KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

SIGs Aren't Silos

A Case Study Into Solving Inter-Domain Problems In Kubernetes Development

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

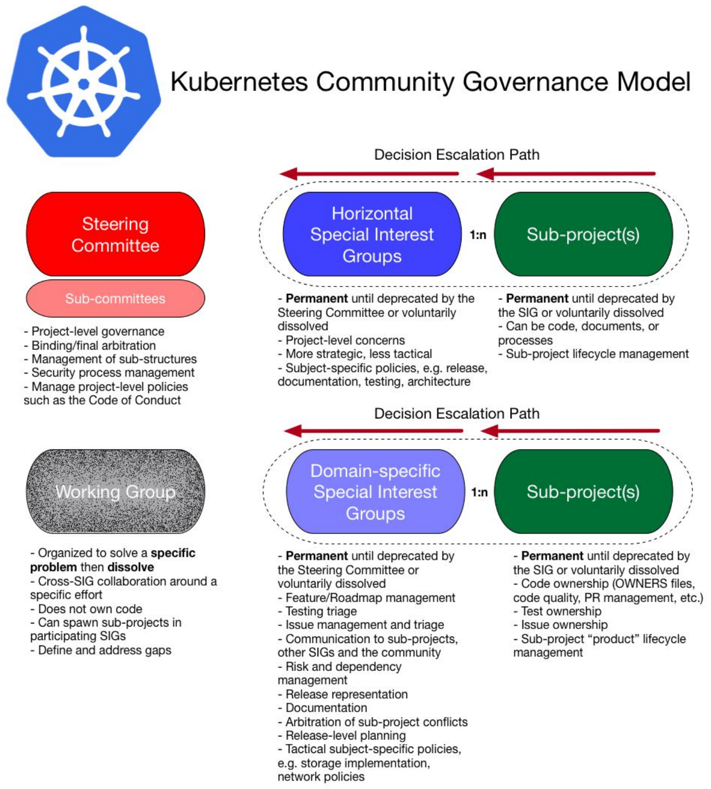
DETROIT 2022

October 24-28, 2021

Swetha Repakula
Software Engineer
Google

Antonio Ojea
Software Engineer
Google

Kubernetes community: Governance



Steering Committee / Sub-committees
- Project-level governance
- Binding/final arbitration
- Management of sub-structures
- Security process management
- Manage project-level policies such as the Code of Conduct

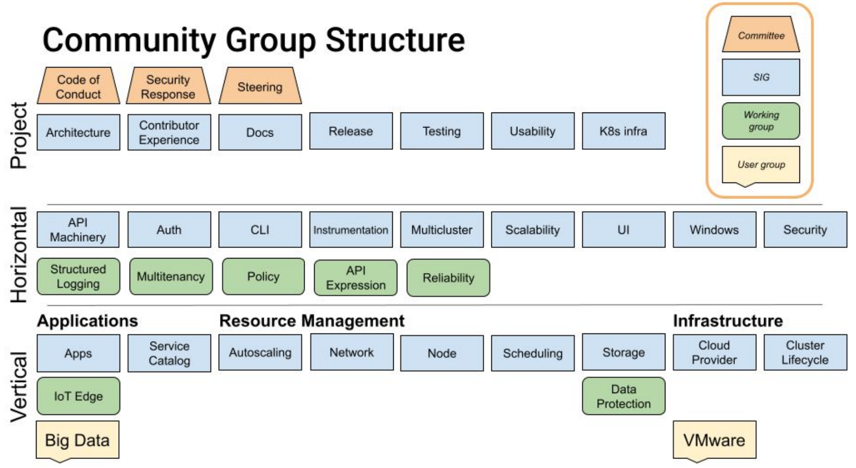
Horizontal Special Interest Groups
- **Permanent** until deprecated by the Steering Committee or voluntarily dissolved
- Project-level concerns
- More strategic, less tactical
- Subject-specific policies, e.g. release, documentation, testing, architecture

Sub-project(s)
- **Permanent** until deprecated by the SIG or voluntarily dissolved
- Can be code, documents, or processes
- Sub-project lifecycle management

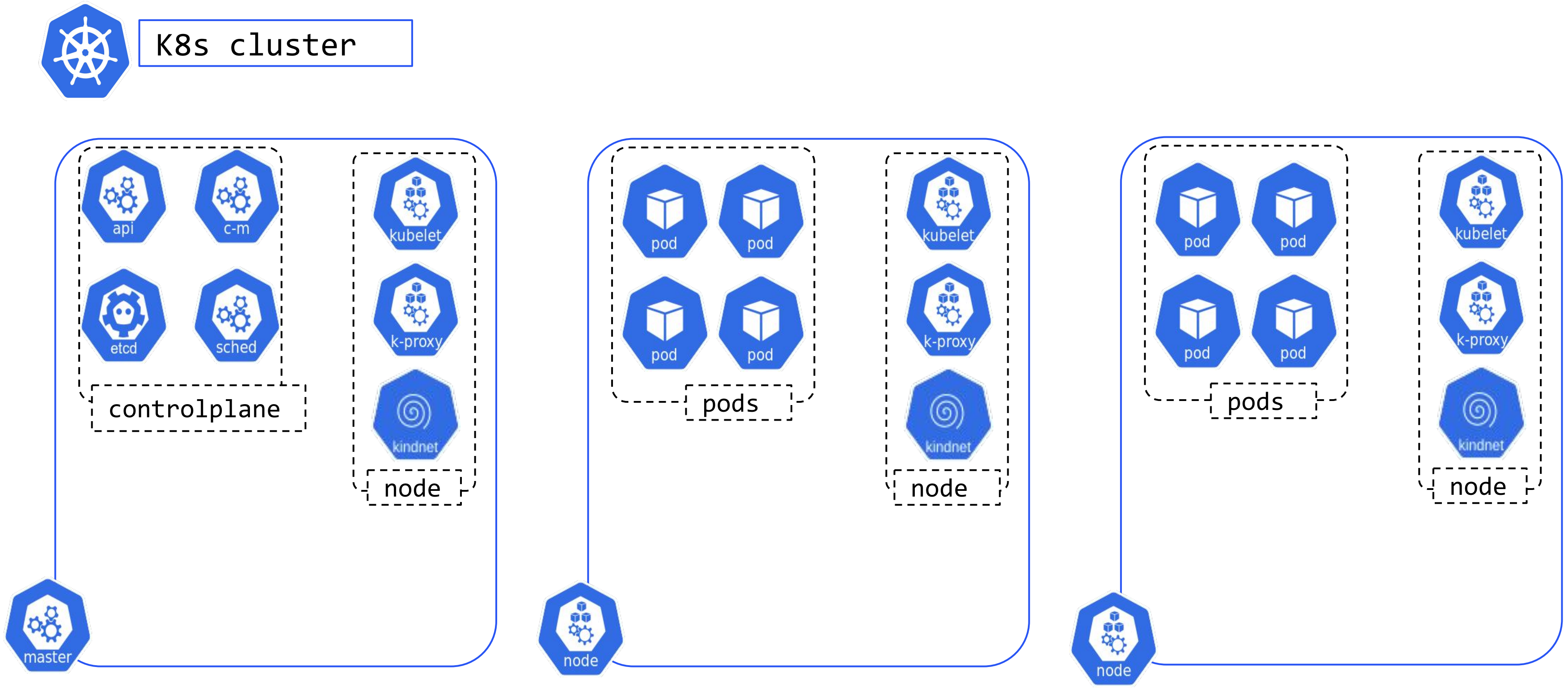
Working Group
- Organized to solve a **specific problem** then **dissolve**
- Cross-SIG collaboration around a specific effort
- Does not own code
- Can spawn sub-projects in participating SIGs
- Define and address gaps

Domain-specific Special Interest Groups
- **Permanent** until deprecated by the Steering Committee or voluntarily dissolved
- Feature/Roadmap management
- Testing triage
- Issue management and triage
- Communication to sub-projects, other SIGs and the community
- Risk and dependency management
- Release representation
- Documentation
- Arbitration of sub-project conflicts
- Release-level planning
- Tactical subject-specific policies, e.g. storage implementation, network policies

Sub-project(s)
- **Permanent** until deprecated by the SIG or voluntarily dissolved
- Code ownership (OWNERS files, code quality, PR management, etc.)
- Test ownership
- Issue ownership
- Sub-project "product" lifecycle management

