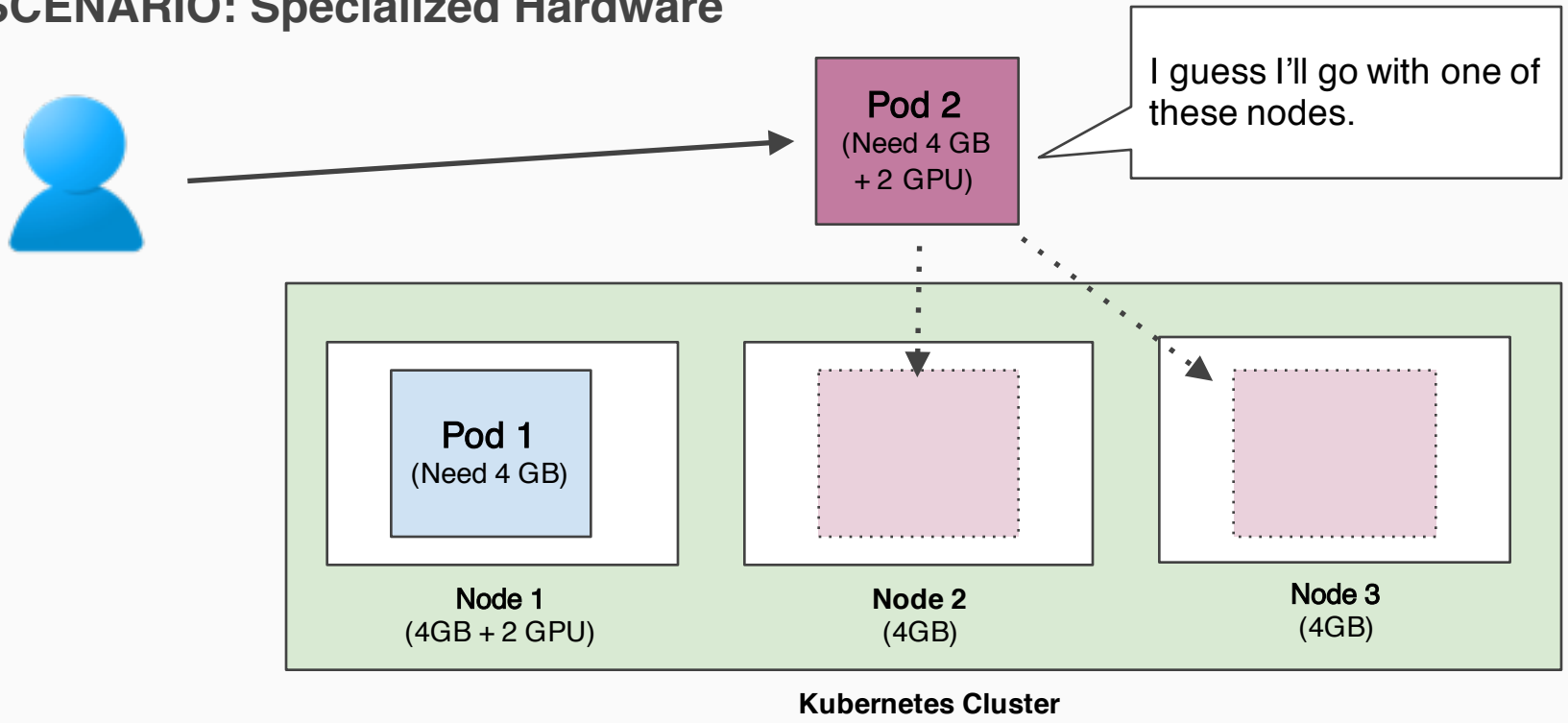
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



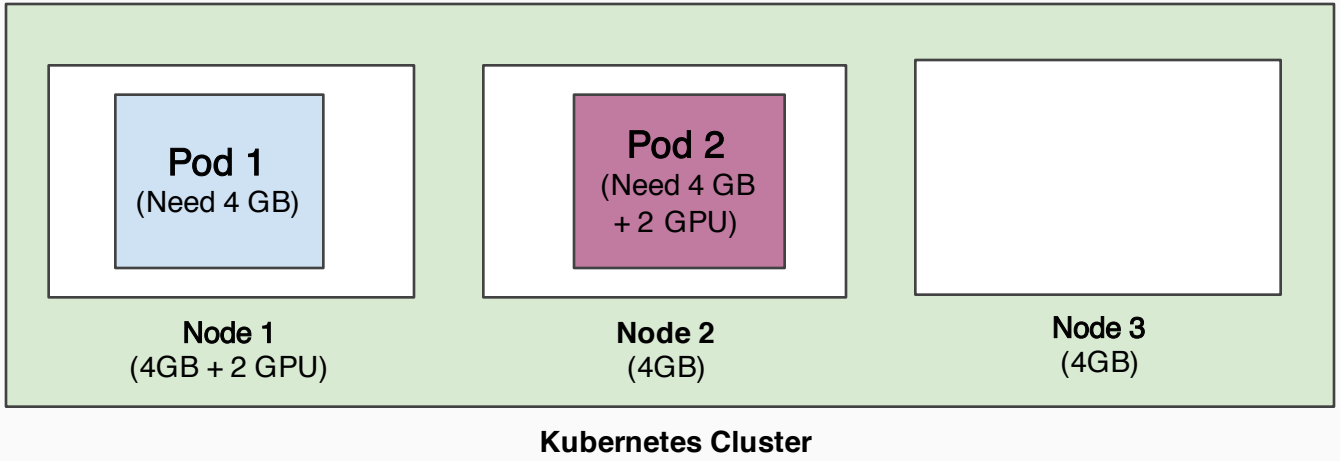
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



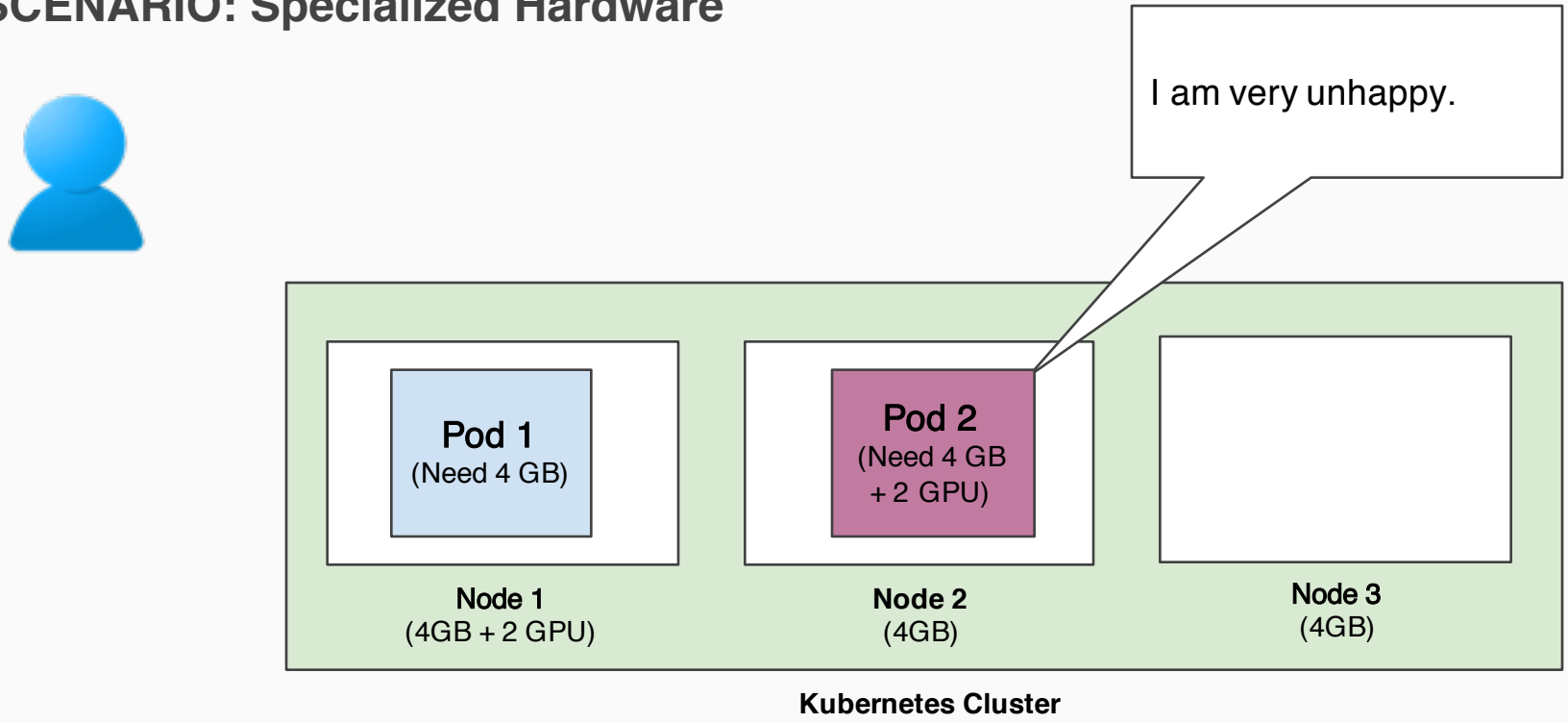
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



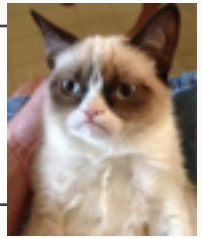
Sophisticated Scheduling: Taints/Toleration

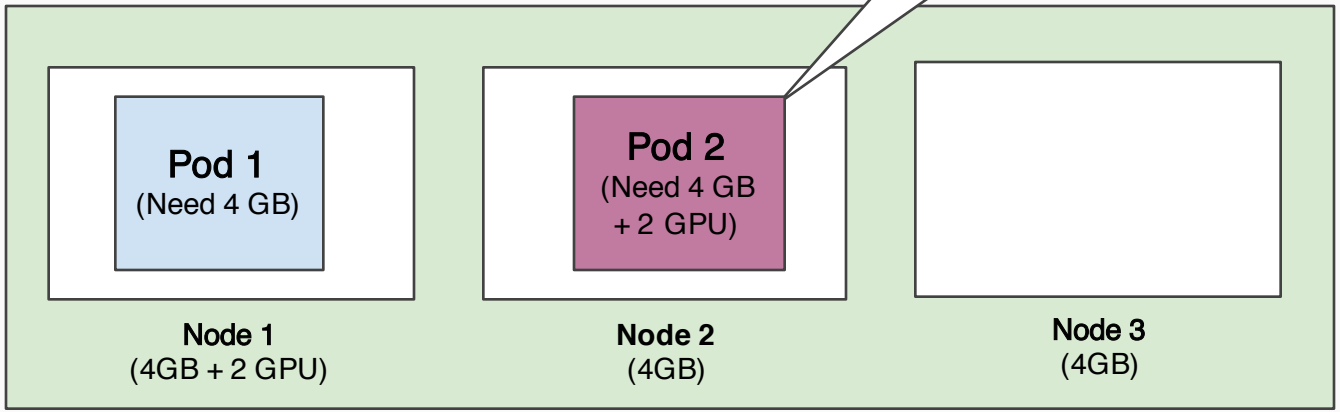
SCENARIO: Specialized Hardware



Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware

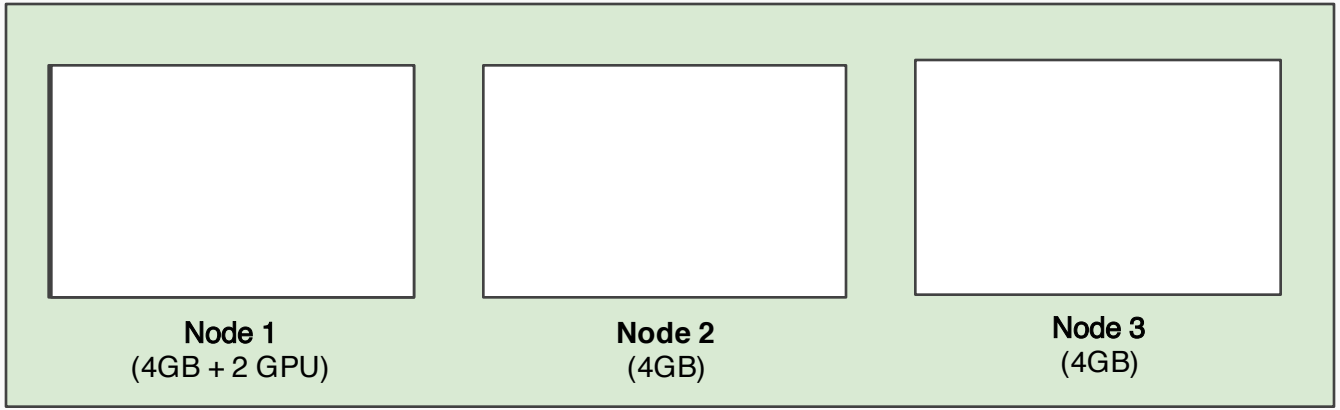




Sophisticated Scheduling: Taints/Toleration

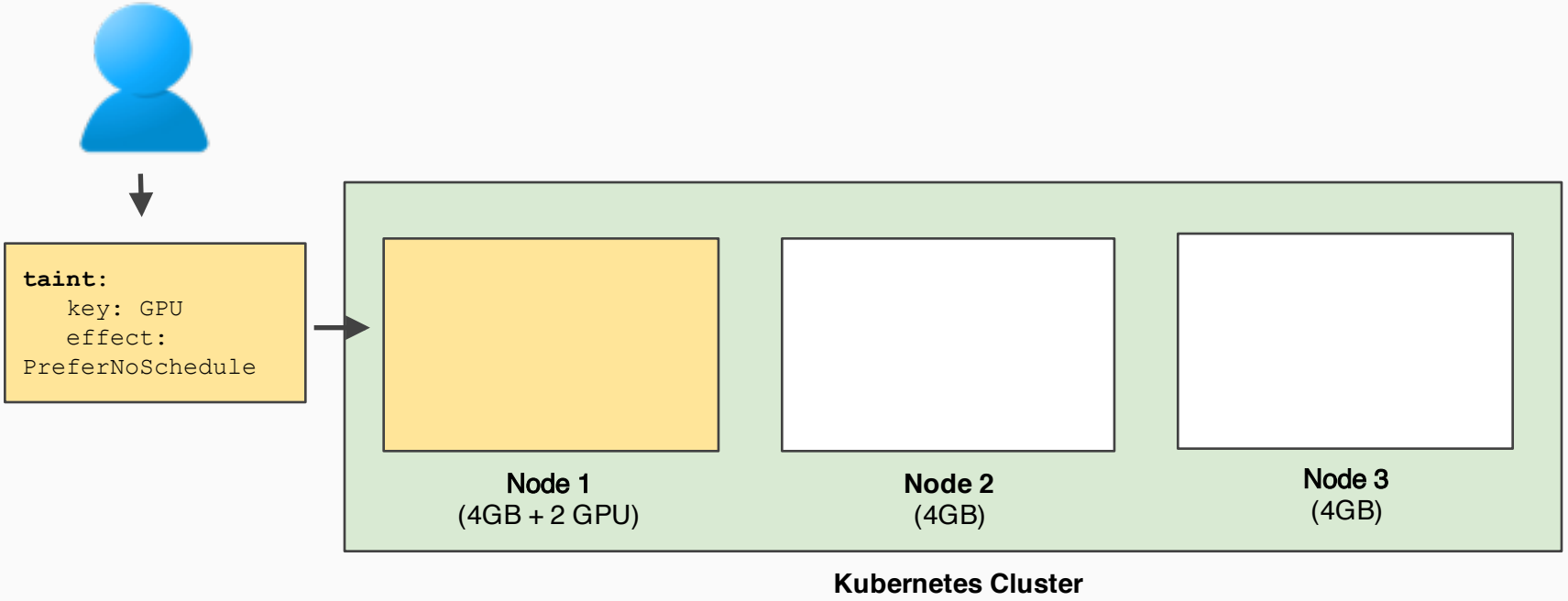
SCENARIO: Specialized Hardware


```
taint:
   key: GPU
   effect:
PreferNoSchedule
```



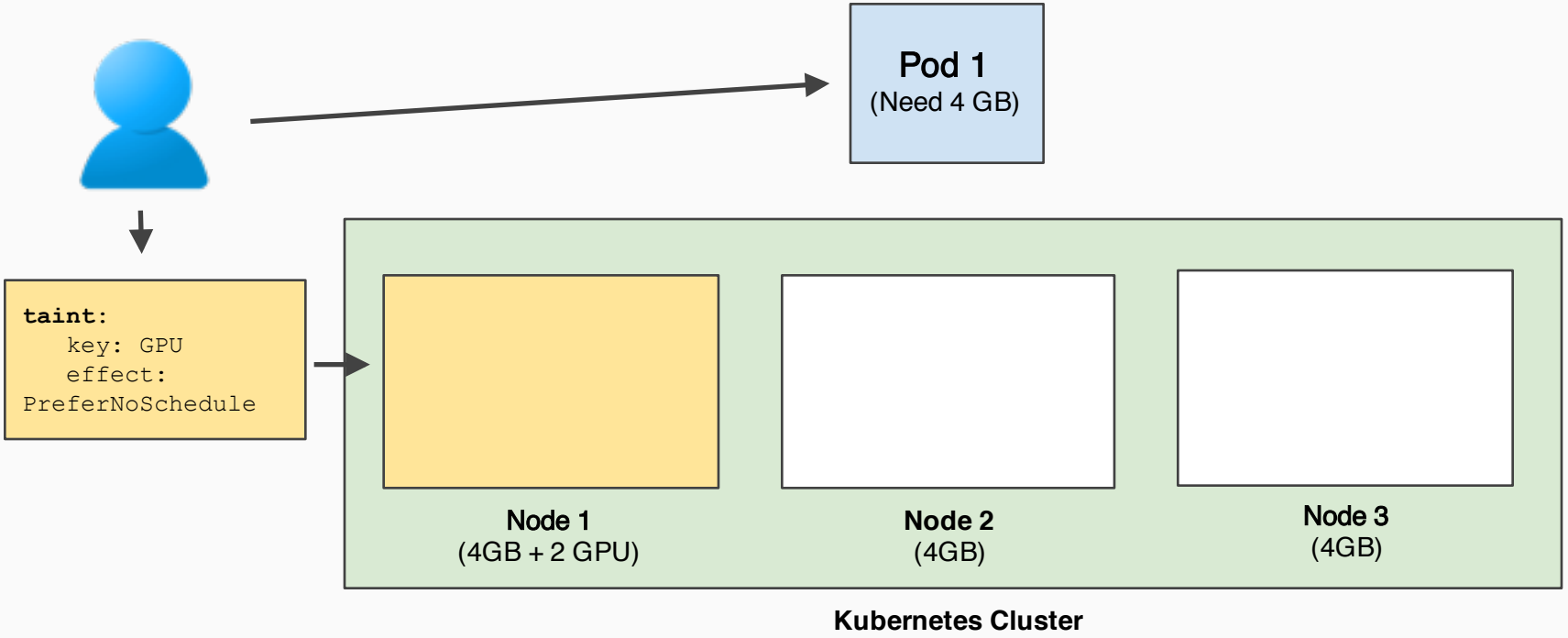
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



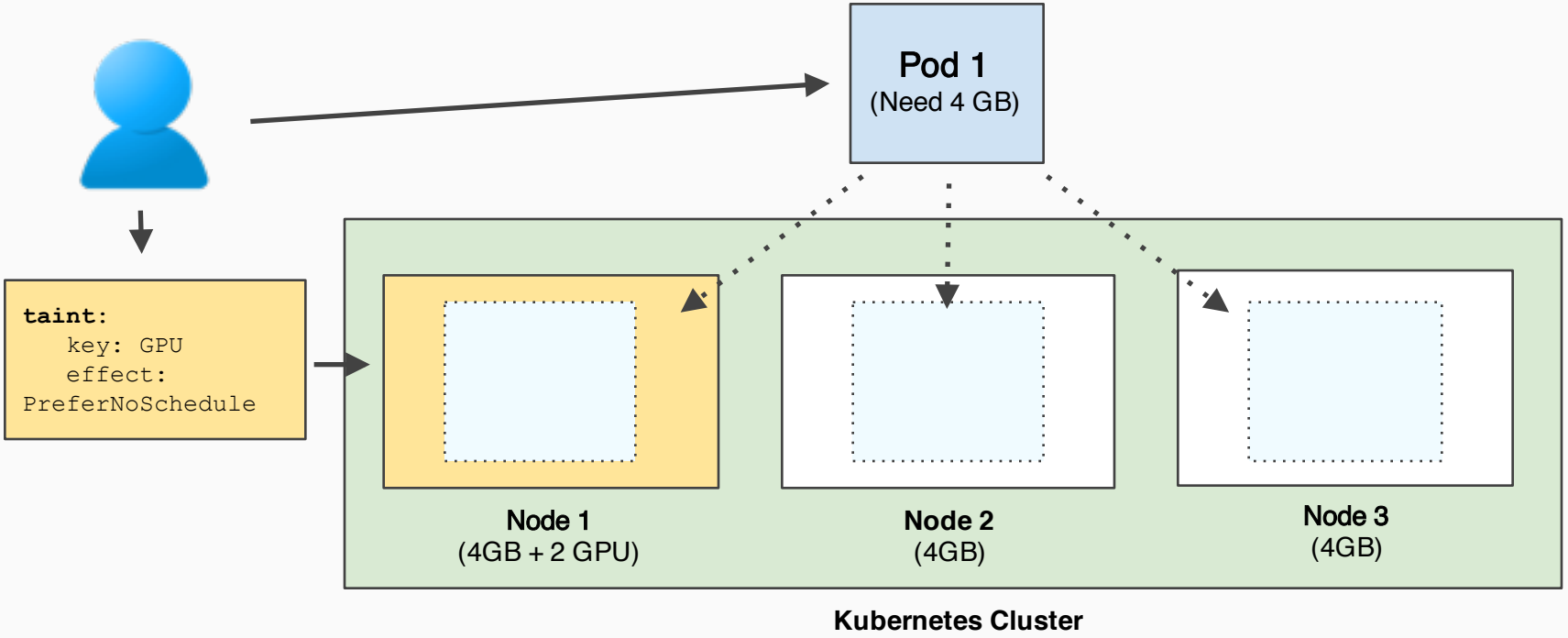
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware

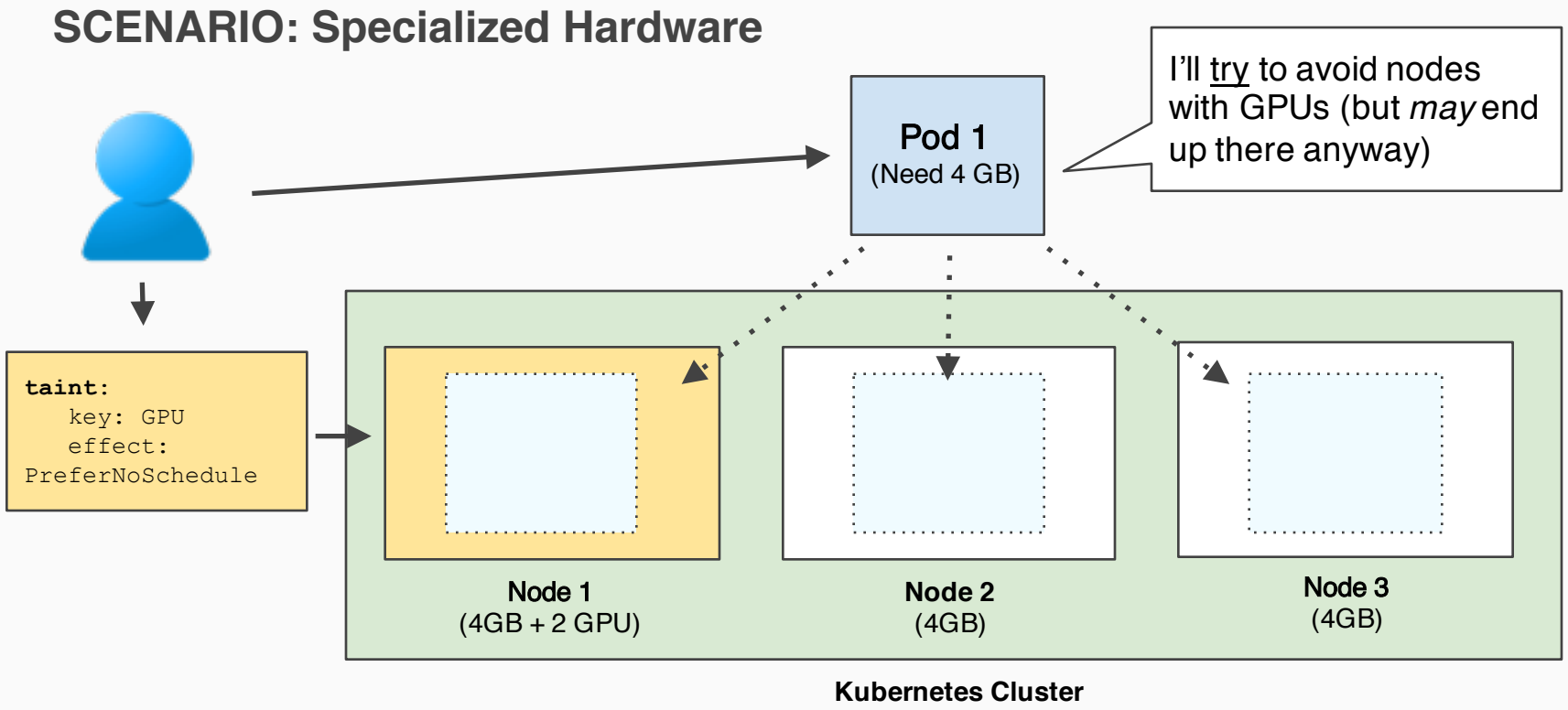


Sophisticated Scheduling: Taints/Toleration

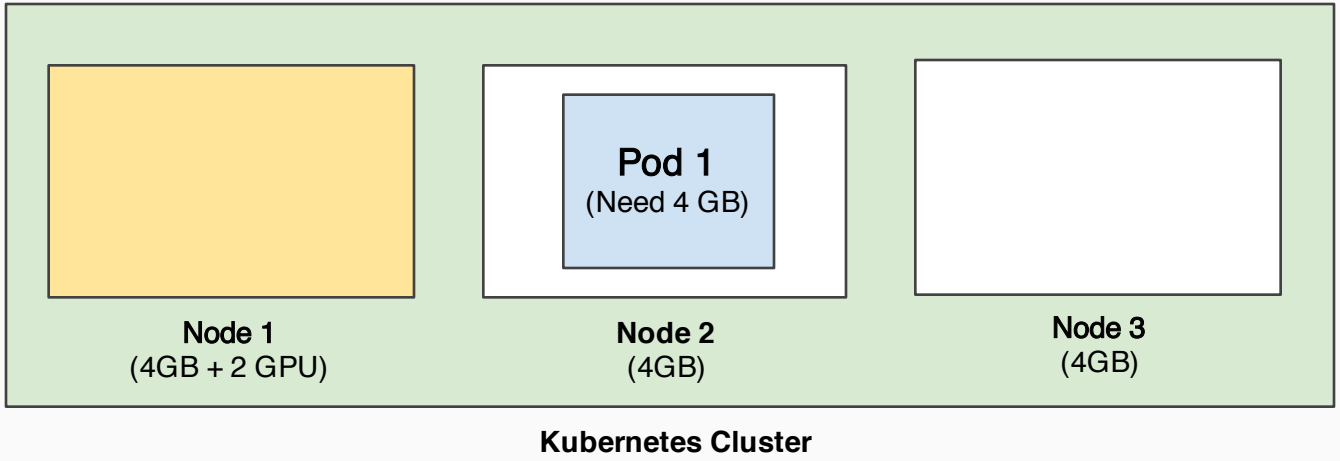
SCENARIO: Specialized Hardware



Sophisticated Scheduling: Taints/Toleration

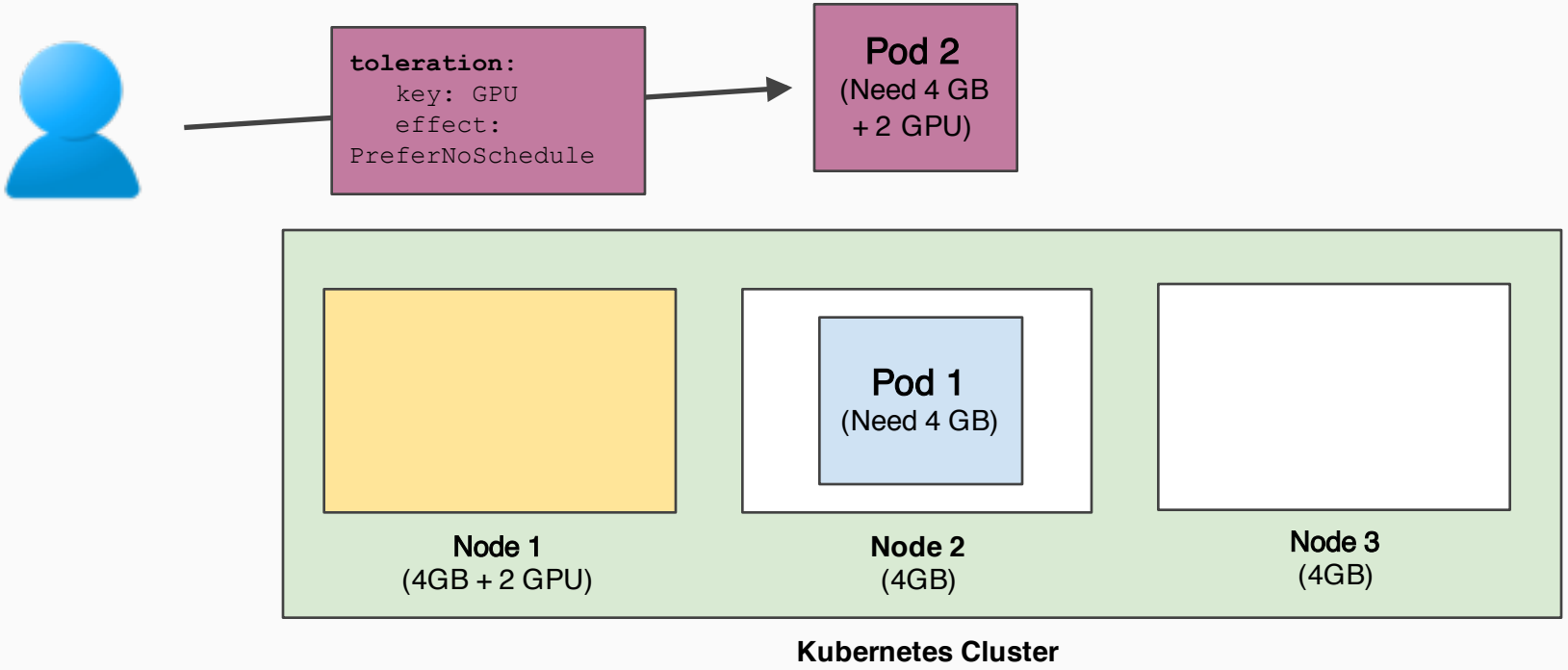


Sophisticated Scheduling: Taints/Toleration



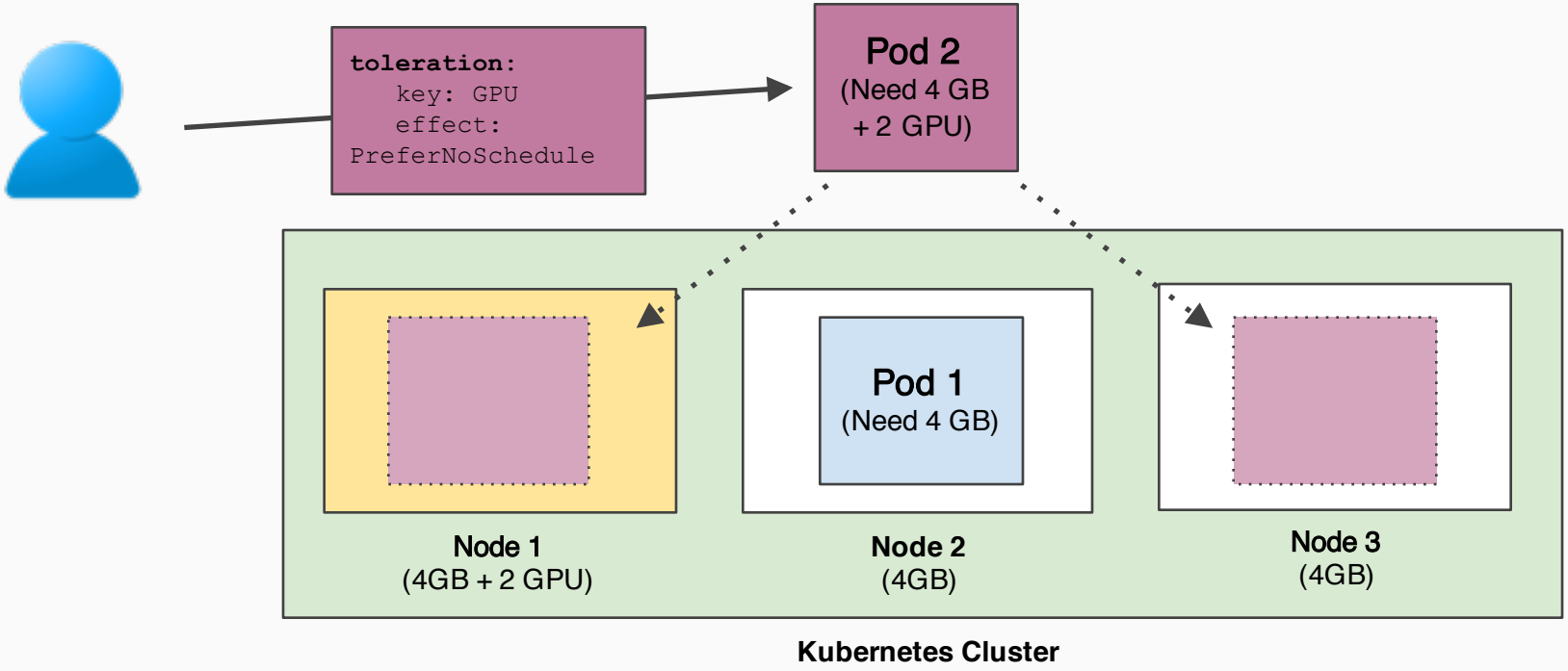
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



