Walls Within Walls:

What if your attacker knows parkour?

Greg Castle

GKE Security Tech Lead
Twitter: @mrgcastle
Github: @destijl
Google

Tim Allclair

GKE Security Engineer
Twitter: @tallclair
Github: @tallclair
Google

A Tale of Two Containers

Node Isolation Setup

Workload Steering Attack

Node vs. Pod Isolation

Takeaways

A Tale of Two Containers

A Niche Webhosting Company

“Webhosting for parkour gyms”

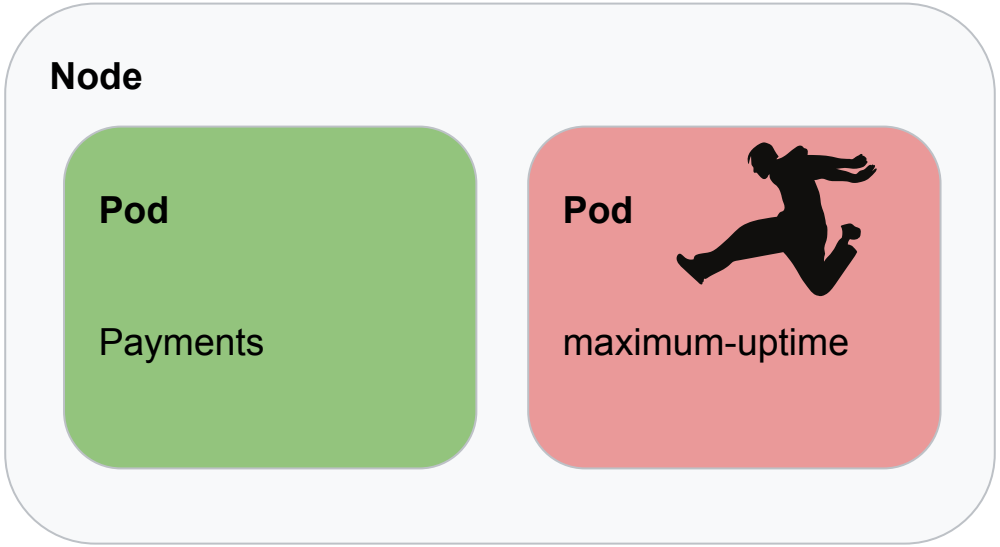
A Tale of Two Containers

Prod payments processing

Customer website for “maximum-uptime” parkour gym

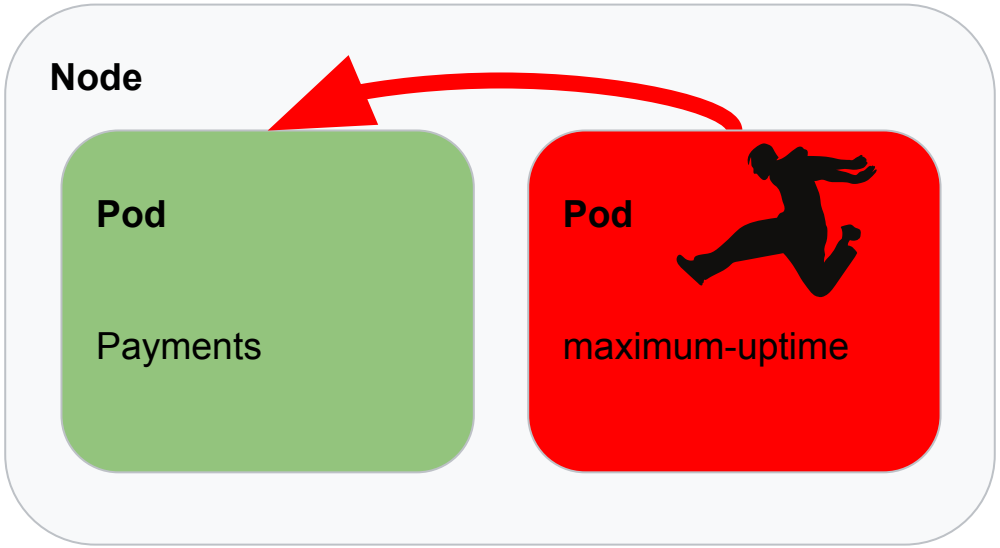
Do Nothing

Sensitive containers scheduled next to untrusted workloads.



Threat Model

Expect low security system to be compromised and escape container.



Are container breakouts a thing?

- Yes, see runc CVE-2019-5736
- Bugs are inevitable
- Not enough to separate untrusted workloads from high value workloads

App-Specific Hardening?

Seccomp, app-armor, selinux:

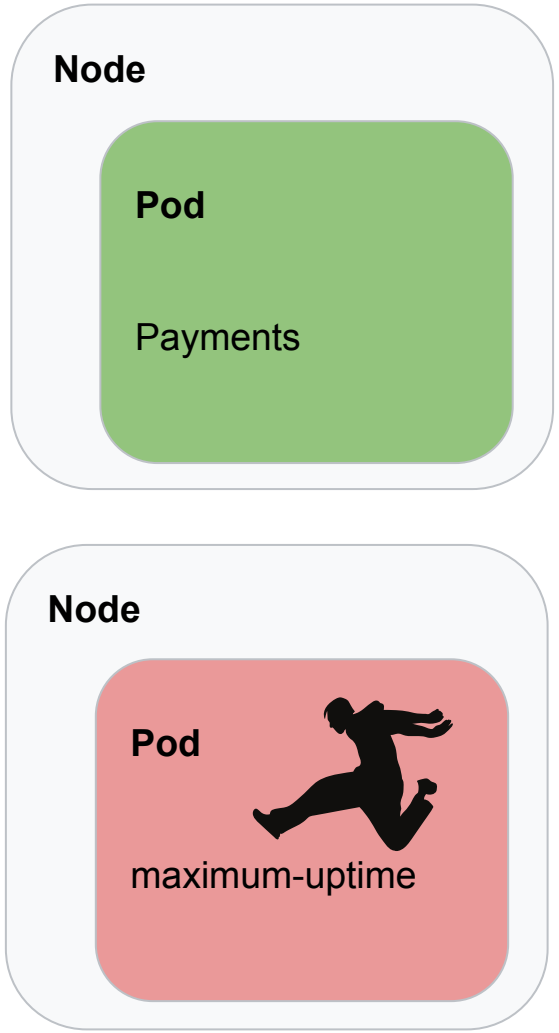
- Difficult to learn and maintain
- Hard to fully exercise applications in test
- Customer website needs may vary
- Beaten by Dirty COW-like vulnerability (CVE-2016-5195)

Separate Nodes

Payments on different nodes to customer workloads

Non-security benefits:

- Separate failure domains
- Resource isolation (disk iops, network)



But is it good enough?