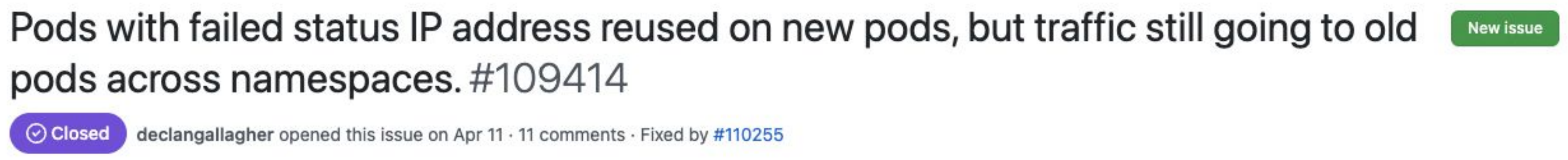


Kubernetes components

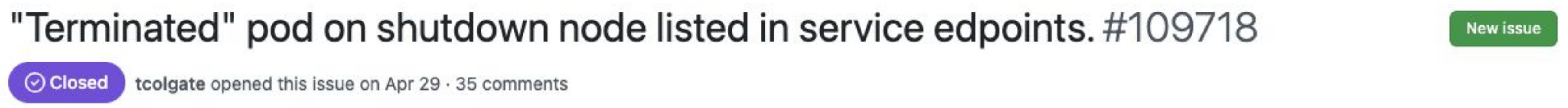


Kubernetes community: People/Friends

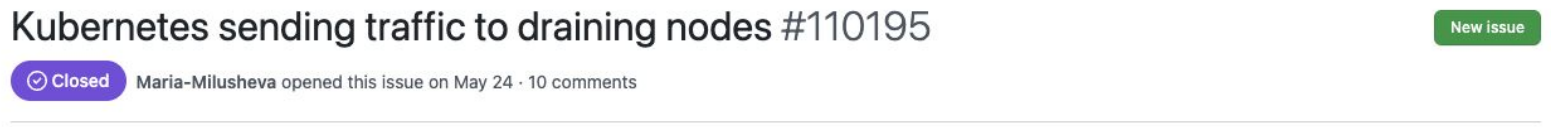
SIG Network and Node bugs


Pods with failed status IP address reused on new pods, but traffic still going to old pods across namespaces. #109414

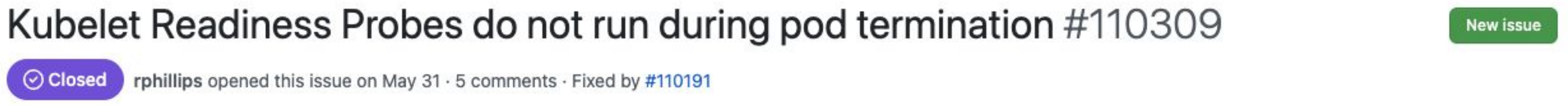
Closed **declangallagher** opened this issue on Apr 11 · 11 comments · Fixed by #110255


"Terminated" pod on shutdown node listed in service edpoints. #109718

Closed **tcolgate** opened this issue on Apr 29 · 35 comments


Kubernetes sending traffic to draining nodes #110195

Closed **Maria-Milusheva** opened this issue on May 24 · 10 comments

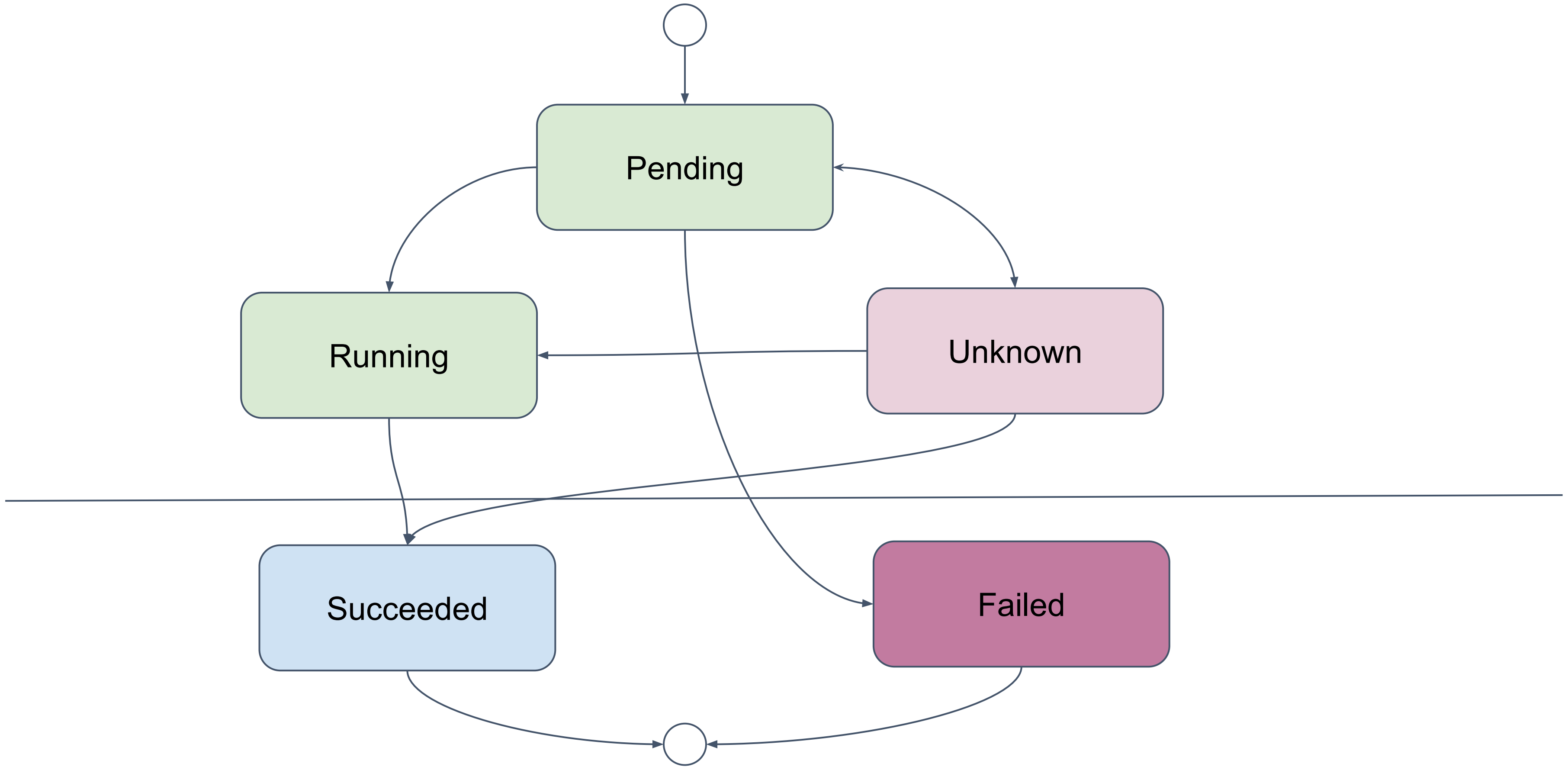

Kubelet Readiness Probes do not run during pod termination #110309

Closed **rphillips** opened this issue on May 31 · 5 comments · Fixed by #110191

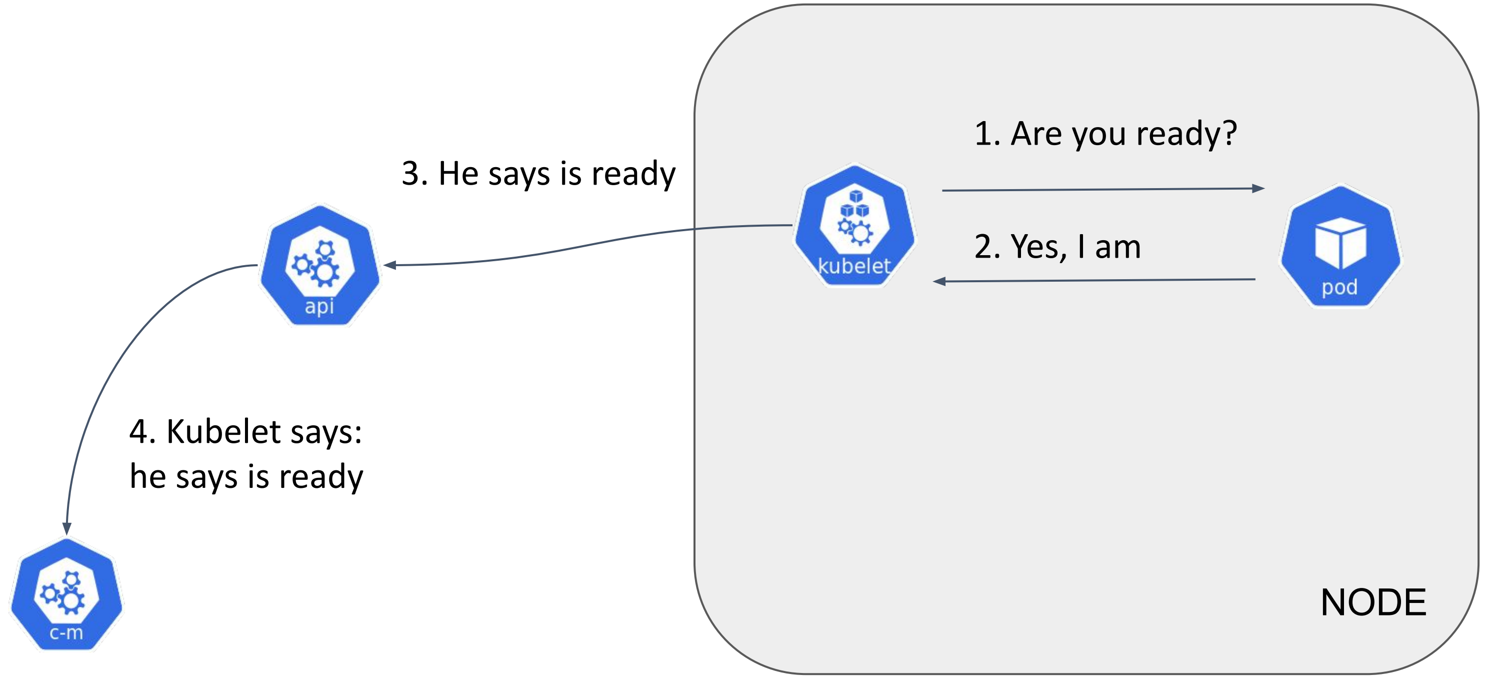
Symptoms

- Traffic was being routed to non-existent pods
- Traffic was being routed to the wrong pods
- IPs in EndpointSlices were not matching the Node Pod CIDR