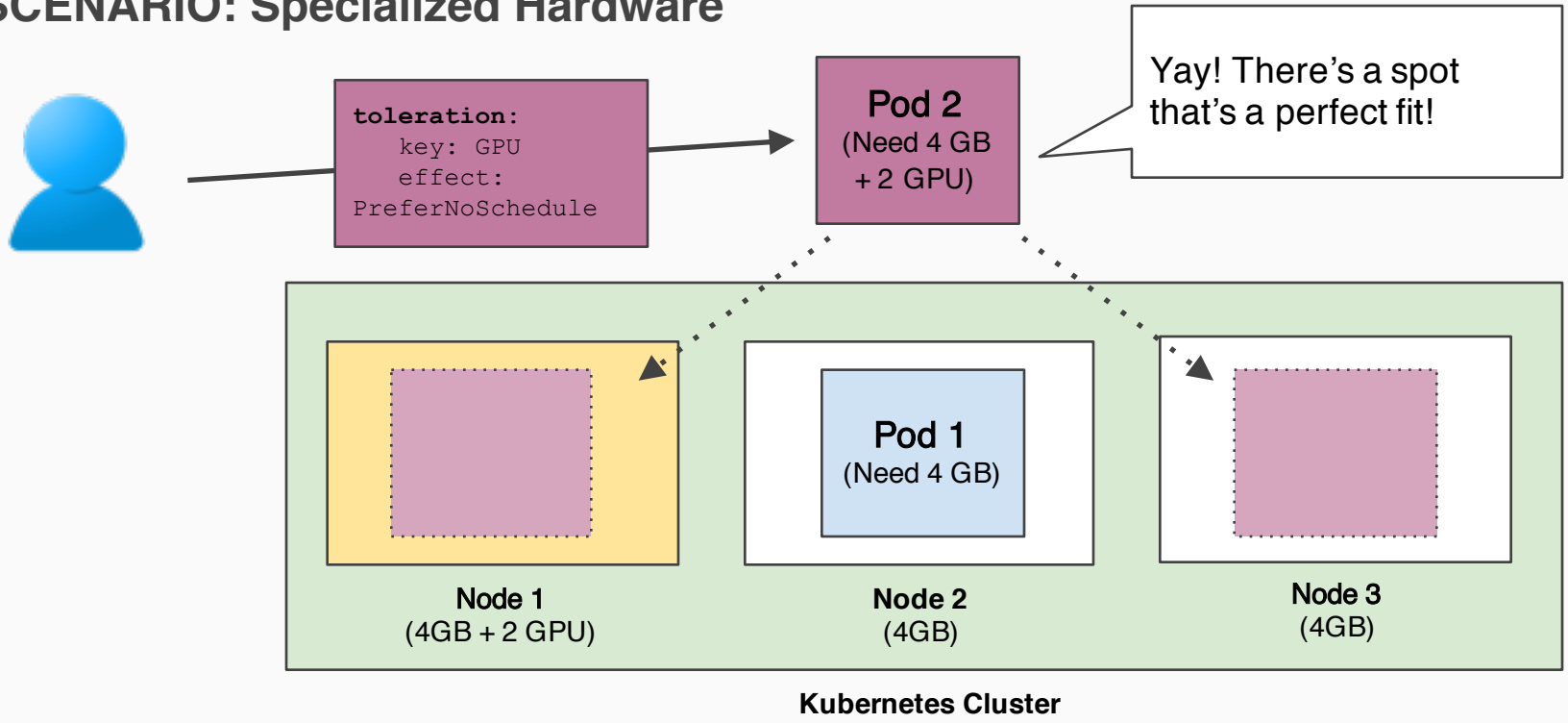
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



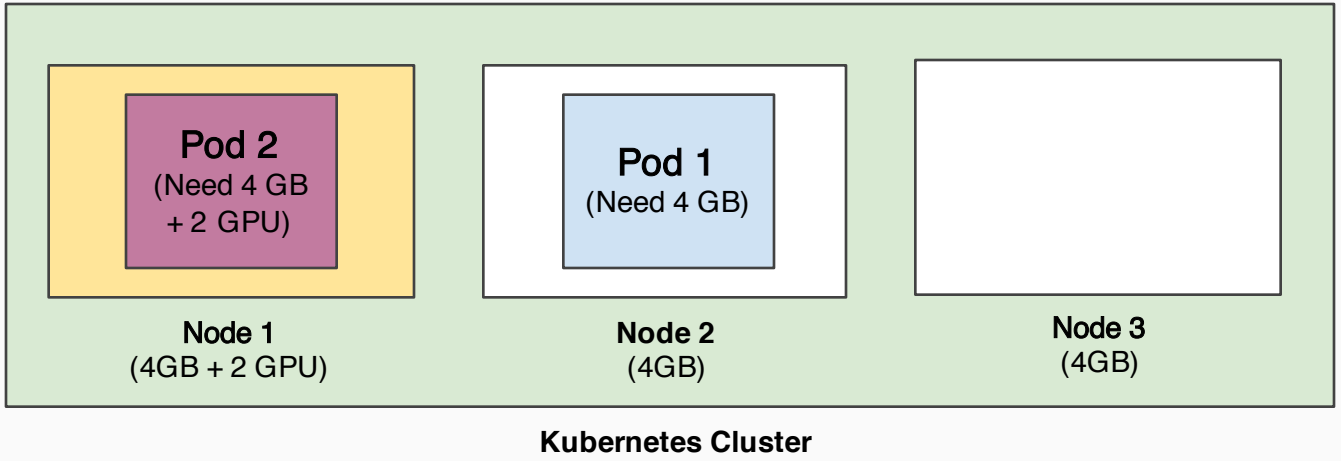
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



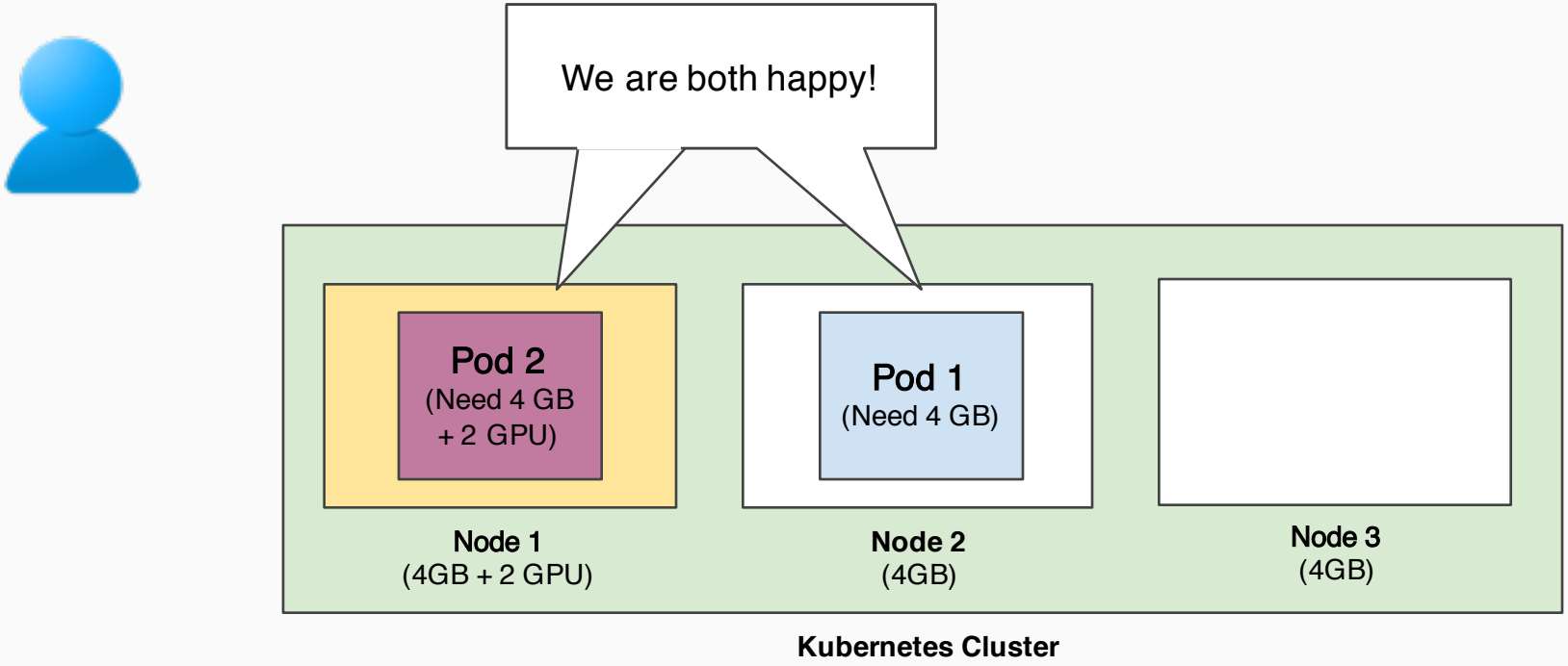
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware



Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware

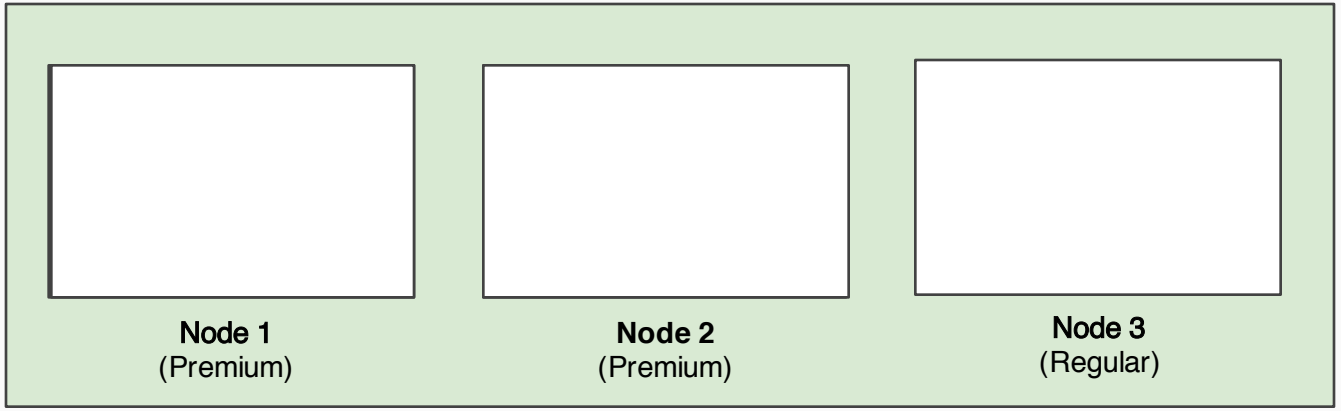


Sophisticated Scheduling: Taints/Toleration

SCENARIO: Specialized Hardware

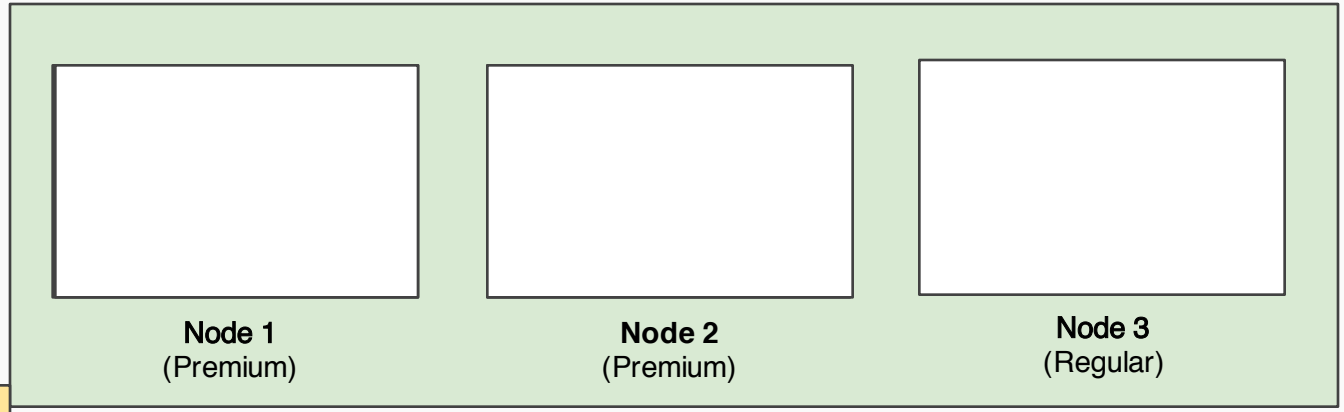
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Reserved instances



Sophisticated Scheduling: Taints/Toleration

SCENARIO: Reserved instances

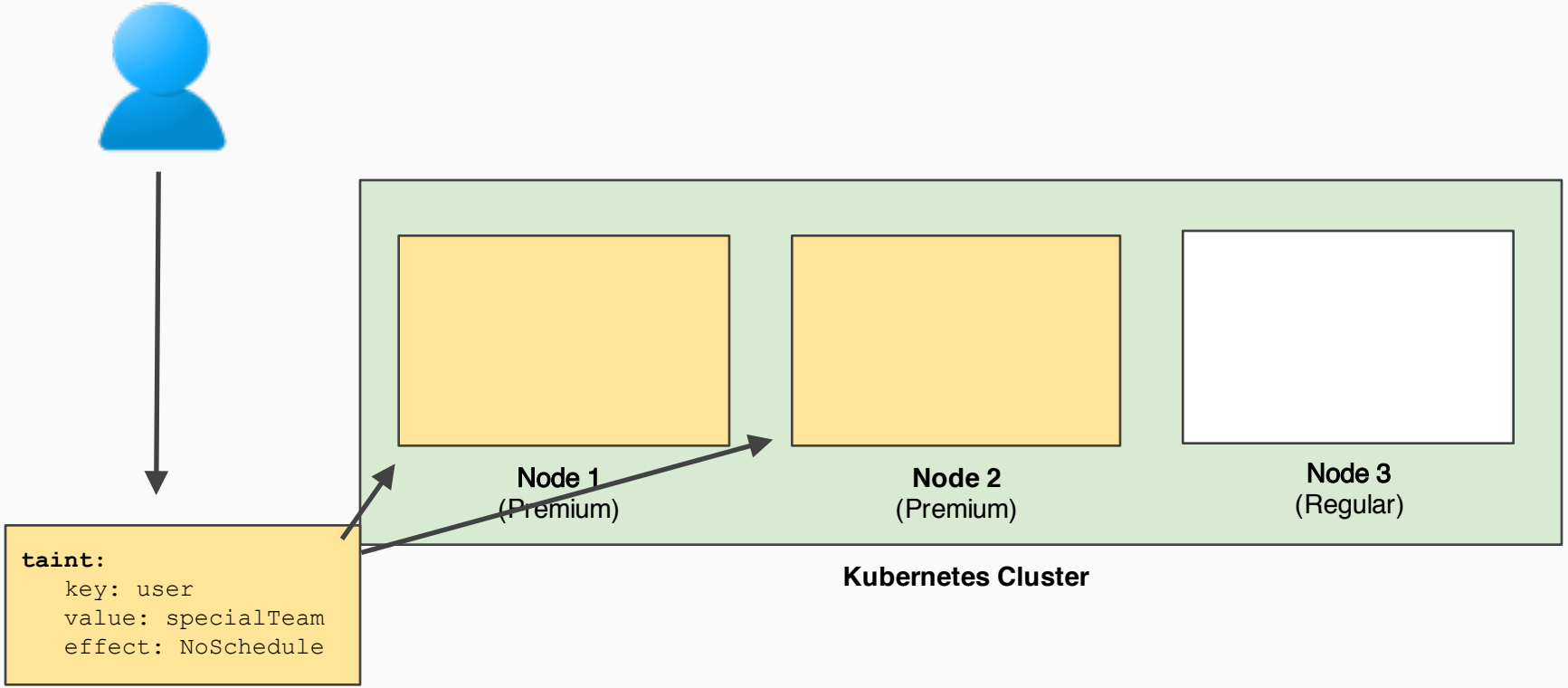


Kubernetes Cluster

```
taint:
   key: user
   value: specialTeam
   effect: NoSchedule
```

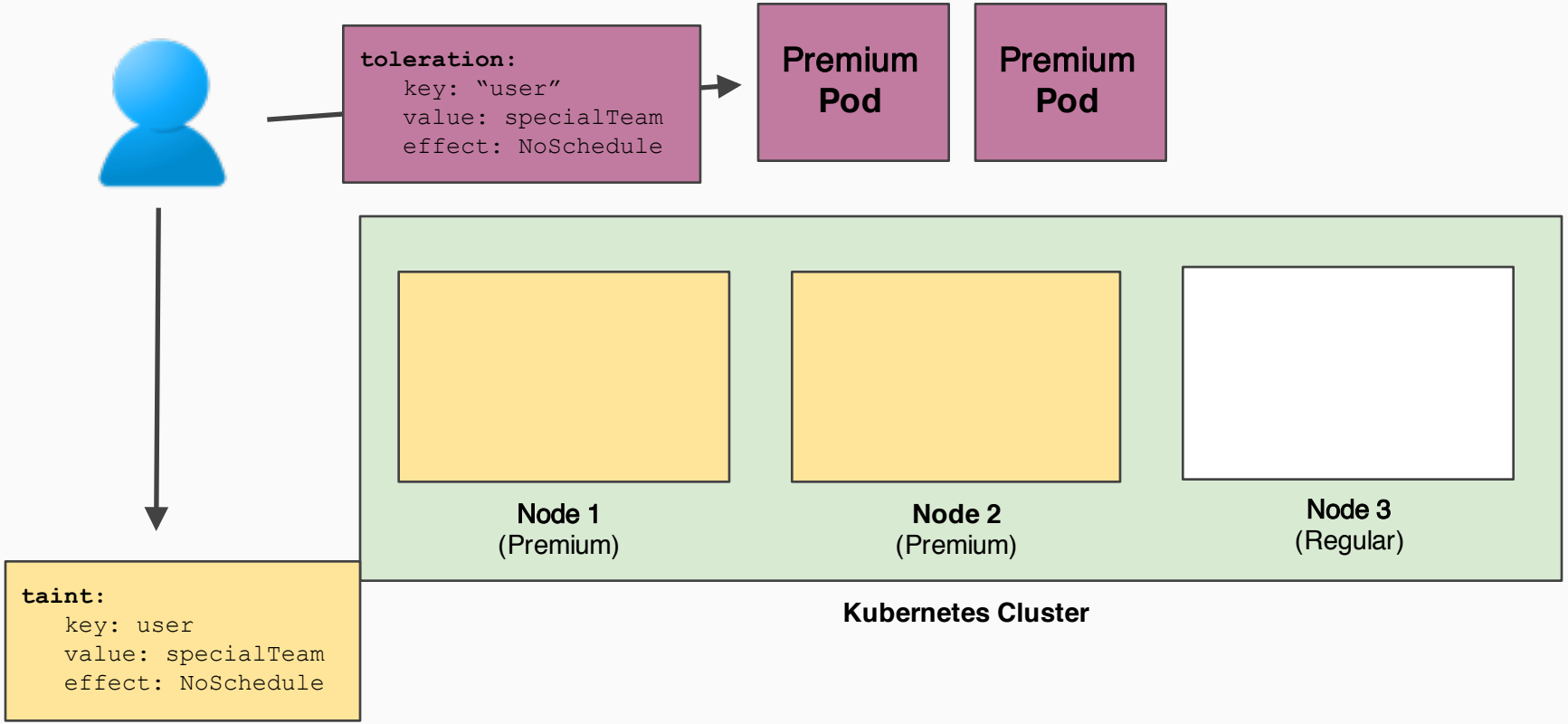
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Reserved instances



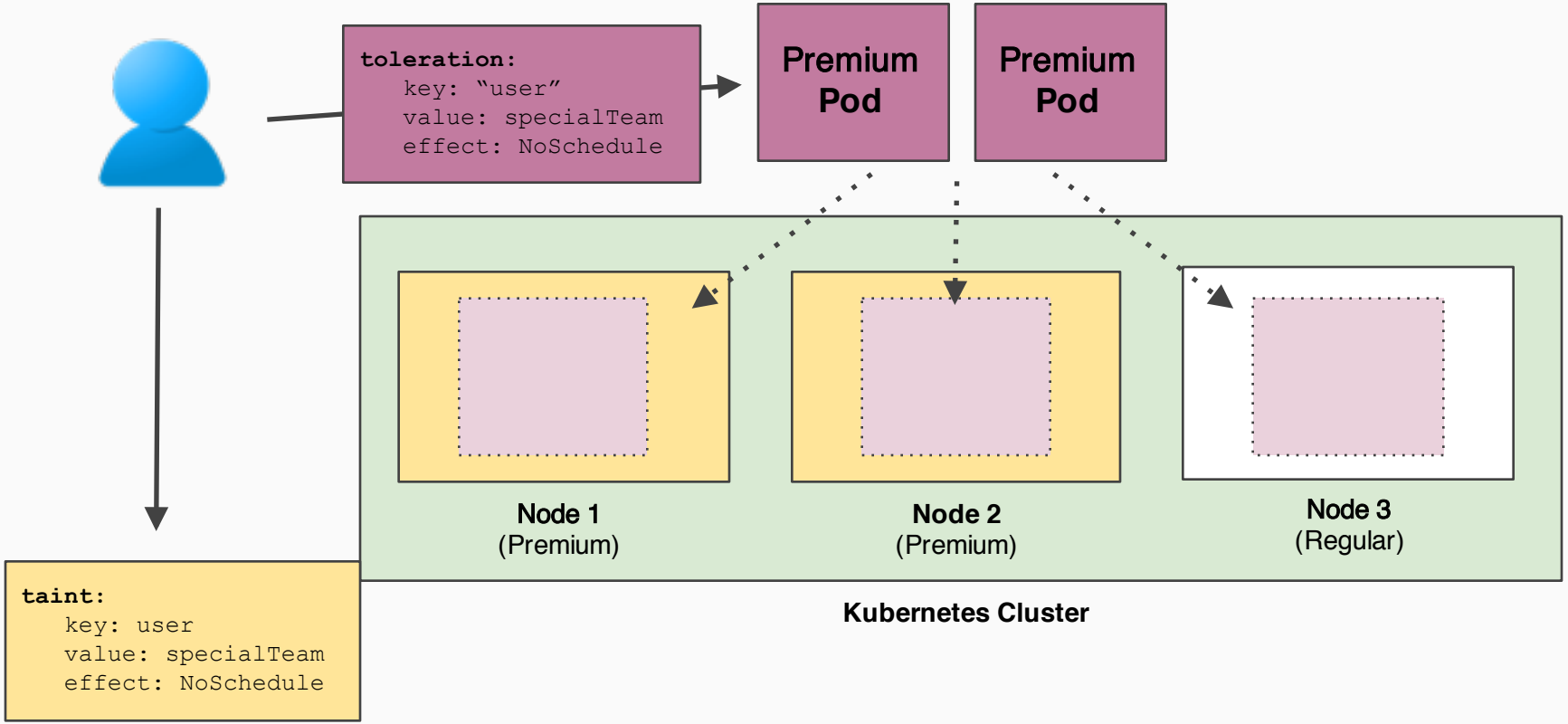
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Reserved instances

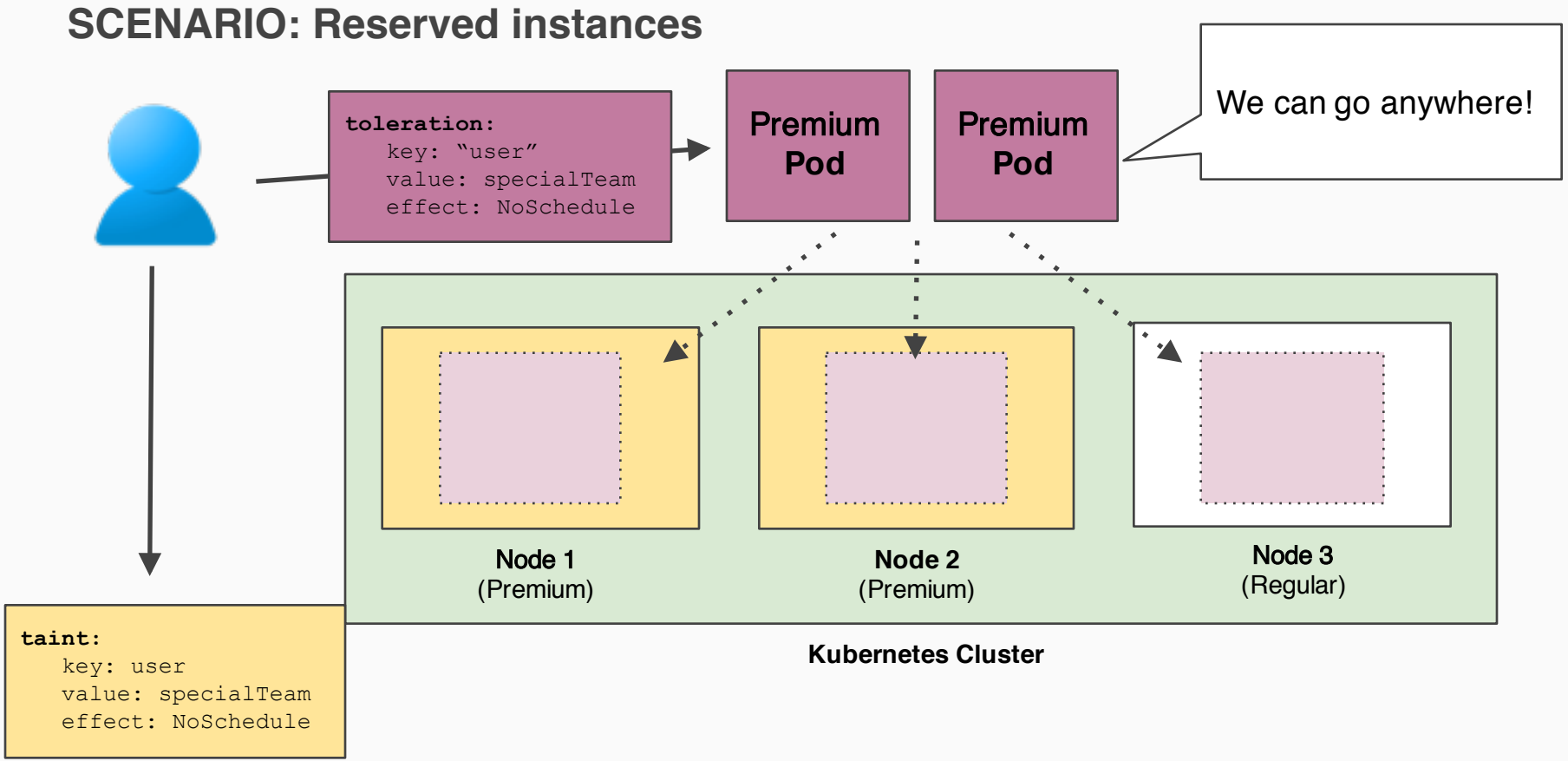


Sophisticated Scheduling: Taints/Toleration

SCENARIO: Reserved instances

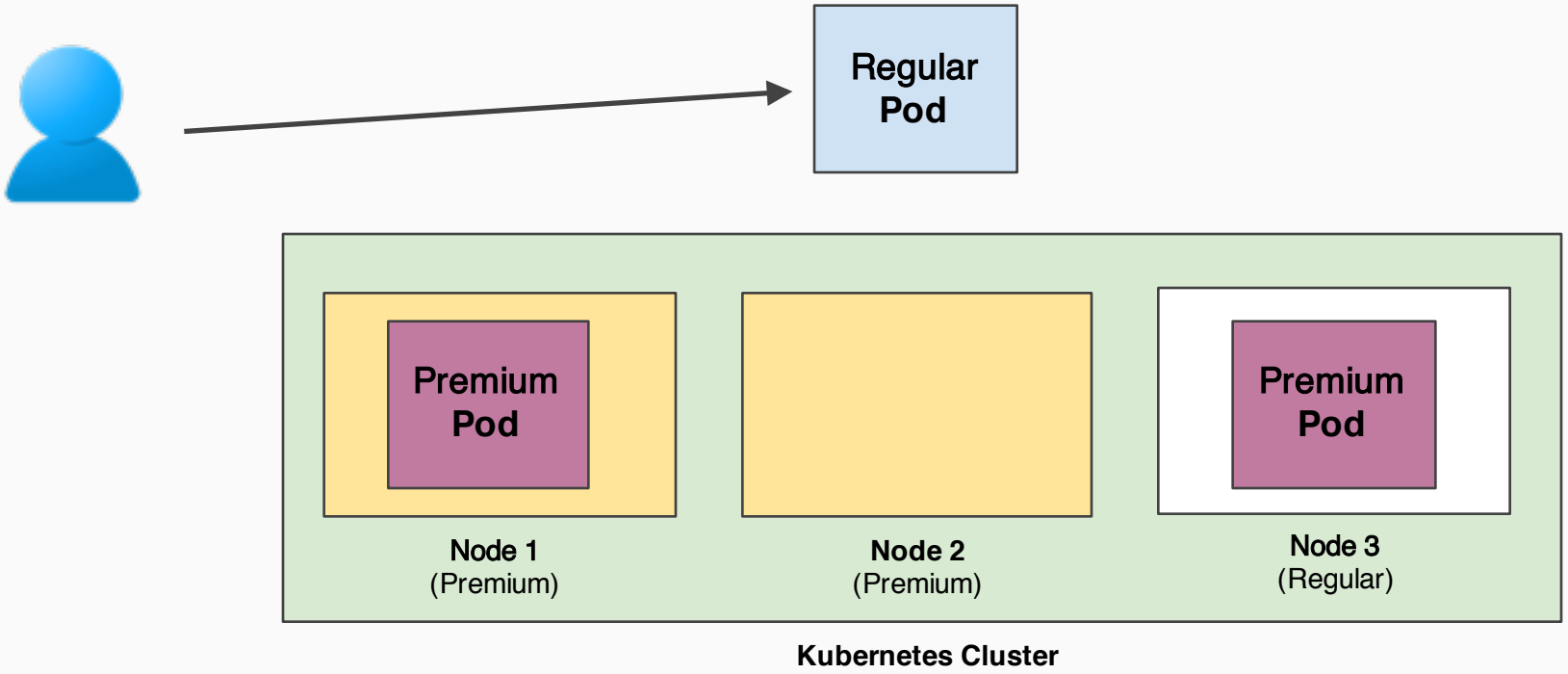


Sophisticated Scheduling: Taints/Toleration



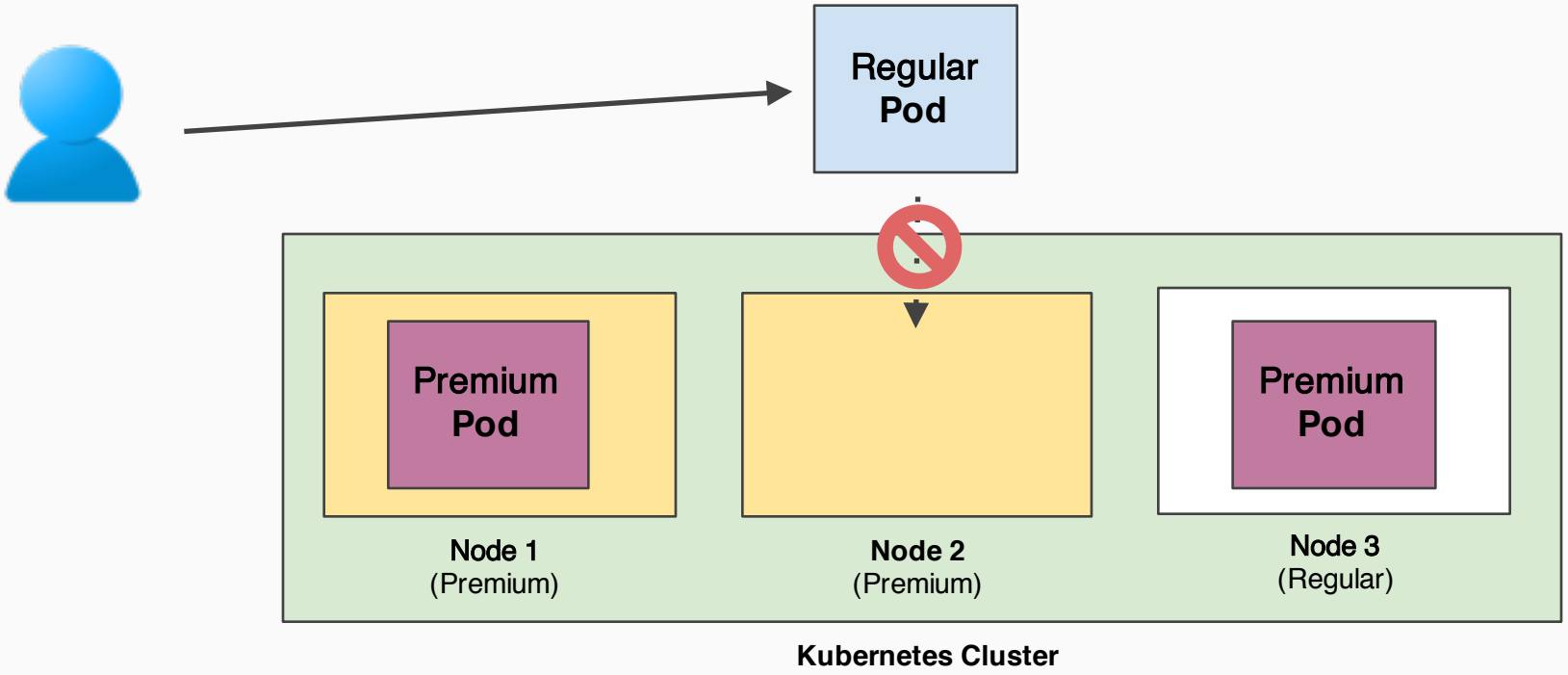
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Reserved instances



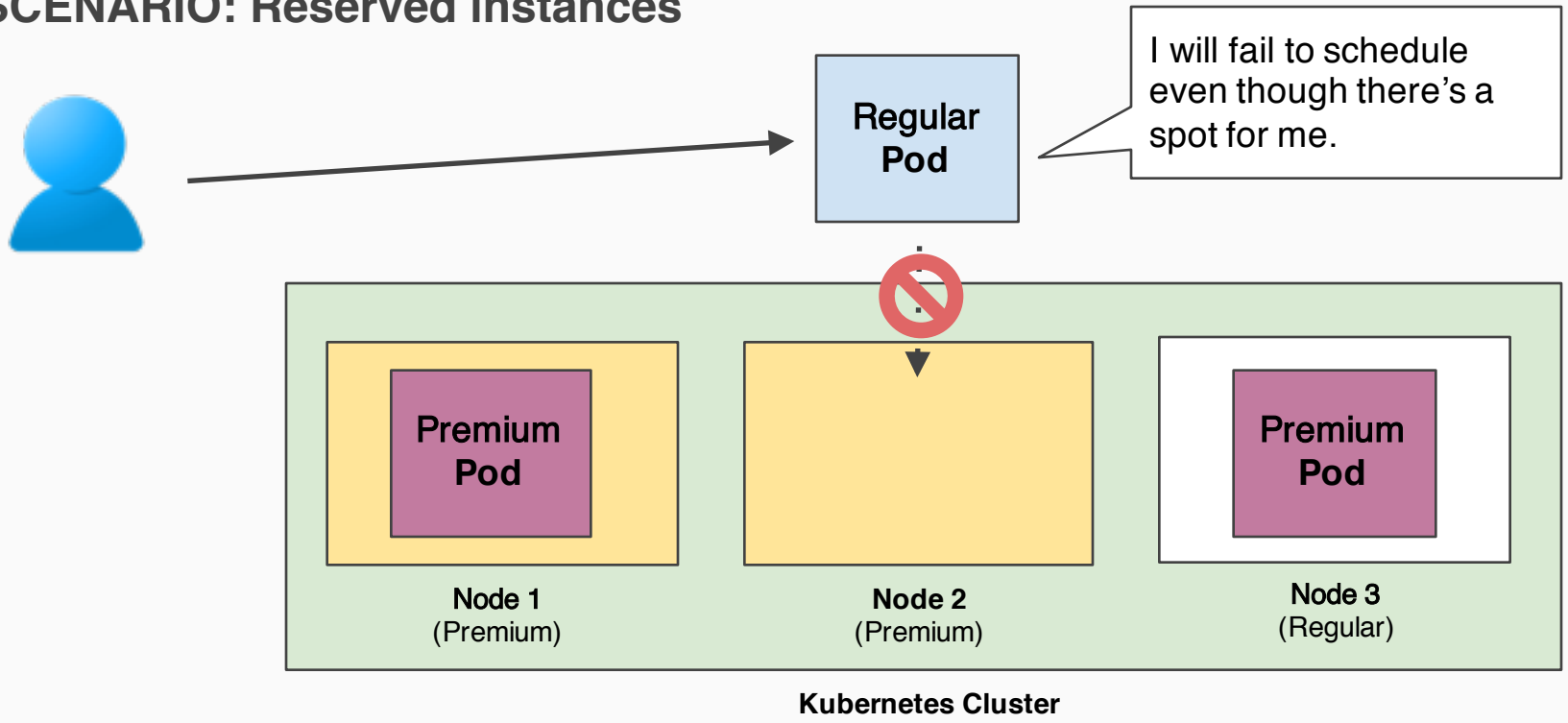
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Reserved instances