We’ll focus here for the rest of the talk.

Assume container escape has happened.

Node isolation setup

Node Isolation: Overview

Configuration:

- labels
- taints

De-privilege kubelet:

- node authorizer
- node restriction

Node setup

Label: target payments pods for payments nodes

```
kubectl label nodes $NODES class=payments
```

Taint: repel non-payments workloads

```
kubectl taint nodes $NODES \
class=payments:NoSchedule
```

Pod Labels

Pod targets label with nodeSelector

Tells scheduler: I **only** run on payments nodes

```
spec:
   nodeSelector:
      class: payments
```

Pod tolerations

Tells scheduler: I can tolerate the payments taint

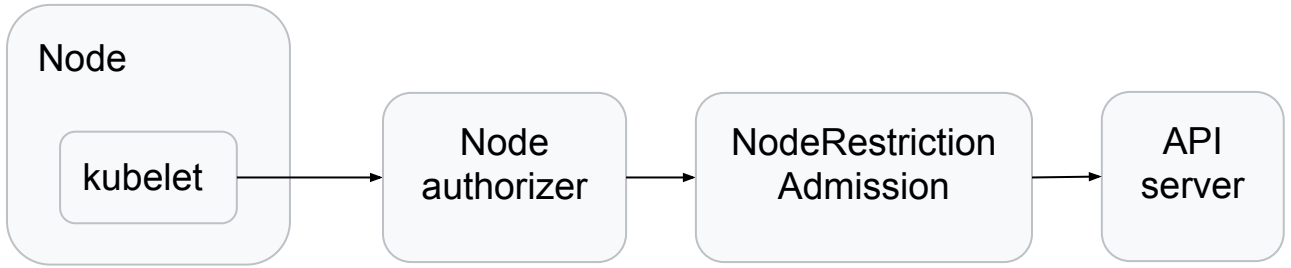
```
spec:
   tolerations:
     - key: class
       operator: "Equal"
       value: "payments"
```

Node Authorizer

Limit kubelet to least privilege, e.g:

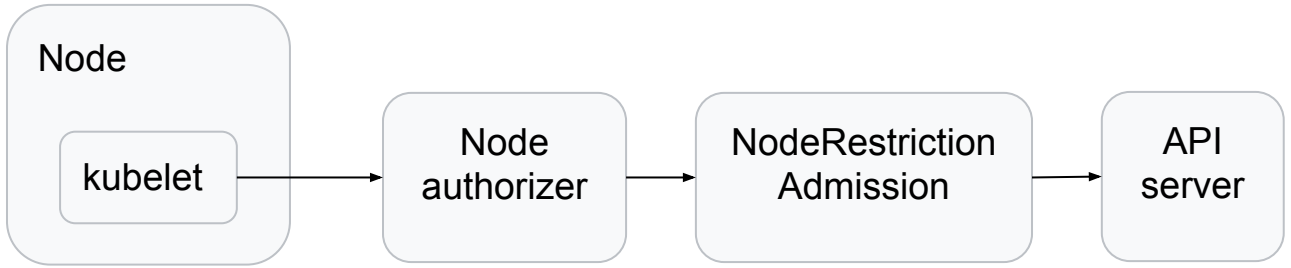
write node, pod objects

read secrets for pods on the node



NodeRestriction Admission

More fine-grained control over kubelet **mutating** operations



Node Isolation: Full Picture

Configuration:

- labels: target payments pods to payments nodes
- taints: keep non-payments workloads off payments nodes

De-privilege kubelet:

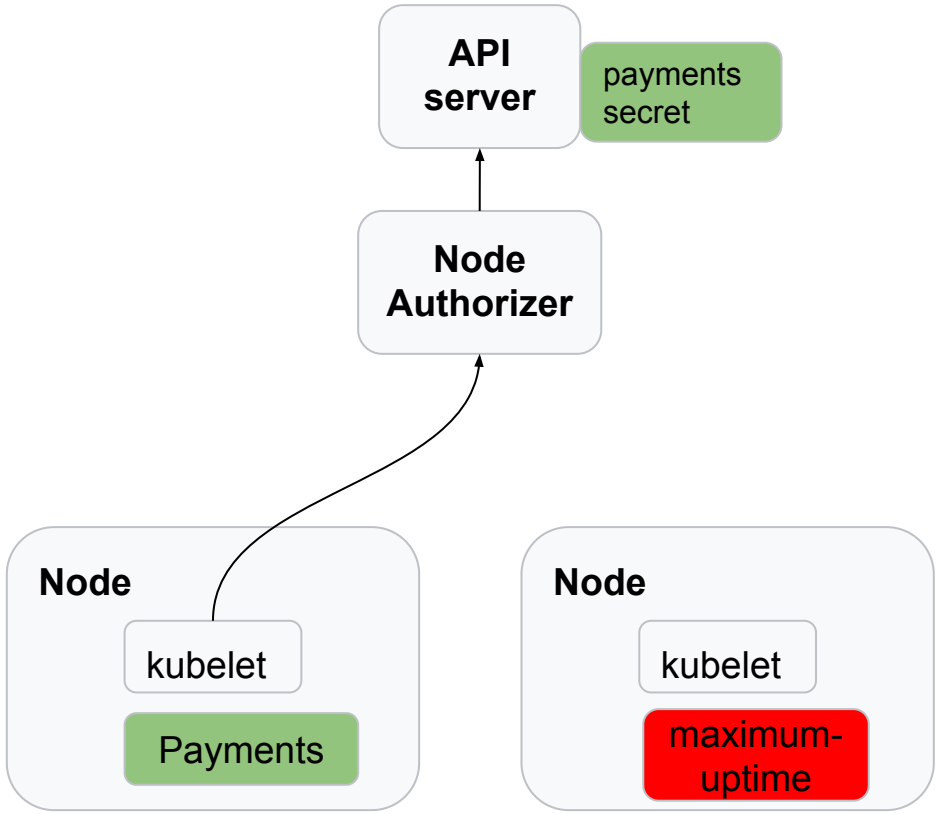
- node authorizer
- node restriction

Workload steering attack

Workload Steering Attack

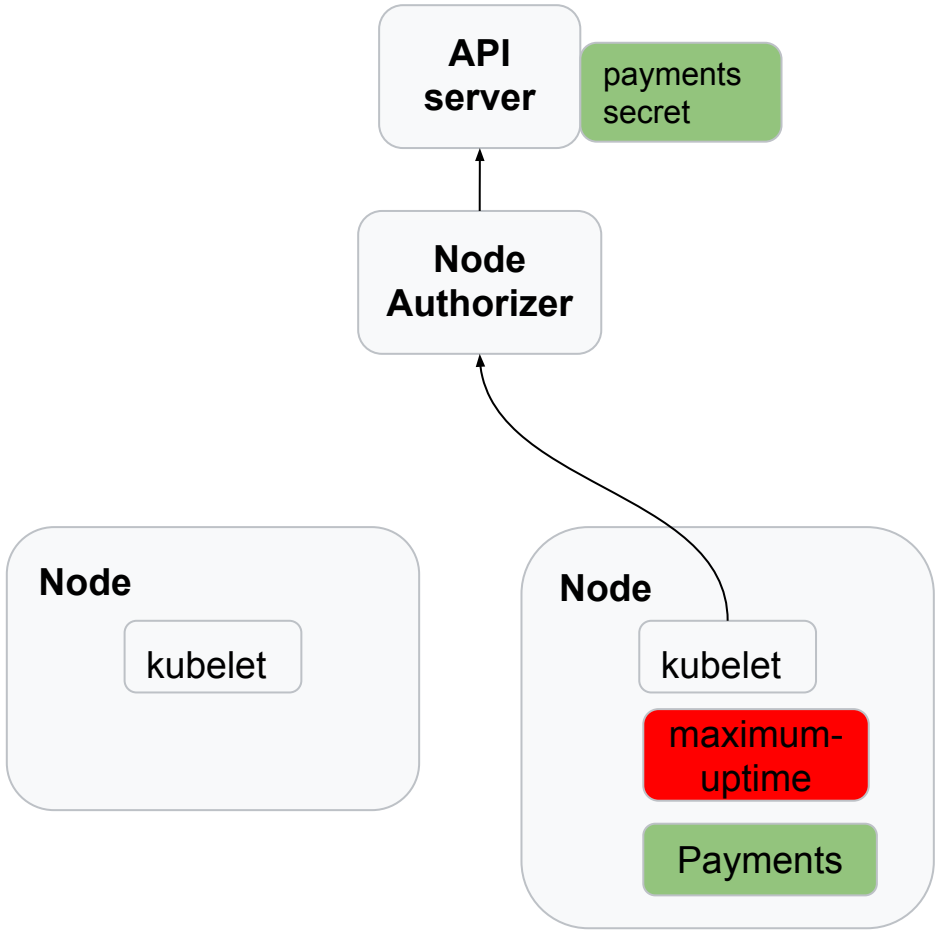
Goal: access secret

Current setup only allows nodes with payments to access



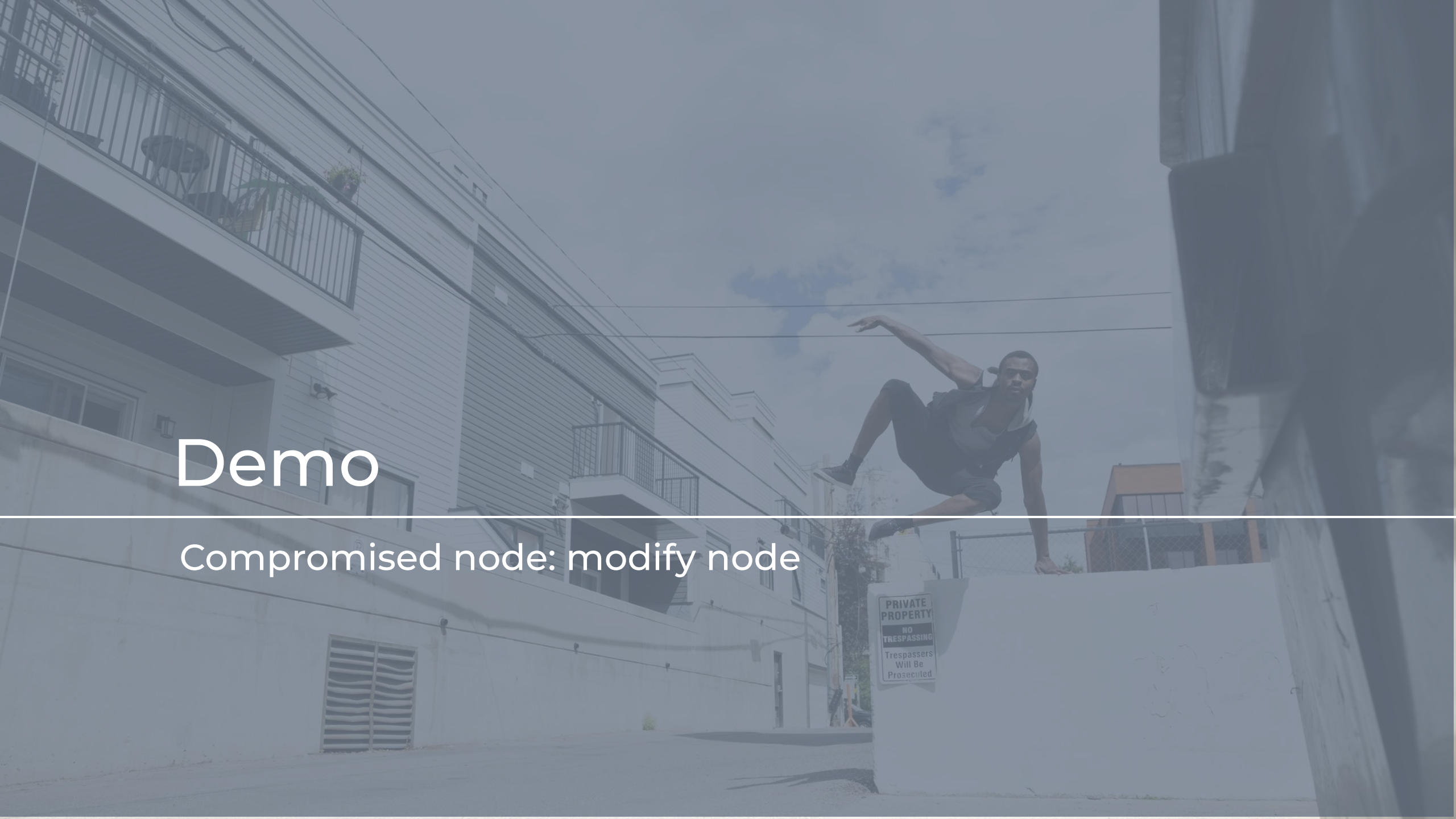
Workload Steering Attack

1. **Modify node**
2. Kill real payments pod
3. Get payments scheduled on our node



1: Modify Node

1. **Modify node**
    a. Remove customer taint
    b. Add payments label
2. Kill real payments pod
3. Get payments scheduled on our node



Demo

Compromised node: modify node

Node is ready for payments

Stop here and hope payments gets scheduled on us?