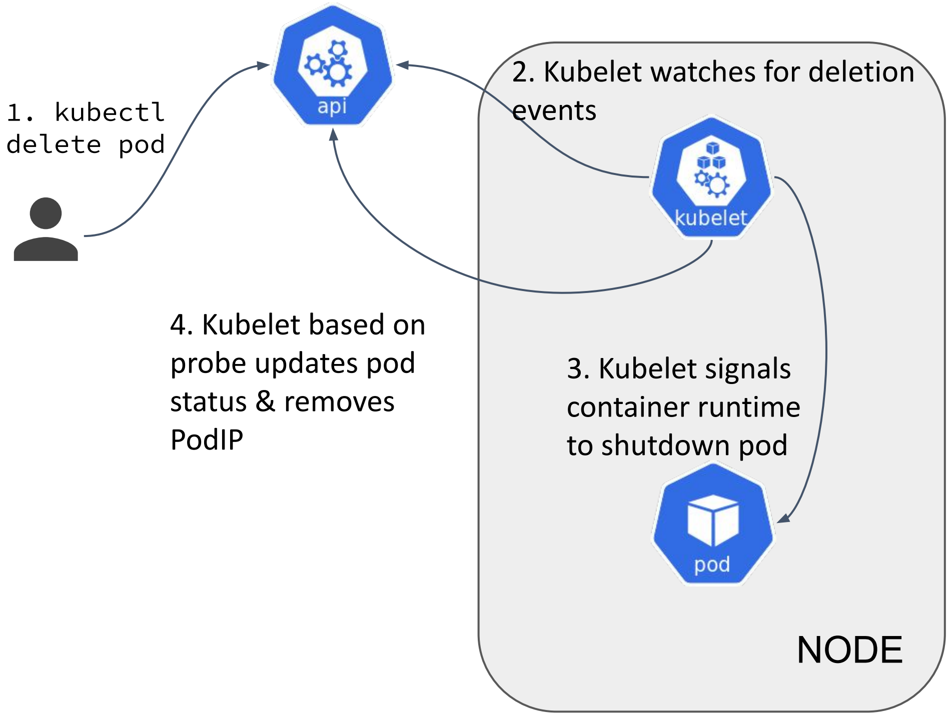
Pod Lifecycle: Pod Phases



Pod Lifecycle: Pod Readiness

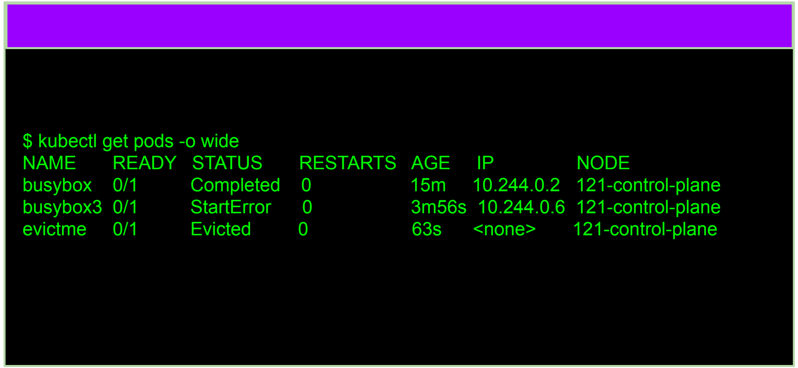


Pod Shutdown - Before Refactor



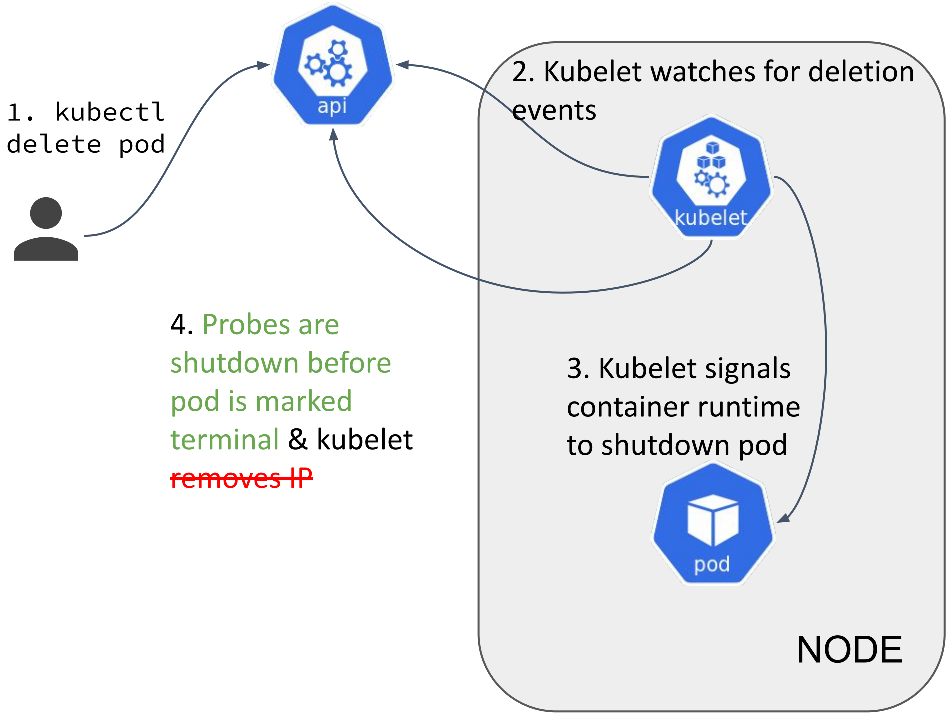
api
1. kubectl delete pod
2. Kubelet watches for deletion events
kubelet
3. Kubelet signals container runtime to shutdown pod
pod
4. Kubelet based on probe updates pod status & removes PodIP
NODE

When nodes are shutdown, or if pods are evicted, pods will not be deleted until PodGC kicks in (based on threshold).

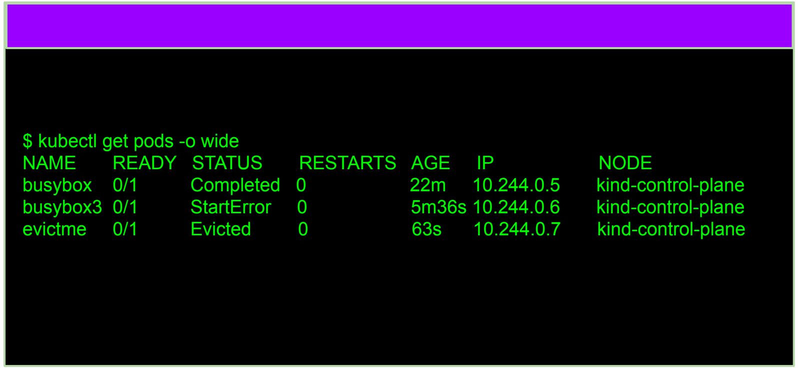


```
$ kubectl get pods -o wide
NAME      READY  STATUS     RESTARTS  AGE    IP          NODE
busybox   0/1    Completed  0         15m    10.244.0.2  121-control-plane
busybox3  0/1    StartError 0         3m56s  10.244.0.6  121-control-plane
evictme   0/1    Evicted    0         63s    <none>      121-control-plane
```