Sophisticated Scheduling: Taints/Toleration

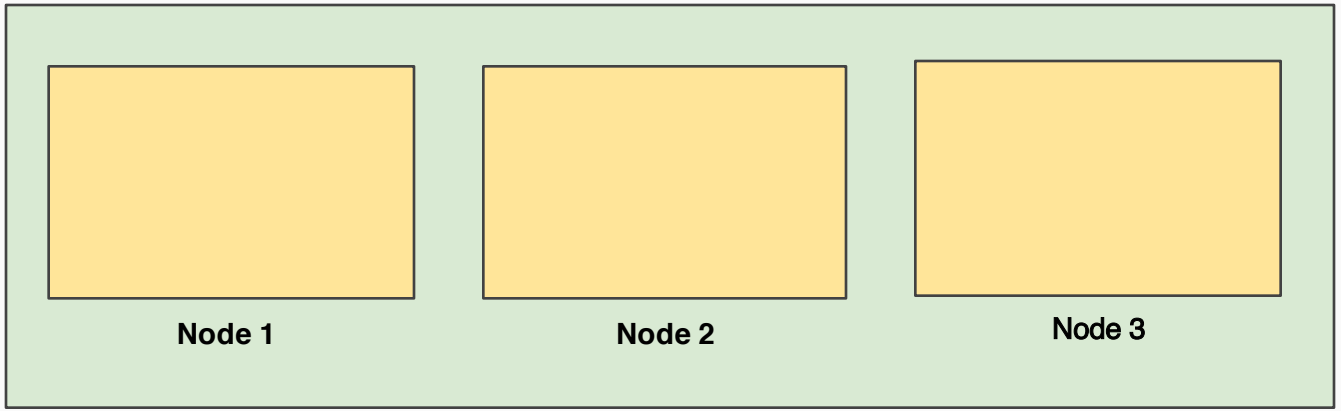
SCENARIO: Reserved instances



Sophisticated Scheduling: Taints/Toleration

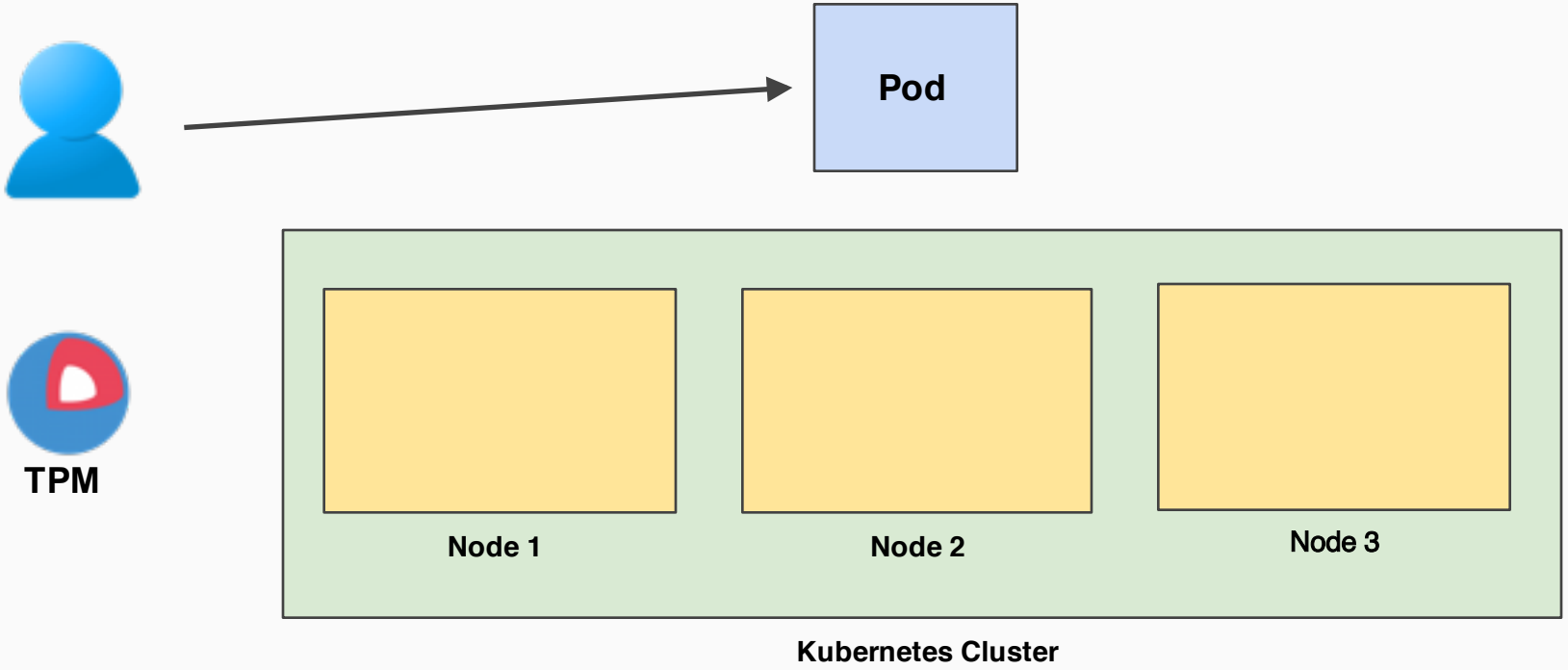
SCENARIO: Ensuring node meets spec

TPM



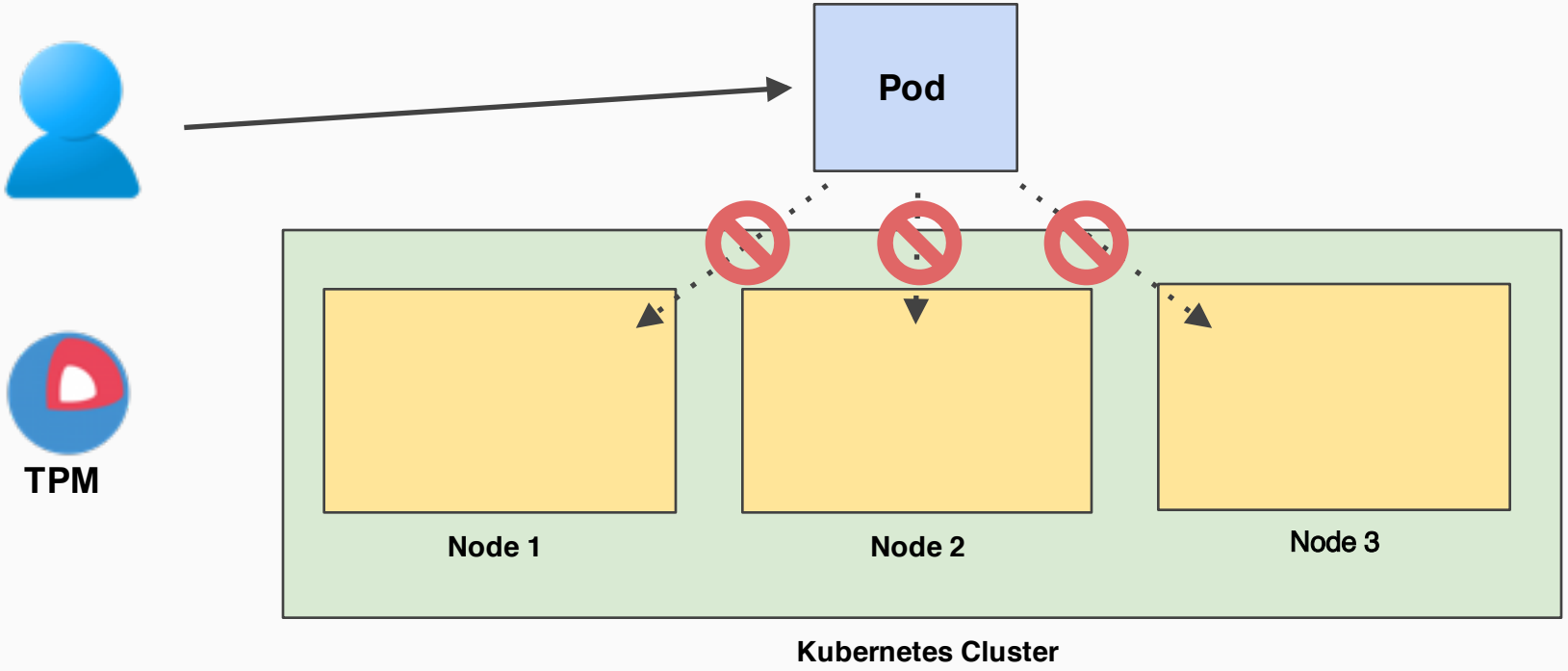
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Ensuring node meets spec



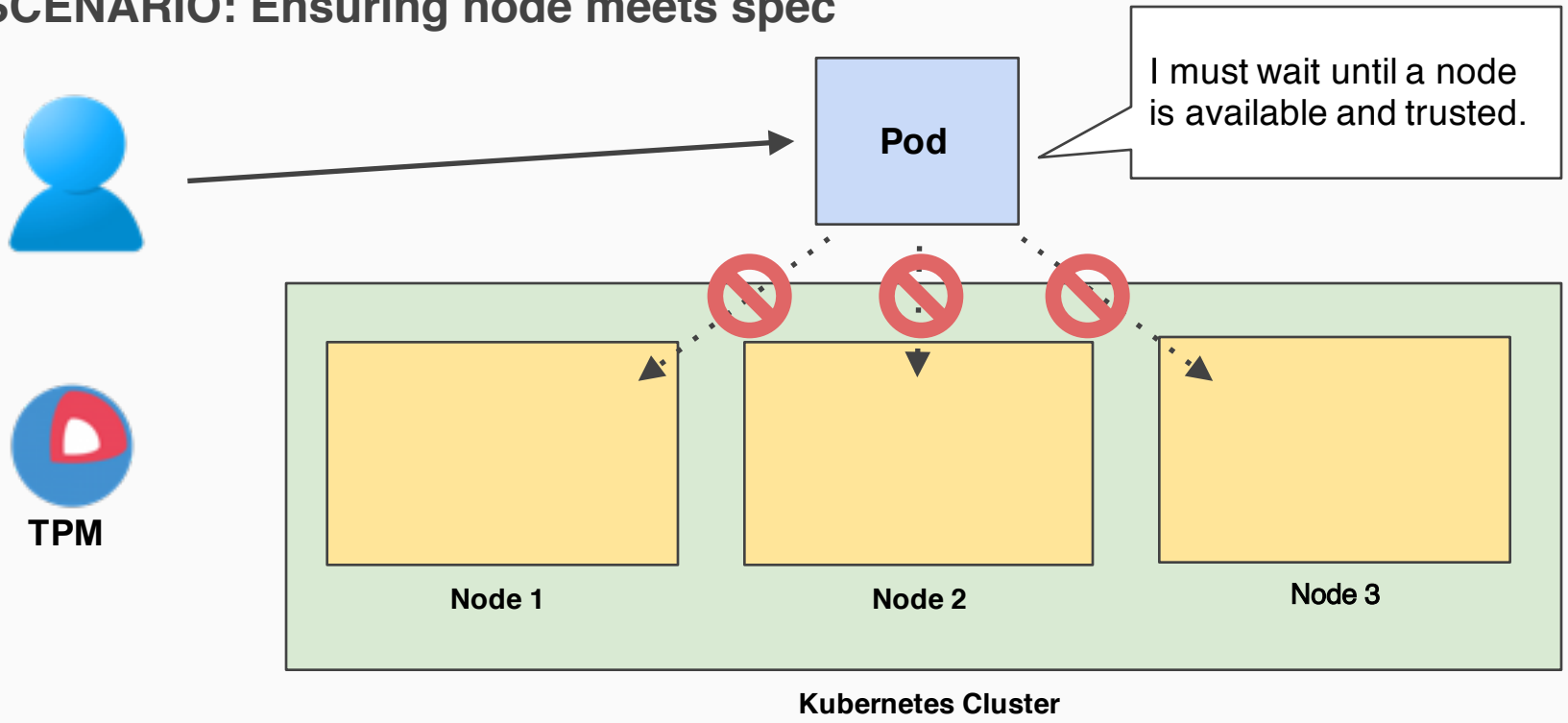
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Ensuring node meets spec



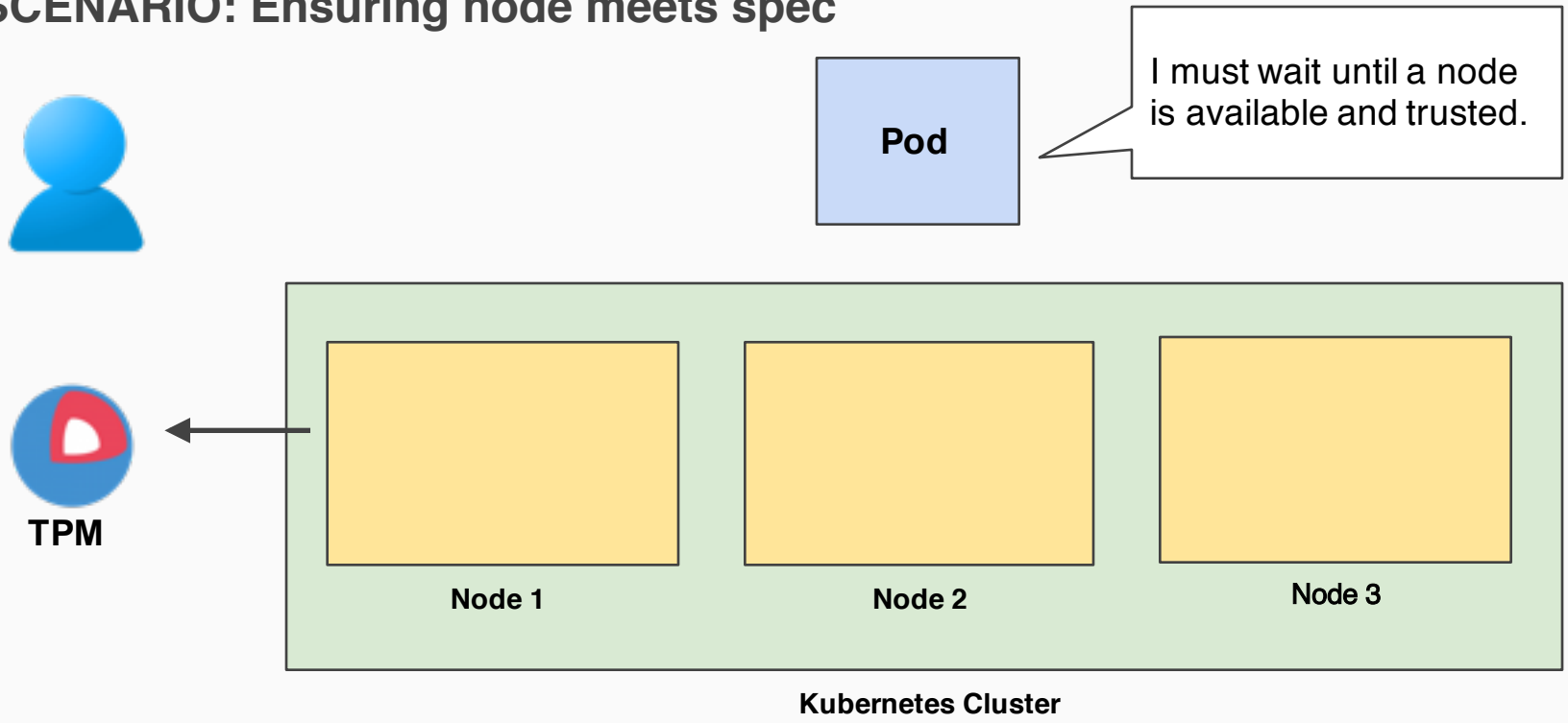
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Ensuring node meets spec



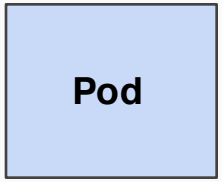
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Ensuring node meets spec



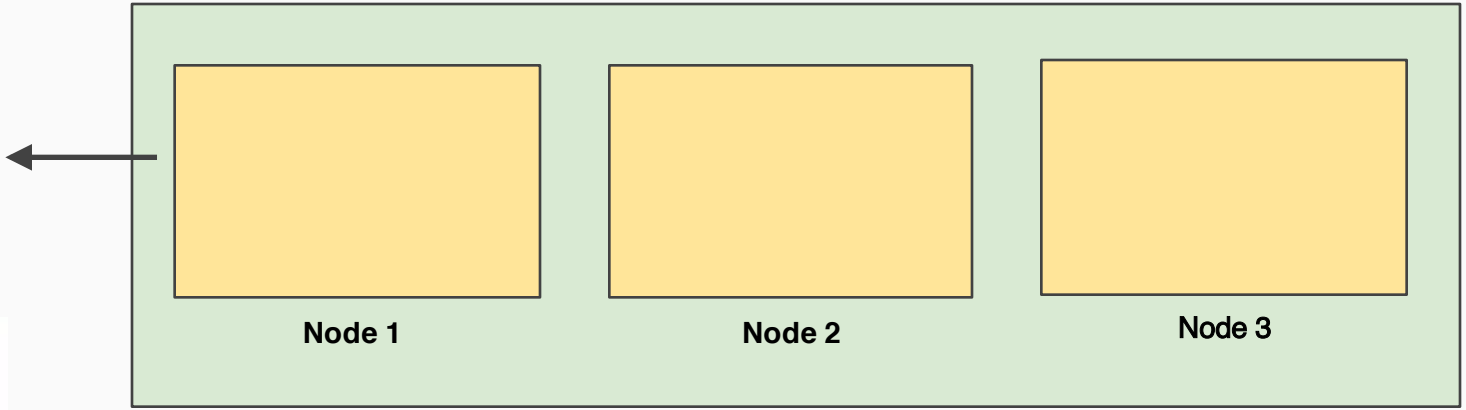
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Ensuring node meets spec



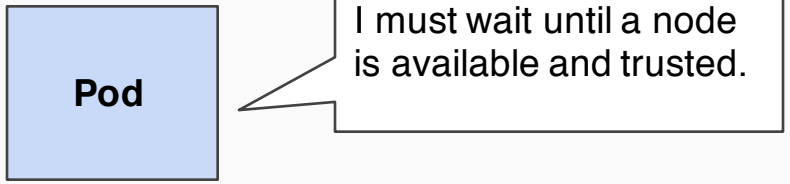
I must wait until a node is available and trusted.

TPM

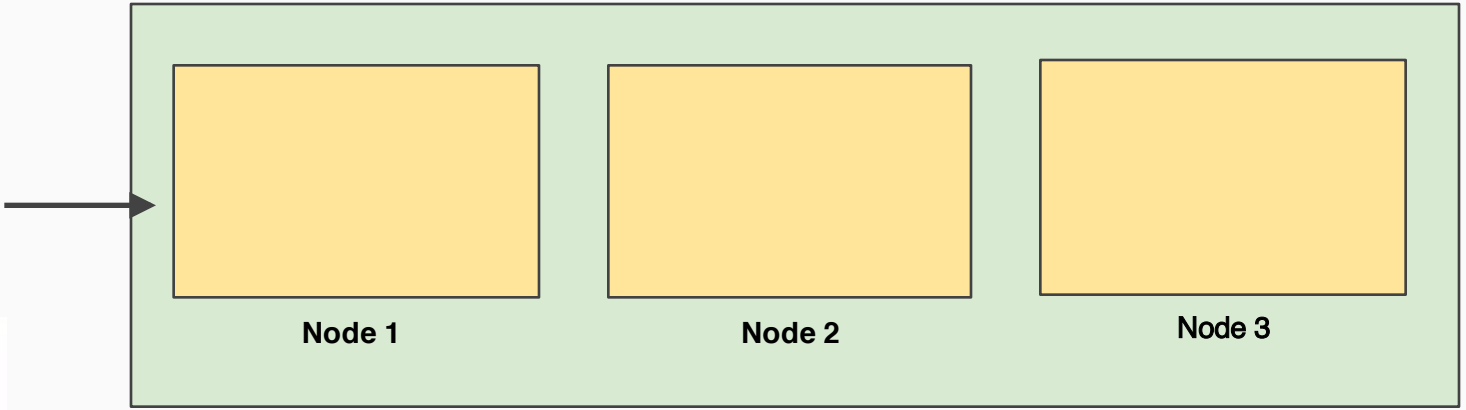


Sophisticated Scheduling: Taints/Toleration

SCENARIO: Ensuring node meets spec

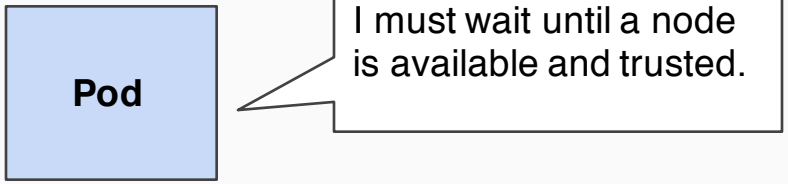


TPM

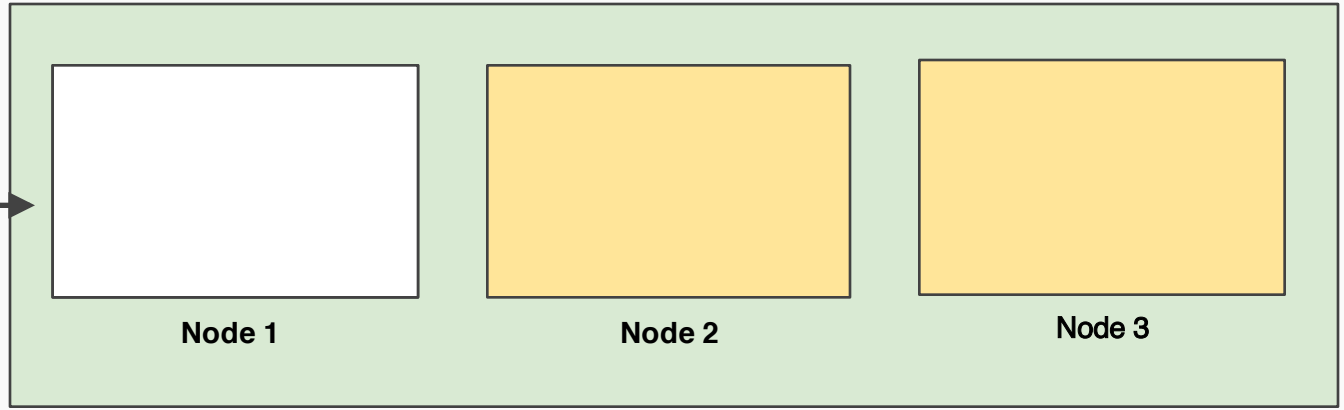


Sophisticated Scheduling: Taints/Toleration

SCENARIO: Ensuring node meets spec



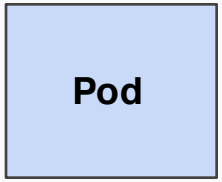
TPM

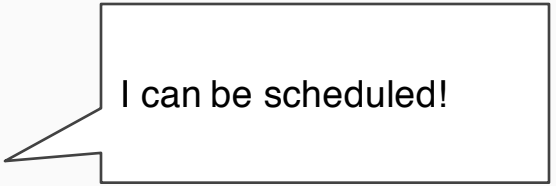


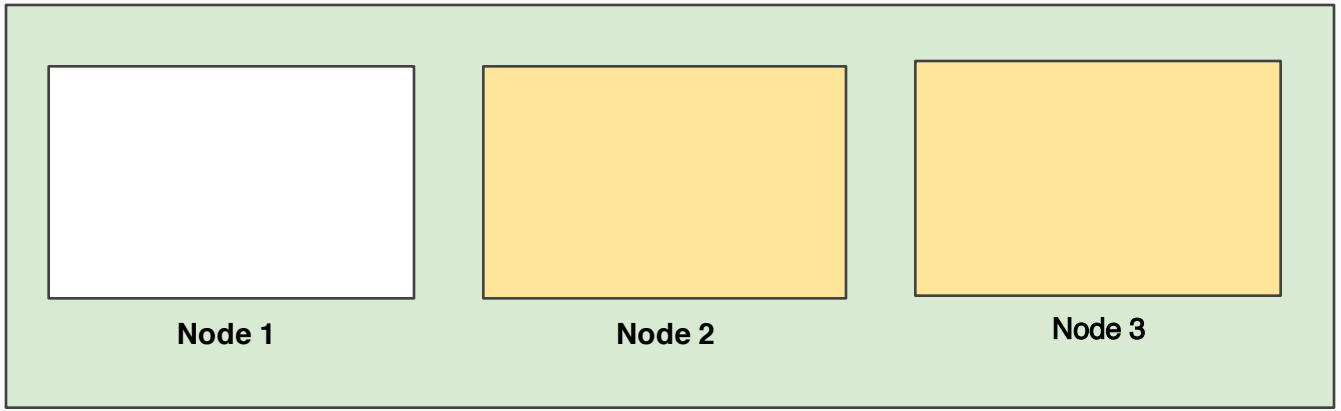
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Ensuring node meets spec

TPM



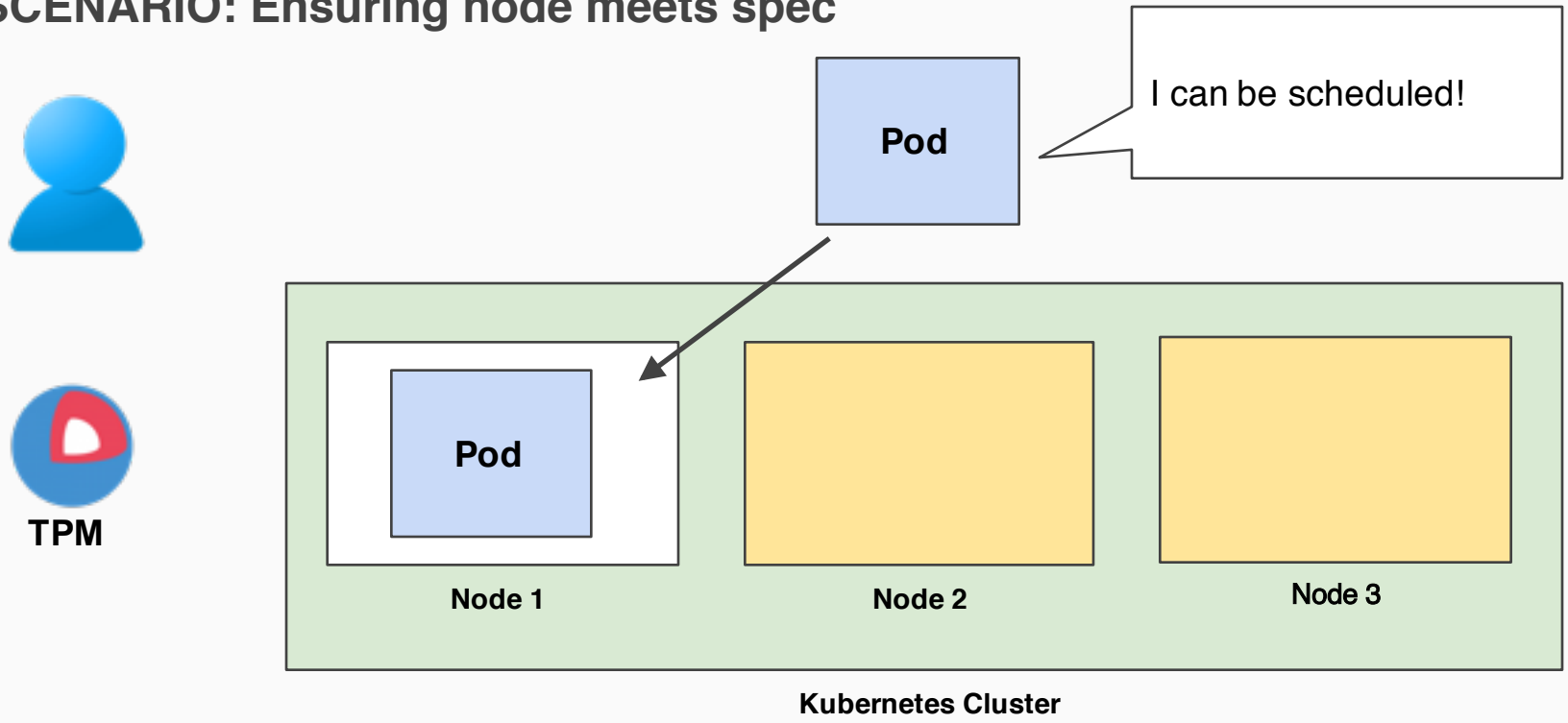




Kubernetes Cluster

Sophisticated Scheduling: Taints/Toleration

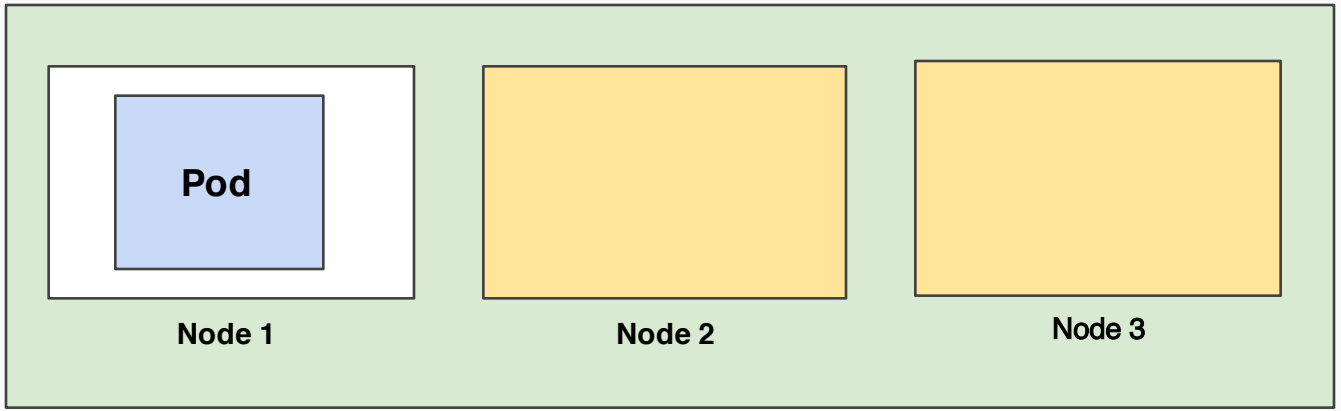
SCENARIO: Ensuring node meets spec



Sophisticated Scheduling: Taints/Toleration

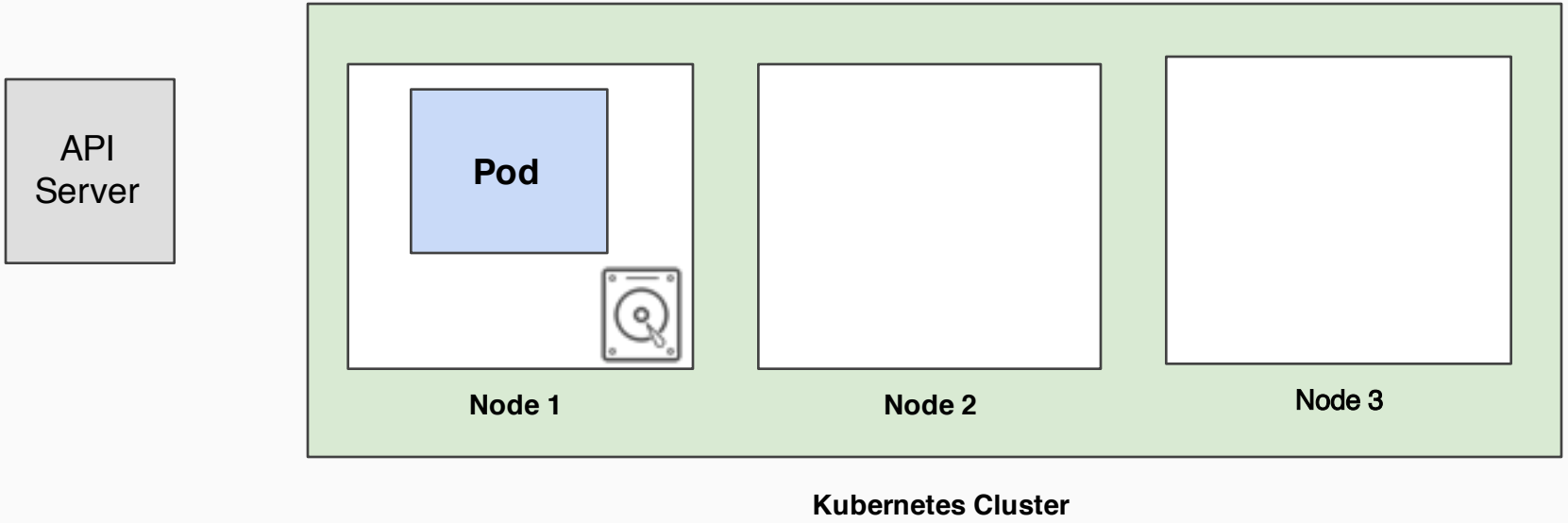
SCENARIO: Ensuring node meets spec

TPM



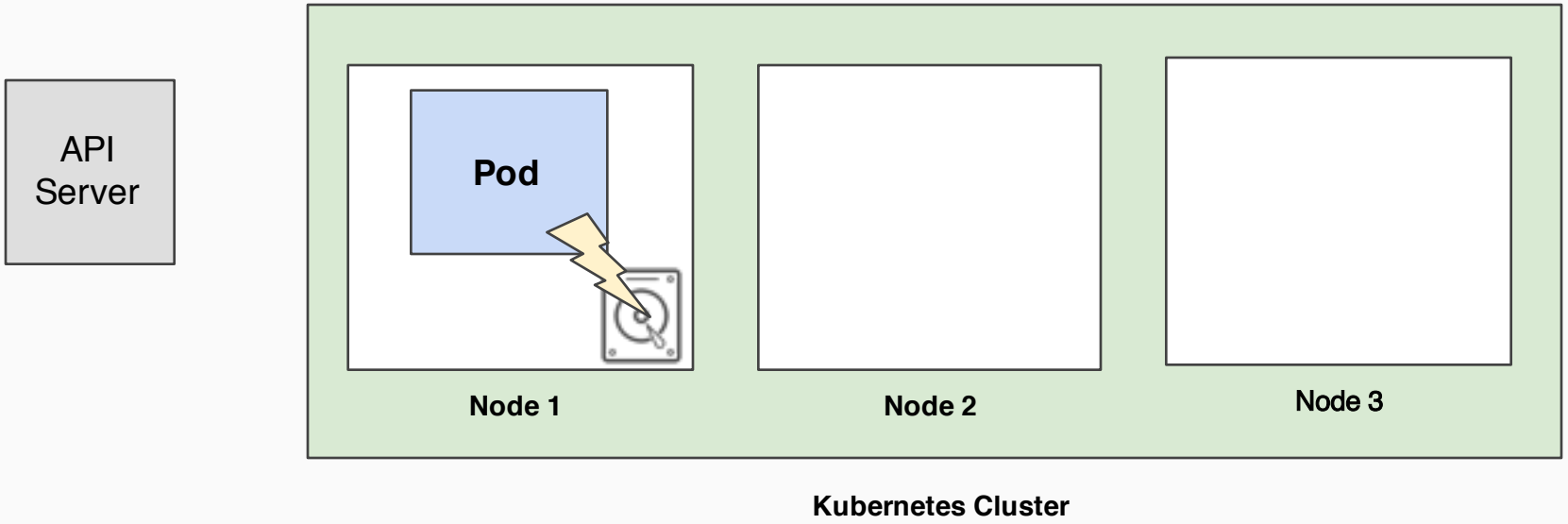
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)



