Every Kubernetes Pod Eviction Path EXPLAINED

Ahmet Alp Balkan
Software Engineer, LinkedIn
@ahmetb

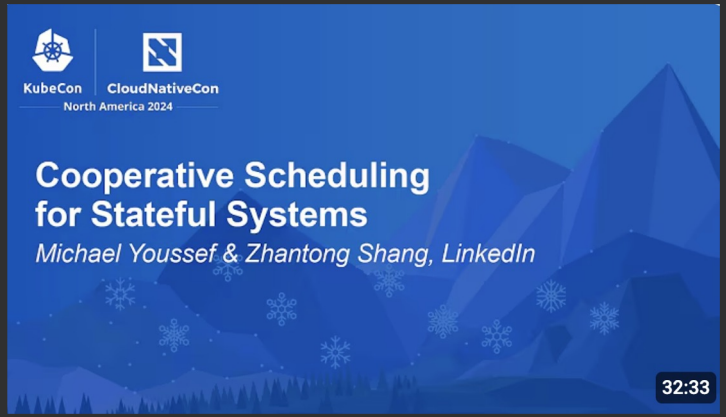
about me

containers & kubernetes (2014-)
developed various tools (kubectx, kubens, krew…)
6th time presenting at kubecon

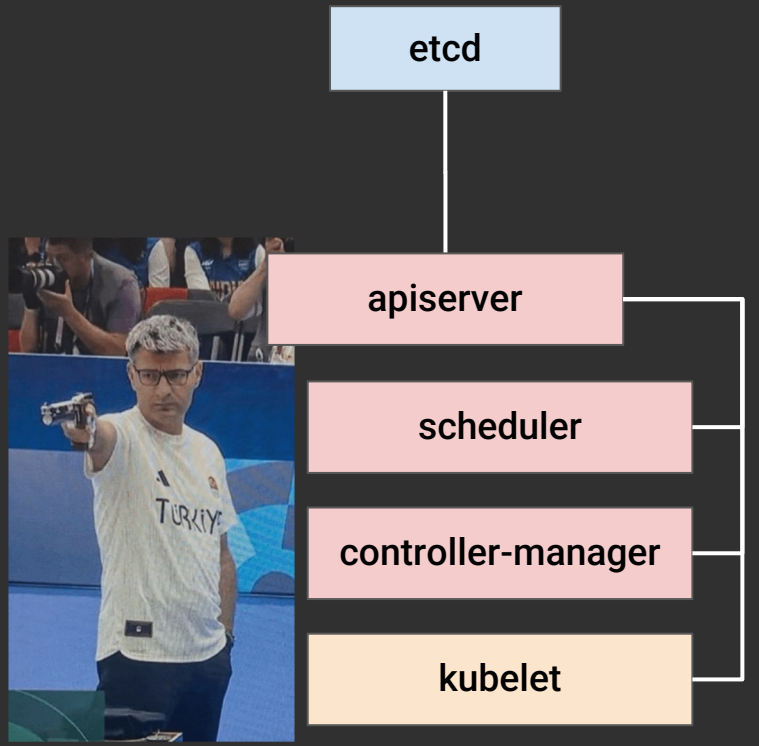
about us

one of the largest Kubernetes installations
fully on bare-metal (500,000+ nodes), thousands of services,
large scale stateful-on-bare-metal, batch jobs, …





everybody is trying to kill your pods



All the ways Kubernetes can kill your pods*

1. Pod deletion
2. Eviction API
3. Node pressure
4. Kubelet admission
5. Kubelet storage eviction
6. Pod Preemption
7. Taint-based Eviction
8. Pod Garbage collection

Most important reliability property

Keep running what's been running.

Why care about Pod evictions?

Kubernetes eviction features might be a “bug” for you.

- ☑ Availability
- ☑ Non-graceful terminations
- ☑ Stateful apps
- ☑ Eviction-averse apps (large ephemeral state on disk, ML training jobs…)

Why do we care about disruptions?