...we can do better

Node

- taint "customer=maximum-uptime:NoSchedule"

+ label “class=payments”

2: Kill Payments

1. Modify node
    a. Remove customer taint
    b. Add payments label
2. **Kill real payments pod**
    a. Create fake payments static pod
    b. Make fake pod older
    c. Put fake pod in ReplicaSet
    d. Have ReplicaSet kill the newest
3. Get payments scheduled on our node

Create fake payments

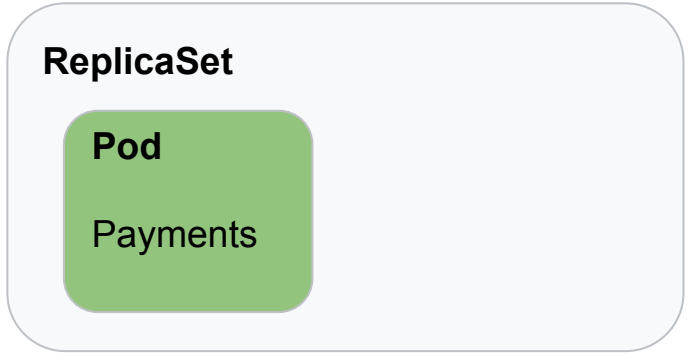
Kubelet not allowed to create regular pods

Can create static (kubelet managed) pods

These are “mirrored” as pods in the API

Abuse ReplicaSet

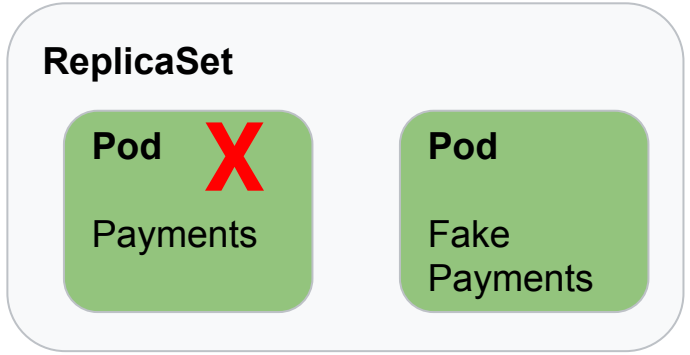
ReplicaSet: keep one copy of payments running



Abuse ReplicaSet

ReplicaSet controller:
Too many copies!

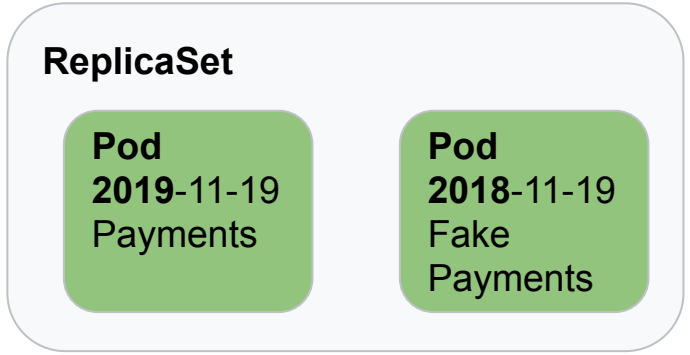
Kill one



Abuse ReplicaSet

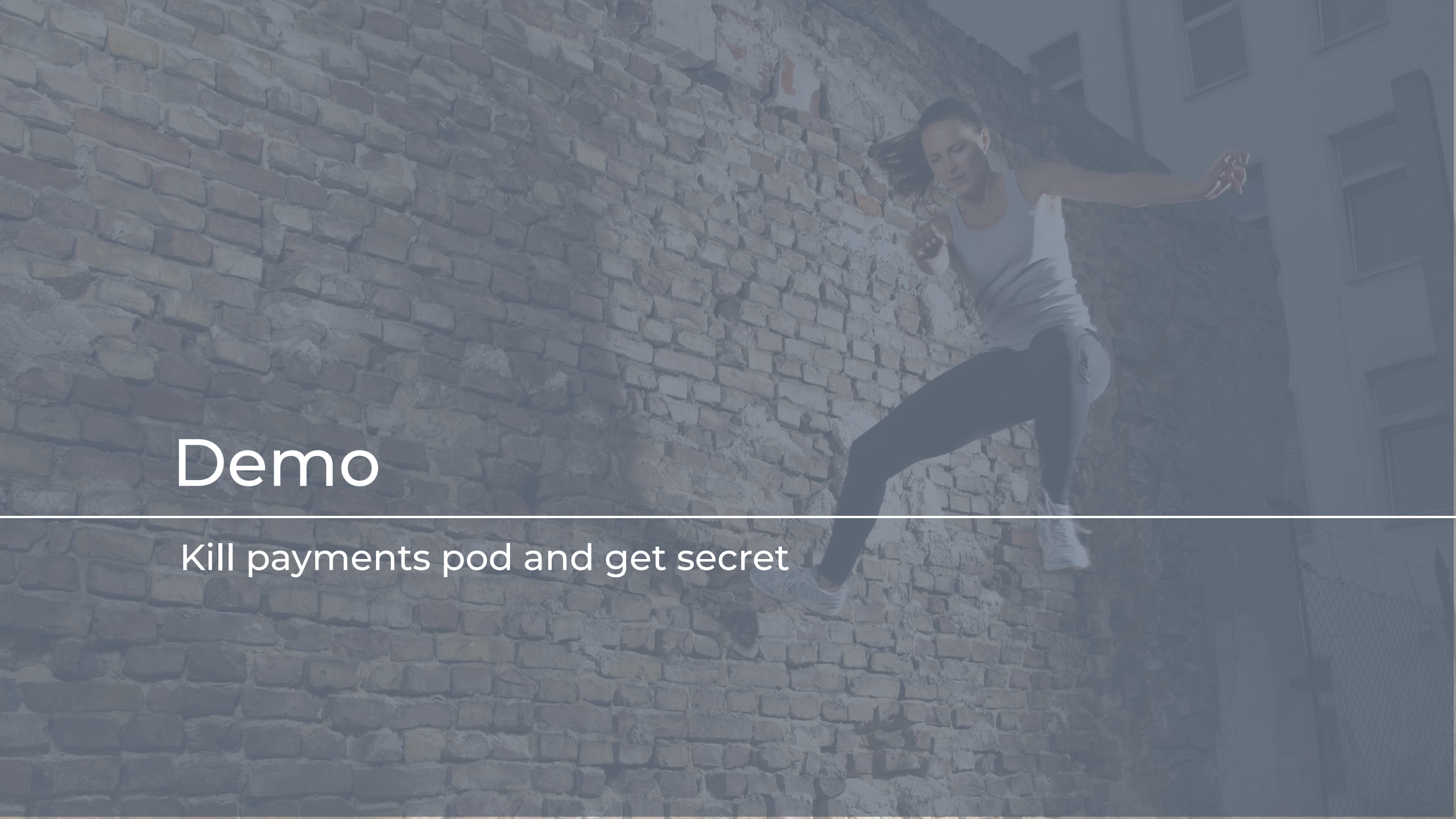
ReplicaSet controller actually kills the newest pod