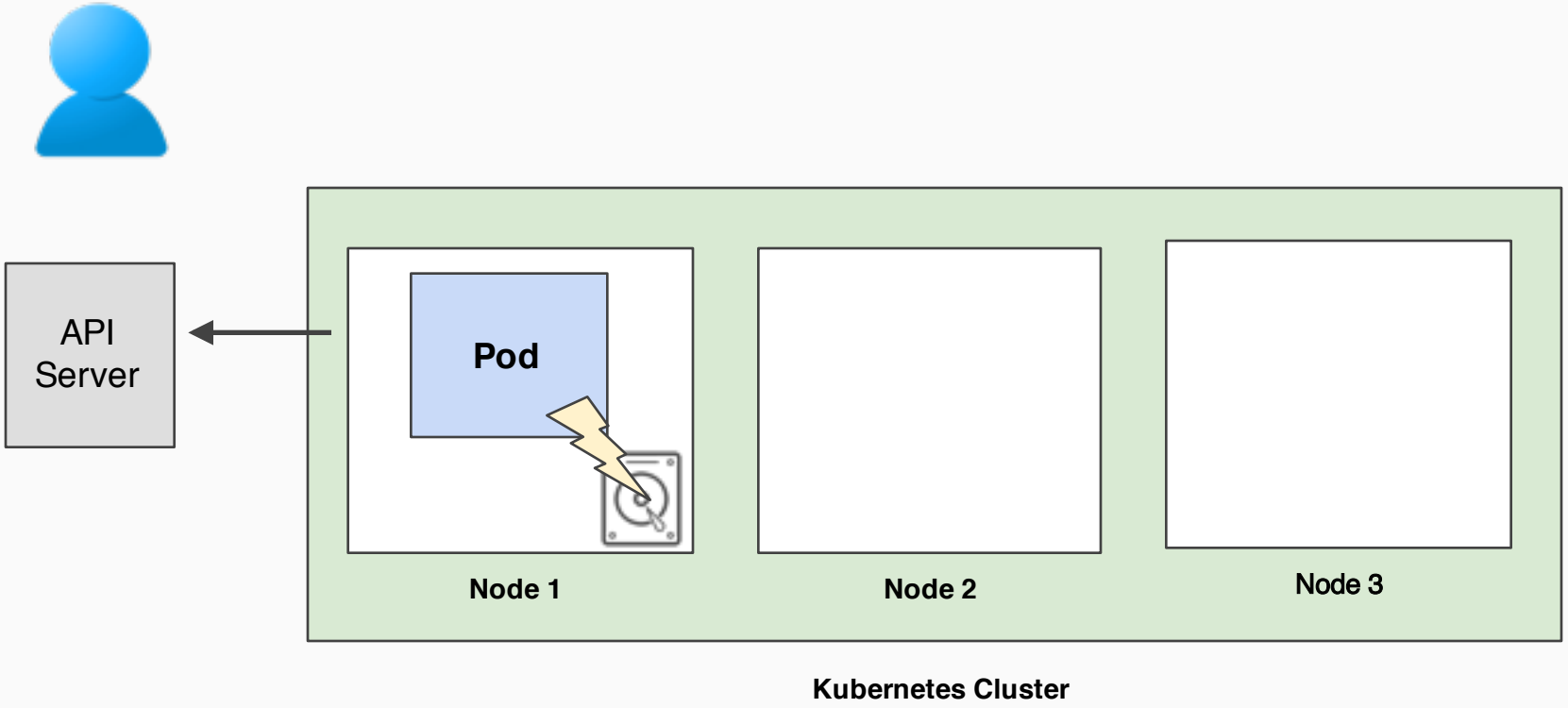
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)



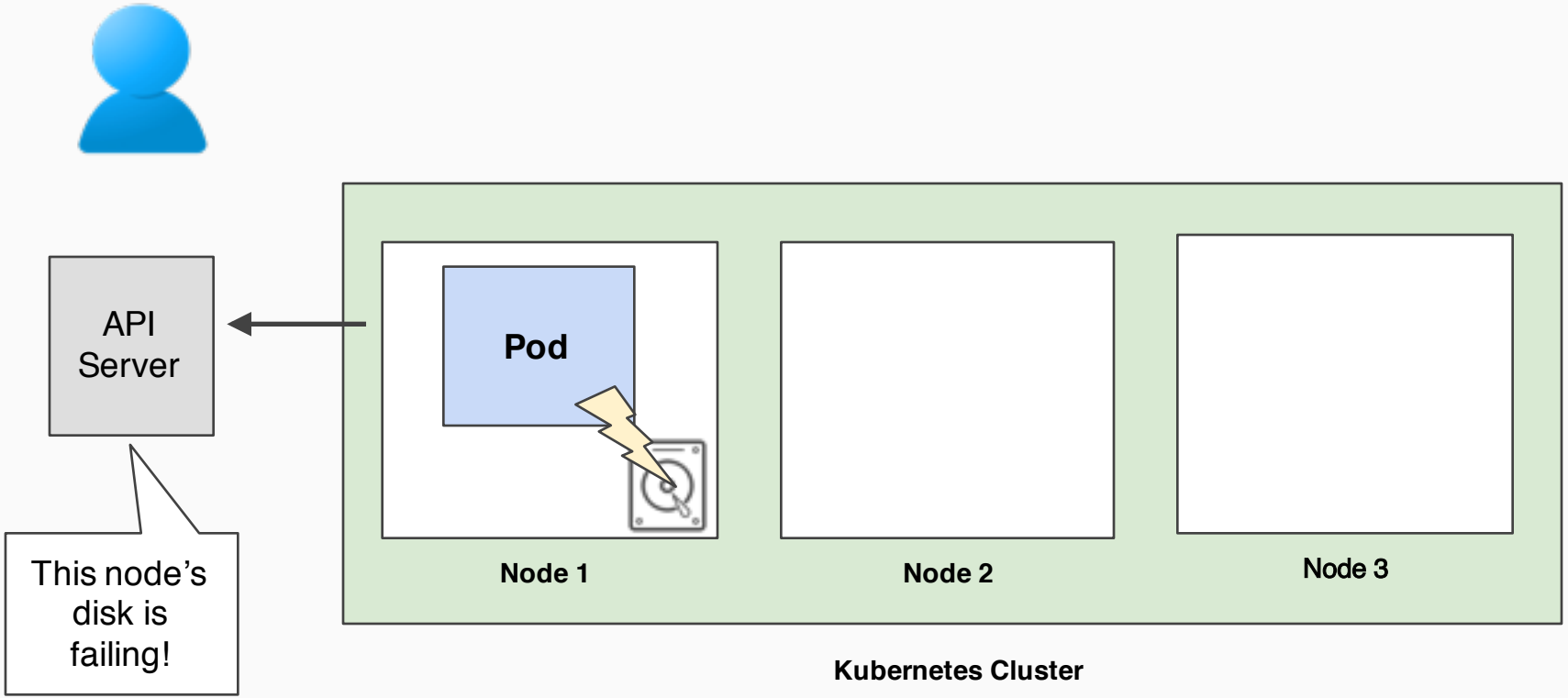
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)



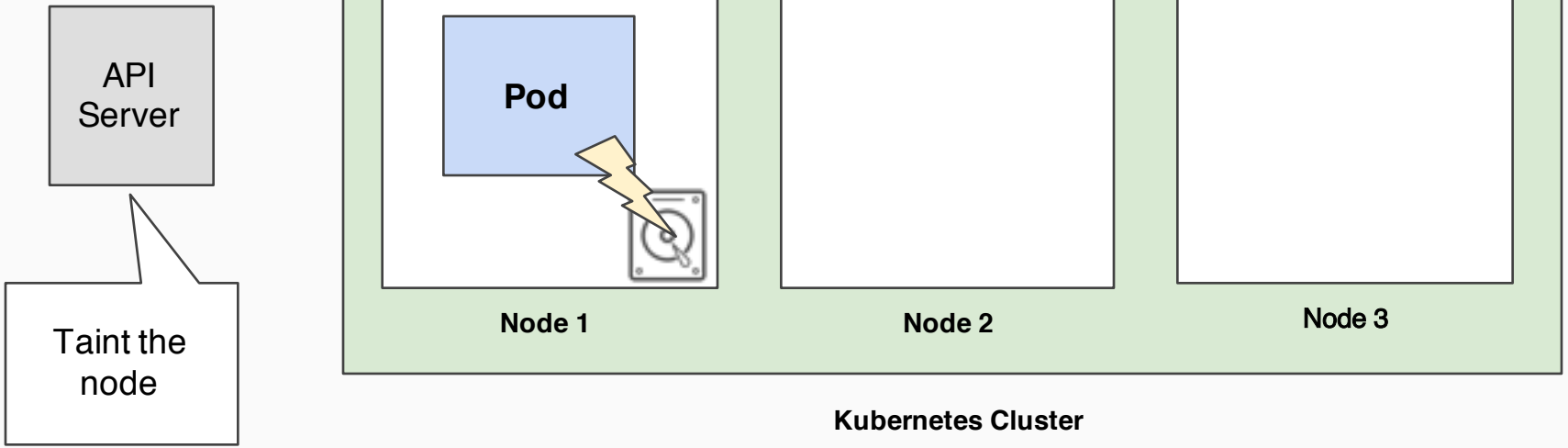
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)



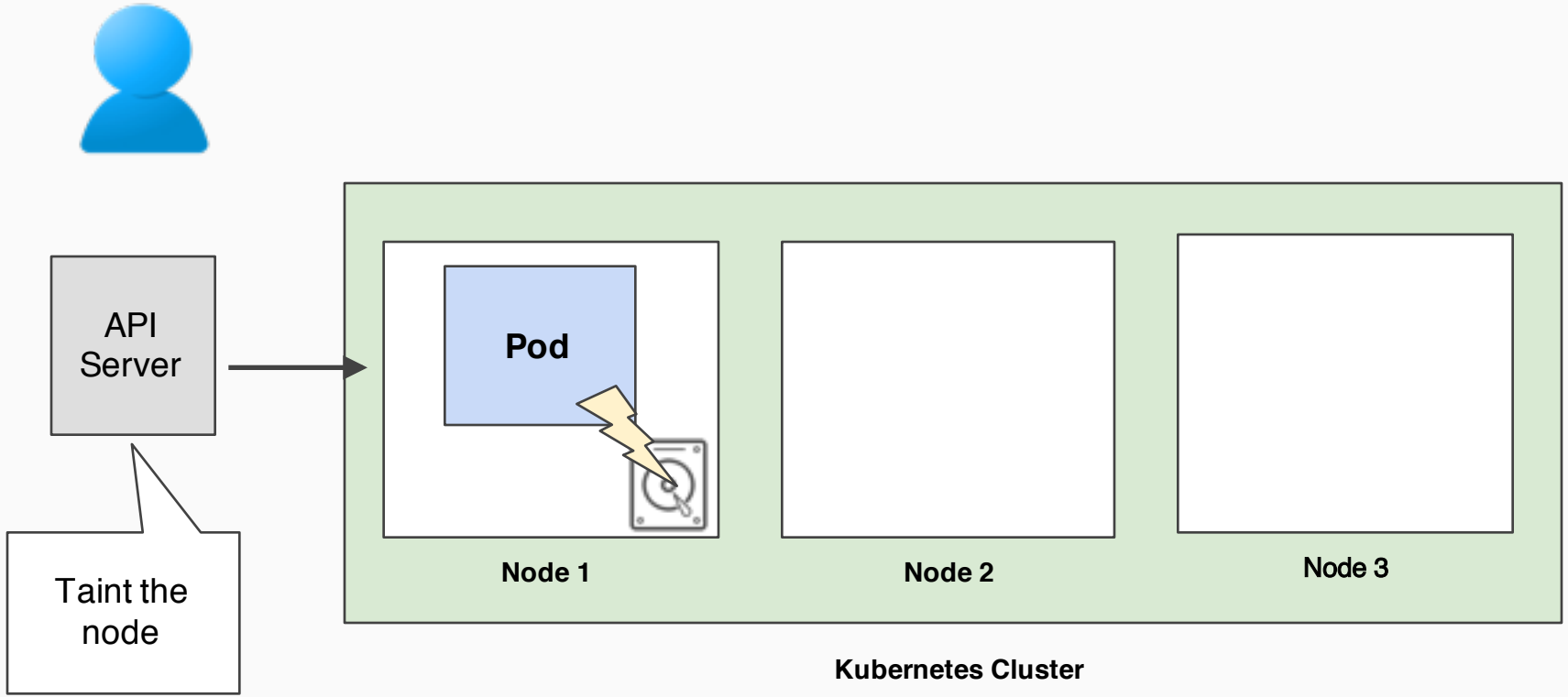
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)



Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)

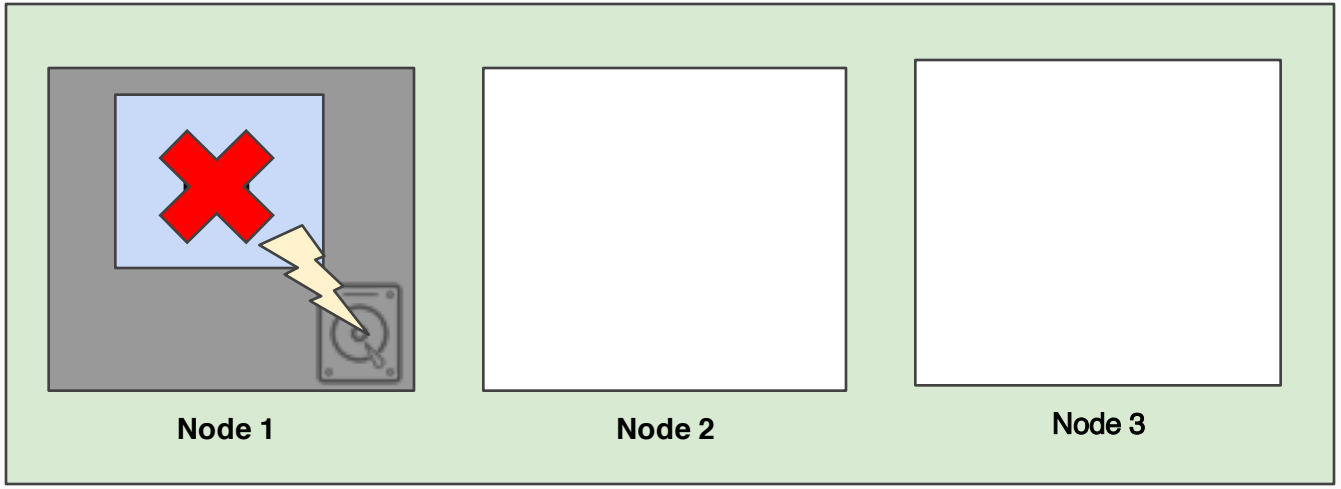


Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)

API Server

Taint the node

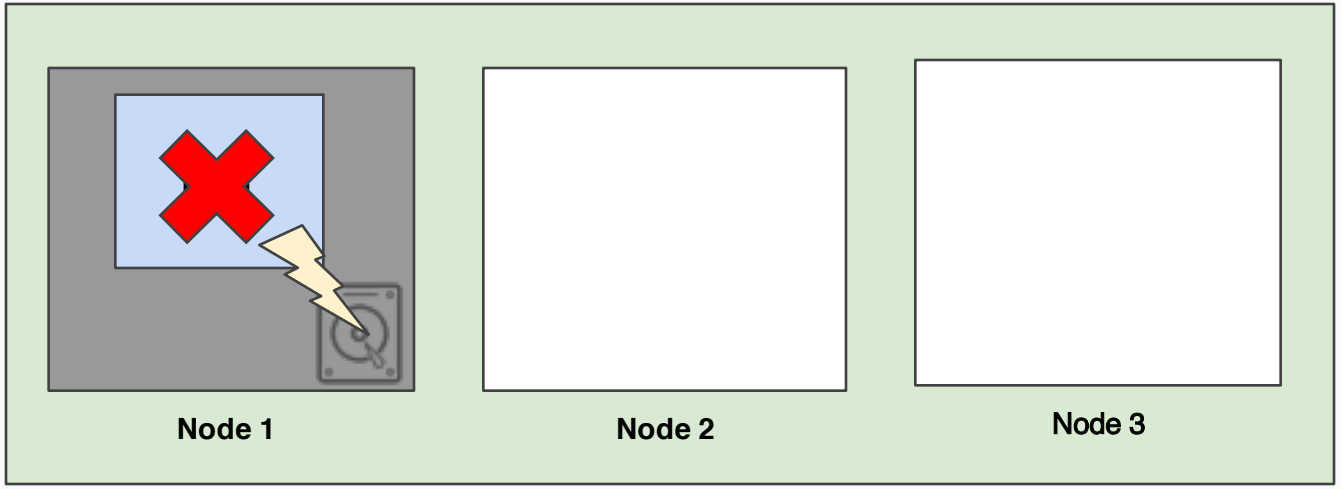


Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)

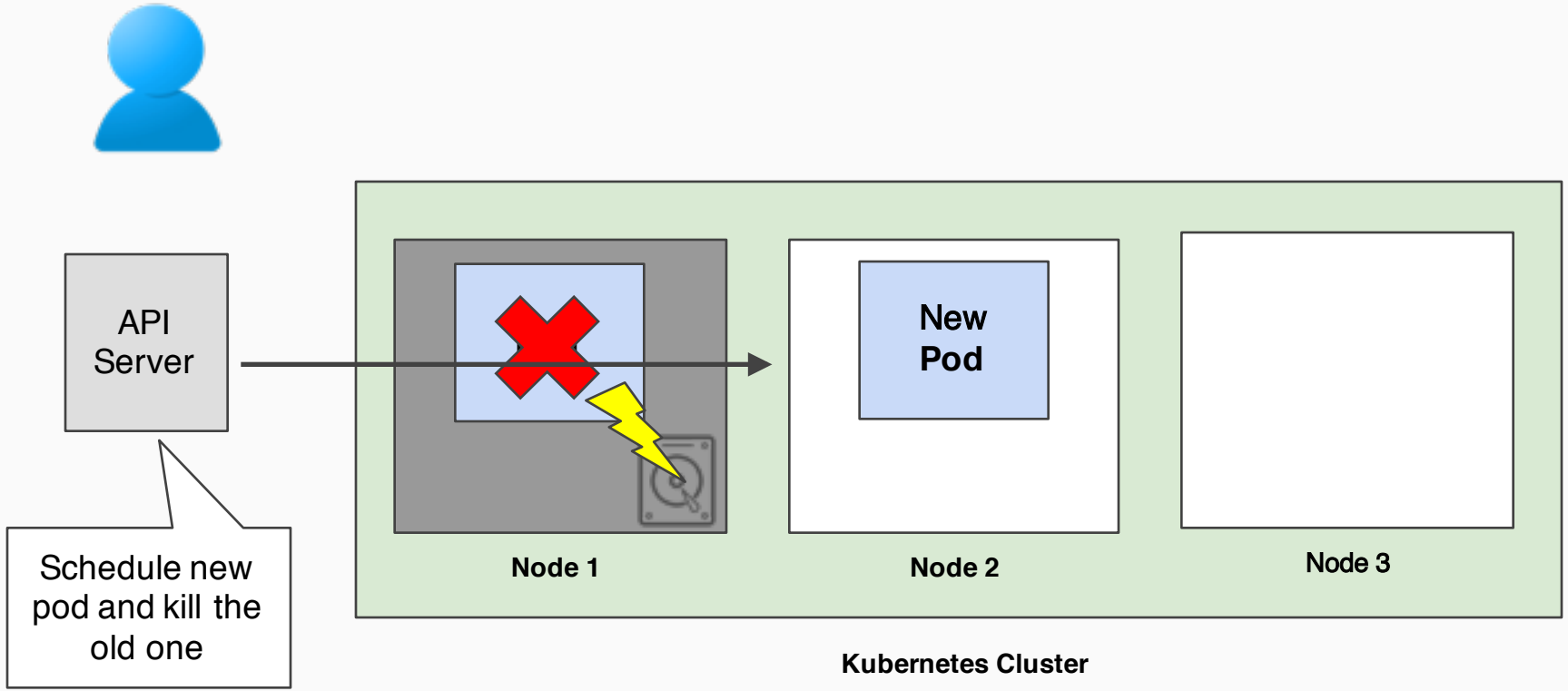
API Server

Schedule new pod and kill the old one



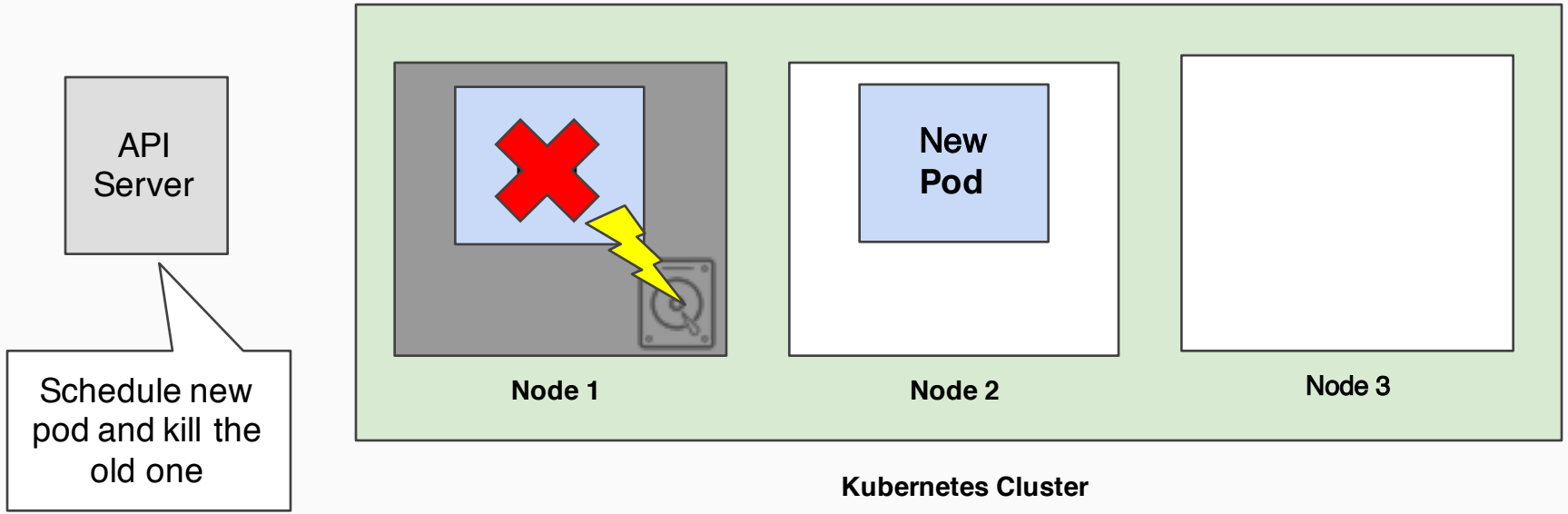
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)



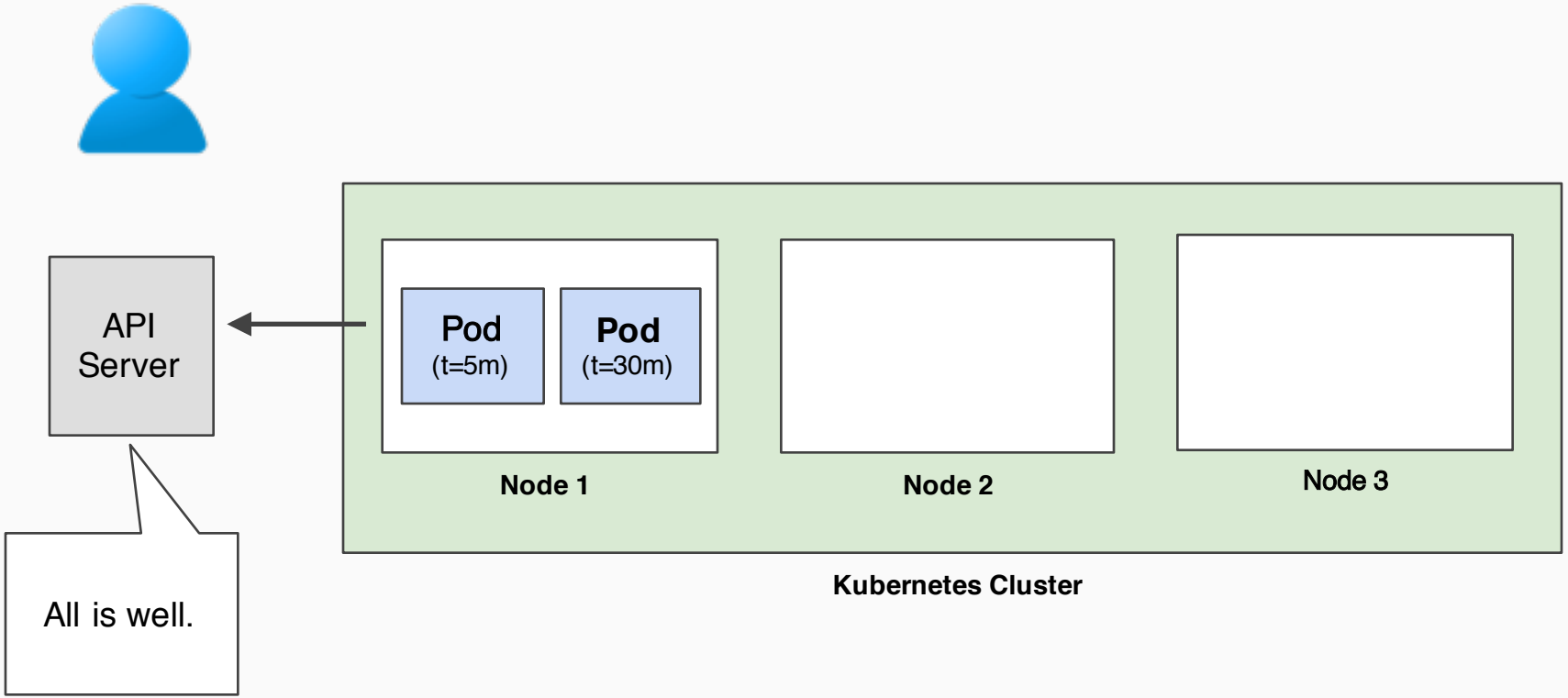
Sophisticated Scheduling: Taints/Toleration

SCENARIO: Hardware failing (but not failed)



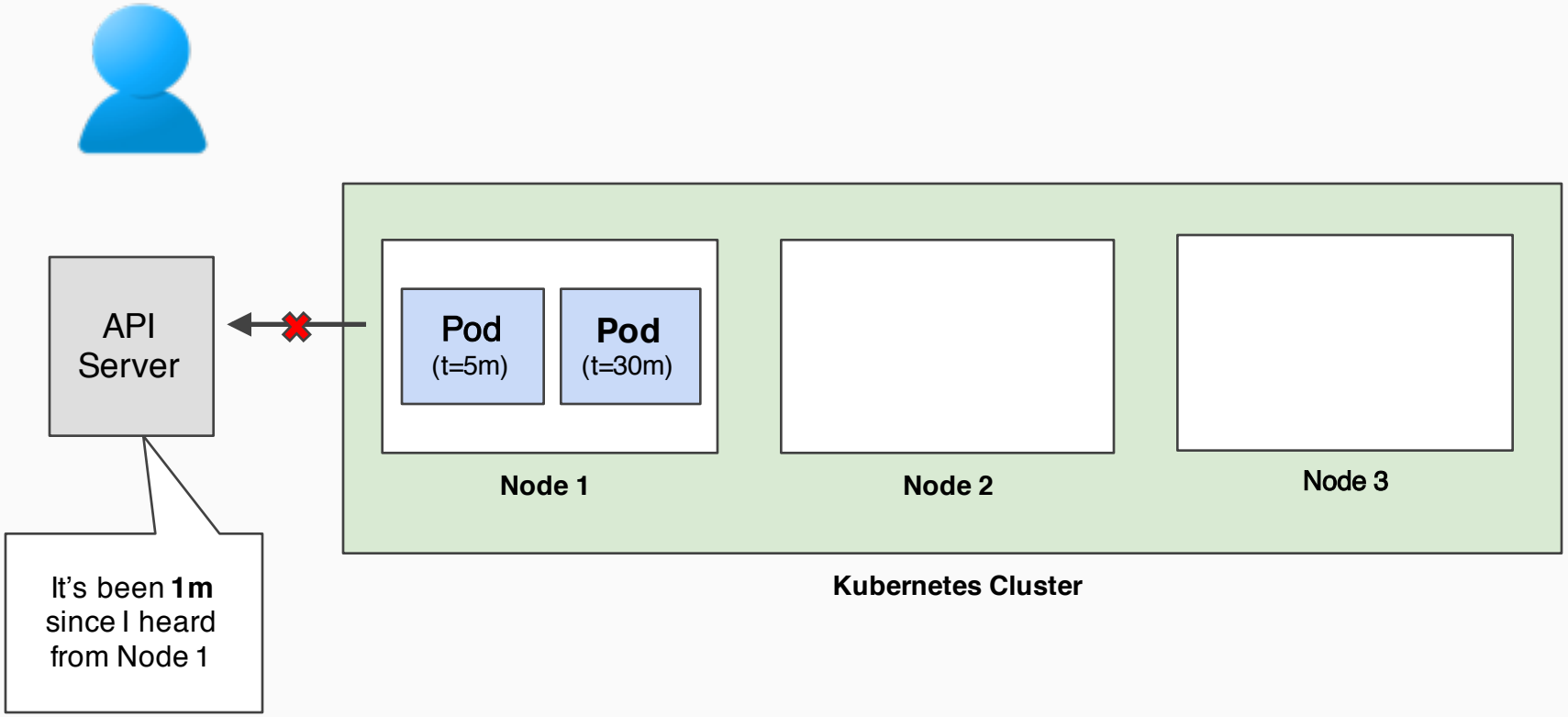
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



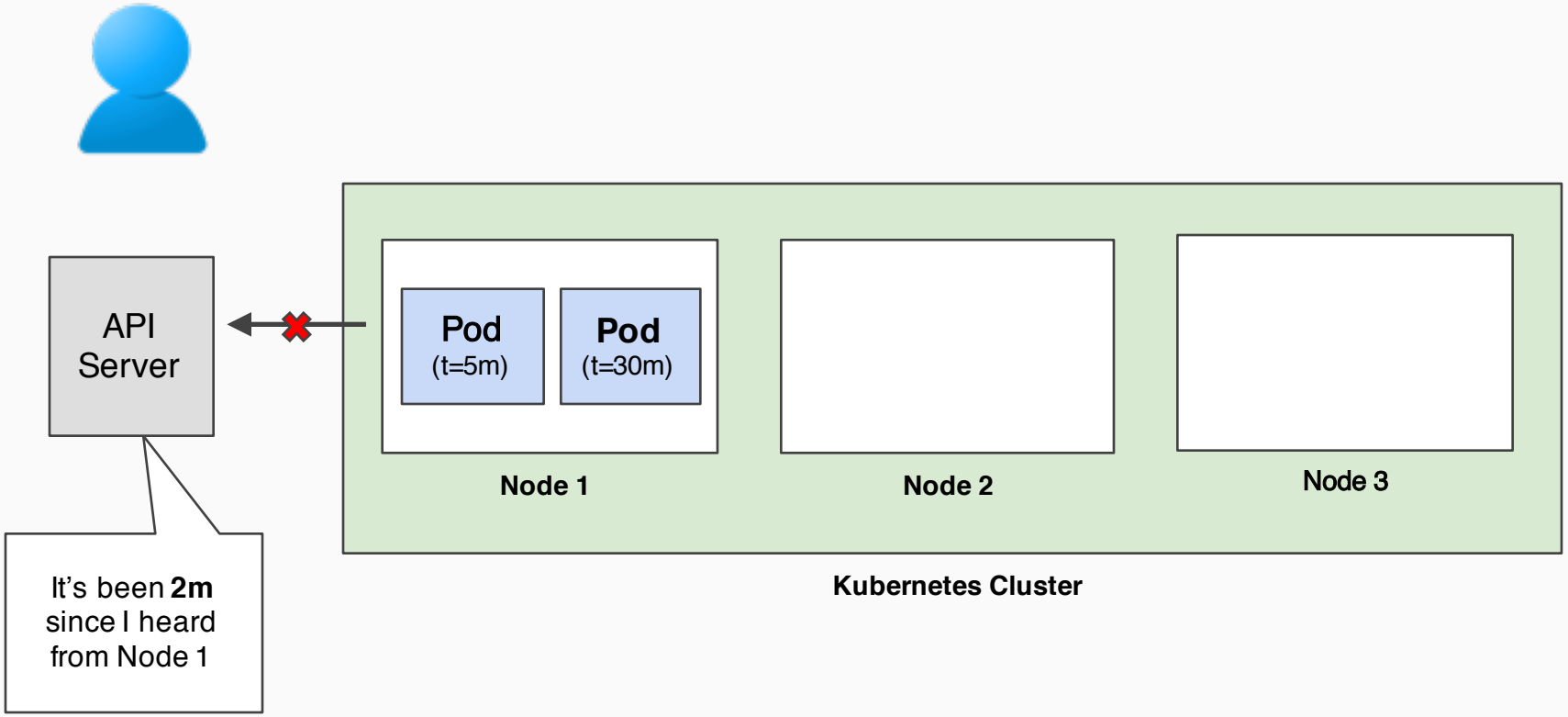
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