We run on on bare-metal:

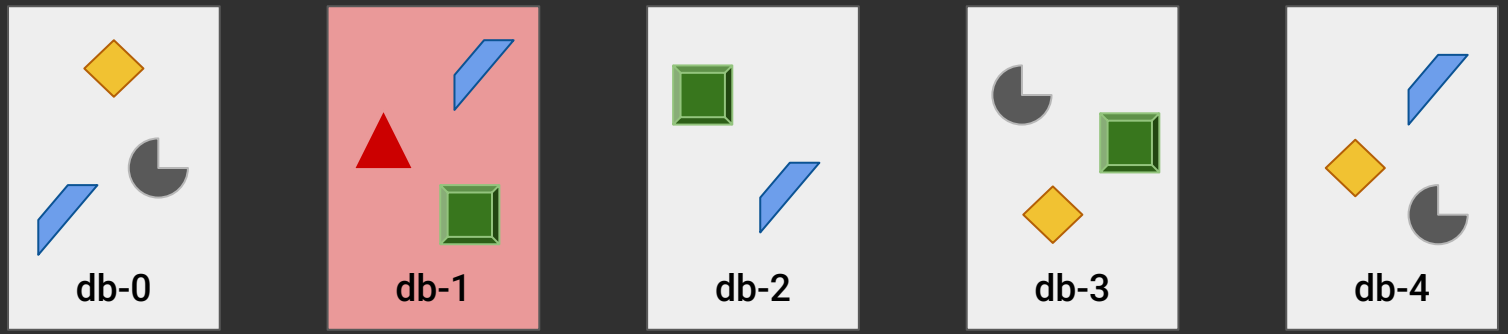
- hardware failures happen
- no live migrations of VMs/disks
- mission-critical stateful systems using local disks

We need observability, and control over who kills our apps (and push back on evictions).

Eviction Controls in Kubernetes

Very few knobs (mostly on/off).
Limited extensibility but more coming soon!

Pod Disruption Budgets (PDB): Limited expressivity, application-agnostic