...so we need to convince the API our fake payments is older



3: Get Payments Scheduled

1. Modify node
    a. Remove customer taint
    b. Add payments label
2. Kill real payments pod
    a. Create fake payments static pod
    b. Make fake pod older
    c. Put fake pod in ReplicaSet
    d. Have ReplicaSet kill the newest
3. **Get payments scheduled on our node**
    a. Delete fake pod
    b. ReplicaSet puts real pod on our node

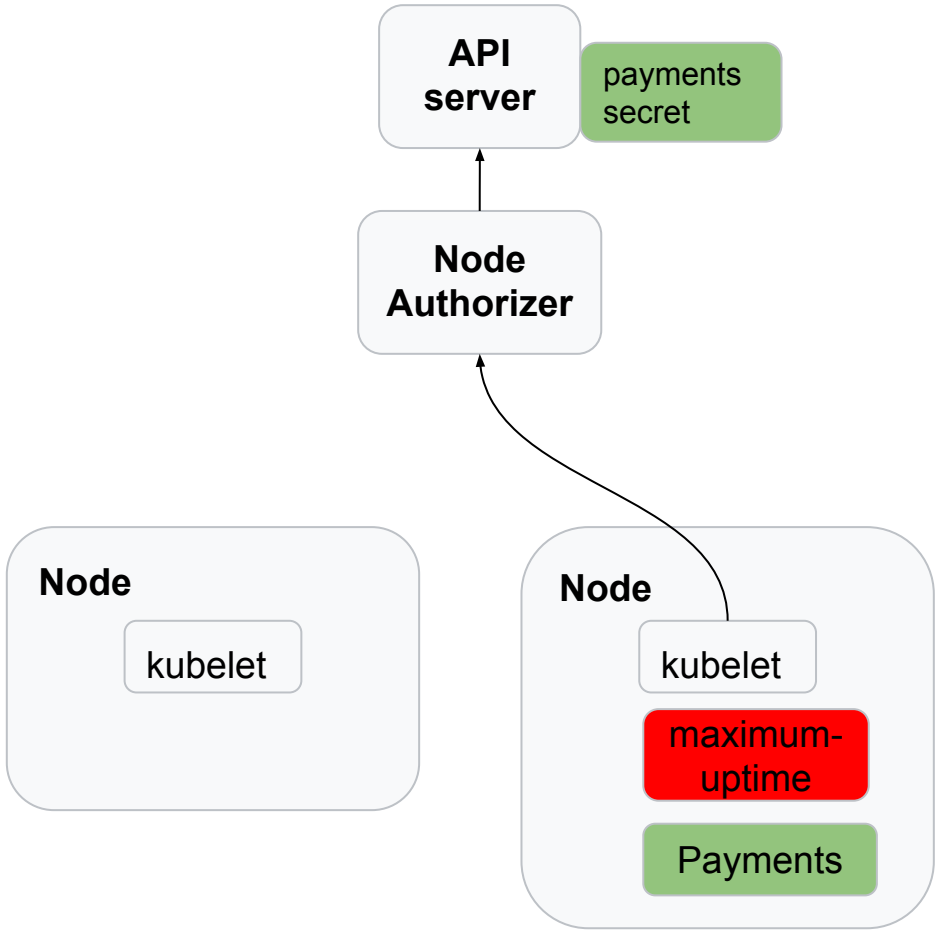


Demo

Kill payments pod and get secret

What happened?