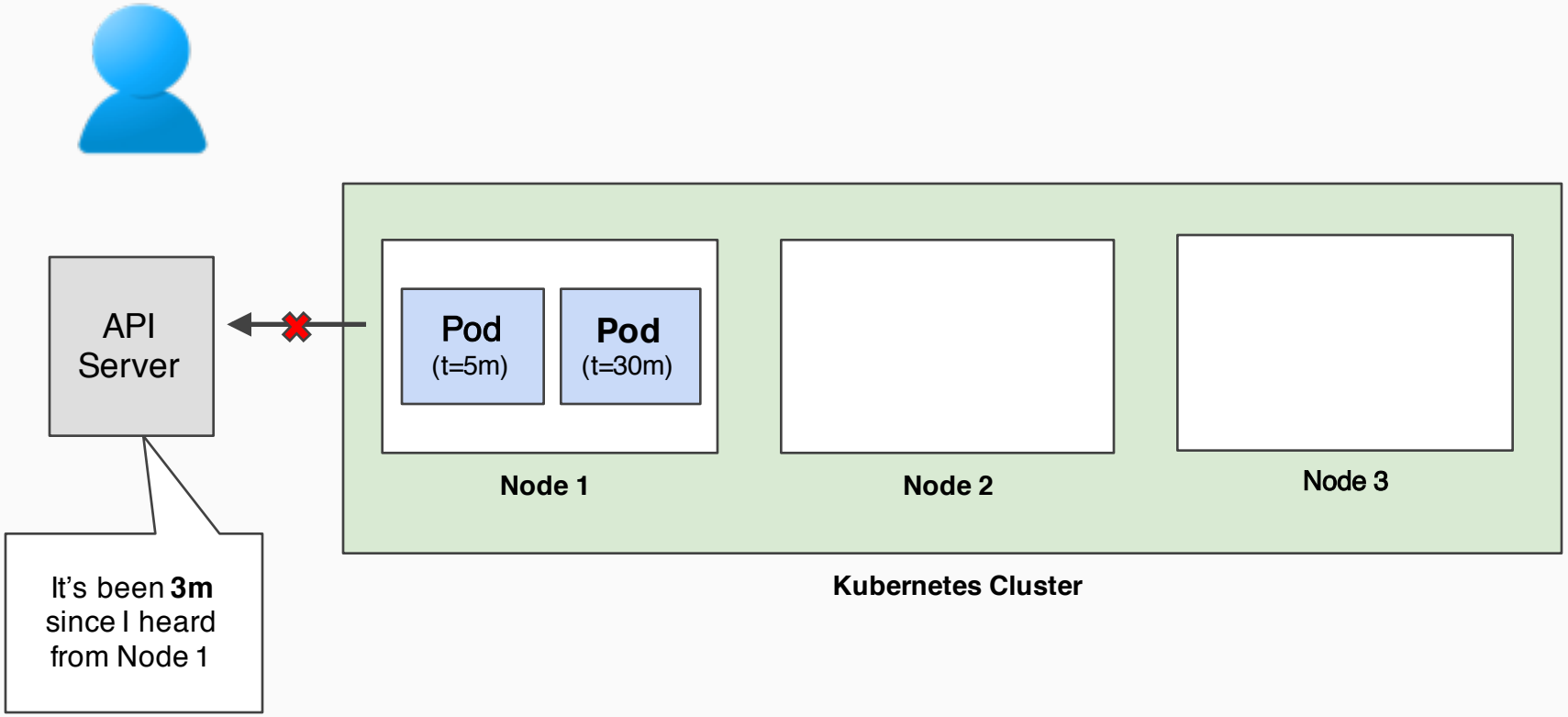
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



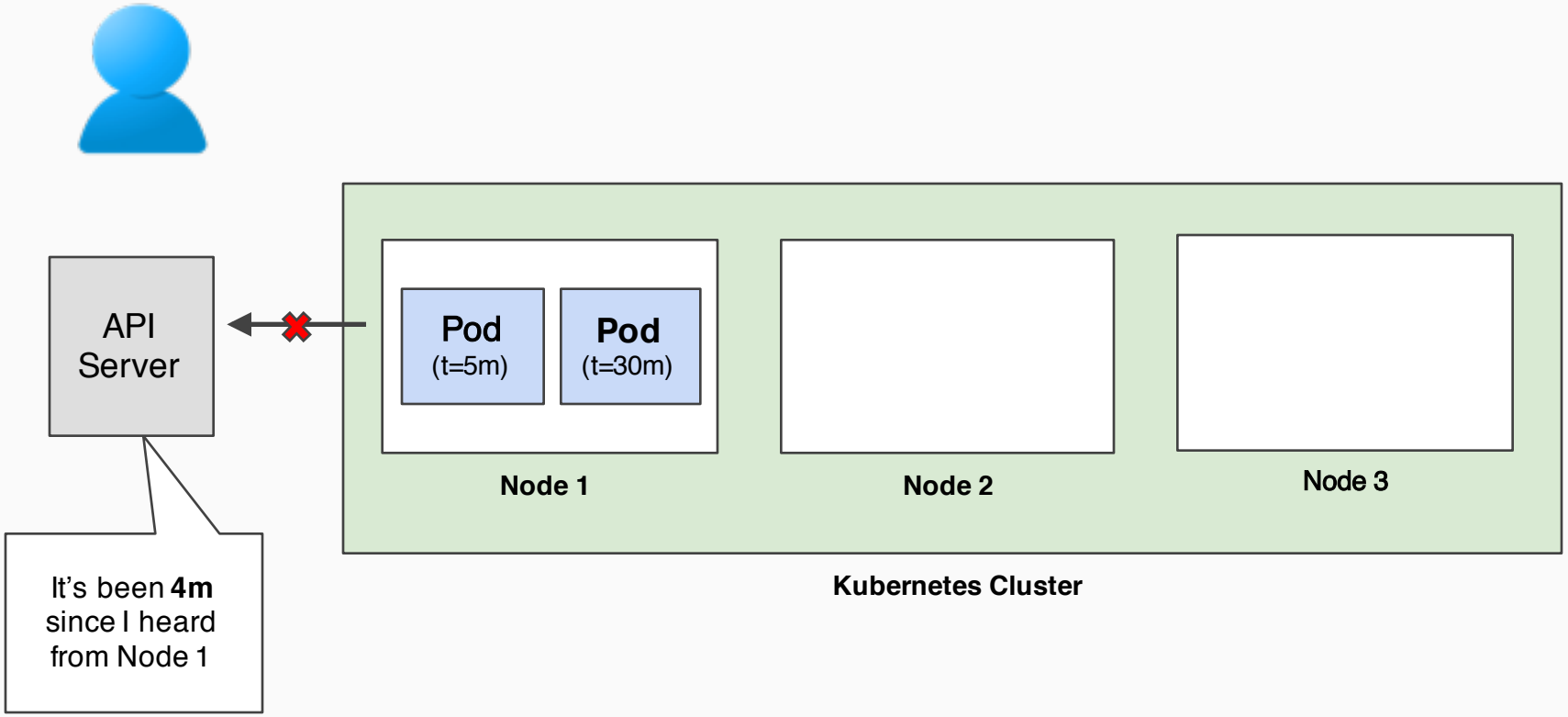
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



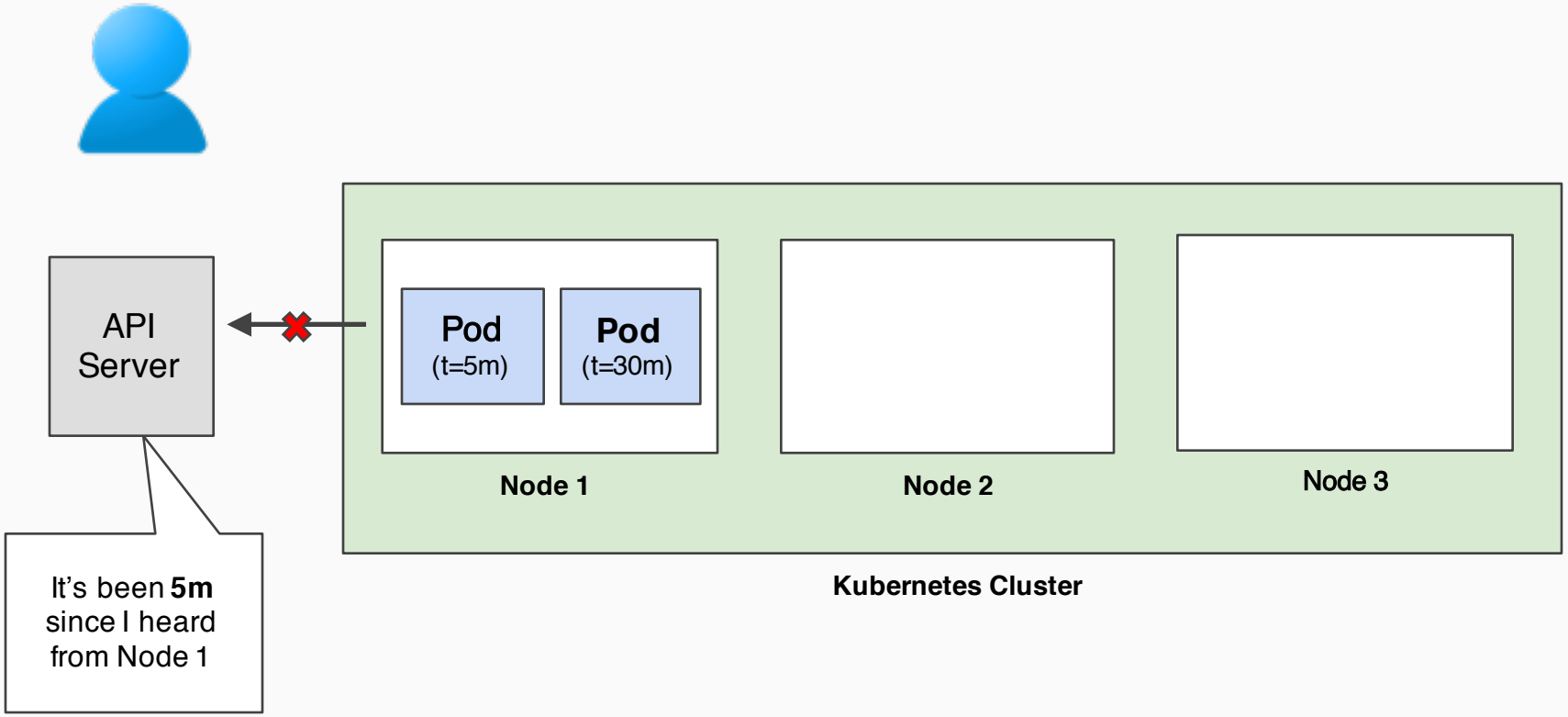
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



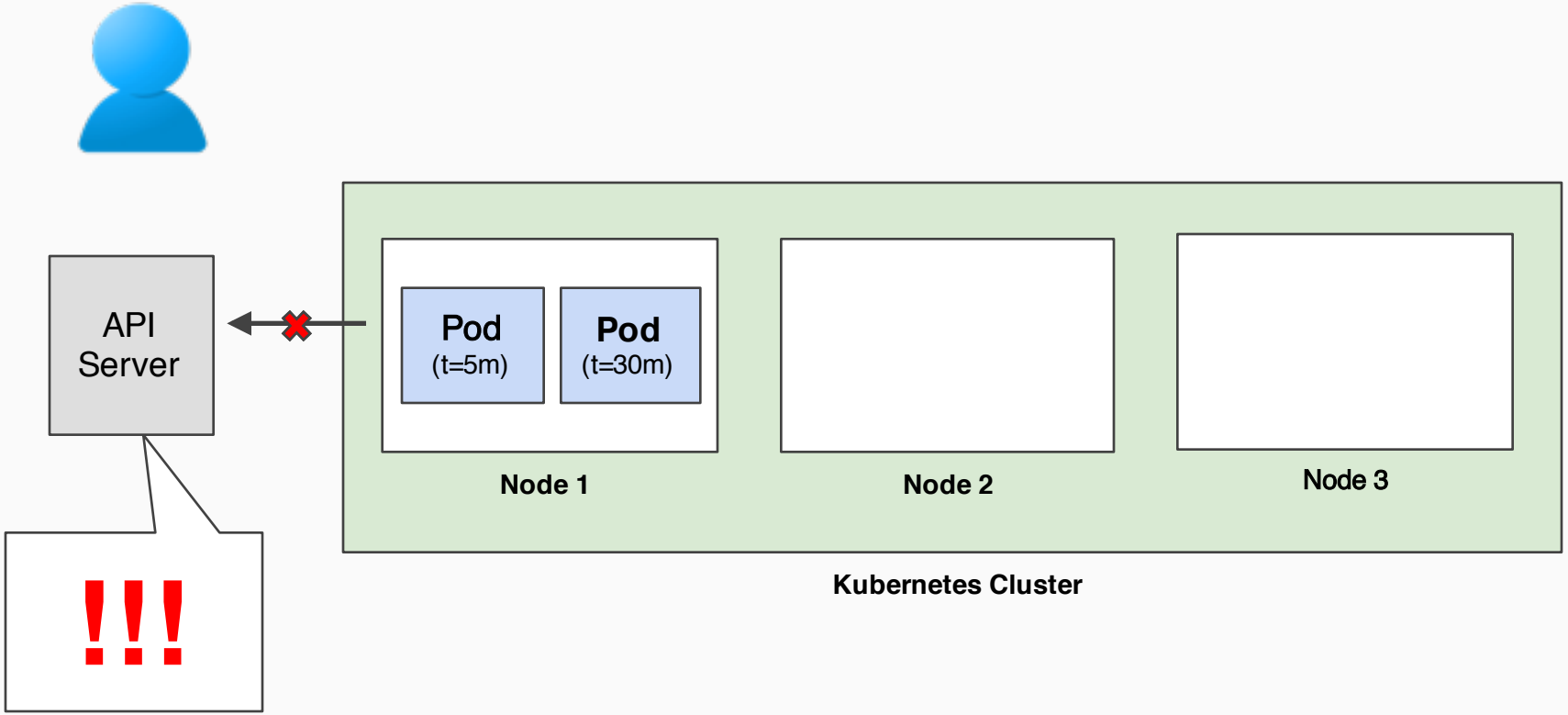
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



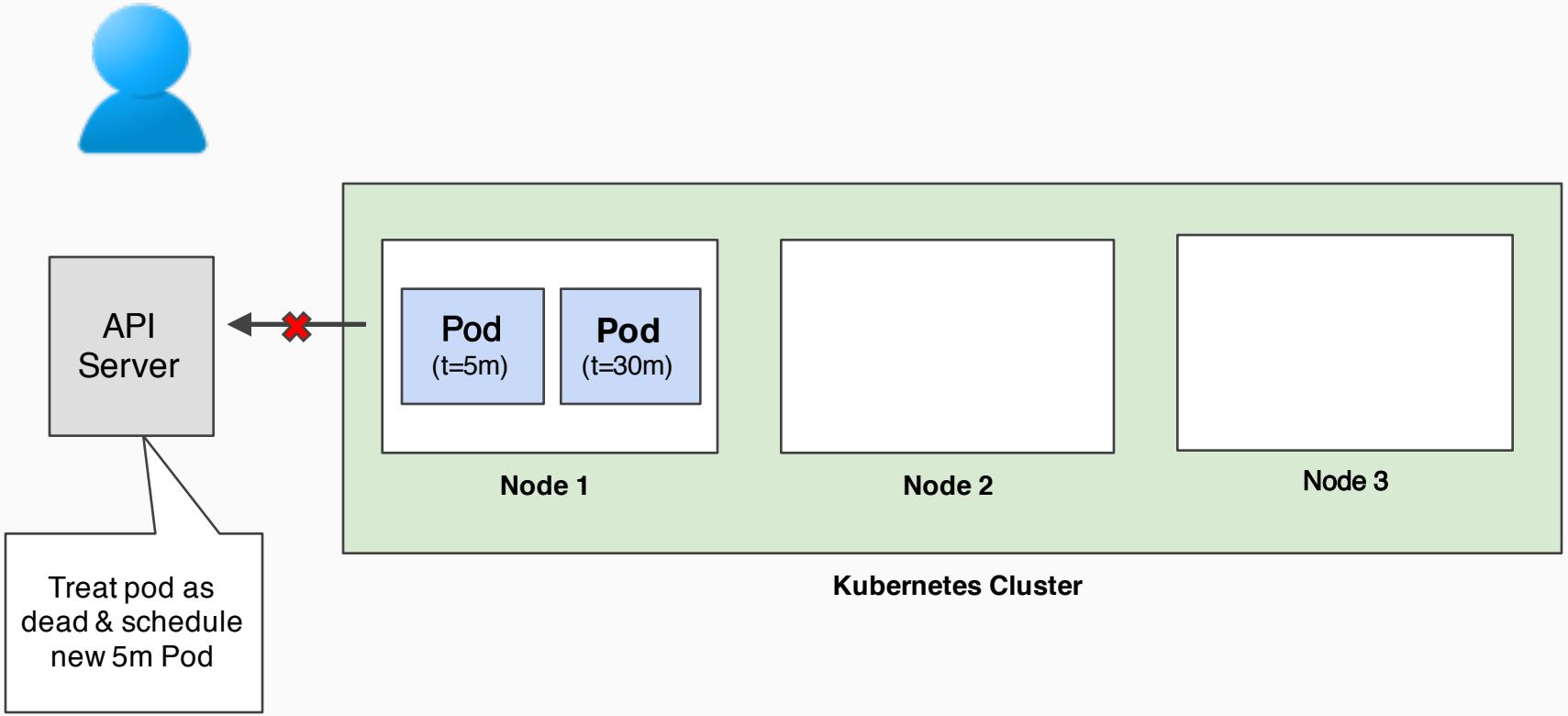
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



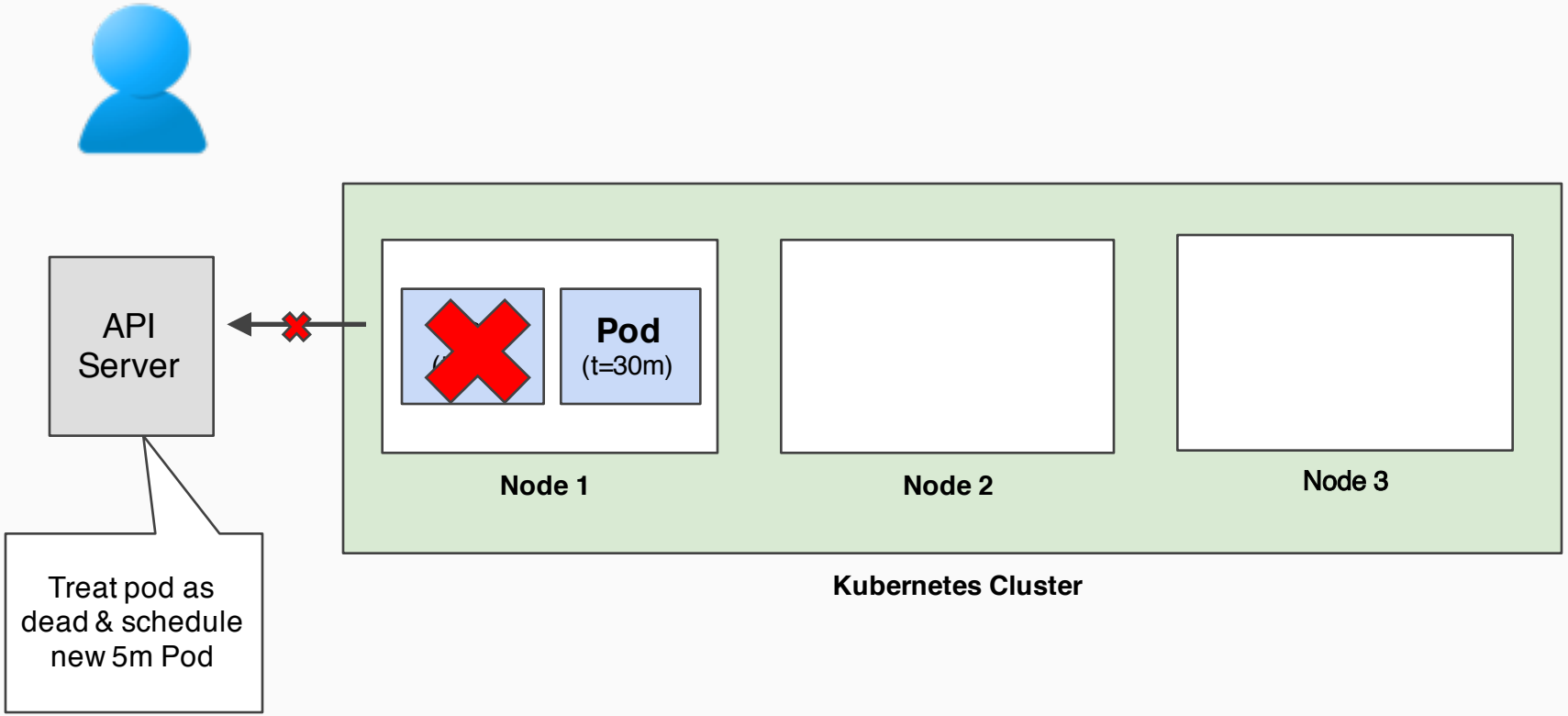
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



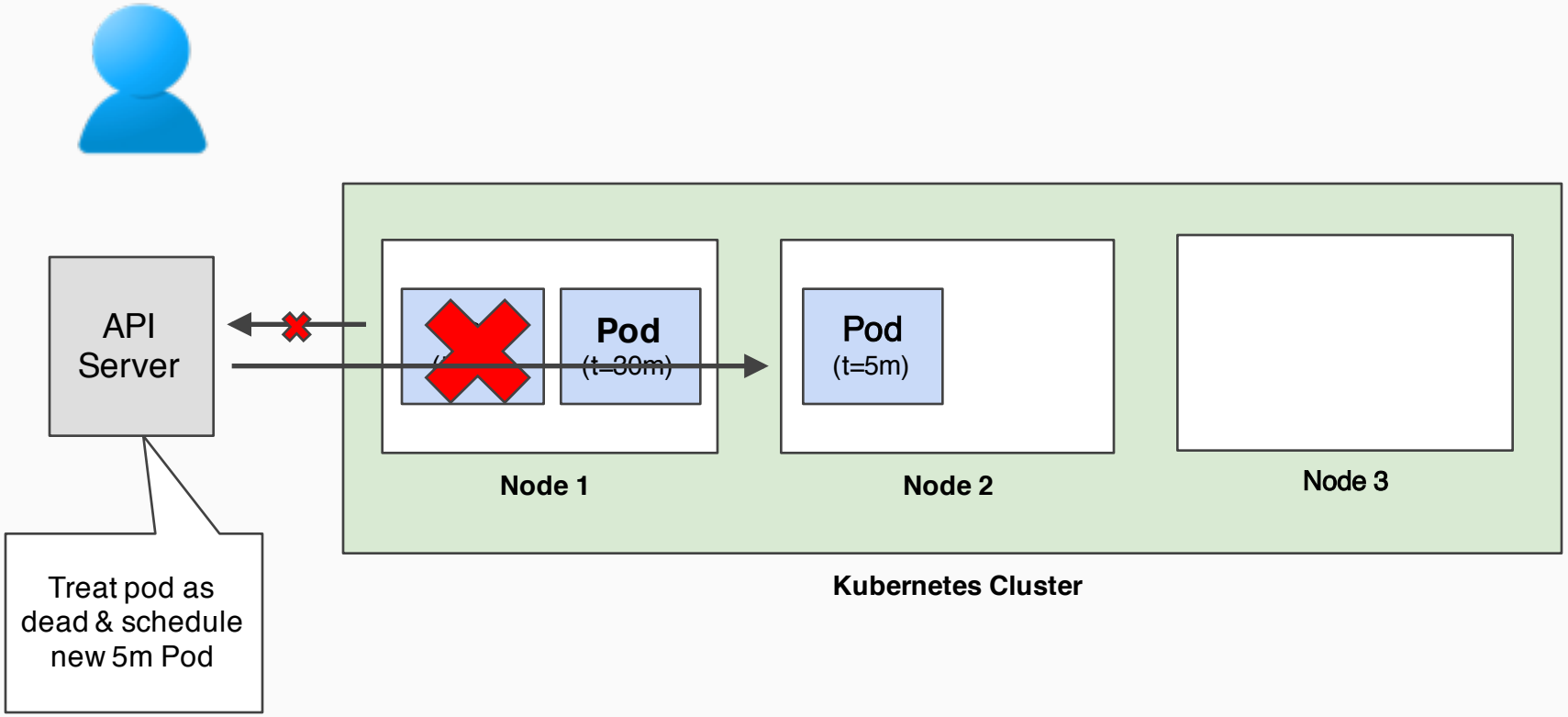
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



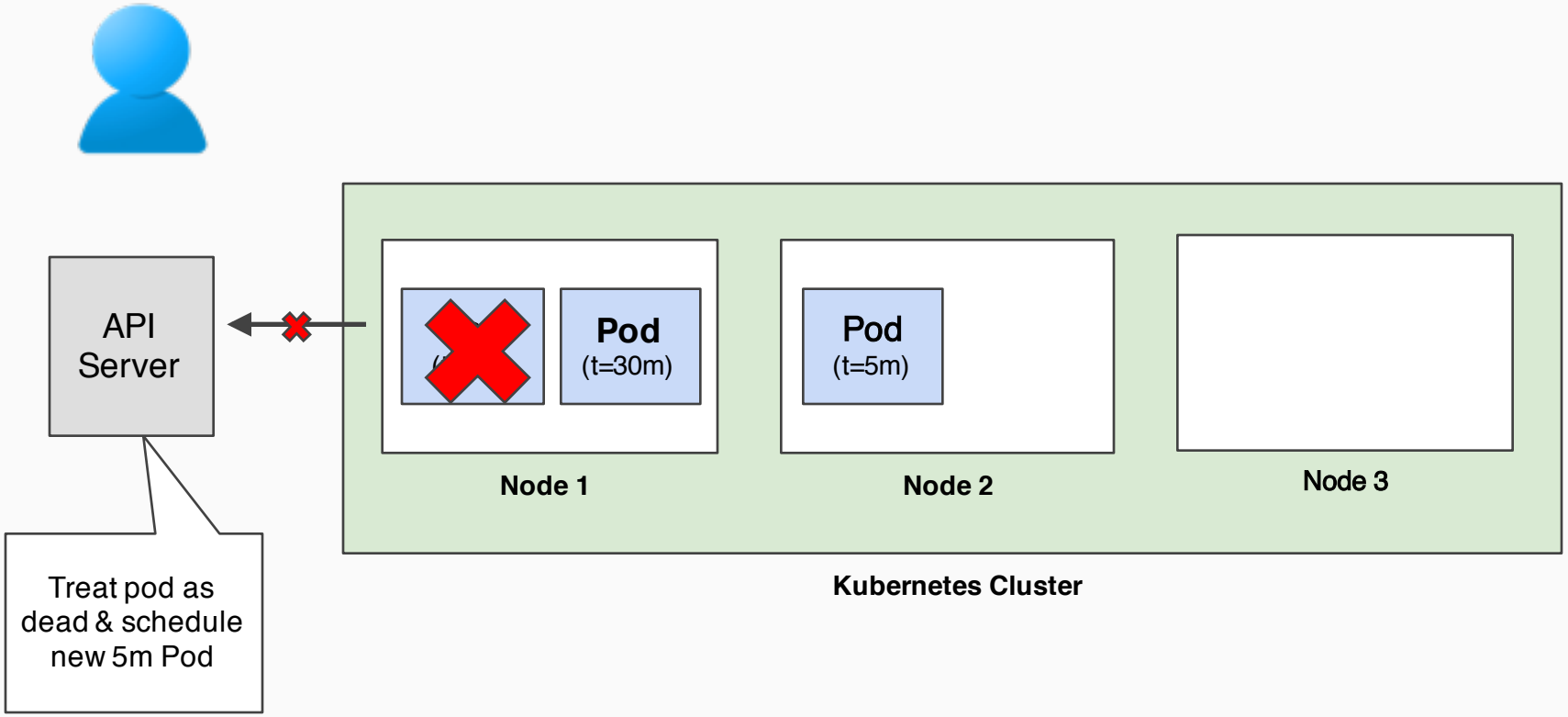
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



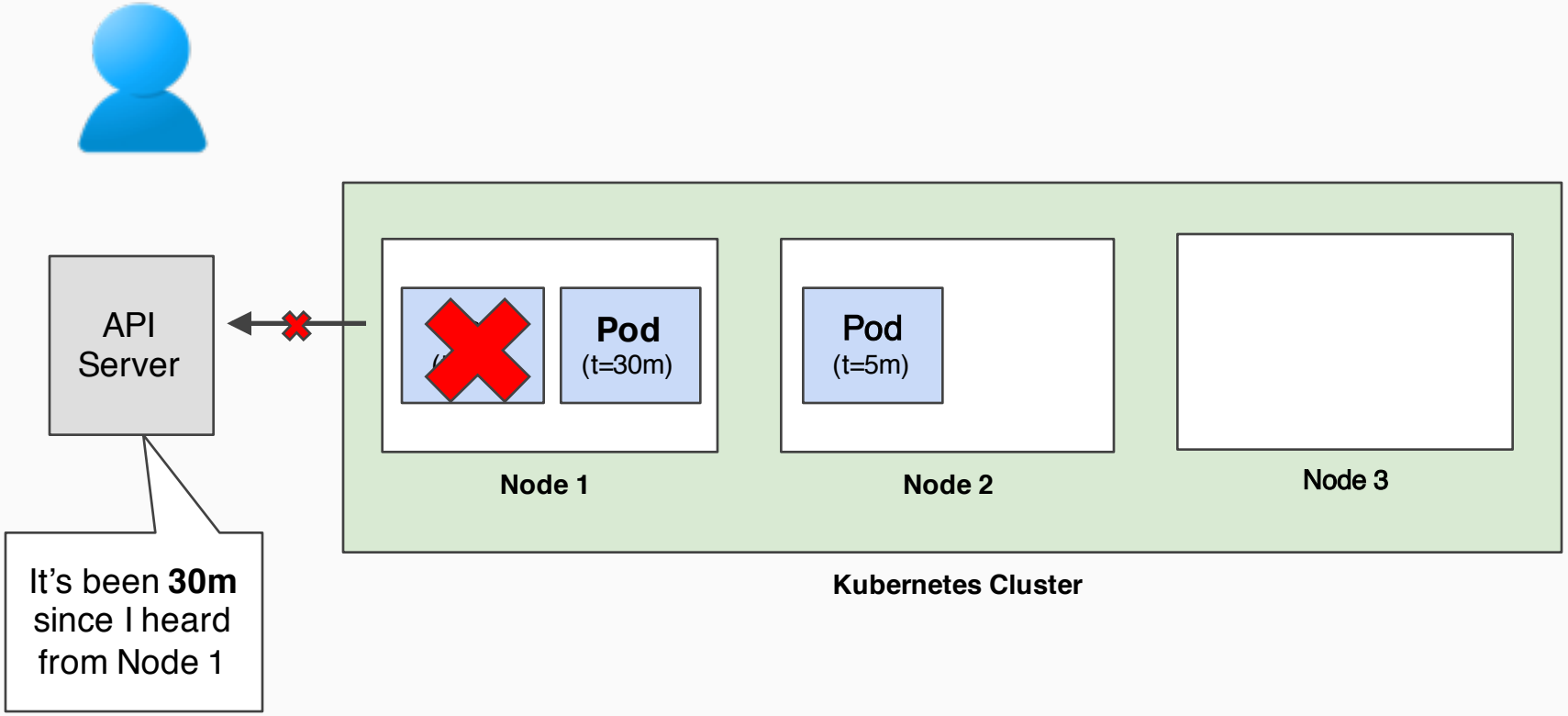
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



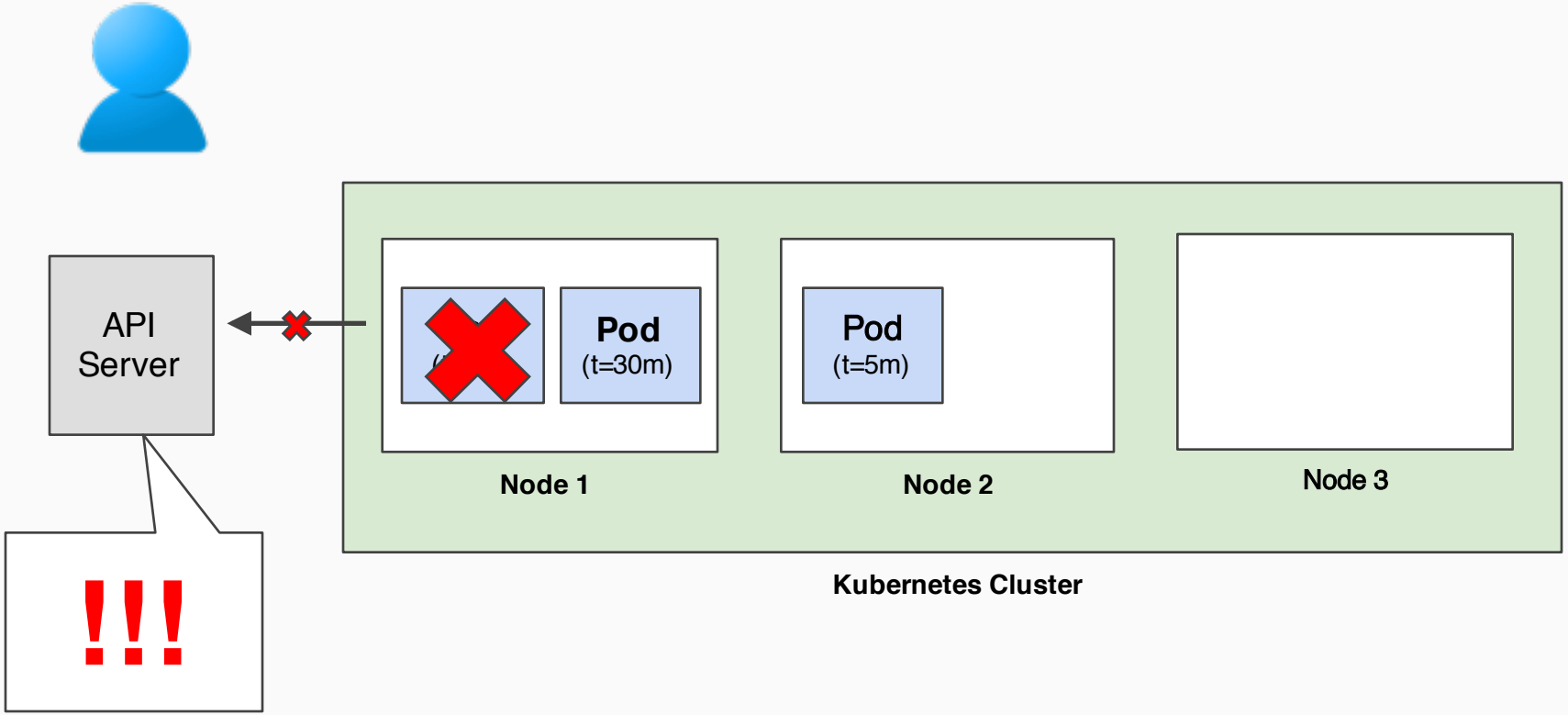
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