API Evictions

1. Pod Delete API

`kubectl delete pod`

- Triggers graceful pod deletion
  - but you can override it

- rolling update/scaledown RS/Deployment/StatefulSet etc.
  - doesn’t do availability checks (PDBs)

2. Pod Eviction API

Nicer `kubectl delete pod`: respects PodDisruptionBudgets (PDB).

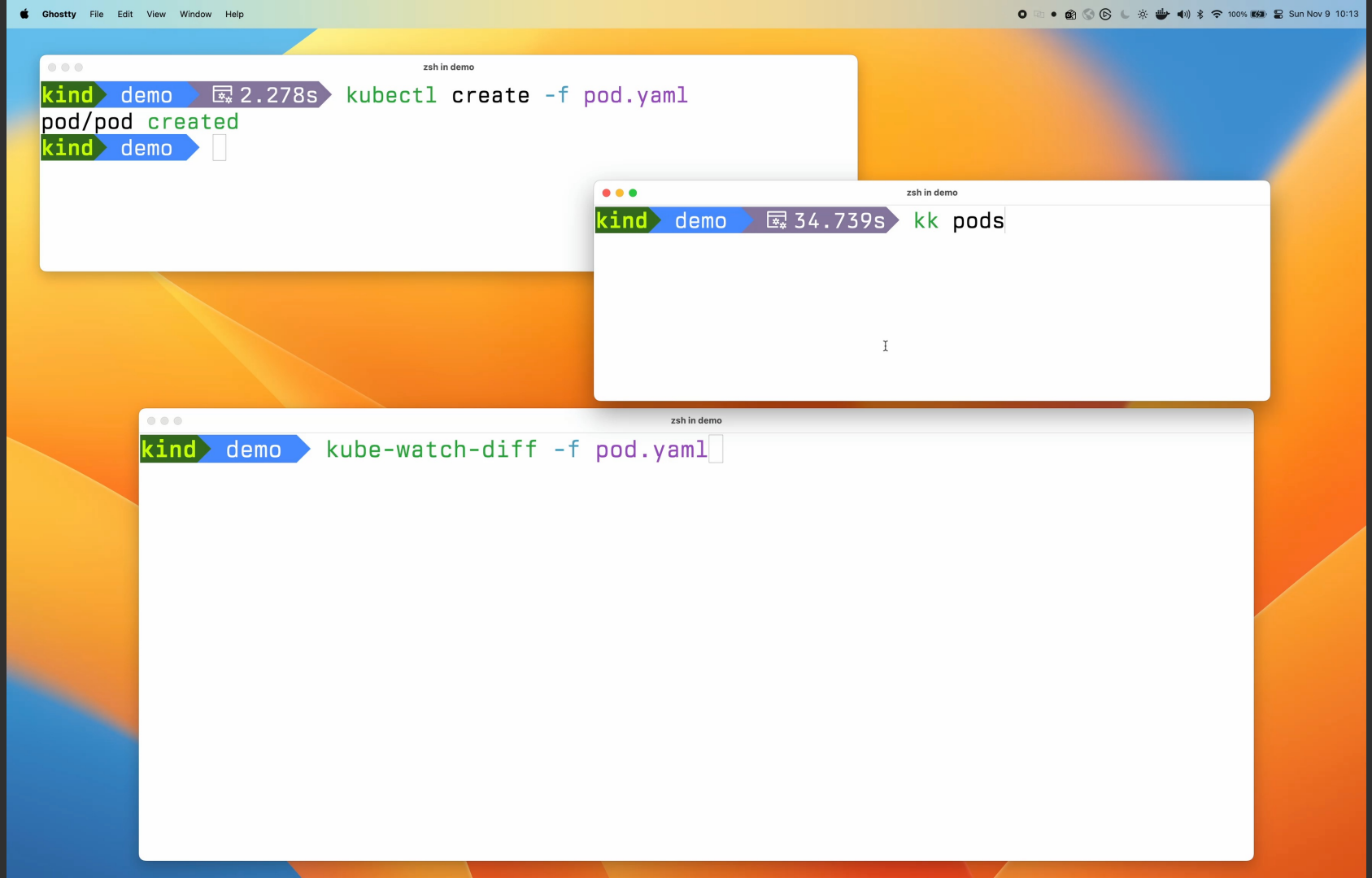
- `kubectl drain`
- *Your cloud provider probably*
- Cluster API