1. Modify node
2. Kill real payments pod
3. Get payments scheduled on our node
4. Get secret



Building up the walls

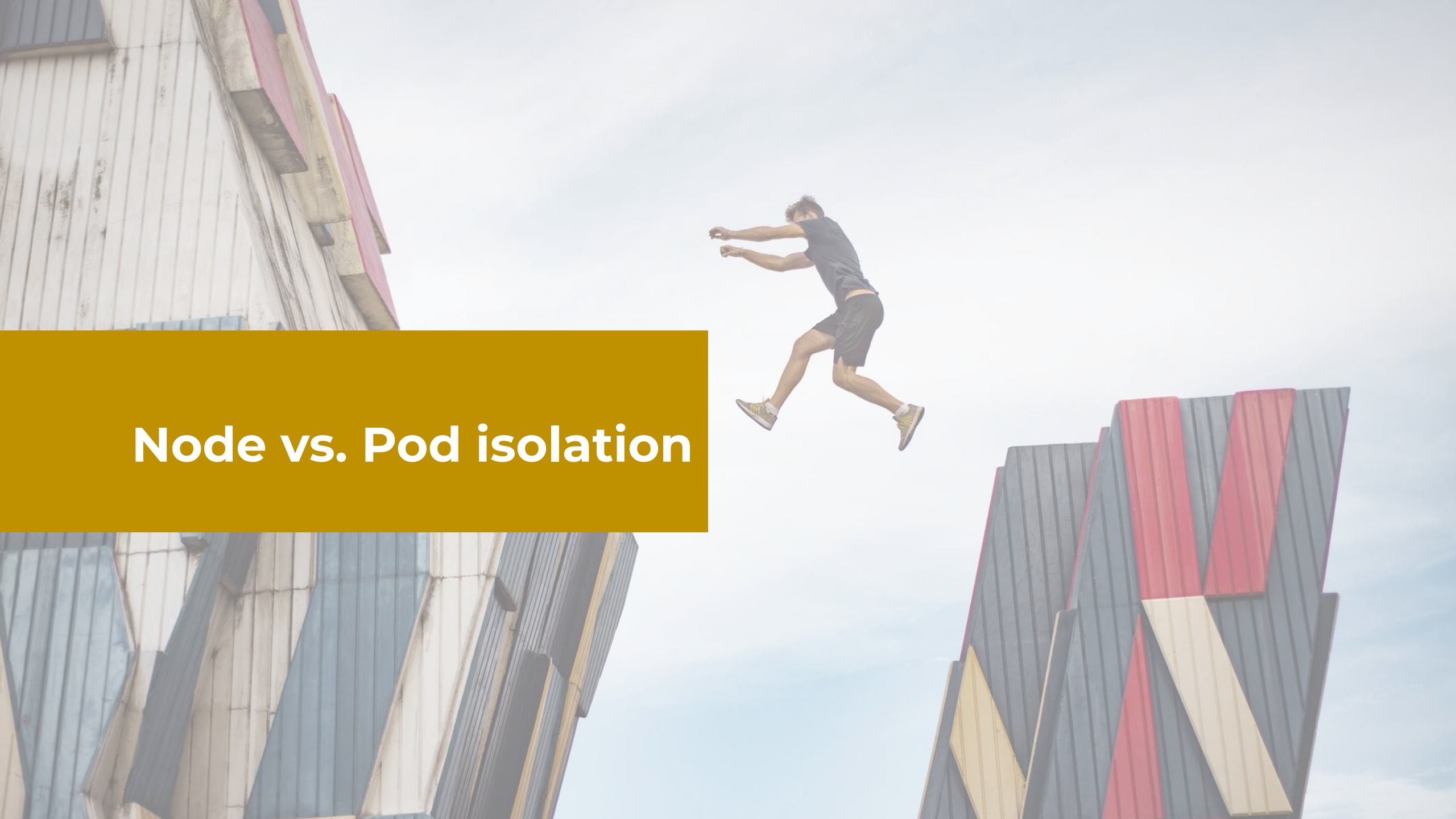
v1.11	Nodes cannot update or remove taints.
	Labels with the restricted prefix can no longer be added or modified by nodes. **`(*.)node-restriction.kubernetes.io/*`**
v1.13	The node authorizer no longer allows nodes to delete themselves.

More on the way:

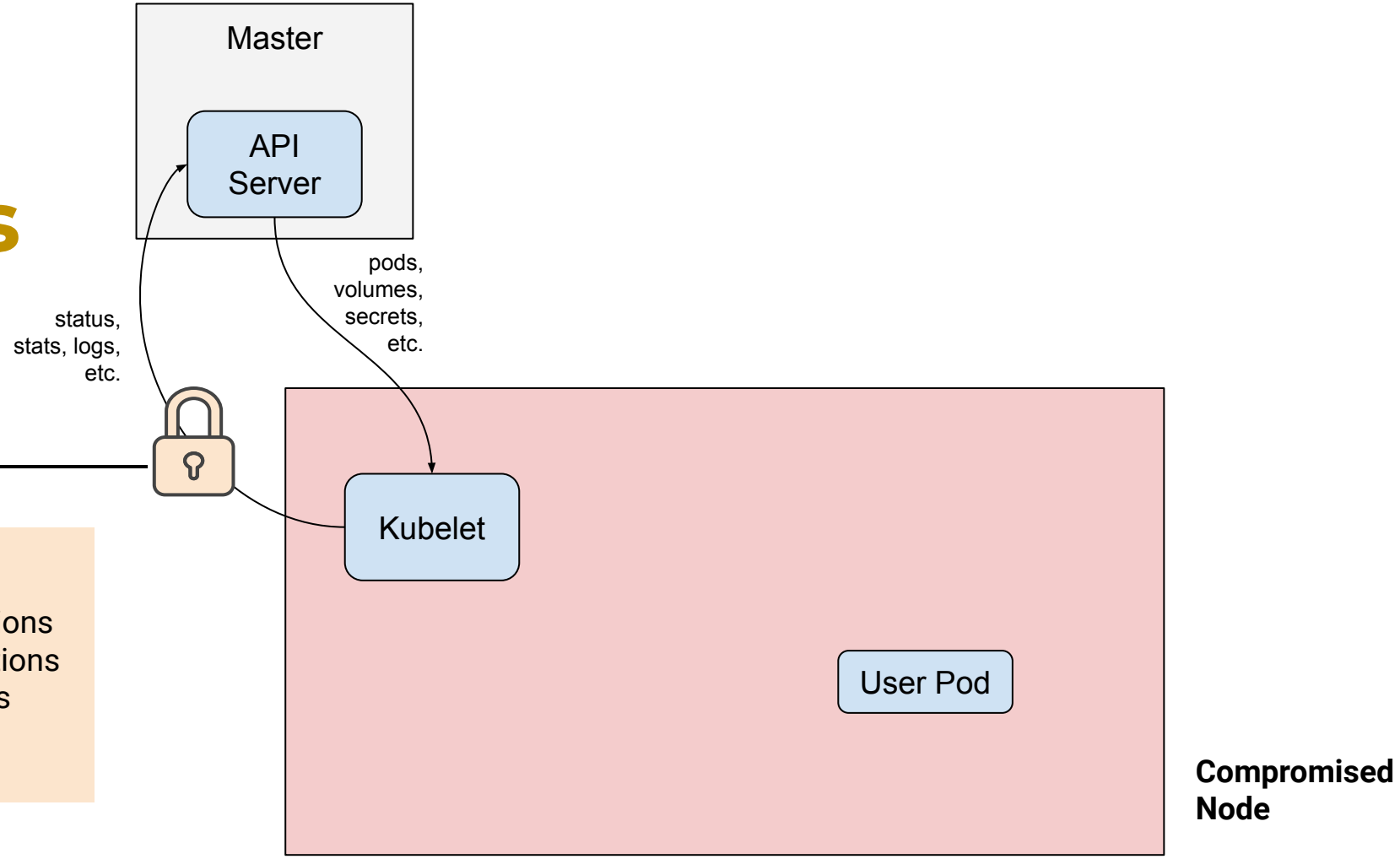
Extended NodeRestrictions for Pods: https://bit.ly/2XdeWOF