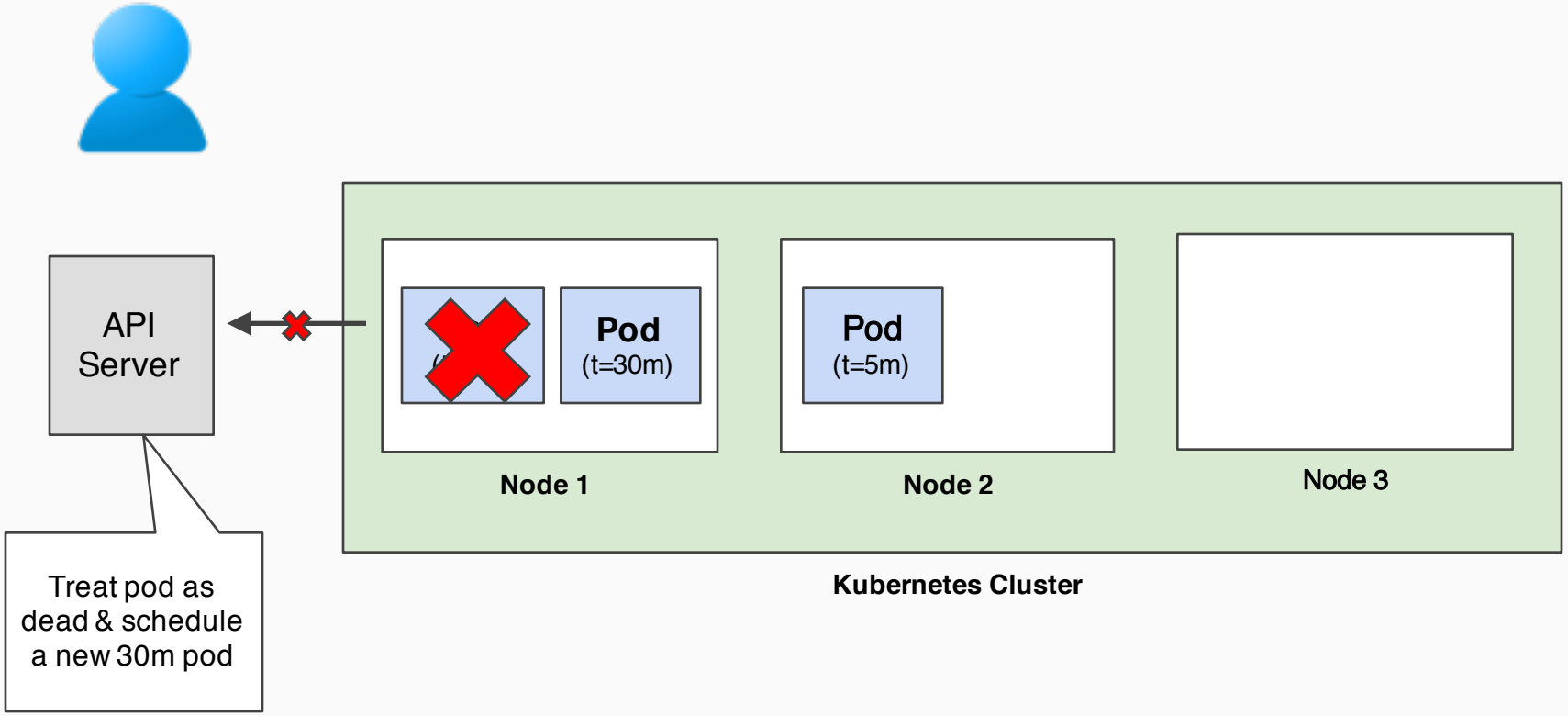
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



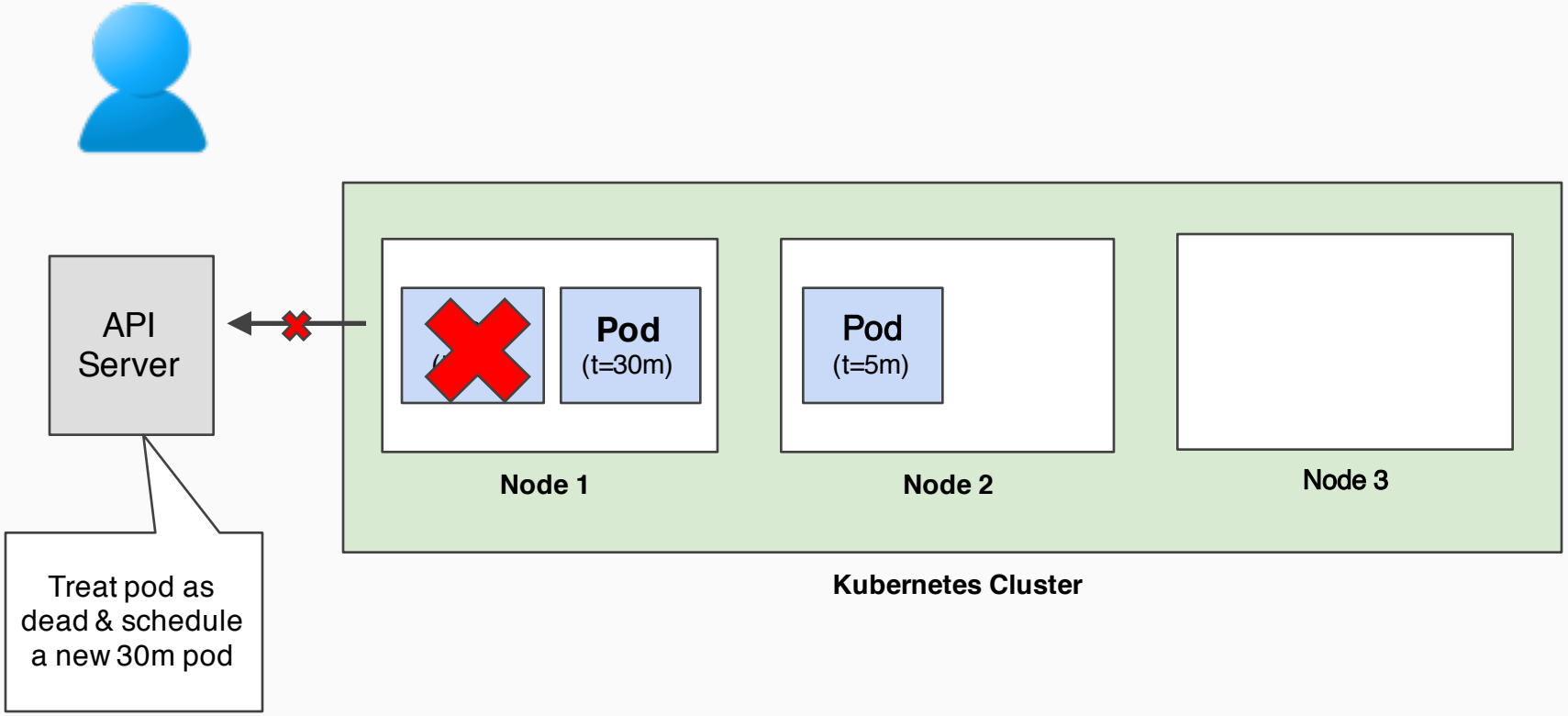
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



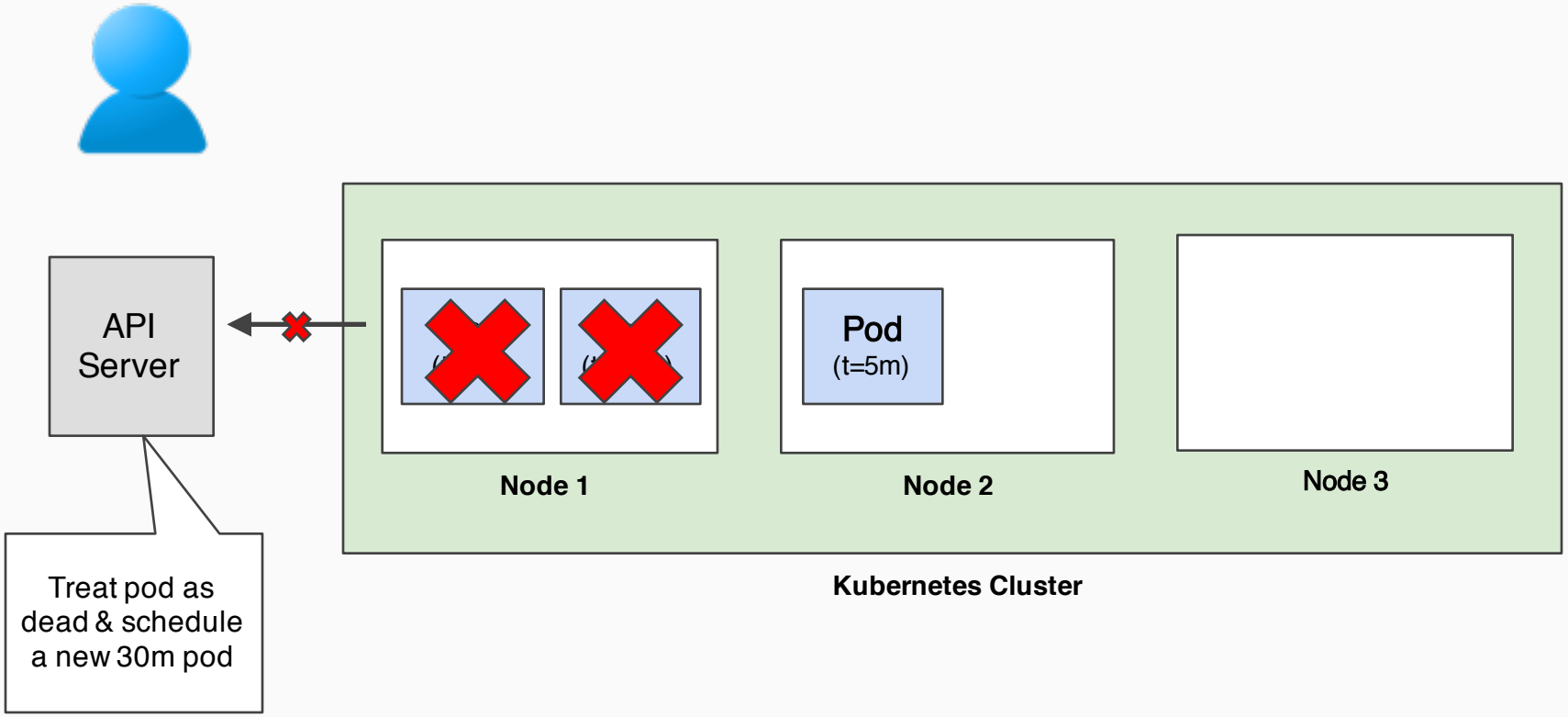
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



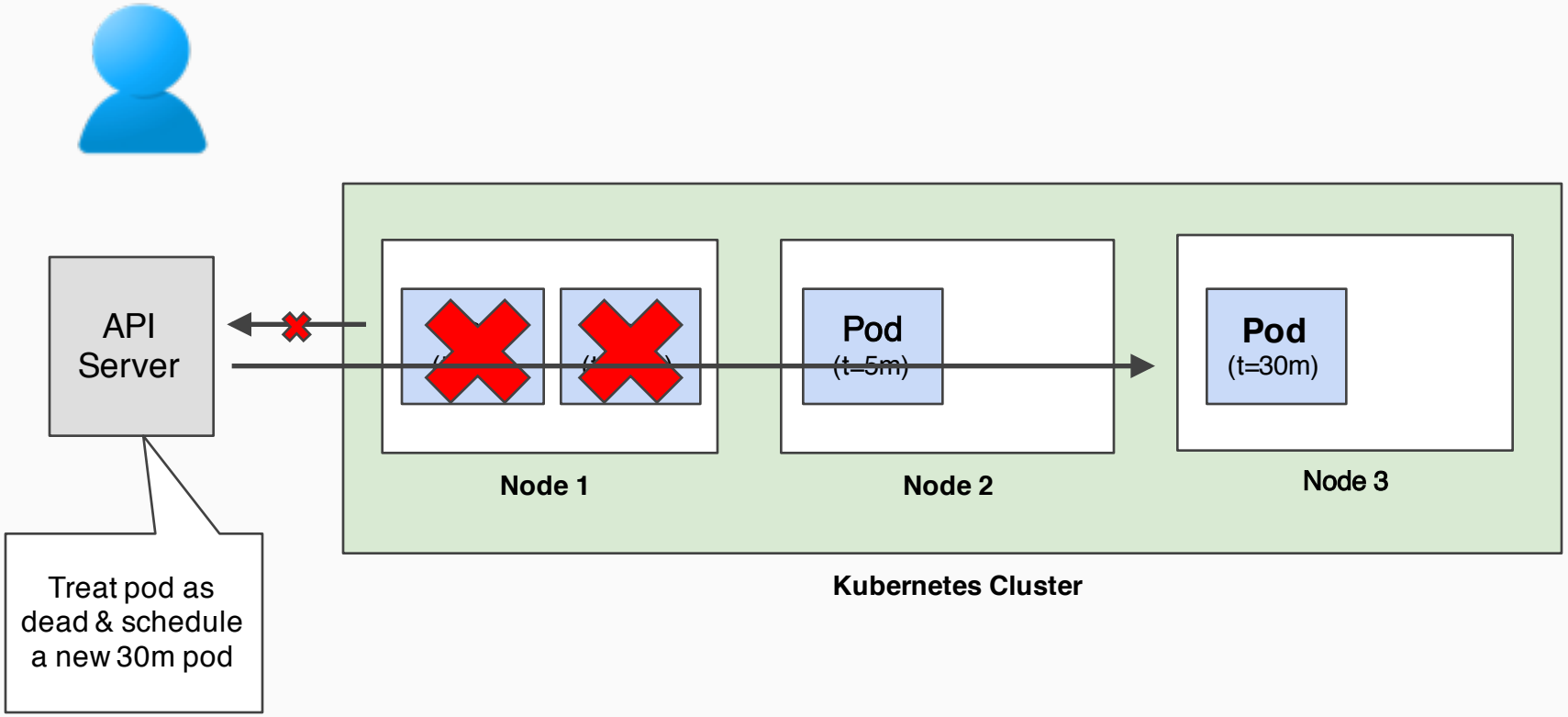
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



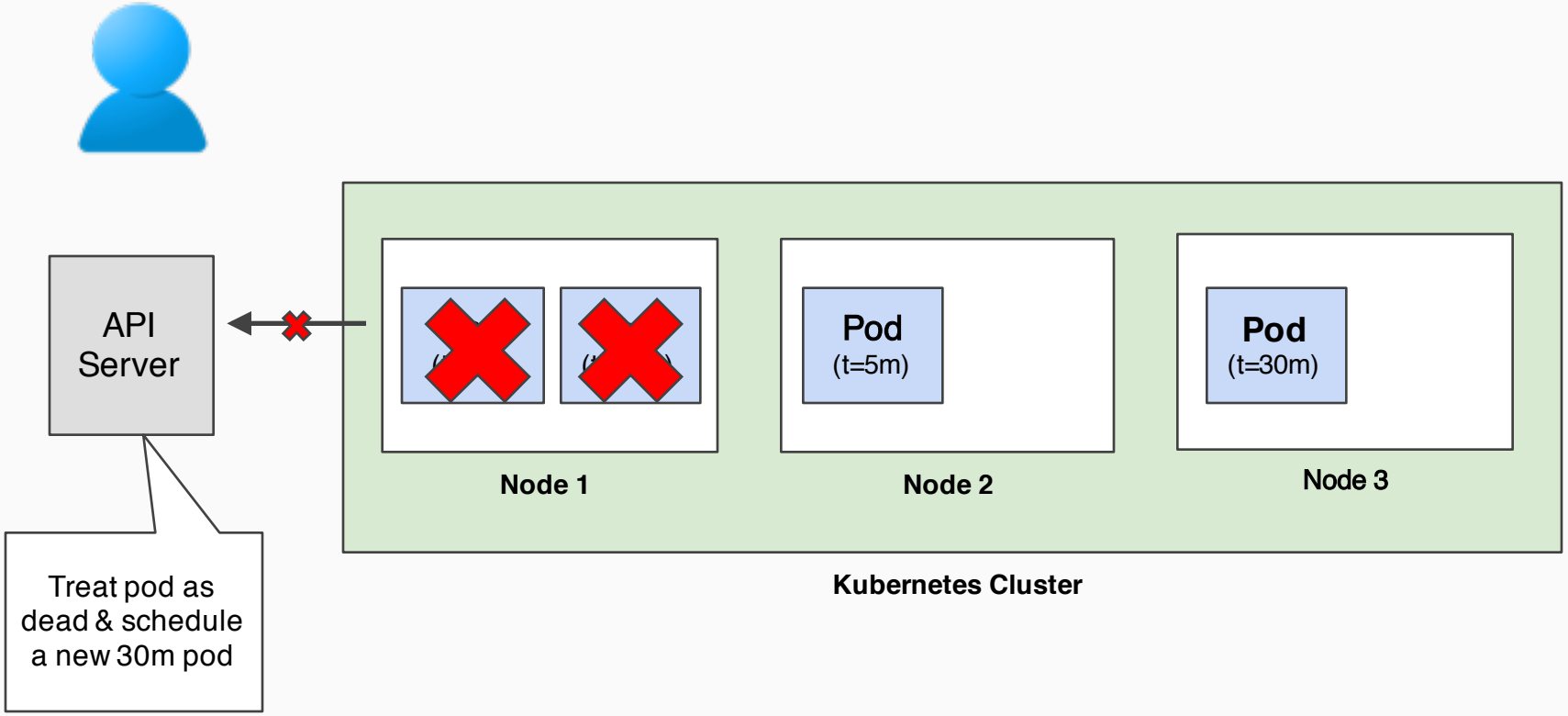
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



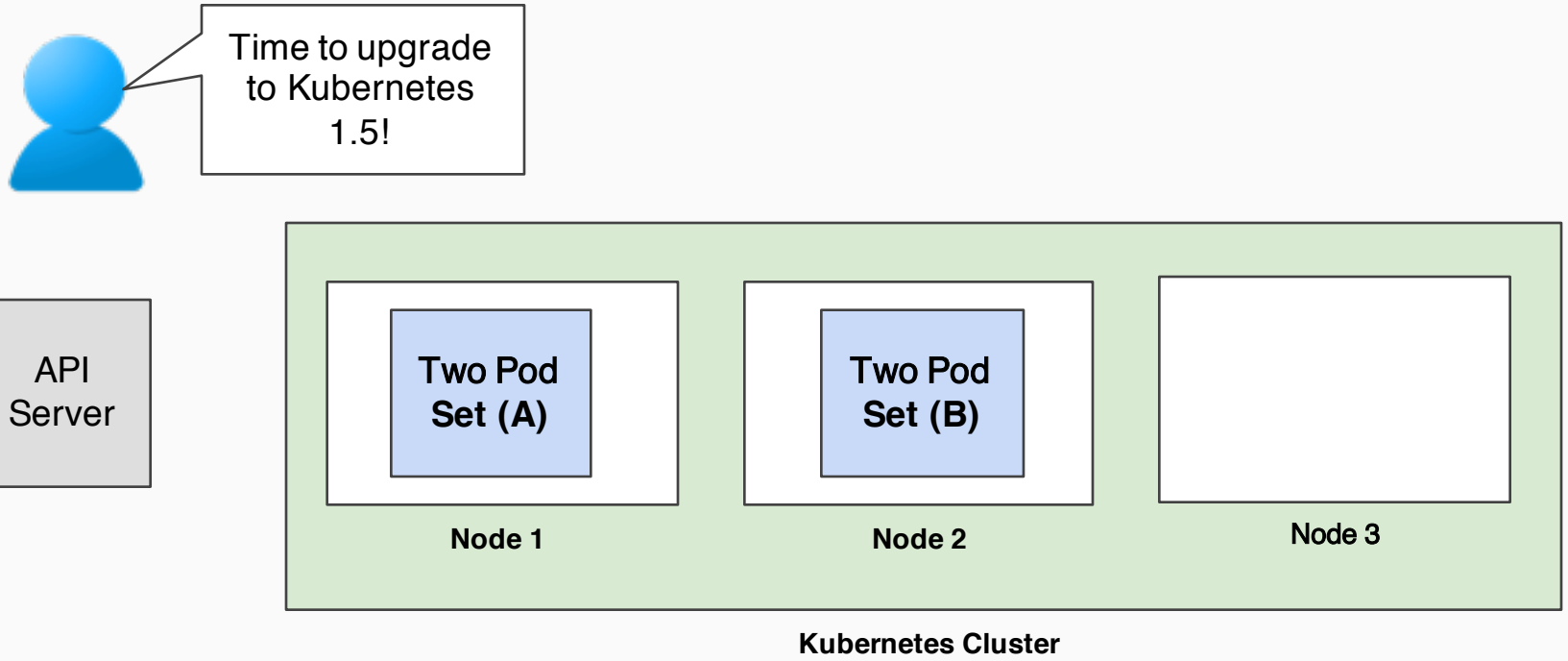
Sophisticated Scheduling: Forgiveness

SCENARIO: Supporting network failure



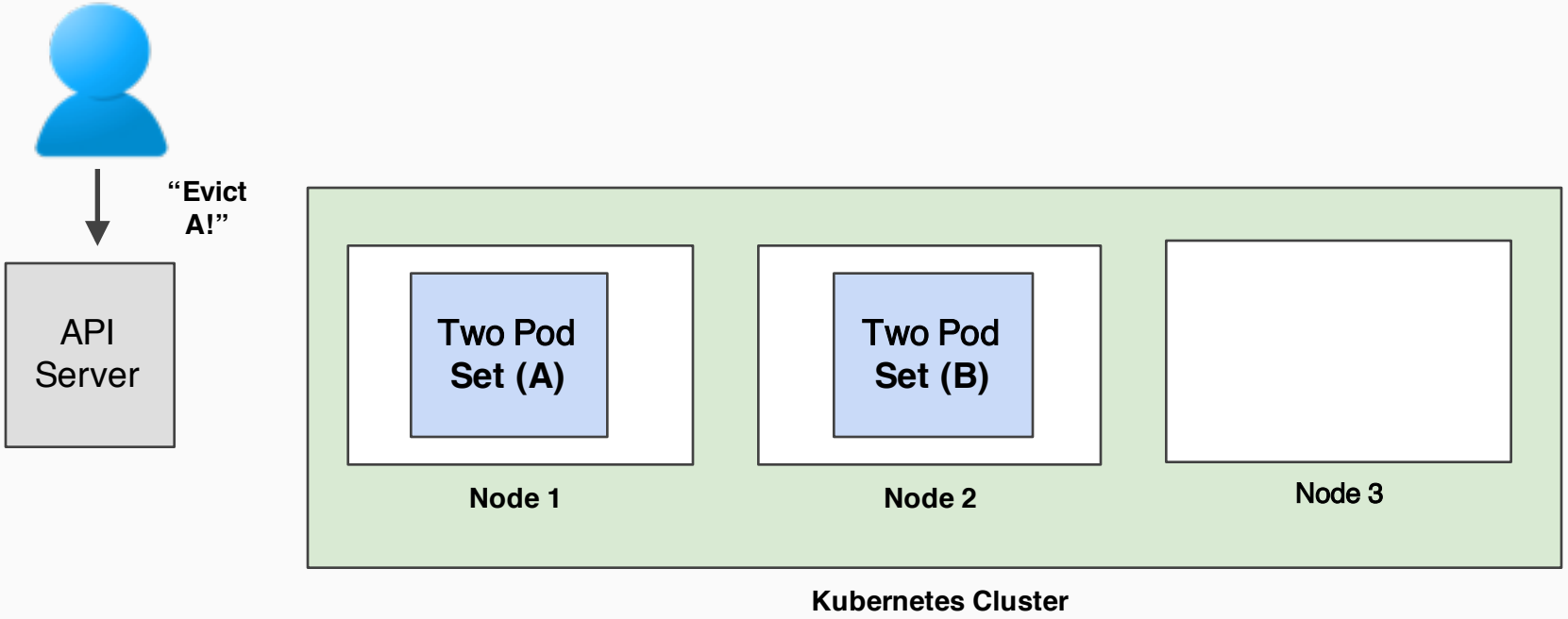
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



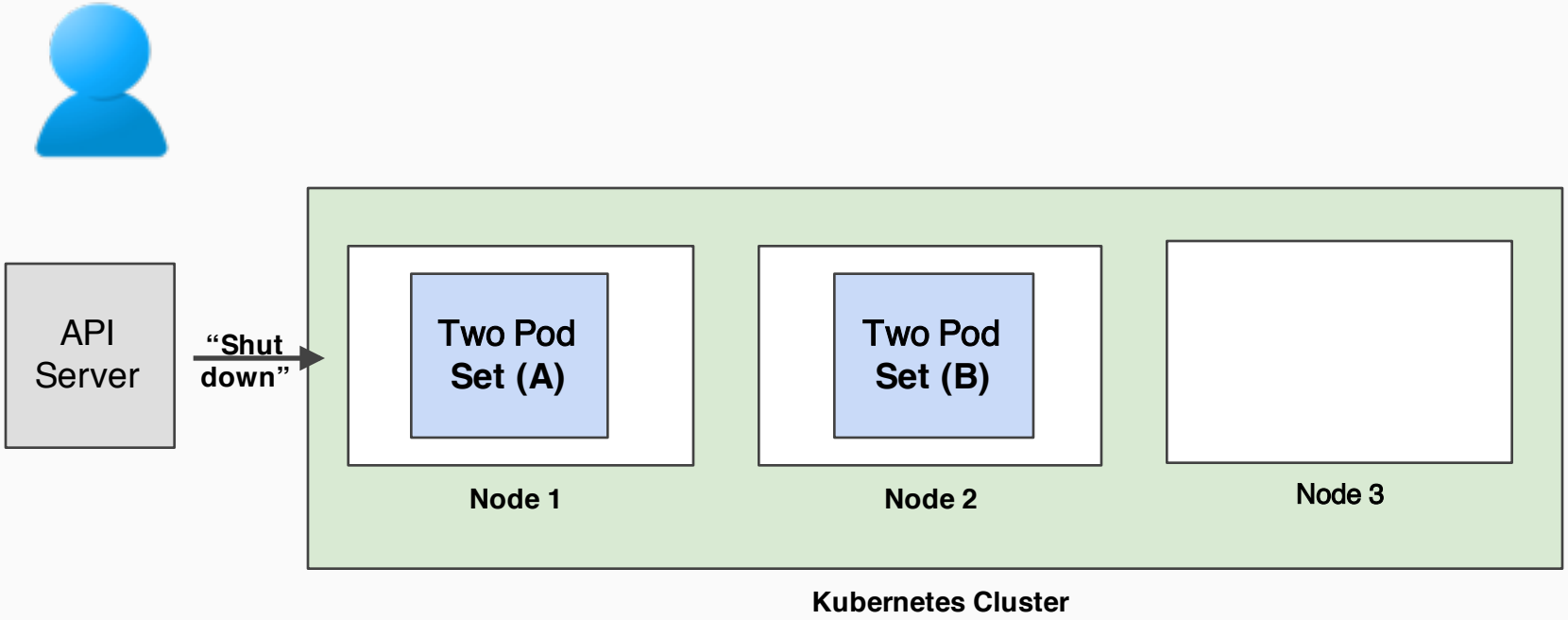
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



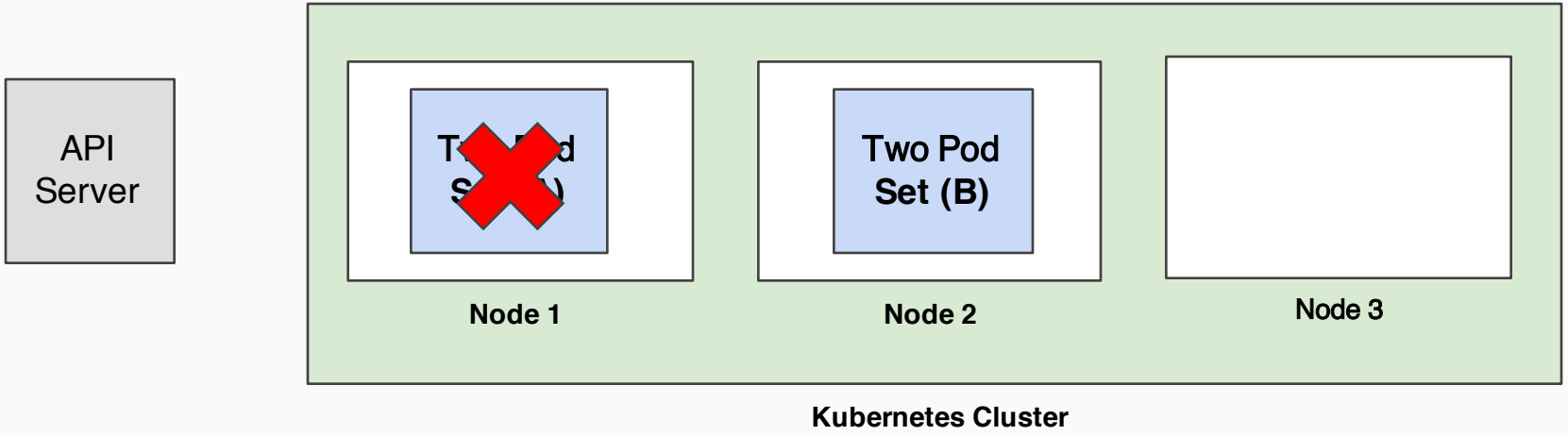
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



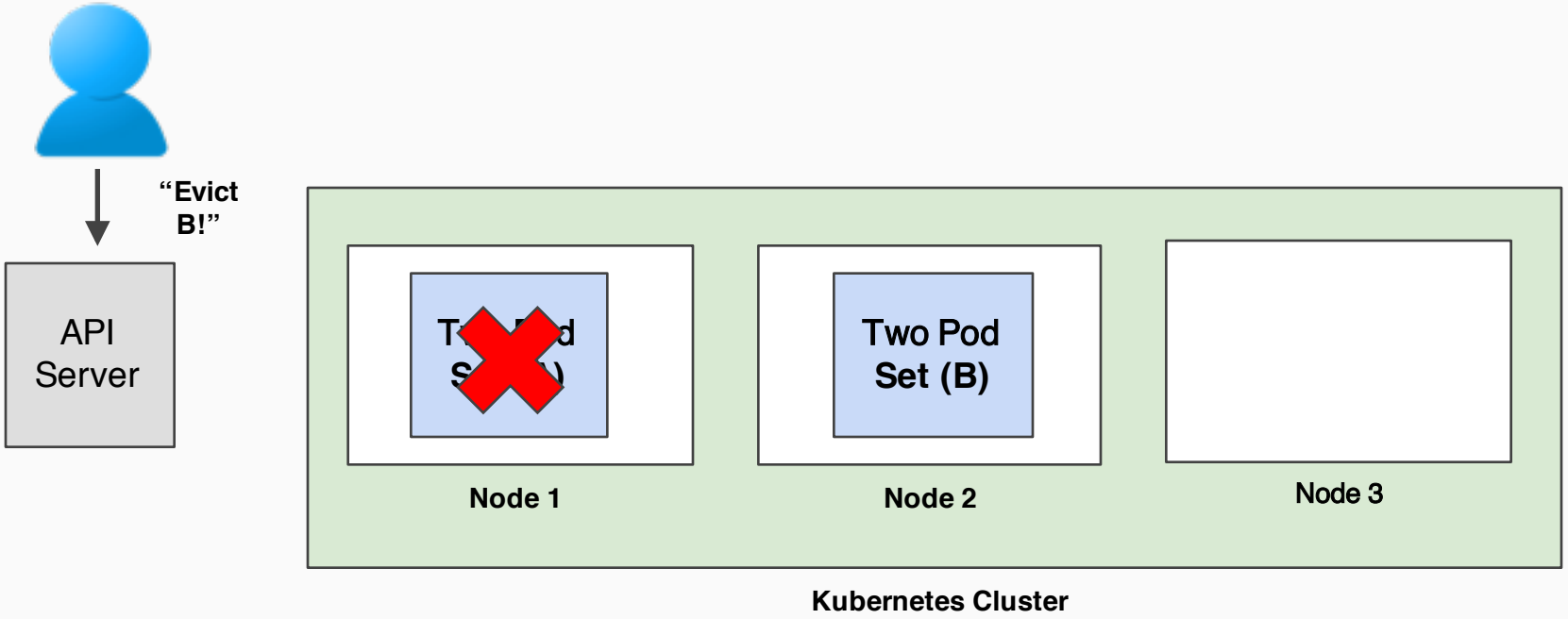
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



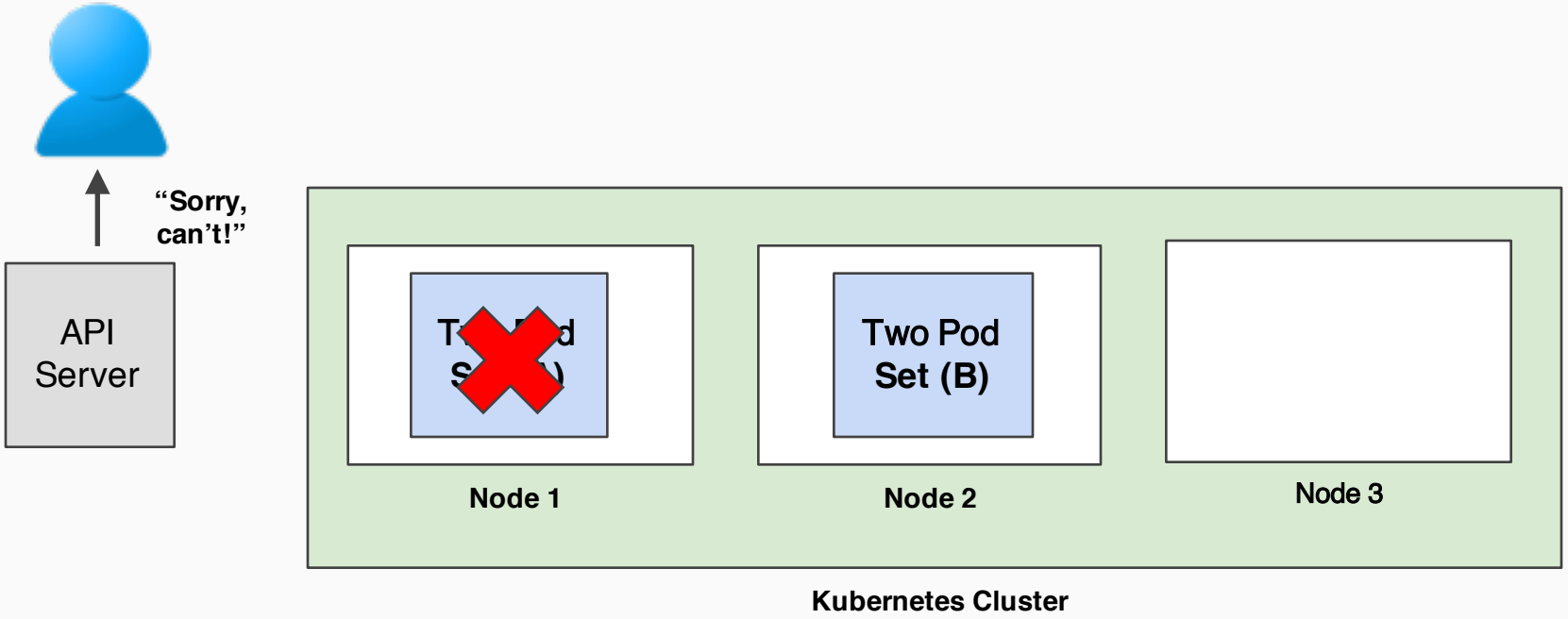
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