Pod Shutdown - After Refactor

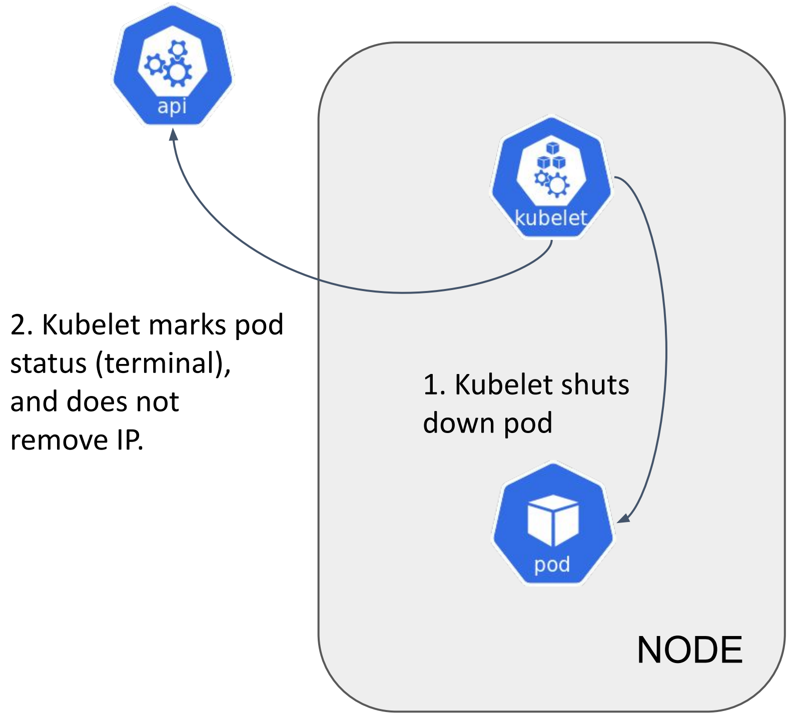


1. kubectl delete pod
api
2. Kubelet watches for deletion events
kubelet
3. Kubelet signals container runtime to shutdown pod
pod
NODE
4. Probes are shutdown before pod is marked terminal & kubelet ~~removes IP~~



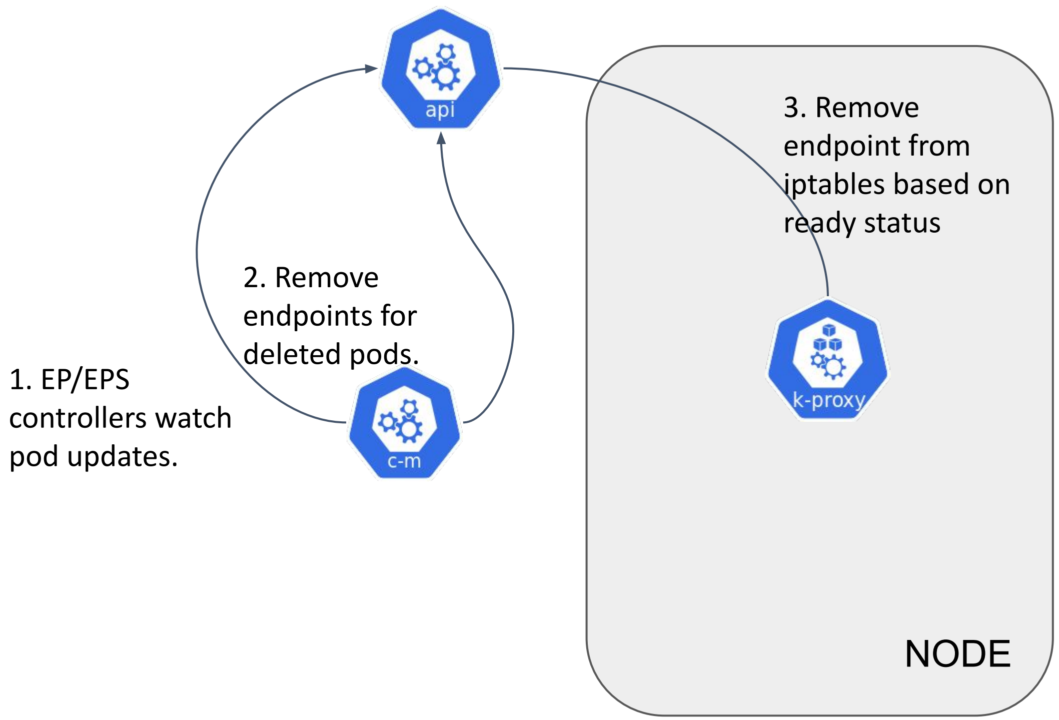
```
$ kubectl get pods -o wide
NAME       READY  STATUS      RESTARTS  AGE    IP           NODE
busybox    0/1    Completed   0         22m    10.244.0.5   kind-control-plane
busybox3   0/1    StartError  0         5m36s  10.244.0.6   kind-control-plane
evictme    0/1    Evicted     0         63s    10.244.0.7   kind-control-plane
```

Pod Shutdown - Pod Eviction


api
kubelet
pod
NODE
2. Kubelet marks pod status (terminal), and does not remove IP.
1. Kubelet shuts down pod

When nodes are shutdown, or if pods are evicted, pods will not be deleted until PodGC kicks in (based on threshold).

Service/Endpoints Removal

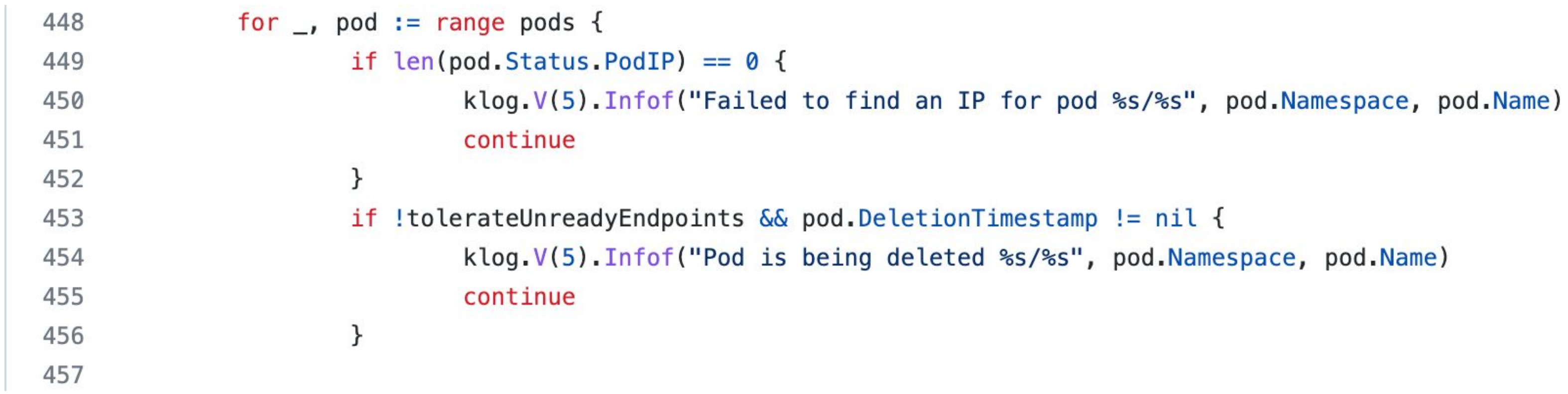


1. EP/EPS controllers watch pod updates.

2. Remove endpoints for deleted pods.

3. Remove endpoint from iptables based on ready status

The bug …

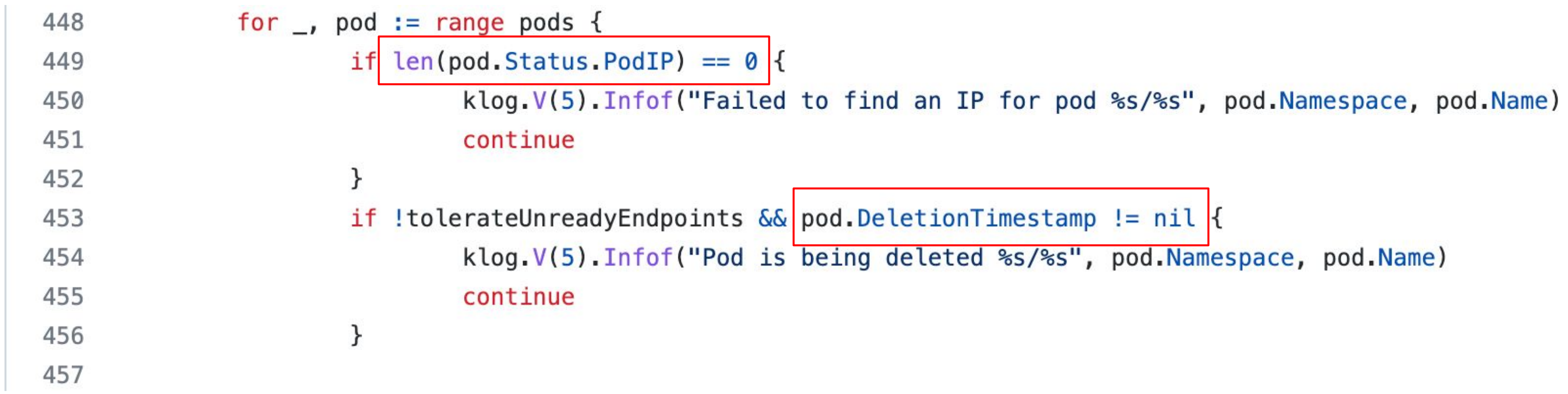


```go
for _, pod := range pods {
        if len(pod.Status.PodIP) == 0 {
                klog.V(5).Infof("Failed to find an IP for pod %s/%s", pod.Namespace, pod.Name)
                continue
        }
        if !tolerateUnreadyEndpoints && pod.DeletionTimestamp != nil {
                klog.V(5).Infof("Pod is being deleted %s/%s", pod.Namespace, pod.Name)
                continue
        }

```

pkg/controller/endpoint/endpoints_controller.go (v1.22.0)

The bug ...