Catch: Most eviction paths in Kubernetes don’t use this. 😞

2. Pod Eviction API mechanics

```
curl -XPOST /api/v1/namespaces/<ns>/pods/<pod>/eviction \
  -H "Content-Type: application/json" \
  -d '{
    "apiVersion": "policy/v1",
    "kind": "Eviction",
    "metadata": {
      "name": "<pod>",
      "namespace": "<ns>"
    }
  }'
```




Ghostty File Edit View Window Help
Sun Nov 9 10:13
zsh in demo
kind demo 2.278s kubectl create -f pod.yaml
pod/pod created
kind demo
zsh in demo
kind demo 34.739s kk pods
zsh in demo
kind demo kube-watch-diff -f pod.yaml

kubelet initiated evictions

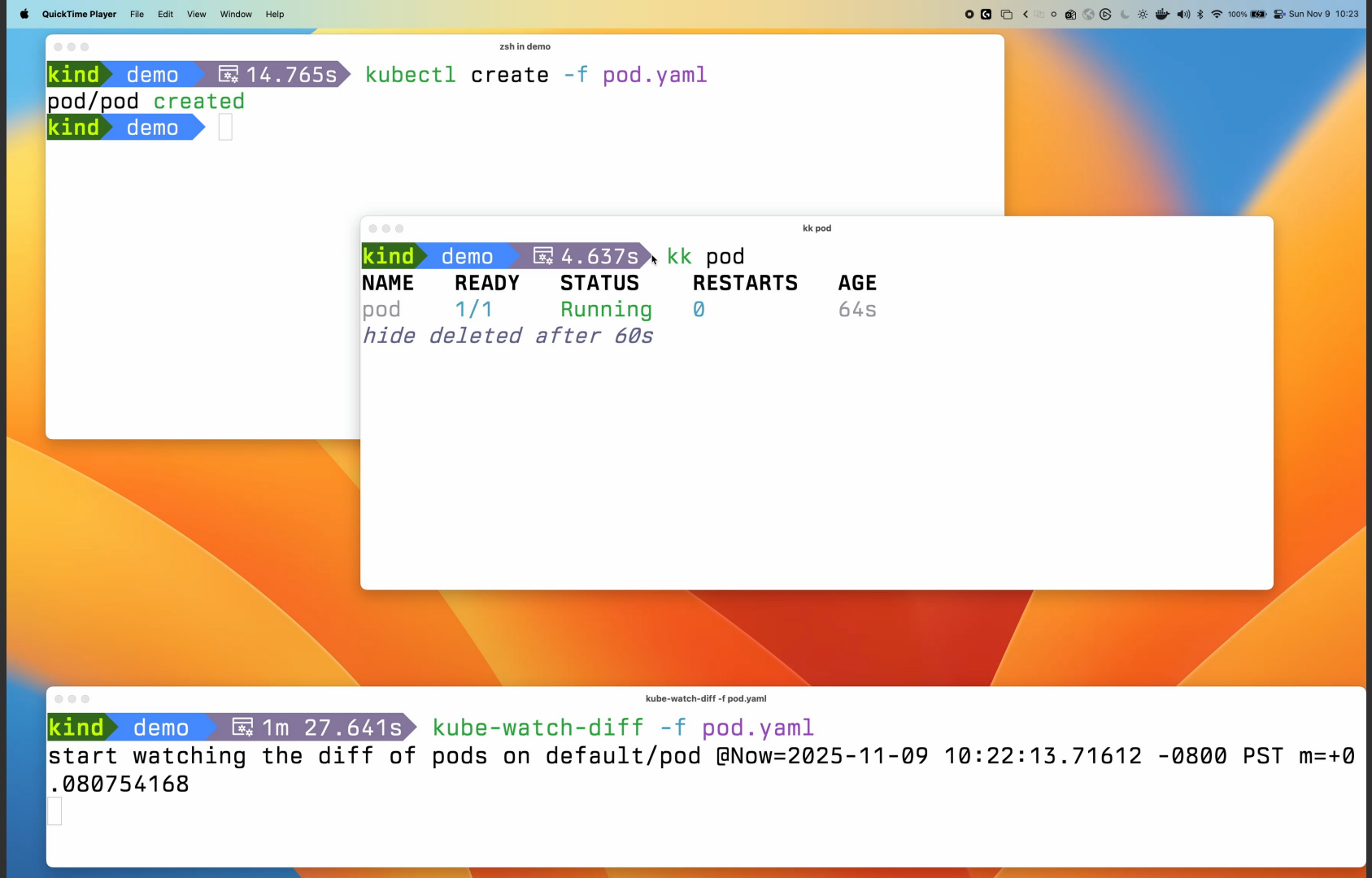
3. Node-pressure Evictions

Remove lower priority pods if the node is under pressure (disk/memory/inodes/PIDs…).

Catch: **Doesn’t respect PDBs.**

Catch: **Hard thresholds directly kill pods** (non-graceful termination)

This feature has many knobs! (We disable this because we have our own node health monitoring and remediation systems.)


QuickTime Player File Edit View Window Help
Sun Nov 9 10:23
zsh in demo
kind demo 14.765s kubectl create -f pod.yaml
pod/pod created
kind demo
kk pod
kind demo 4.637s kk pod
NAME READY STATUS RESTARTS AGE
pod 1/1 Running 0 64s
hide deleted after 60s
kube-watch-diff -f pod.yaml
kind demo 1m 27.641s kube-watch-diff -f pod.yaml
start watching the diff of pods on default/pod @Now=2025-11-09 10:22:13.71612 -0800 PST m=+0
.080754168