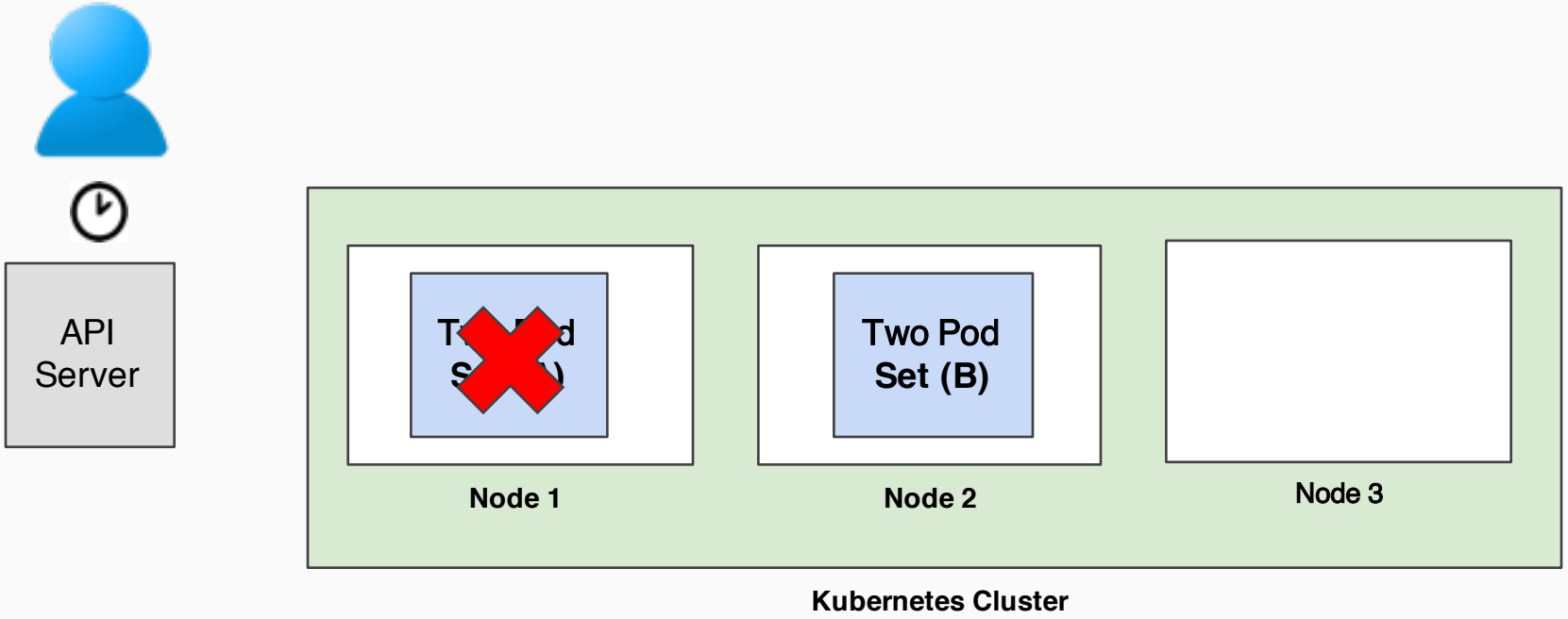
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



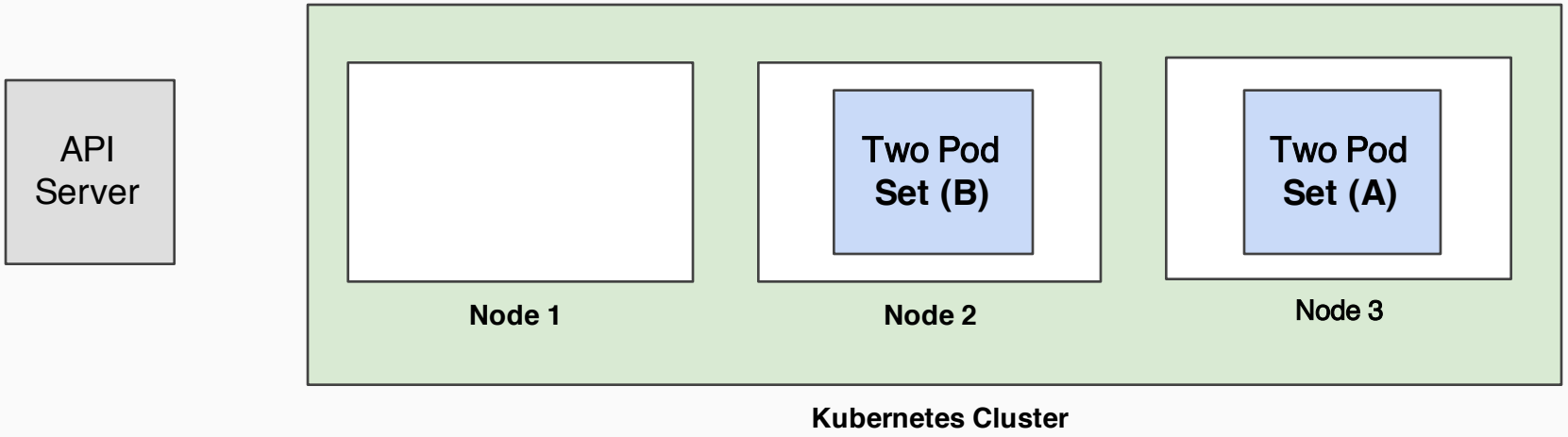
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



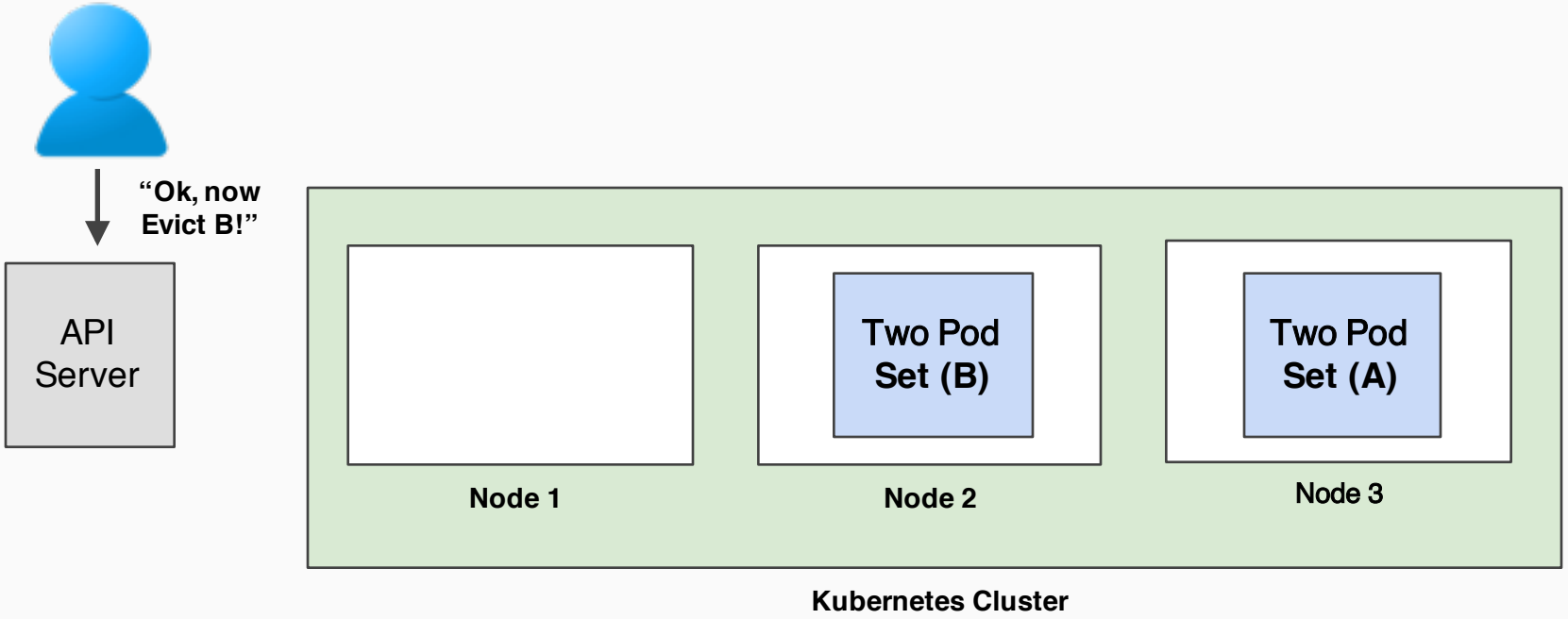
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



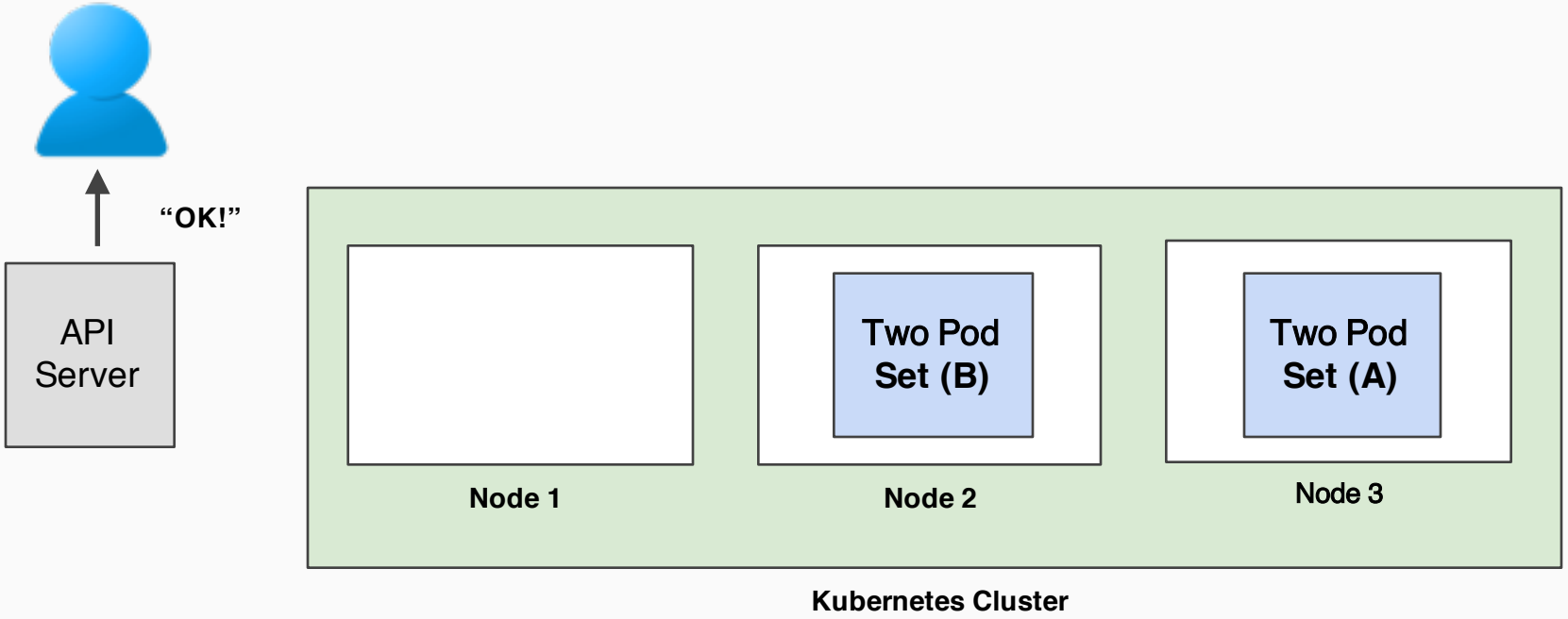
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



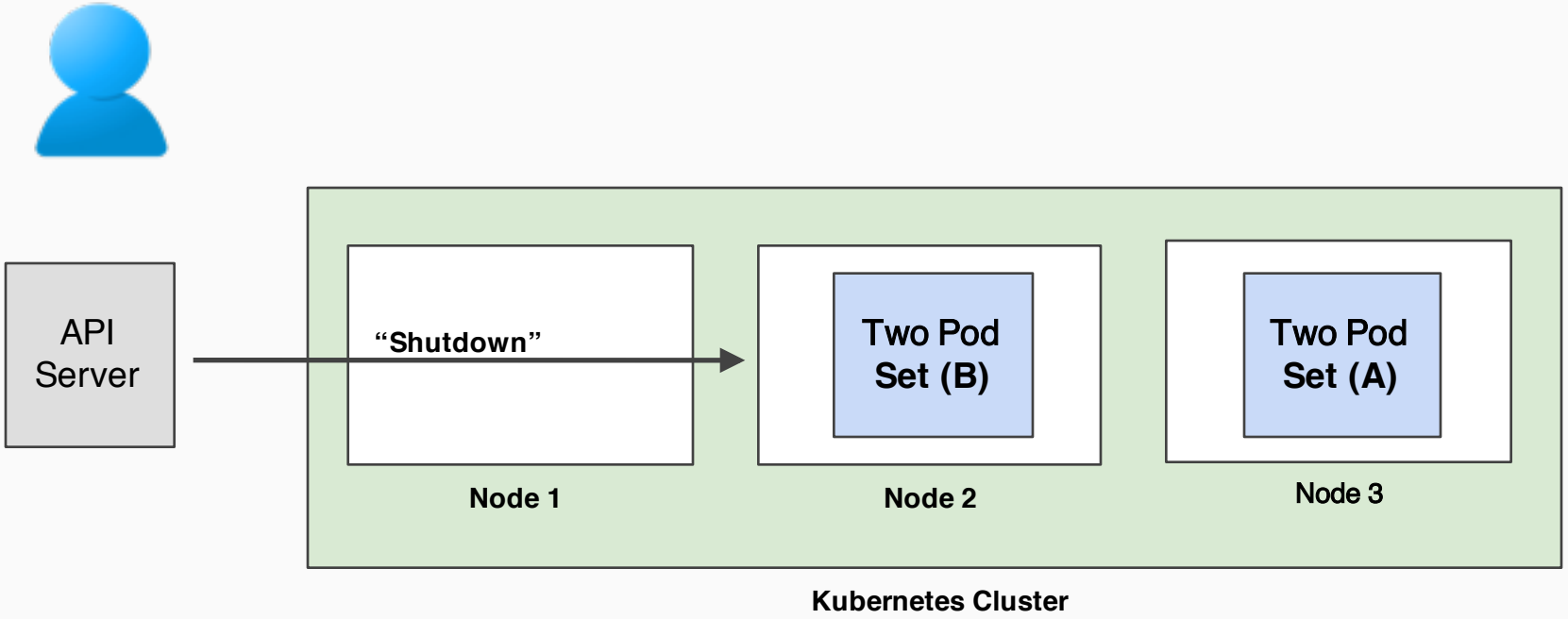
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



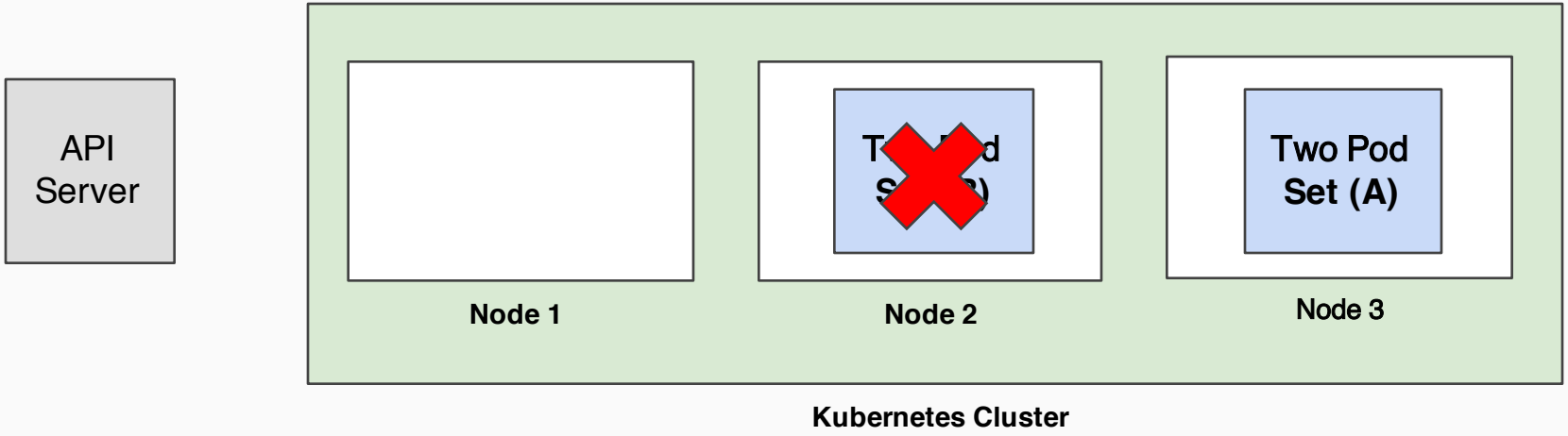
Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



Sophisticated Scheduling: Disruption Budget

SCENARIO: Cluster upgrades with stateful workloads



Network Policy

Problem: Network policy is complicated!

Today:

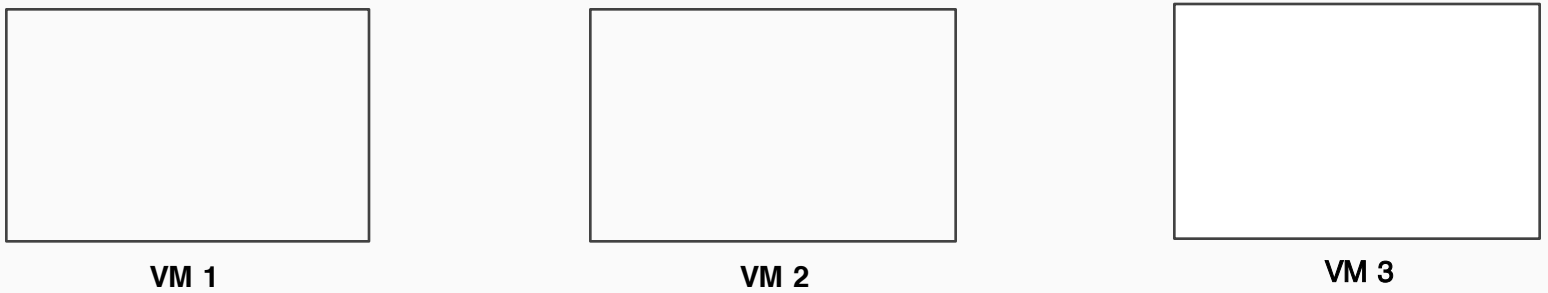
Use VM tooling to support security (but limit VM utilization)
Managing port level security
Proxy-ing everything

Network Policy

Solution: Network Policy Object!

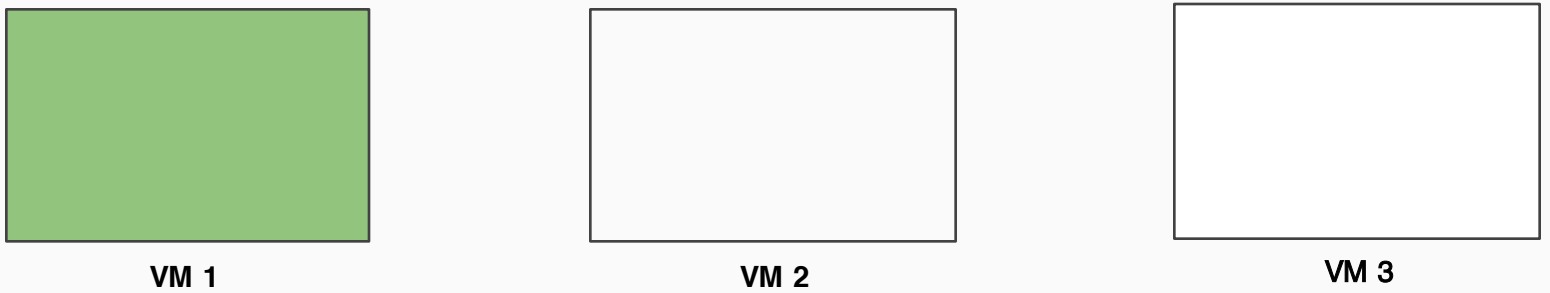
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



Network Policy Object

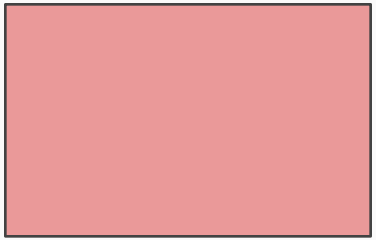
SCENARIO: Two-tier app needs to be locked down



Network Policy Object

SCENARIO: Two-tier app needs to be locked down

VM 1

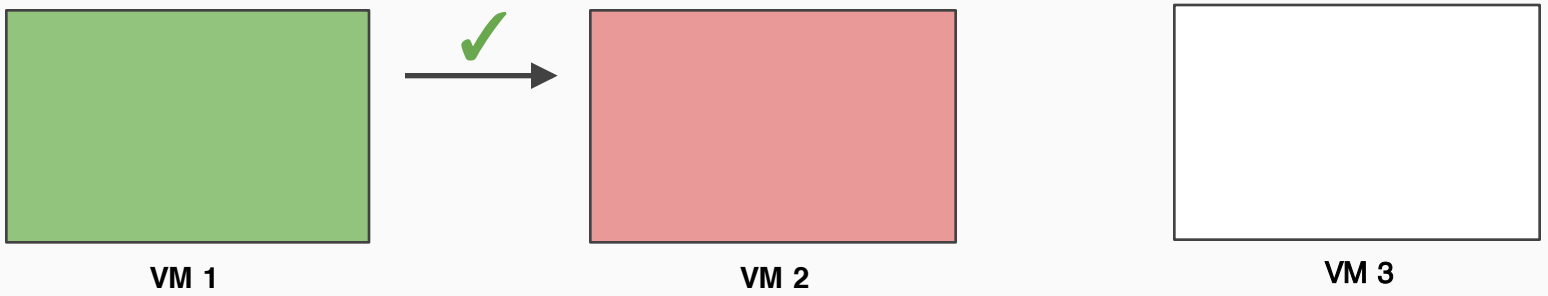


VM 2

VM 3

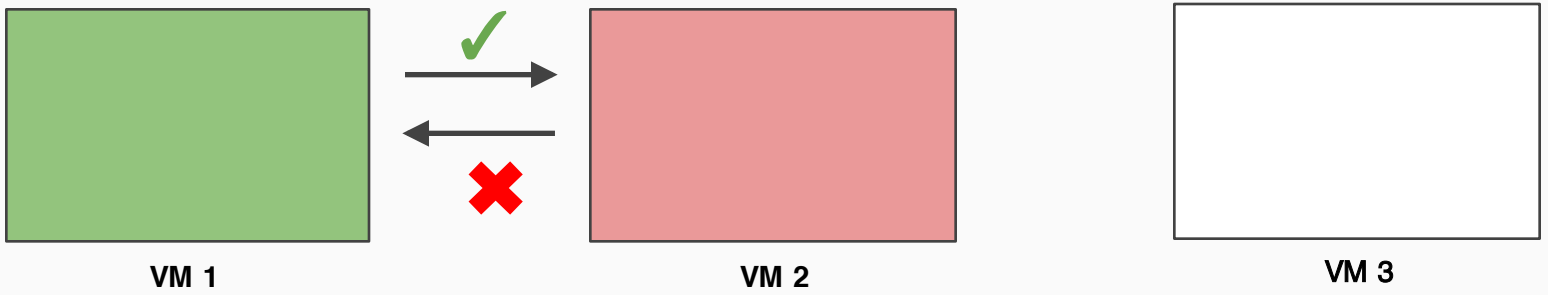
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



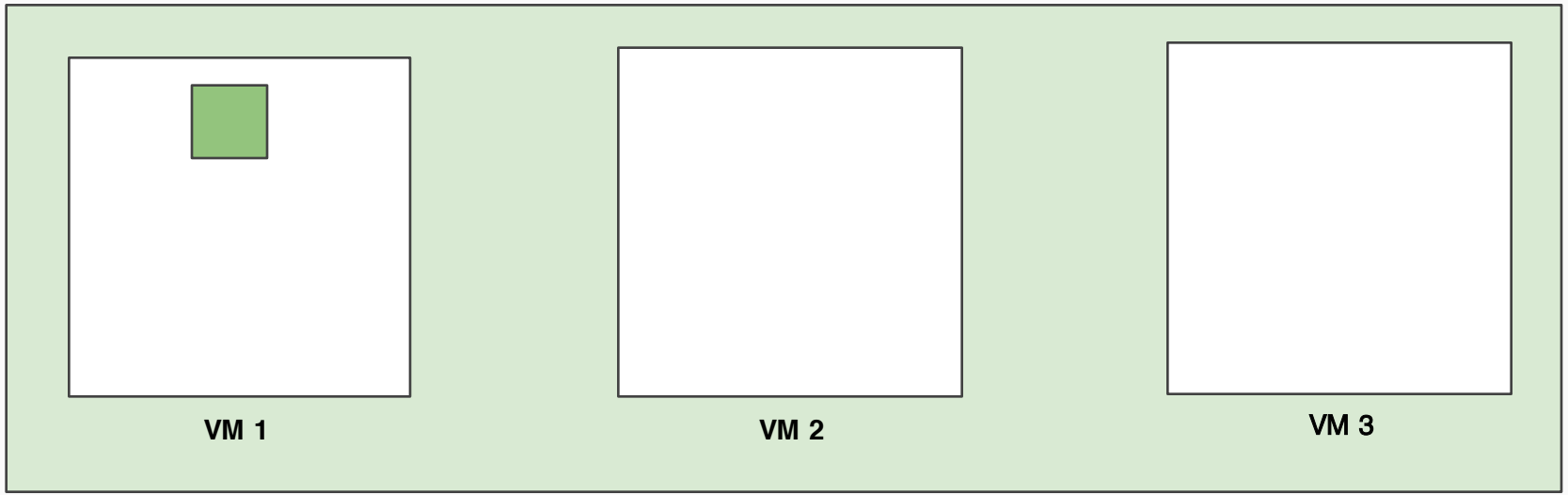
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



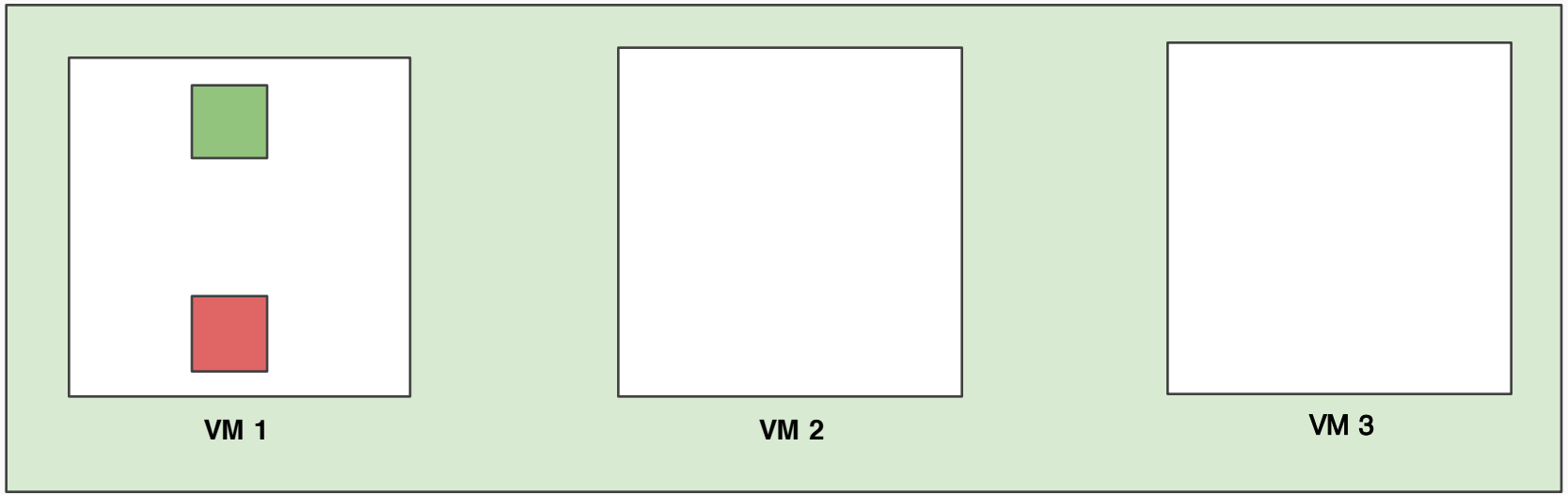
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



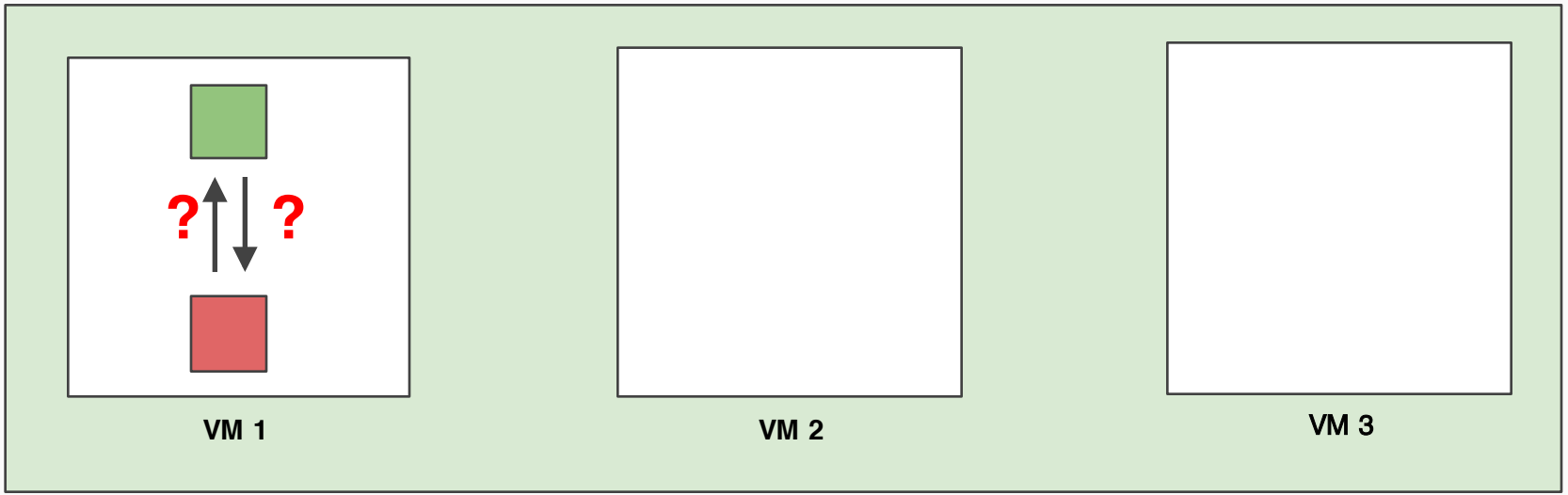
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



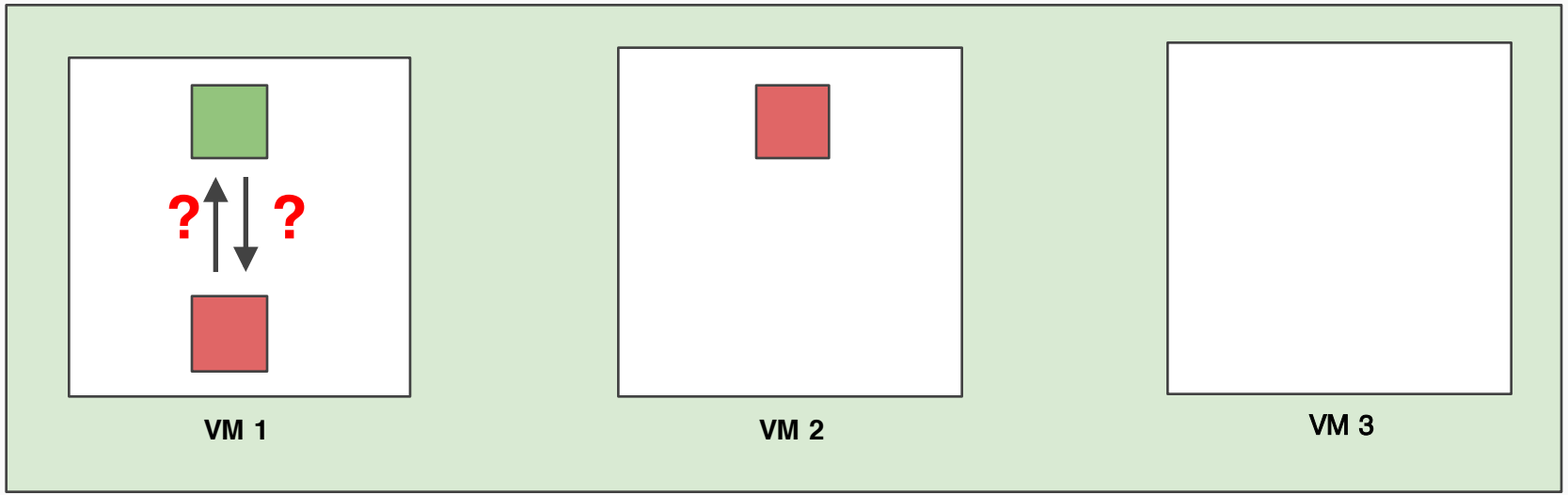
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



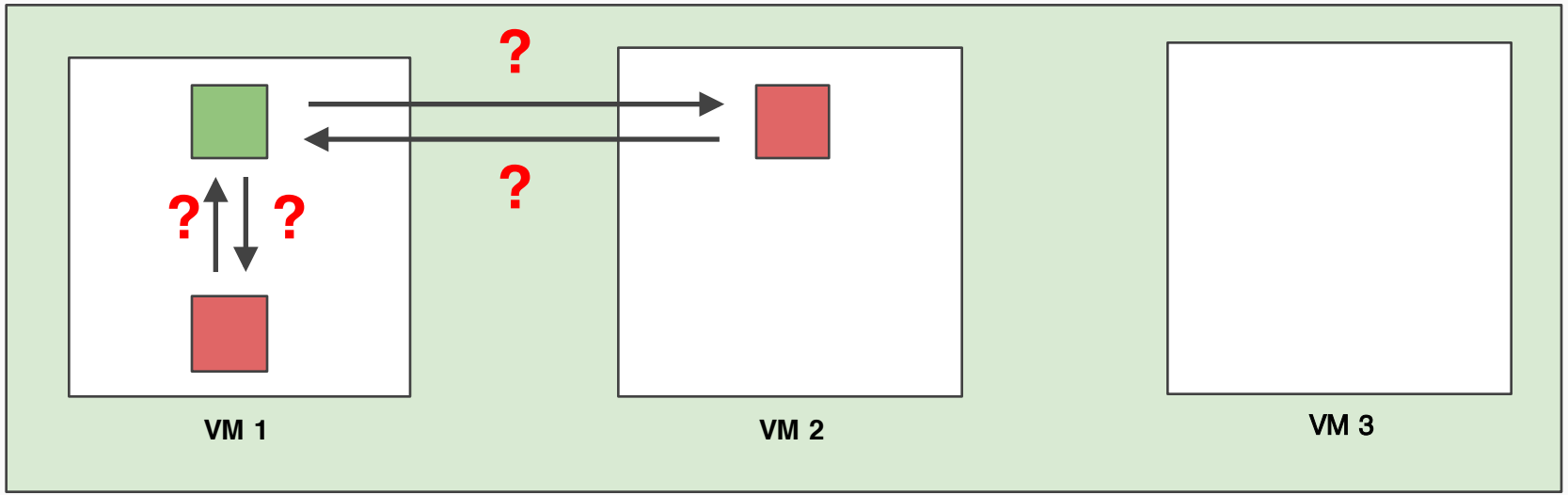
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