Bounding Self-Labeling Kubelets: https://bit.ly/351BaFN

Node vs. Pod isolation



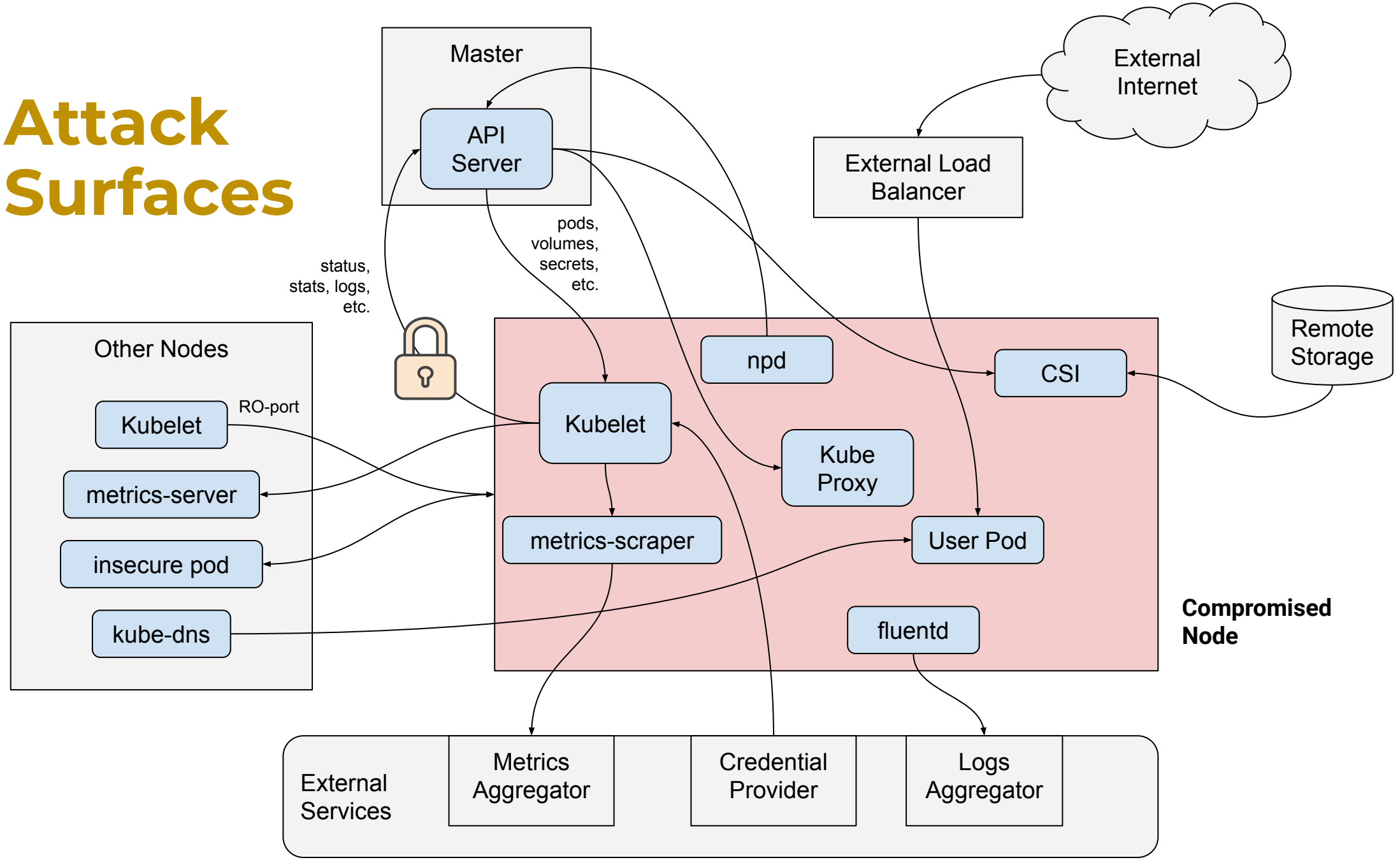
Attack Surfaces



Node Authorizer

- Mirror Pod Restrictions
- Scheduling Restrictions
- Secrets Restrictions
- ...

Attack Surfaces



Node vs. Pod Isolation

	Nodes	Pods

Node vs. Pod Isolation

	Nodes	Pods
Authorization	Union of all the permissions of everything on the node	Only what is needed by containers in the pod

Node vs. Pod Isolation

	Nodes	Pods
Authorization	Union of all the permissions of everything on the node	Only what is needed by containers in the pod
Network Access	Union of all network access required by the node	Can be restricted per-application with NetworkPolicy, Istio, etc.