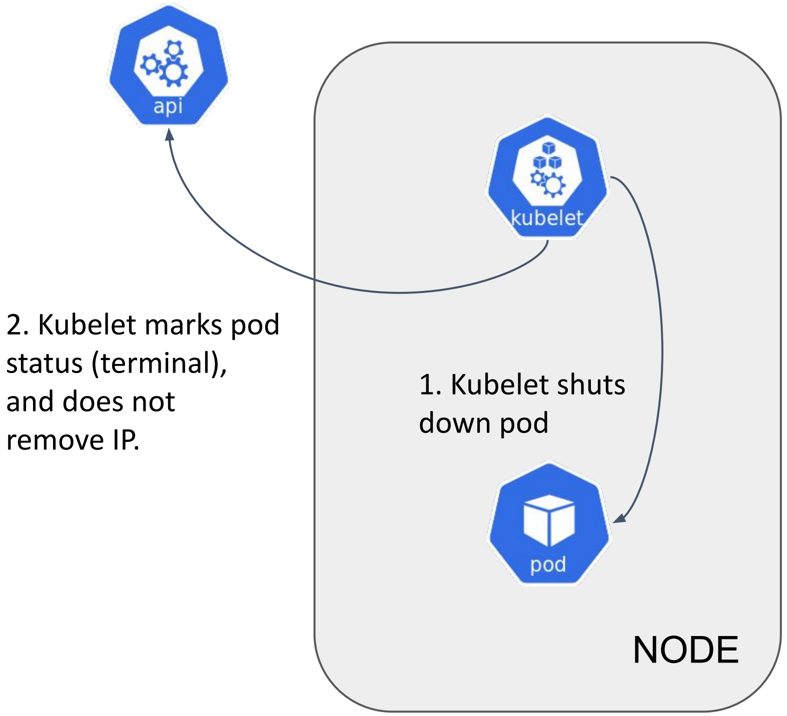


```go
	for _, pod := range pods {
		if len(pod.Status.PodIP) == 0 {
			klog.V(5).Infof("Failed to find an IP for pod %s/%s", pod.Namespace, pod.Name)
			continue
		}
		if !tolerateUnreadyEndpoints && pod.DeletionTimestamp != nil {
			klog.V(5).Infof("Pod is being deleted %s/%s", pod.Namespace, pod.Name)
			continue
		}

```

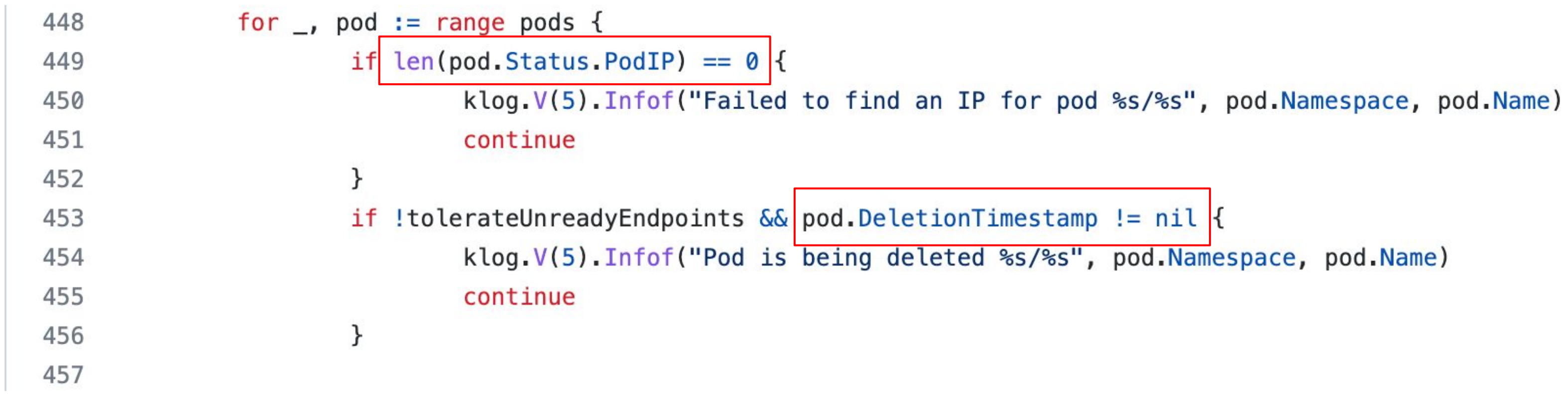
pkg/controller/endpoint/endpoints_controller.go (v1.22.0)

Pod Shutdown - Pod Eviction


api
kubelet
2. Kubelet marks pod status (terminal), and does not remove IP.
1. Kubelet shuts down pod
pod
NODE

When nodes are shutdown, or if pods are evicted, pods will not be deleted until PodGC kicks in (based on threshold).

The bug ...

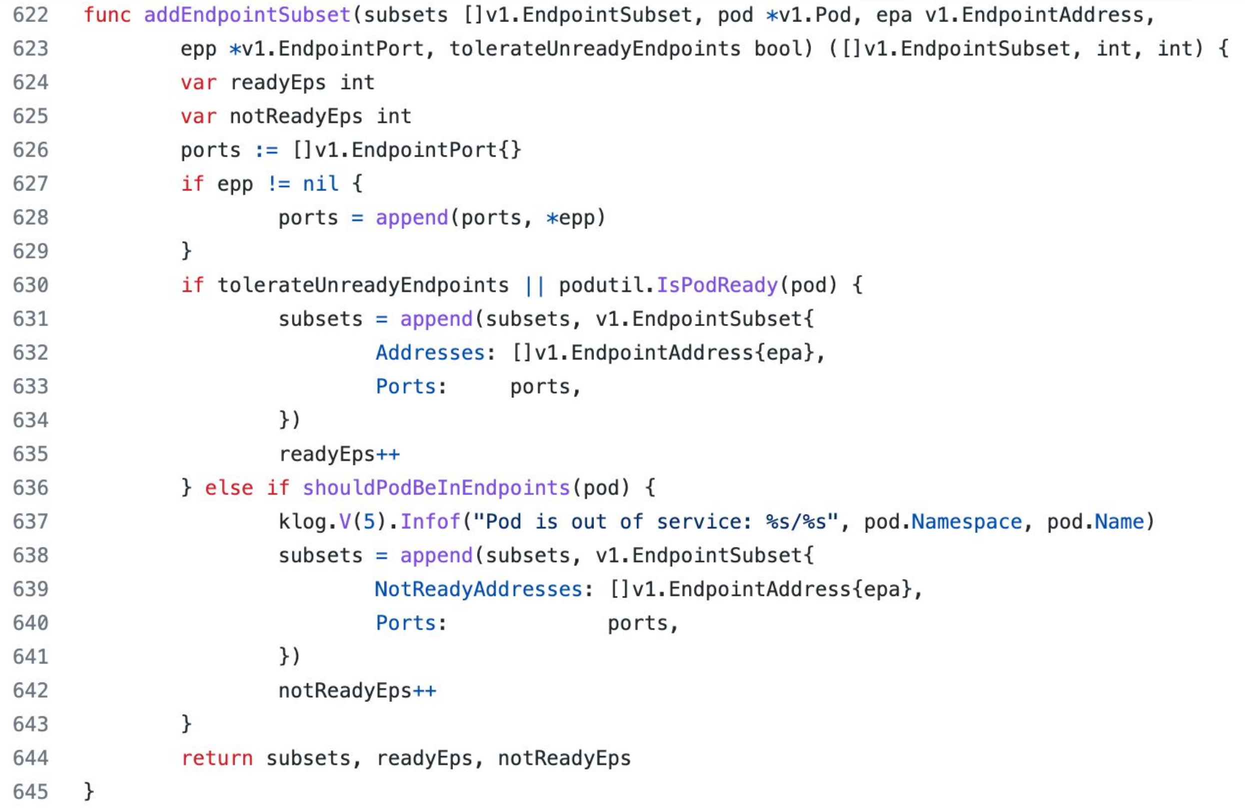


```go
	for _, pod := range pods {
		if len(pod.Status.PodIP) == 0 {
			klog.V(5).Infof("Failed to find an IP for pod %s/%s", pod.Namespace, pod.Name)
			continue
		}
		if !tolerateUnreadyEndpoints && pod.DeletionTimestamp != nil {
			klog.V(5).Infof("Pod is being deleted %s/%s", pod.Namespace, pod.Name)
			continue
		}

```

pkg/controller/endpoint/endpoints_controller.go (v1.22.0)