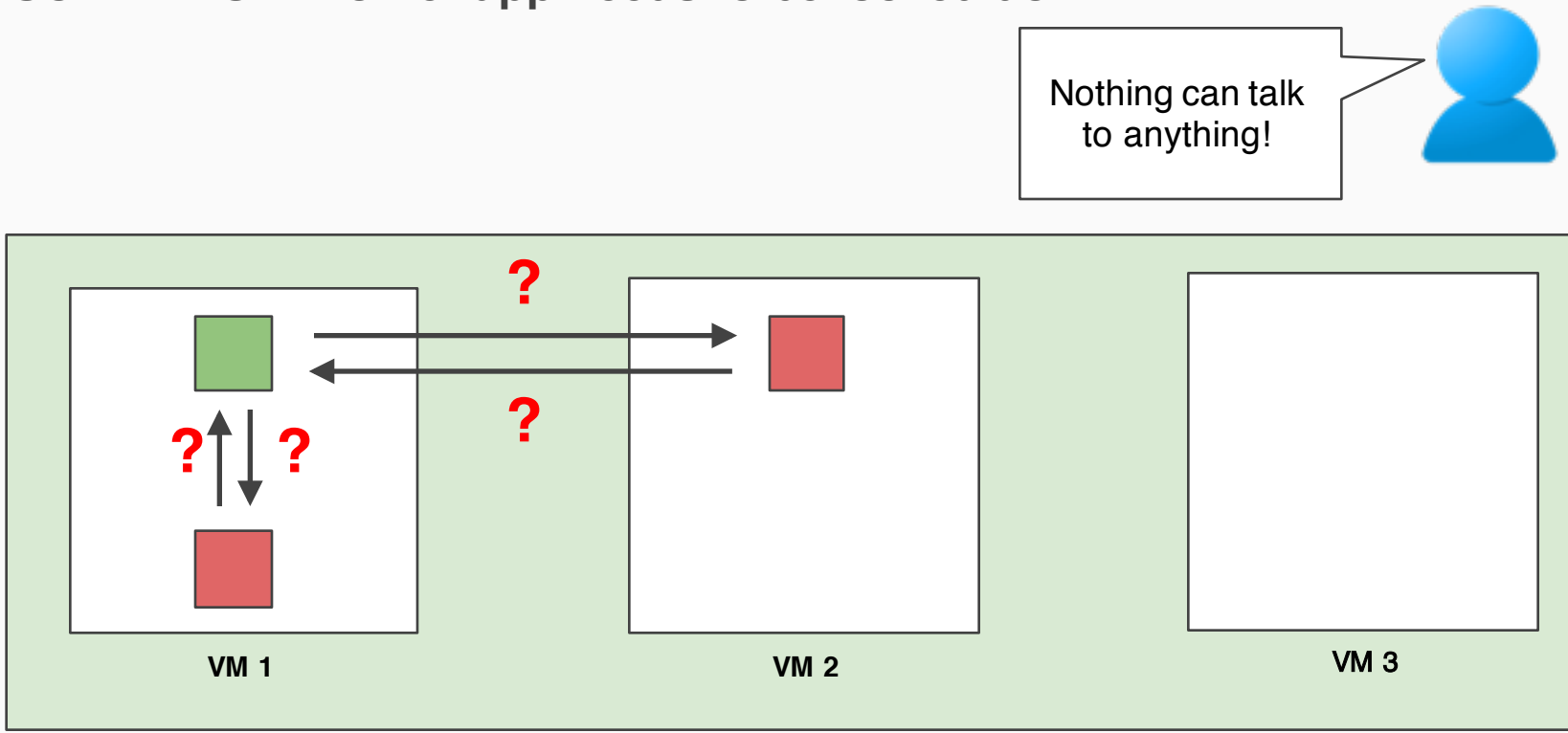
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



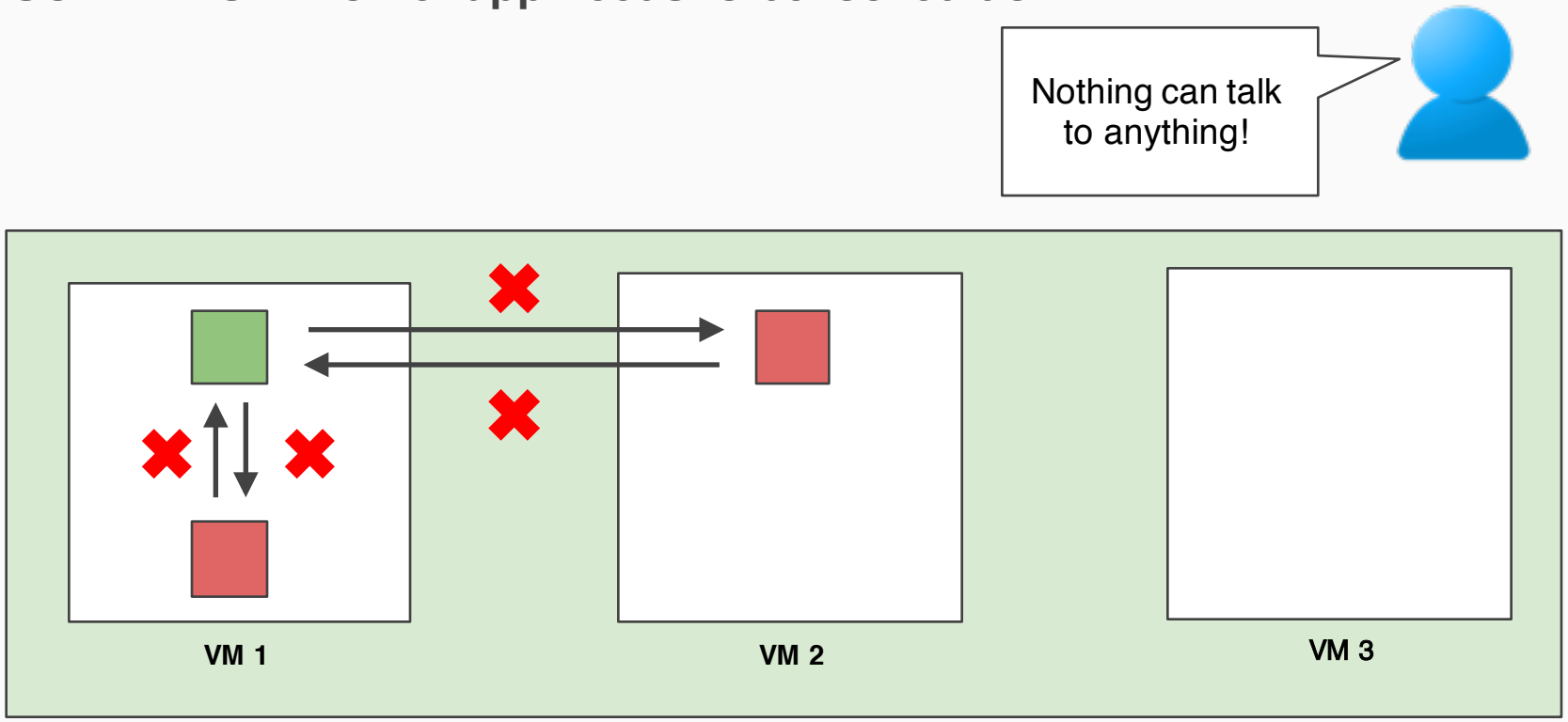
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



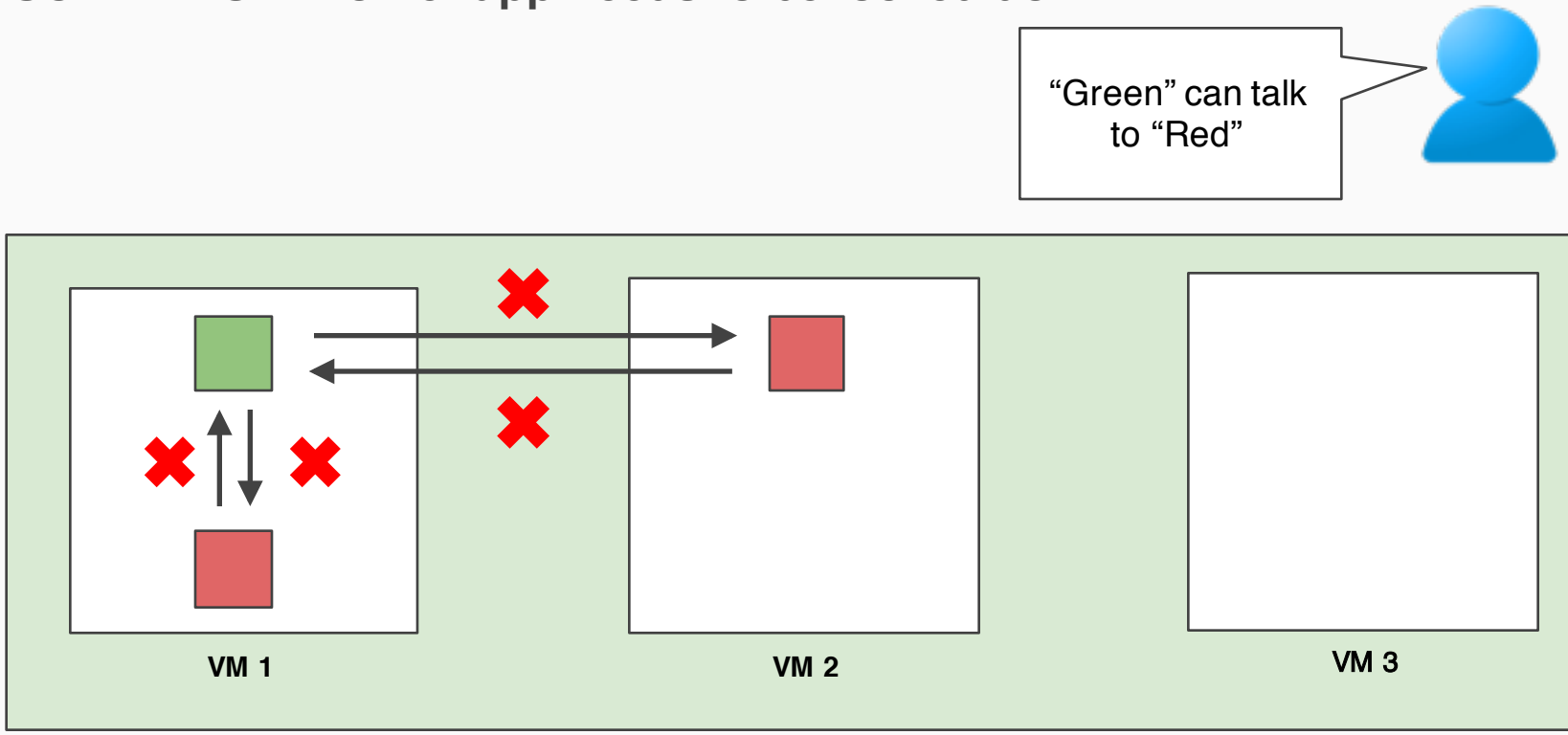
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



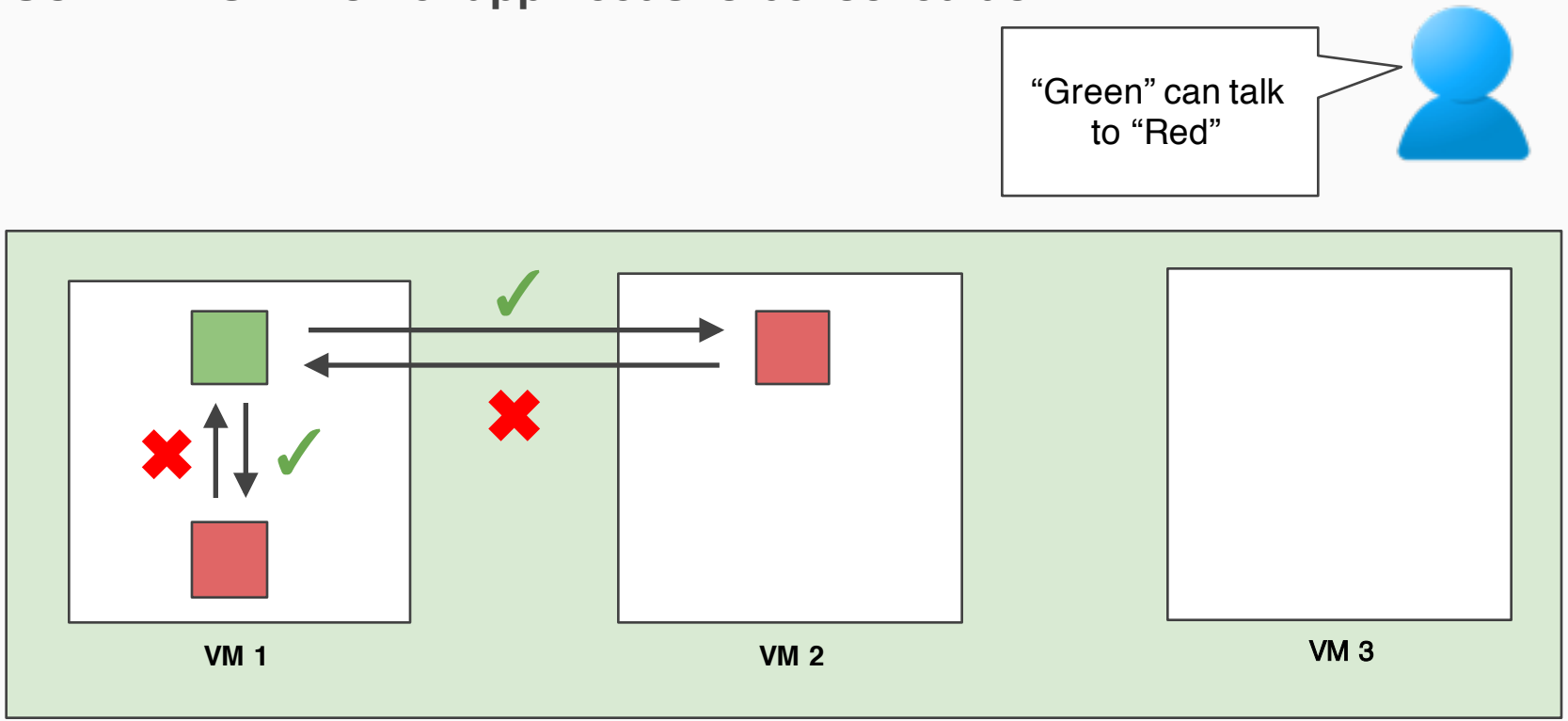
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



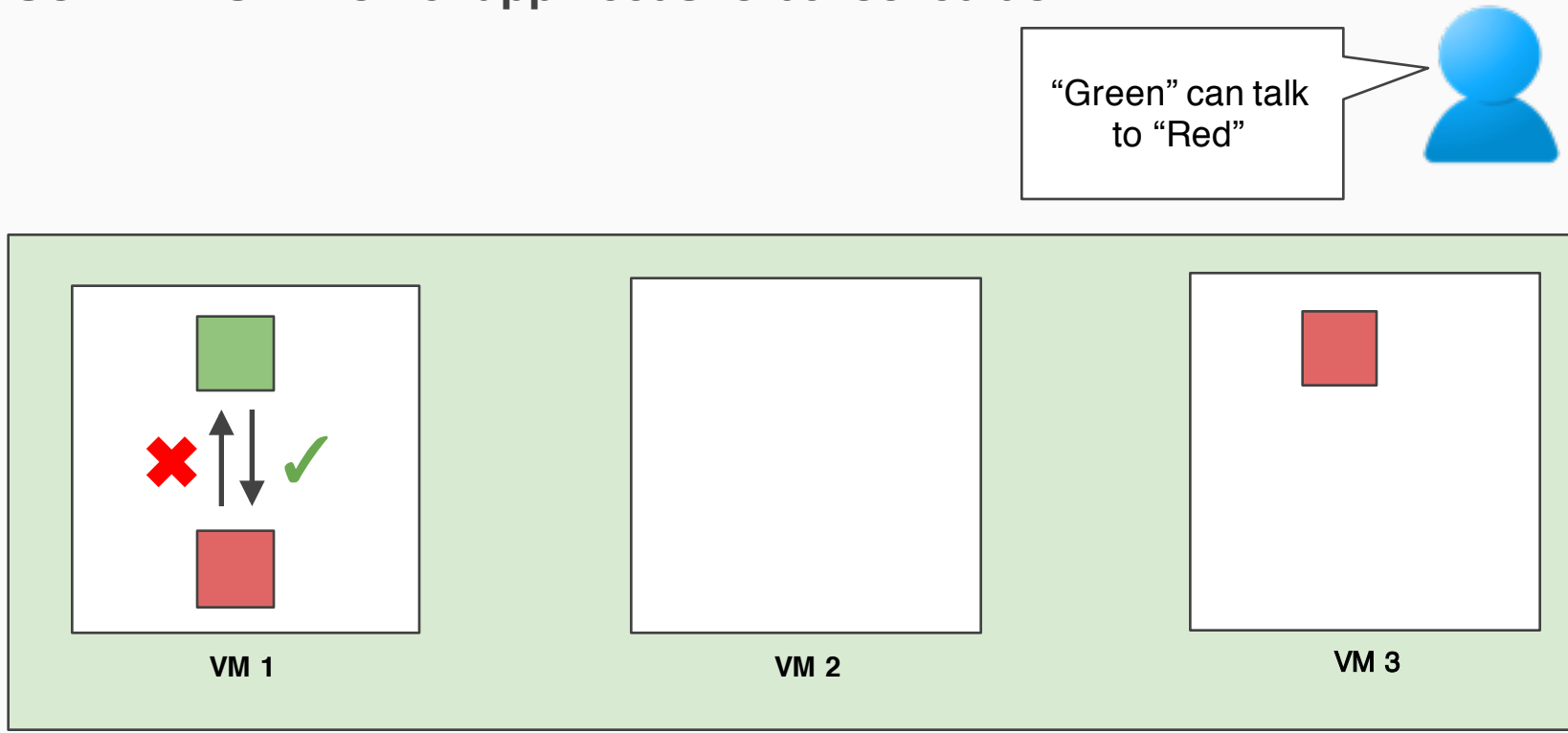
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



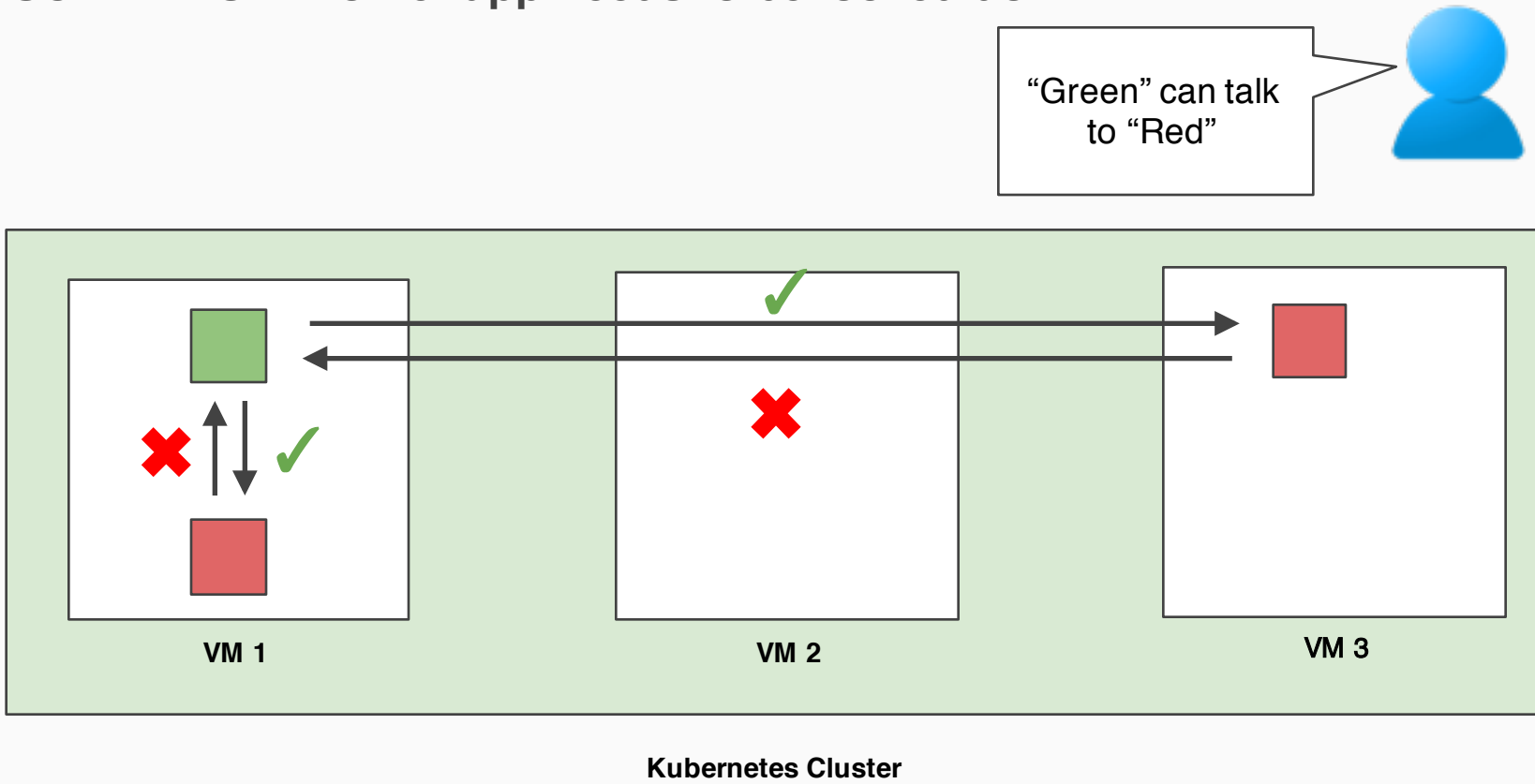
Network Policy Object

SCENARIO: Two-tier app needs to be locked down



Network Policy Object

SCENARIO: Two-tier app needs to be locked down



Helm

Problem: I need to deploy complicated apps!

Today:

Manually deploy applications once per cluster
Manually publish global endpoints and load balance
Build a control plane for monitoring application

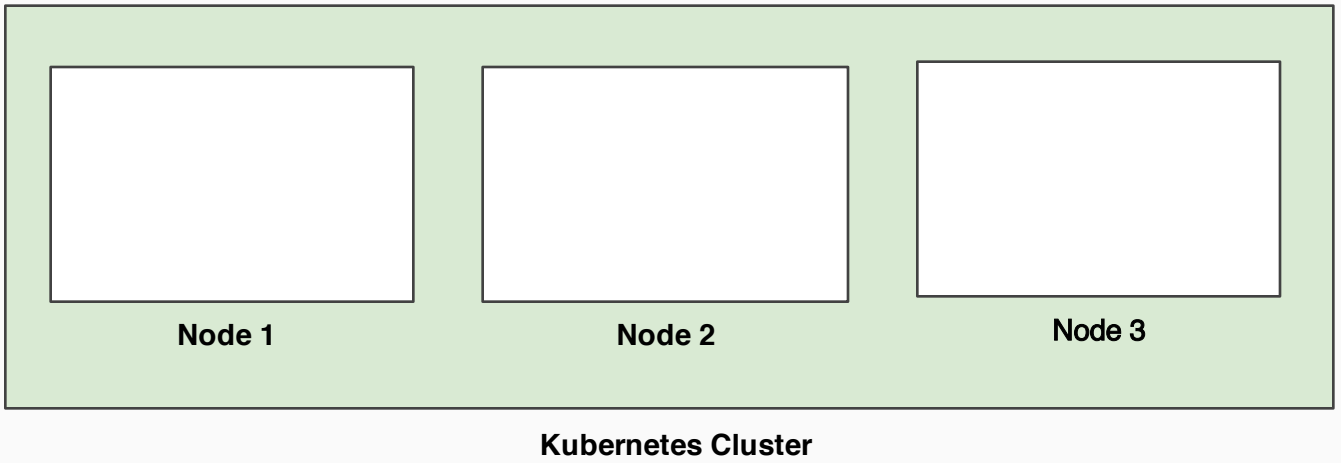
Helm

Solution: Helm - The Package manager for Kubernetes

Think “apt-get/yum”
Supports Kubernetes objects natively
- Deployments
- DaemonSets
- Secrets & config
- Multi-tier apps
- Upgrades

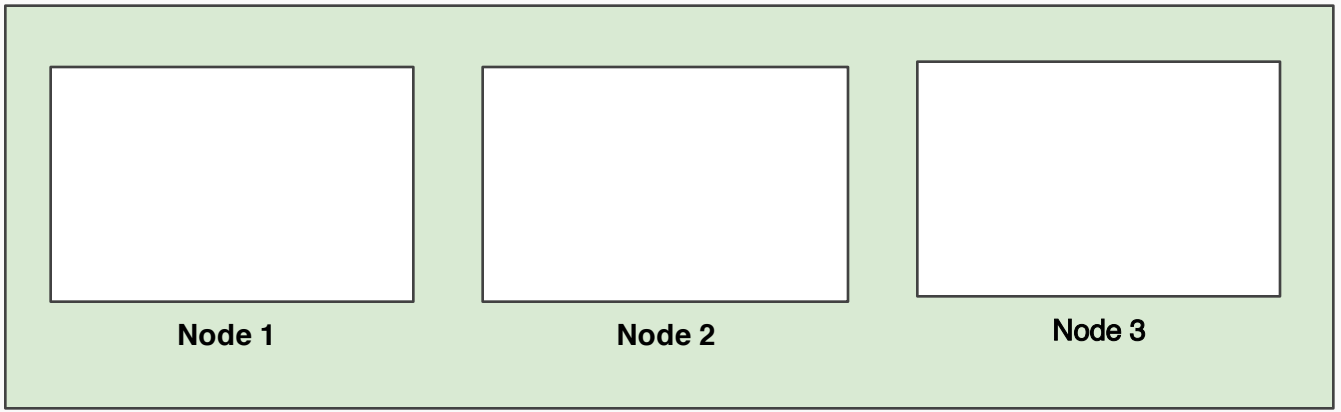
Helm

DaemonSets: DataDog



Helm

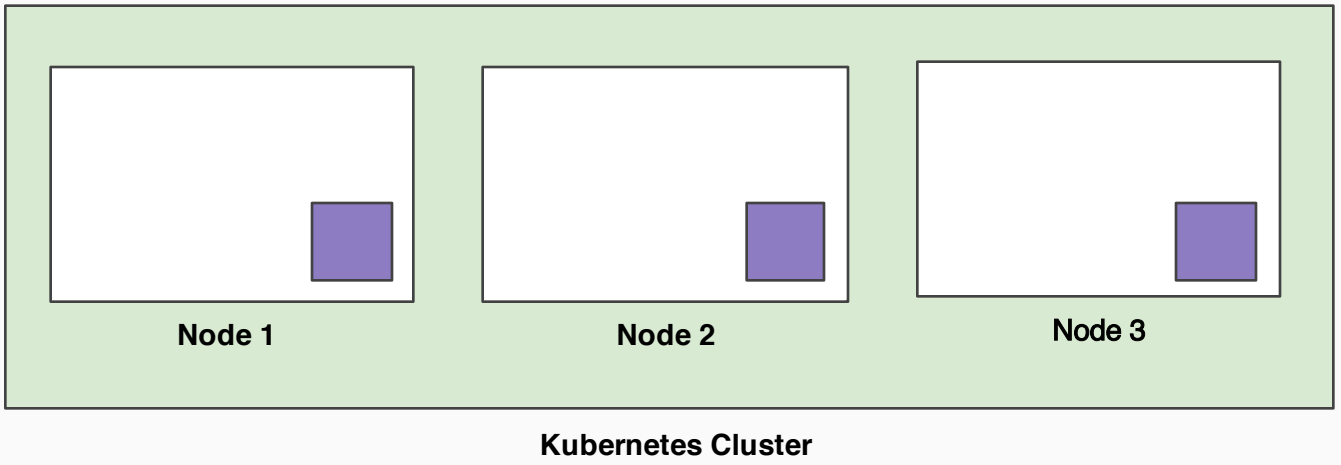
DaemonSets: DataDog




```
helm install --name  datadog --set datadog.apiKey=<APIKEY> stable/datadog
```

Helm

DaemonSets: DataDog



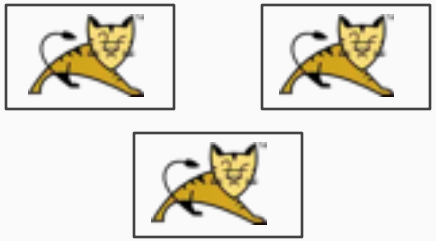

```
helm install --name  datadog --set datadog.apiKey=<APIKEY> stable/datadog
```

Helm

Solution: Helm - The Package manager for Kubernetes

Helm

Solution: Helm - The Package manager for Kubernetes



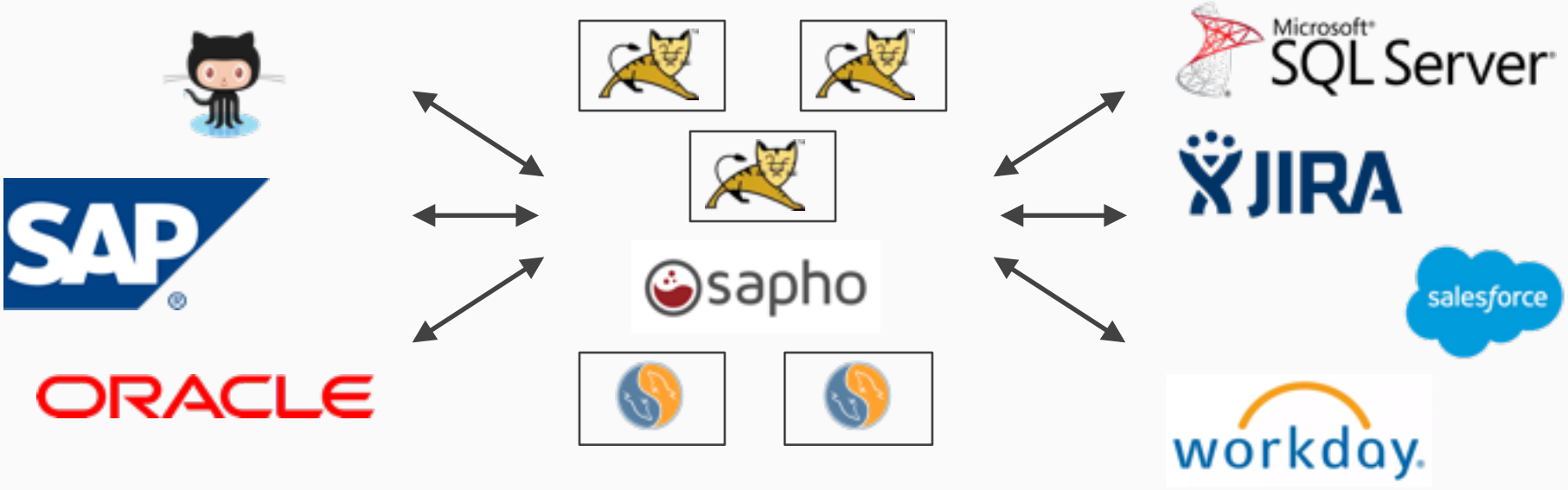
sapho

Helm

Solution: Helm - The Package manager for Kubernetes

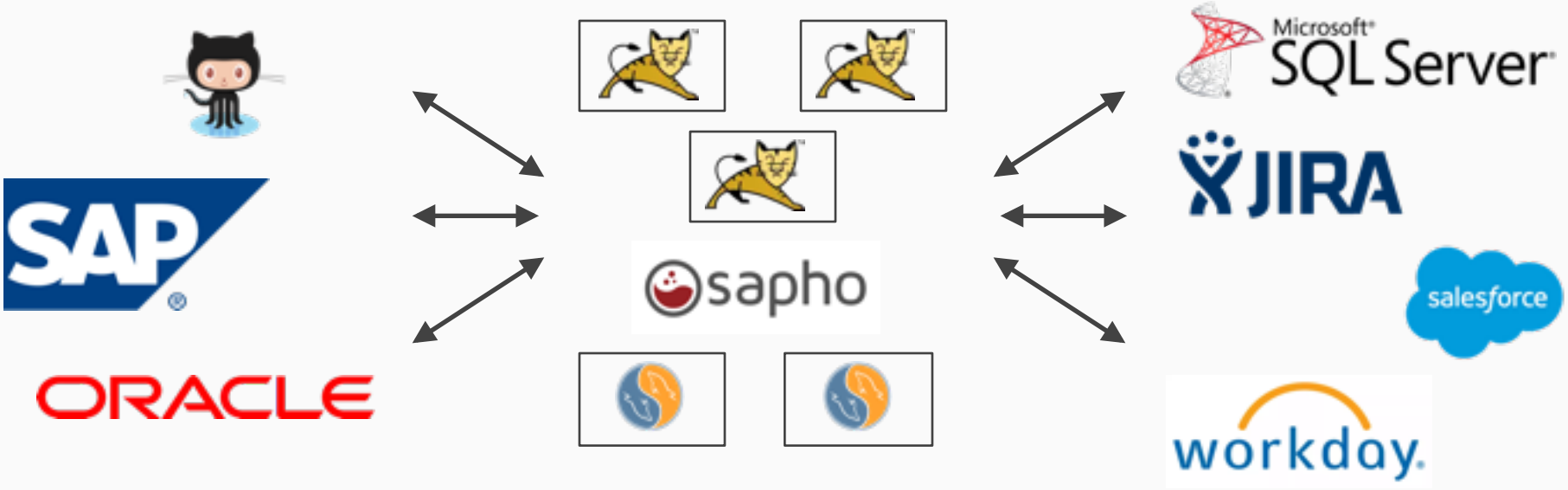
Helm

Solution: Helm - The Package manager for Kubernetes



Helm

Solution: Helm - The Package manager for Kubernetes




```
helm install sapho
```

Accelerating Stateful Applications

Accelerating Stateful Applications

Management of storage and data for stateful applications on Kubernetes

Accelerating Stateful Applications

Management of storage and data for stateful applications on Kubernetes

Management of Kubernetes at enterprise scale

Accelerating Stateful Applications

Management of storage and data for stateful applications on Kubernetes

Management of Kubernetes at enterprise scale

Container-optimized servers for compute and storage

Accelerating Stateful Applications

Management of storage and data for stateful applications on Kubernetes

Management of Kubernetes at enterprise scale

\+ 

Container-optimized servers for compute and storage

Accelerating Stateful Applications

Management of storage and data for stateful applications on Kubernetes

Management of Kubernetes at enterprise scale

\+ 

Container-optimized servers for compute and storage

Automated Stateful Apps on K8S

What’s Next

What’s Next

What’s Next

Nothing!*

What's Next

Nothing!*

* for large values of "Nothing"

What's Next

Nothing!*

Bringing many features from alpha to beta & GA, including:

- Federated deployments and daemon sets
- Improved RBAC
- StatefulSet upgrades

Improved scaling & etcd 3

Easy cluster setup for high availability configuration

Integrated Metrics API

* for large values of "Nothing"

Kubernetes is Open

- open community
- open design
- open source
- open to ideas

- kubernetes.io
- github.com/kubernetes/kubernetes
- slack.kubernetes.io
- twitter: @kubernetesio

Twitter: @aronchick
Email: aronchick@google.com