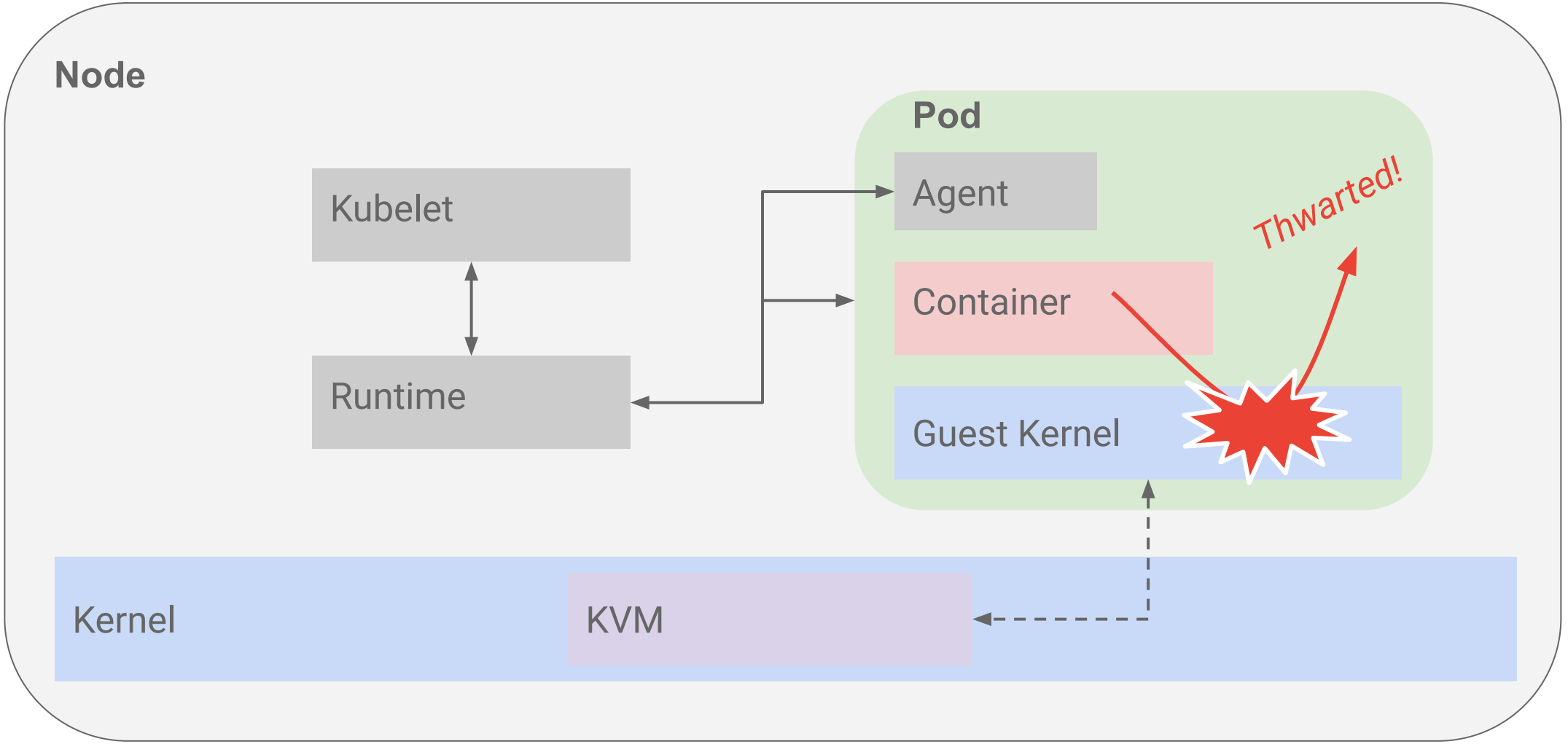
Node vs. Pod Isolation

	Nodes	Pods
Authorization	Union of all the permissions of everything on the node	Only what is needed by containers in the pod
Network Access	Union of all network access required by the node	Can be restricted per-application with NetworkPolicy, Istio, etc.
Monitoring	Measurements are made from within the node	Measurements may be made from outside the pod

Node vs. Pod Isolation

	Nodes	Pods
Authorization	Union of all the permissions of everything on the node	Only what is needed by containers in the pod
Network Access	Union of all network access required by the node	Can be restricted per-application with NetworkPolicy, Istio, etc.
Monitoring	Measurements are made from within the node	Measurements may be made from outside the pod
Resource Usage	Strong isolation, depending on underlying infrastructure	Some isolation through cgroups, subject to noisy neighbors

Sandboxes


Node
Pod
Kubelet
Runtime
Agent
Container
Guest Kernel
Thwarted!
Kernel
KVM

Takeaways

Node Isolation Isn’t Your Only Defense

Compromise Application

Remote Code Execution

Escape Container

And Escalate to Root

Escape Node

Attack Cluster

What can you do?

Harden the application:

1. Patch, patch, and patch some more!
2. Choose a secure base image
3. Apply application specific hardening

What can you do?

Harden the container:

1. Run as non root!
   + AllowPrivilegeEscalation = false
2. Use resource limits
3. Use least privilege authorization
   AutomountServiceAccountToken = false
4. Secure the network access

What can you do?

Sandbox the pod:

- GKE Sandboxes with gVisor
- Per-pod VM with Kata-Containers

Separate kernel per-pod defends against many Linux bugs.

Key Takeaways

1. Nodes are really complicated! There are many known weaknesses in node isolation.

2. Node isolation shouldn't be your only defense.

3. Look at pod isolation and sandboxing for strong isolation.

Links and references

Node Authorizer: https://bit.ly/33XRIPb

Node Restriction: https://bit.ly/2QkRqhk

Kubelet Static Pods: https://bit.ly/2Qj0DGL

Extended NodeRestrictions for Pods: https://bit.ly/2XdeWOF

Bounding Self-Labeling Kubelets: https://bit.ly/351BaFN

GKE hardening guide: g.co/gke/hardening

GKE sandboxes: g.co/gke/sandbox

Kata containers: katacontainers.io

Kubernetes blog on runc vuln: https://bit.ly/2QmbOhU

Dirty COW vuln: dirtycow.ninja

Greg Castle
GKE Security Tech Lead
Twitter: @mrgcastle
Github: @destijl
Google

Tim Allclair
GKE Security Engineer
Twitter: @tallclair
Github: @tallclair
Google

So Many Great Security Talks!

State of Kubernetes Security https://bit.ly/2OdqgWC

CJ Cullen & Tim Allclair: Mon 11:00am

“The Devil in the Details: Kubernetes’ First Security Assessment” https://bit.ly/34VkAr2

Aaron Small, Google & Jay Beale: Tue 10:55am

Walls Within Walls: What If Your Attacker Knows Parkour?” https://bit.ly/33PZiLl

Greg Castle and Tim Allclair: Tue 3:20pm

“Binary Authorization in Kubernetes” https://bit.ly/32L2yqj

Aysylu Greenberg & Liron Levin: Wed 10:55am

“Piloting Around the Rocks: Avoiding Threats in Kubernetes” https://bit.ly/36XLAbc

Robert Tonic and Stefan Edwards : Wed 2:25pm

“Hello from the Other Side: Dispatches from a Kubernetes Attacker” https://bit.ly/2NBpe7Y

Ian Coldwater : Thur 9:22 am

“How Kubernetes Components Communicate Securely in Your Cluster” https://bit.ly/2QrIzKP

Maya Kaczorowski: Thur 11:50am

“Sig-Auth Update” https://bit.ly/2Kk7kEQ

Mike Danese, Tim Allclair, Mo Khan: Thur 2:25pm

“Attacking and Defending Kubernetes Clusters: A Guided Tour” https://bit.ly/36Xb0G0

Brad Geesaman, Jimmy Mesta, Tabitha Sable, Peter Benjamin : Thur 4:25pm