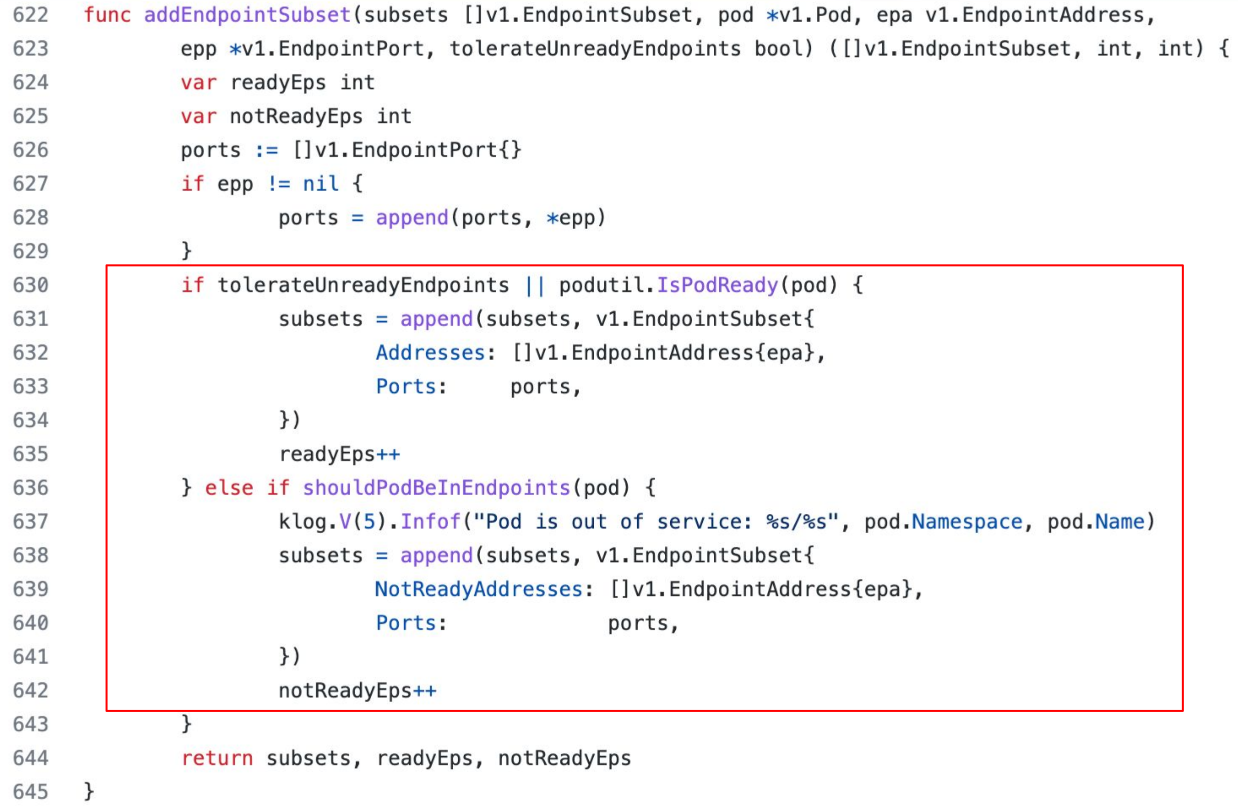
The bug …



```go
func addEndpointSubset(subsets []v1.EndpointSubset, pod *v1.Pod, epa v1.EndpointAddress,
        epp *v1.EndpointPort, tolerateUnreadyEndpoints bool) ([]v1.EndpointSubset, int, int) {
        var readyEps int
        var notReadyEps int
        ports := []v1.EndpointPort{}
        if epp != nil {
                ports = append(ports, *epp)
        }
        if tolerateUnreadyEndpoints || podutil.IsPodReady(pod) {
                subsets = append(subsets, v1.EndpointSubset{
                        Addresses: []v1.EndpointAddress{epa},
                        Ports:     ports,
                })
                readyEps++
        } else if shouldPodBeInEndpoints(pod) {
                klog.V(5).Infof("Pod is out of service: %s/%s", pod.Namespace, pod.Name)
                subsets = append(subsets, v1.EndpointSubset{
                        NotReadyAddresses: []v1.EndpointAddress{epa},
                        Ports:             ports,
                })
                notReadyEps++
        }
        return subsets, readyEps, notReadyEps
}
```

pkg/controller/endpoint/endpoints_controller.go (v1.22.0)

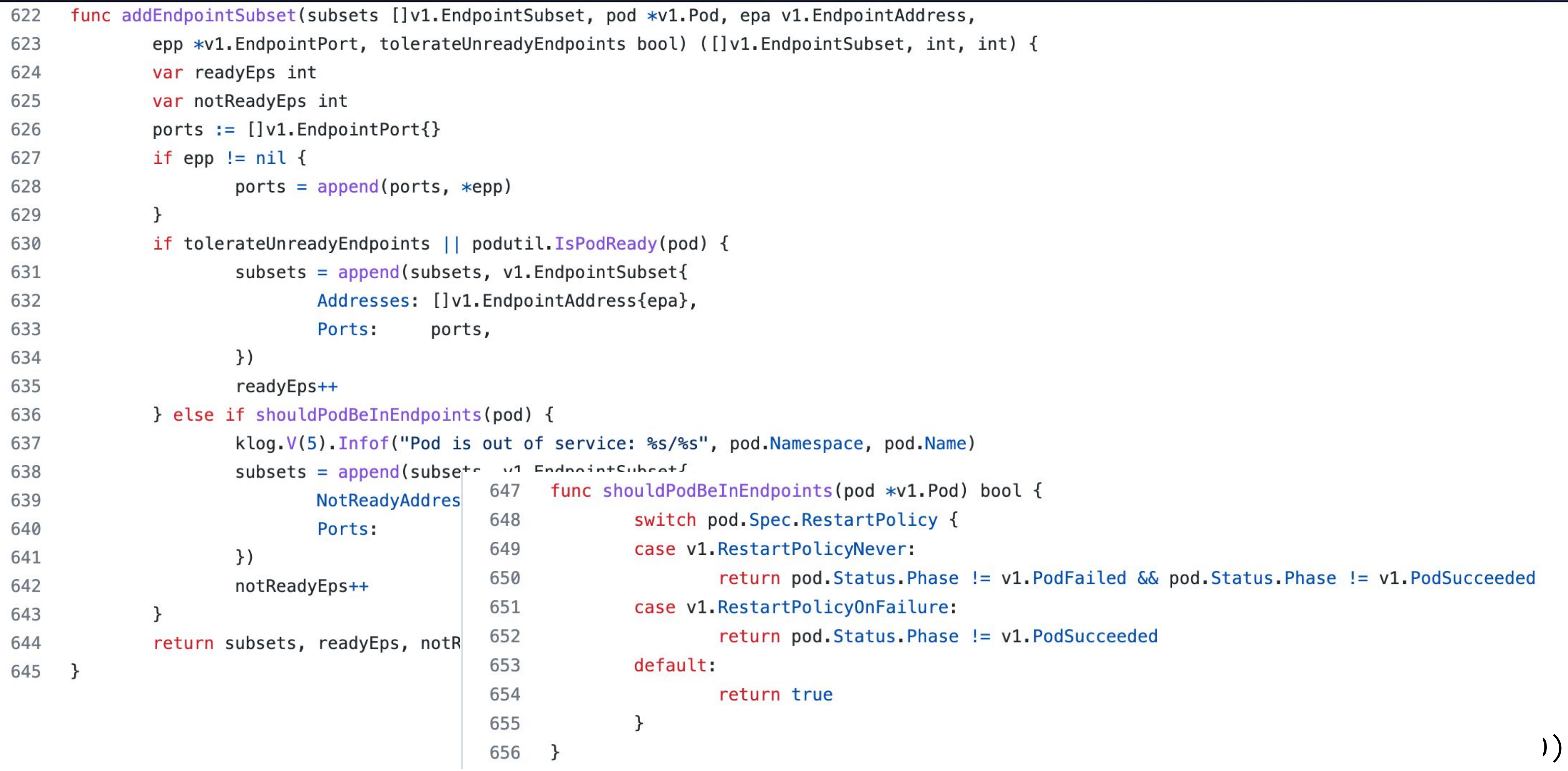
The bug …



```go
func addEndpointSubset(subsets []v1.EndpointSubset, pod *v1.Pod, epa v1.EndpointAddress,
	epp *v1.EndpointPort, tolerateUnreadyEndpoints bool) ([]v1.EndpointSubset, int, int) {
	var readyEps int
	var notReadyEps int
	ports := []v1.EndpointPort{}
	if epp != nil {
		ports = append(ports, *epp)
	}
	if tolerateUnreadyEndpoints || podutil.IsPodReady(pod) {
		subsets = append(subsets, v1.EndpointSubset{
			Addresses: []v1.EndpointAddress{epa},
			Ports:     ports,
		})
		readyEps++
	} else if shouldPodBeInEndpoints(pod) {
		klog.V(5).Infof("Pod is out of service: %s/%s", pod.Namespace, pod.Name)
		subsets = append(subsets, v1.EndpointSubset{
			NotReadyAddresses: []v1.EndpointAddress{epa},
			Ports:             ports,
		})
		notReadyEps++
	}
	return subsets, readyEps, notReadyEps
}
```

pkg/controller/endpoint/endpoints_controller.go (v1.22.0)

The bug ...