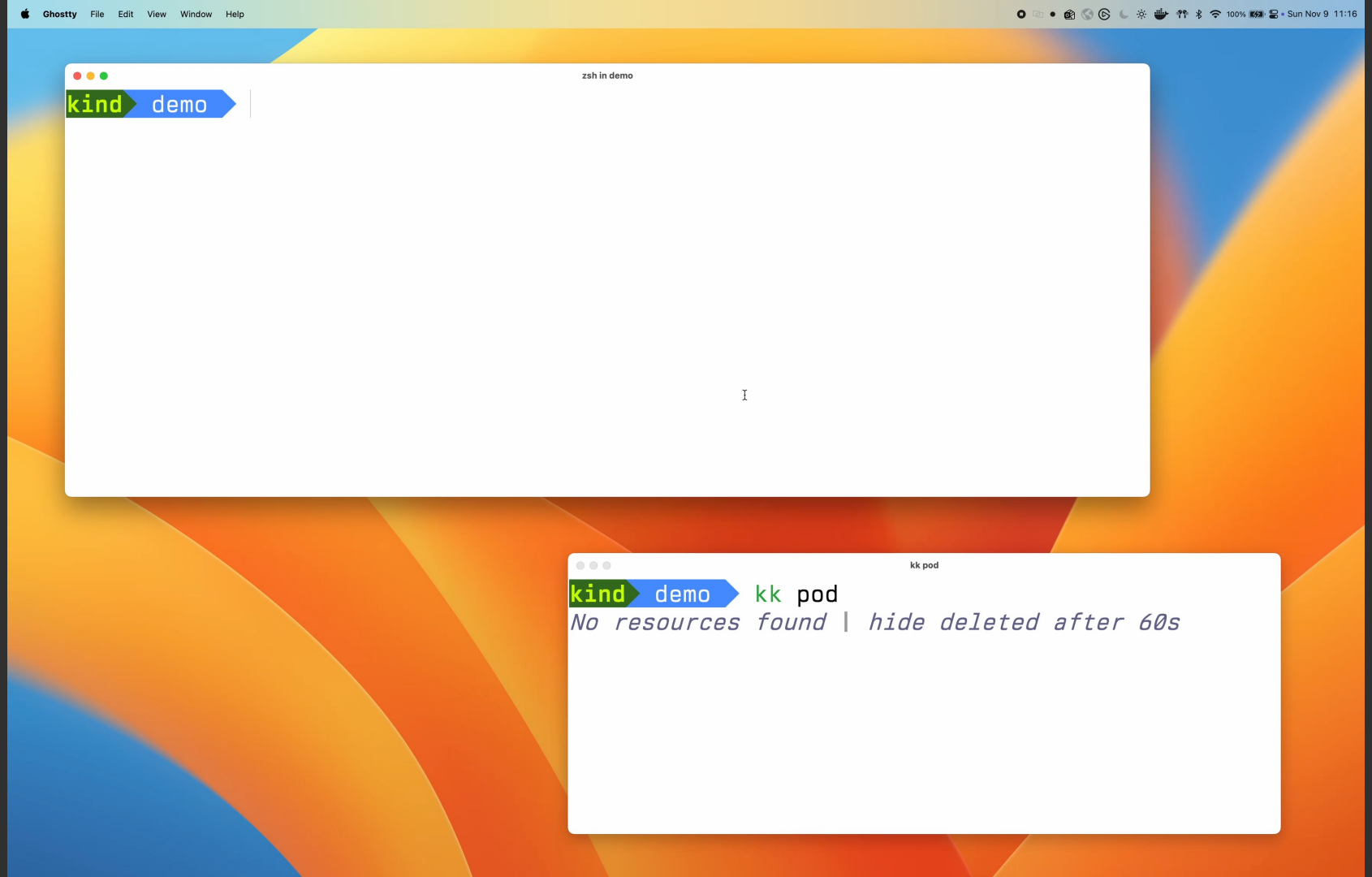
4. Kubelet Admission

Kubelet has admission checks (NodeAffinity, NodeResources…) and can directly kill a Pod assigned to the node by the scheduler.

Can happen if:

- node labels change dynamically (e.g. node-feature-discovery)
- using multiple schedulers concurrently (bad idea)

Restarting kubelets in-place is not safe. Drain before upgrading.