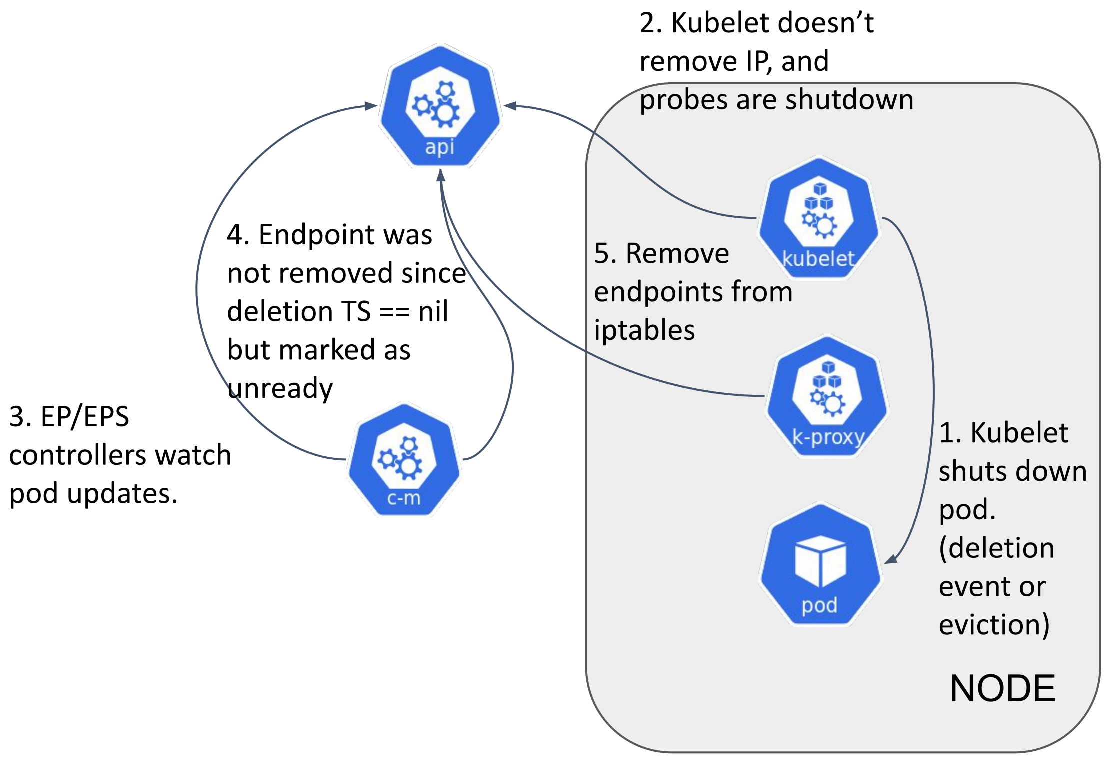


```go
func addEndpointSubset(subsets []v1.EndpointSubset, pod *v1.Pod, epa v1.EndpointAddress,
	epp *v1.EndpointPort, tolerateUnreadyEndpoints bool) ([]v1.EndpointSubset, int, int) {
	var readyEps int
	var notReadyEps int
	ports := []v1.EndpointPort{}
	if epp != nil {
		ports = append(ports, *epp)
	}
	if tolerateUnreadyEndpoints || podutil.IsPodReady(pod) {
		subsets = append(subsets, v1.EndpointSubset{
			Addresses: []v1.EndpointAddress{epa},
			Ports:     ports,
		})
		readyEps++
	} else if shouldPodBeInEndpoints(pod) {
		klog.V(5).Infof("Pod is out of service: %s/%s", pod.Namespace, pod.Name)
		subsets = append(subse
			NotReadyAddres
			Ports:
		})
		notReadyEps++
	}
	return subsets, readyEps, notR
}
```

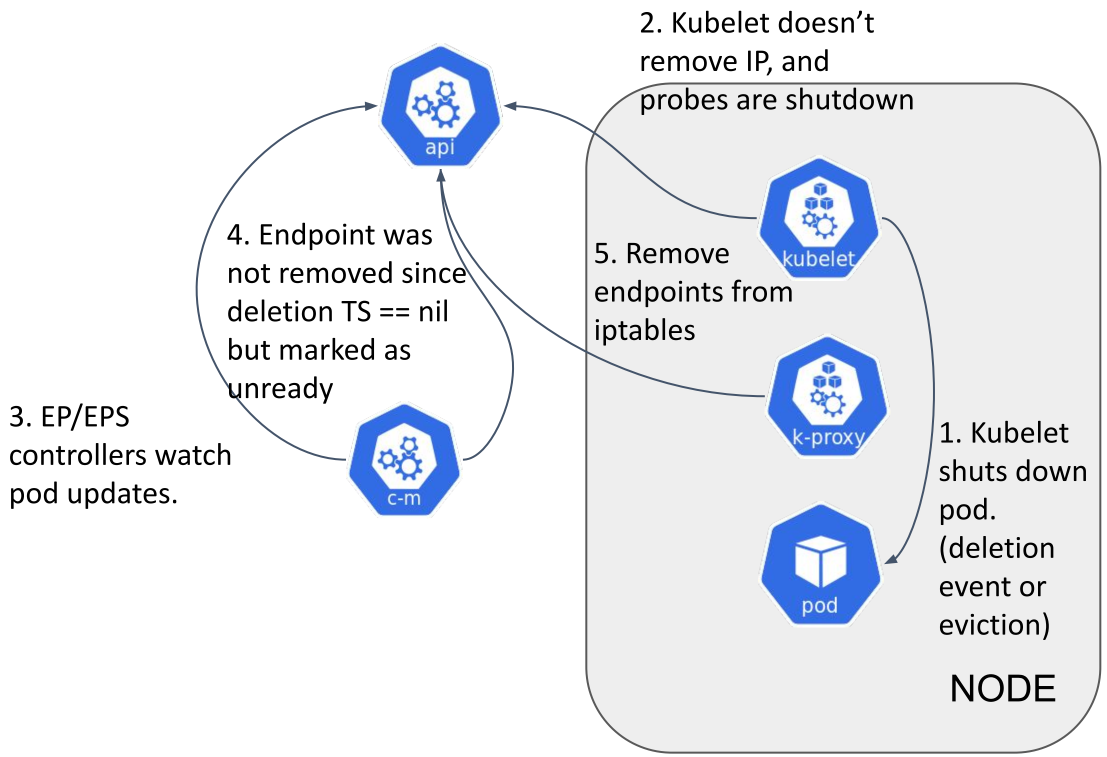
```go
func shouldPodBeInEndpoints(pod *v1.Pod) bool {
	switch pod.Spec.RestartPolicy {
	case v1.RestartPolicyNever:
		return pod.Status.Phase != v1.PodFailed && pod.Status.Phase != v1.PodSucceeded
	case v1.RestartPolicyOnFailure:
		return pod.Status.Phase != v1.PodSucceeded
	default:
		return true
	}
}
```

))

The bug …


2. Kubelet doesn't remove IP, and probes are shutdown
api
kubelet
4. Endpoint was not removed since deletion TS == nil but marked as unready
5. Remove endpoints from iptables
k-proxy
3. EP/EPS controllers watch pod updates.
c-m
1. Kubelet shuts down pod. (deletion event or eviction)
pod
NODE