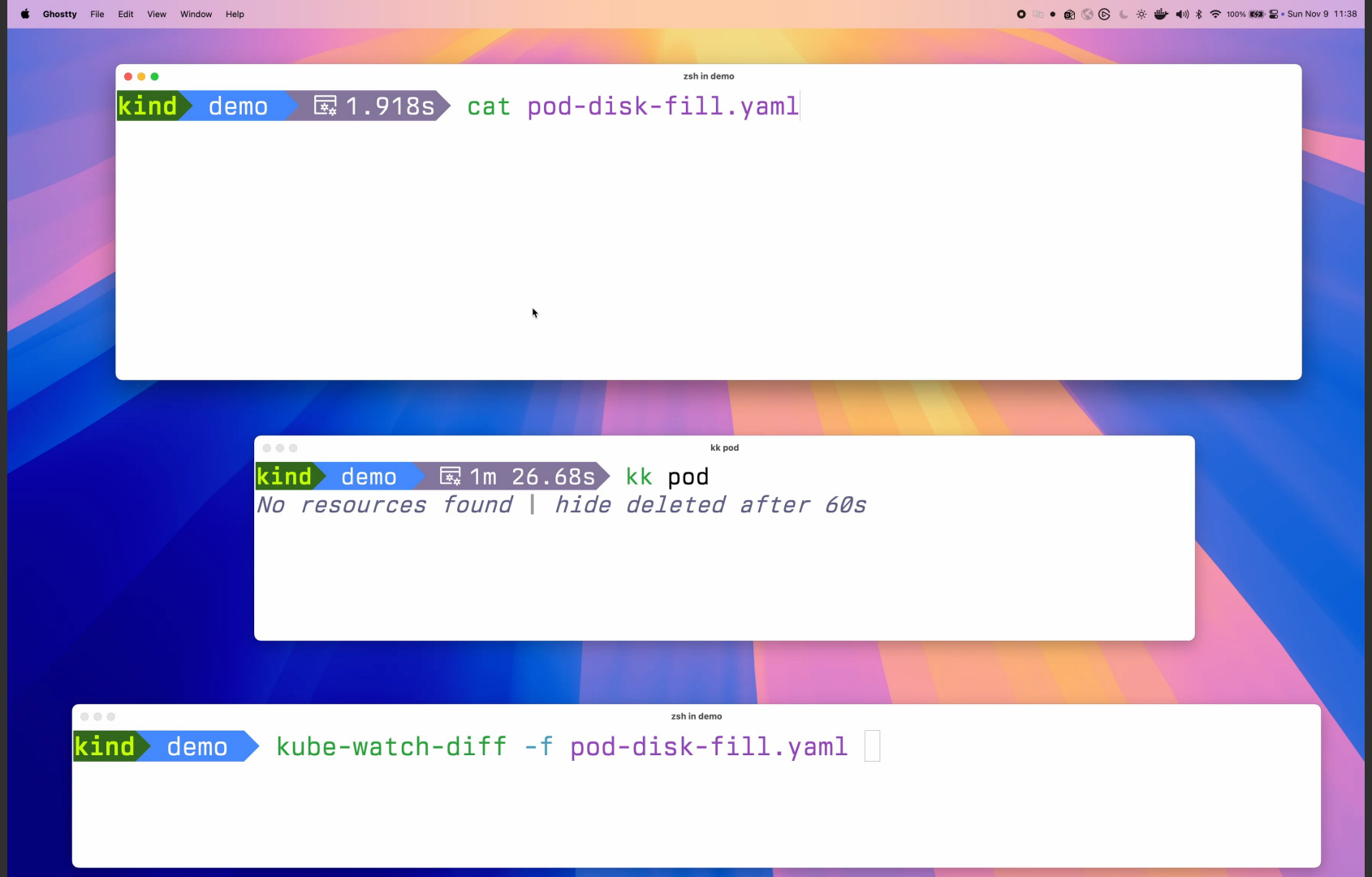

Ghostty File Edit View Window Help
Sun Nov 9 11:16
zsh in demo
kind demo
kk pod
kind demo kk pod
No resources found | hide deleted after 60s

5. Kubelet local storage evictions

You can now set:

- emptyDir volume size limit
- pod/container ephemeral storage limit

kubelet *gracefully* terminates the pod.


Ghostty File Edit View Window Help
Sun Nov 9 11:38
zsh in demo
kind demo 1.918s cat pod-disk-fill.yaml
kk pod
kind demo 1m 26.68s kk pod
No resources found | hide deleted after 60s
zsh in demo
kind demo kube-watch-diff -f pod-disk-fill.yaml