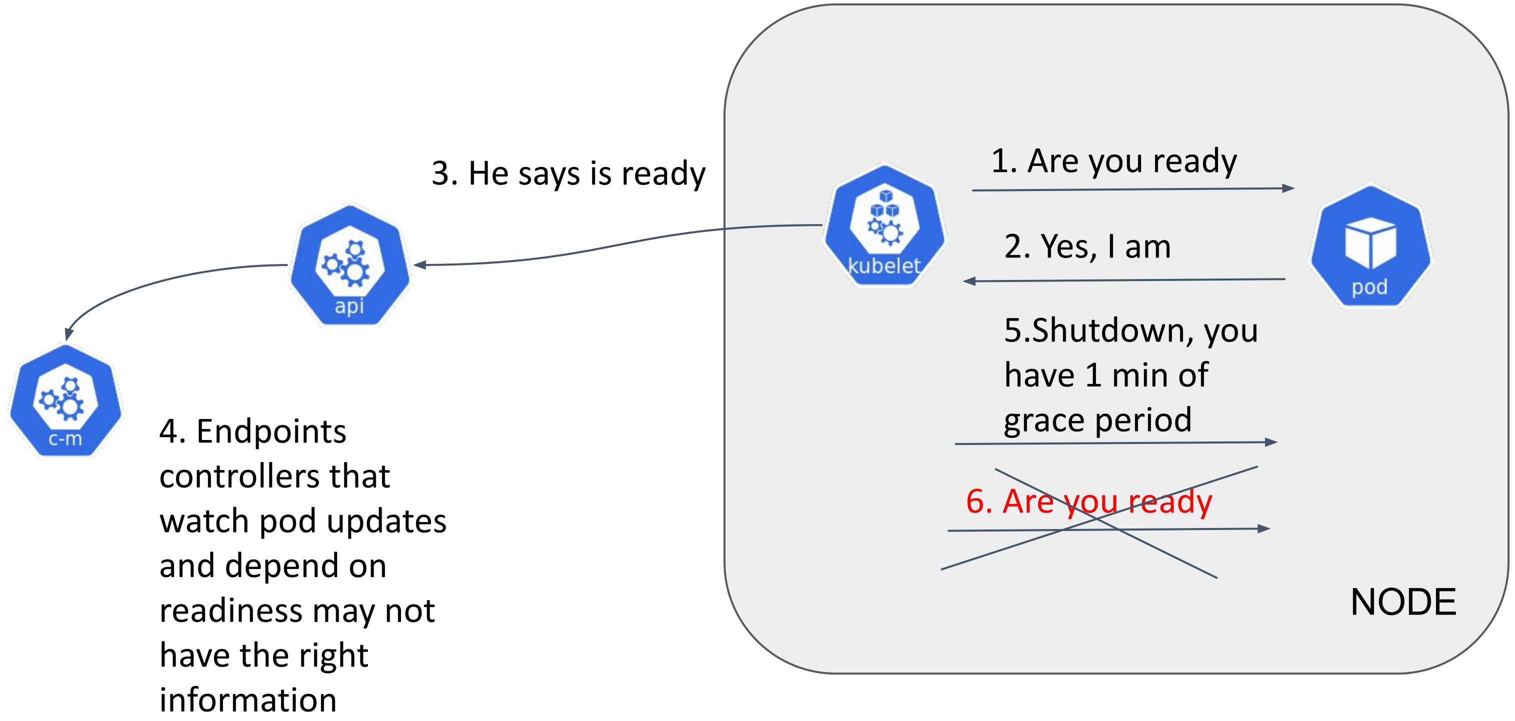
… and its consequences



2. Kubelet doesn't remove IP, and probes are shutdown
api
kubelet
4. Endpoint was not removed since deletion TS == nil but marked as unready
5. Remove endpoints from iptables
k-proxy
3. EP/EPS controllers watch pod updates.
c-m
1. Kubelet shuts down pod. (deletion event or eviction)
pod
NODE

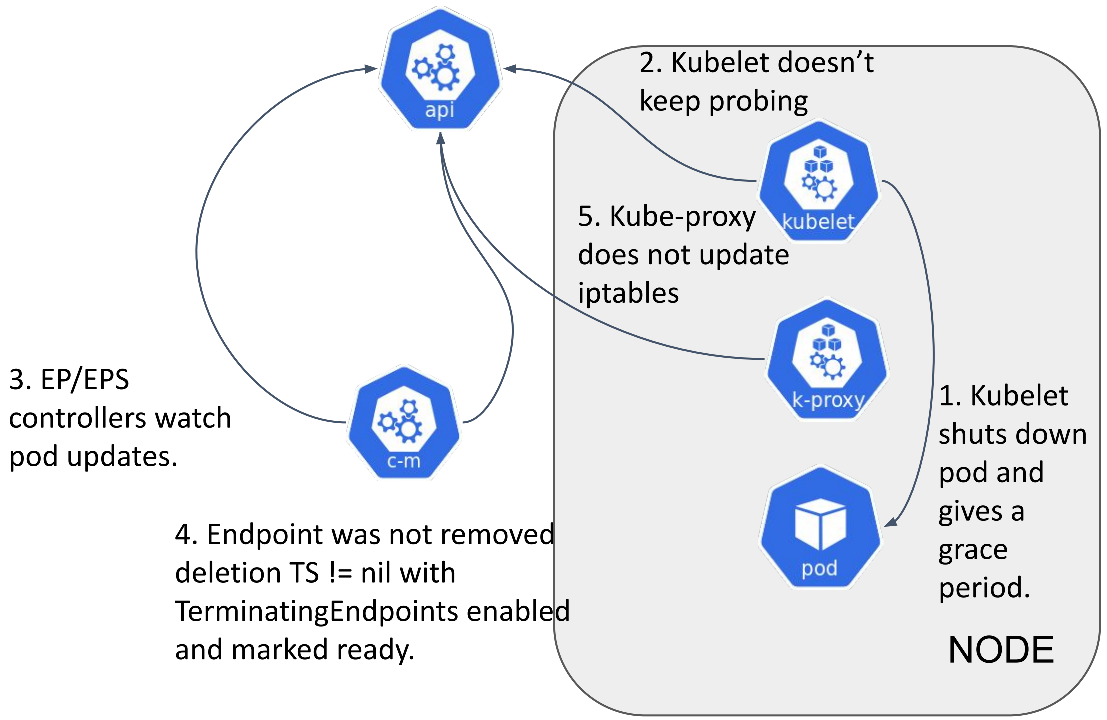
Endpoints and EndpointSlice resources have an endpoint that doesn't point to a current pod but marked as unready, so kube-proxy will removed from iptables.

The other bug … TerminatingEndpoints



3. He says is ready
kubelet
1. Are you ready
2. Yes, I am
pod
api
c-m
4. Endpoints controllers that watch pod updates and depend on readiness may not have the right information
5.Shutdown, you have 1 min of grace period
~~6. Are you ready~~
NODE

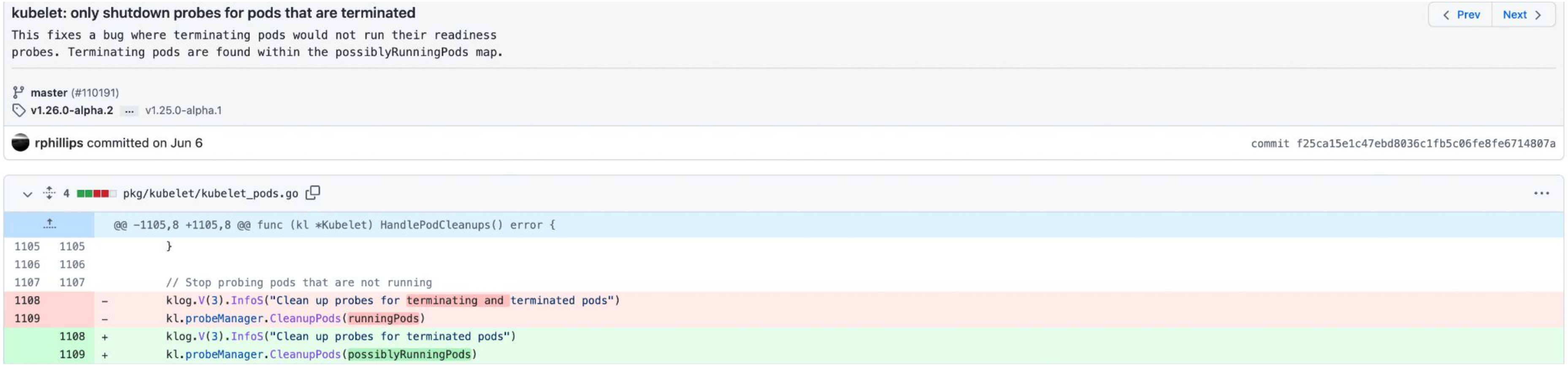
… and its consequences


api
2. Kubelet doesn't keep probing
kubelet
5. Kube-proxy does not update iptables
k-proxy
3. EP/EPS controllers watch pod updates.
c-m
1. Kubelet shuts down pod and gives a grace period.
pod
4. Endpoint was not removed deletion TS != nil with TerminatingEndpoints enabled and marked ready.
NODE

Traffic could be routed pods that area no longer ready. If pod readiness is not checked during the termination grace period, the pod keeps receiving traffic.

The bugfix

KubeCon CloudNativeCon North America 2022



kubelet: only shutdown probes for pods that are terminated

```
This fixes a bug where terminating pods would not run their readiness
probes. Terminating pods are found within the possiblyRunningPods map.
```

< Prev | Next >

master (#110191)

v1.26.0-alpha.2 ... v1.25.0-alpha.1

rphillips committed on Jun 6

commit f25ca15e1c47ebd8036c1fb5c06fe8fe6714807a

4 pkg/kubelet/kubelet_pods.go

```
@@ -1105,8 +1105,8 @@ func (kl *Kubelet) HandlePodCleanups() error {
 	}
 
 	// Stop probing pods that are not running
-	klog.V(3).InfoS("Clean up probes for terminating and terminated pods")
-	kl.probeManager.CleanupPods(runningPods)
+	klog.V(3).InfoS("Clean up probes for terminated pods")
+	kl.probeManager.CleanupPods(possiblyRunningPods)
```

The lessons learned

- All behavior changes have potential to break others
- Networking and Pod/Node lifecycle are tightly coupled
- Behavior Contracts need to be enforced with testing
- “Broken telephone” bugs are hard to solve
- Bugs can become a feature

Thank You!

Thank you!