kube-scheduler Evictions

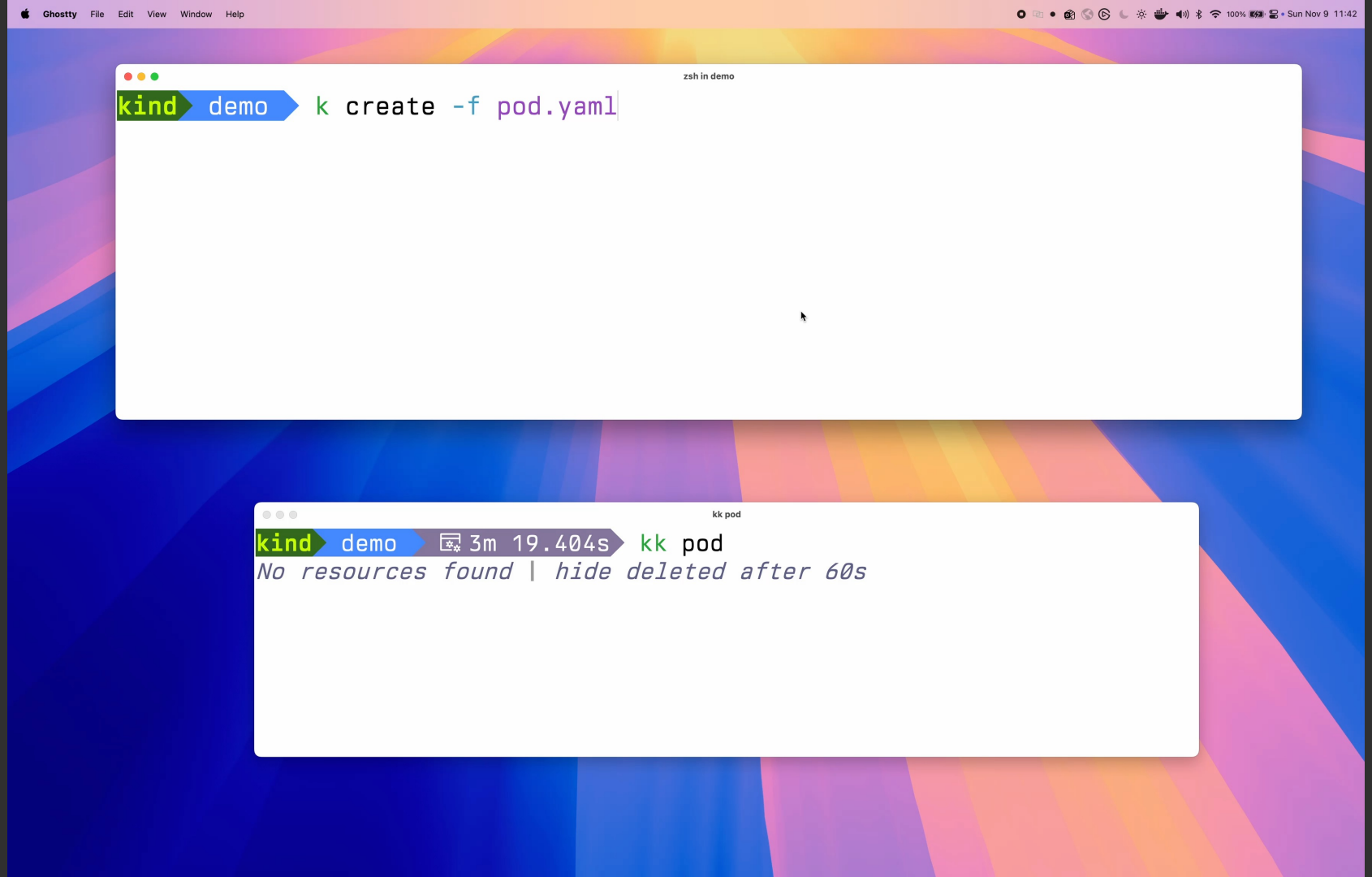
6. Pod Preemption

Fairly well documented. High-priority pod bumps out low-pri pod.

Honoring PDBs is best effort:

- Any nodes where evicting a lower-priority pod doesn't violate PDBs? no:
- Choose a node with pods with least PDB violations.
- Choose a lower-priority Pod with fewest PDB violations
- Evict the pod despite PDB violation.

Graceful termination.


Ghostty File Edit View Window Help
Sun Nov 9 11:42
zsh in demo
kind demo k create -f pod.yaml
kk pod
kind demo 3m 19.404s kk pod
No resources found | hide deleted after 60s

kube-controller-manager Evictions

7. Taint-based eviction

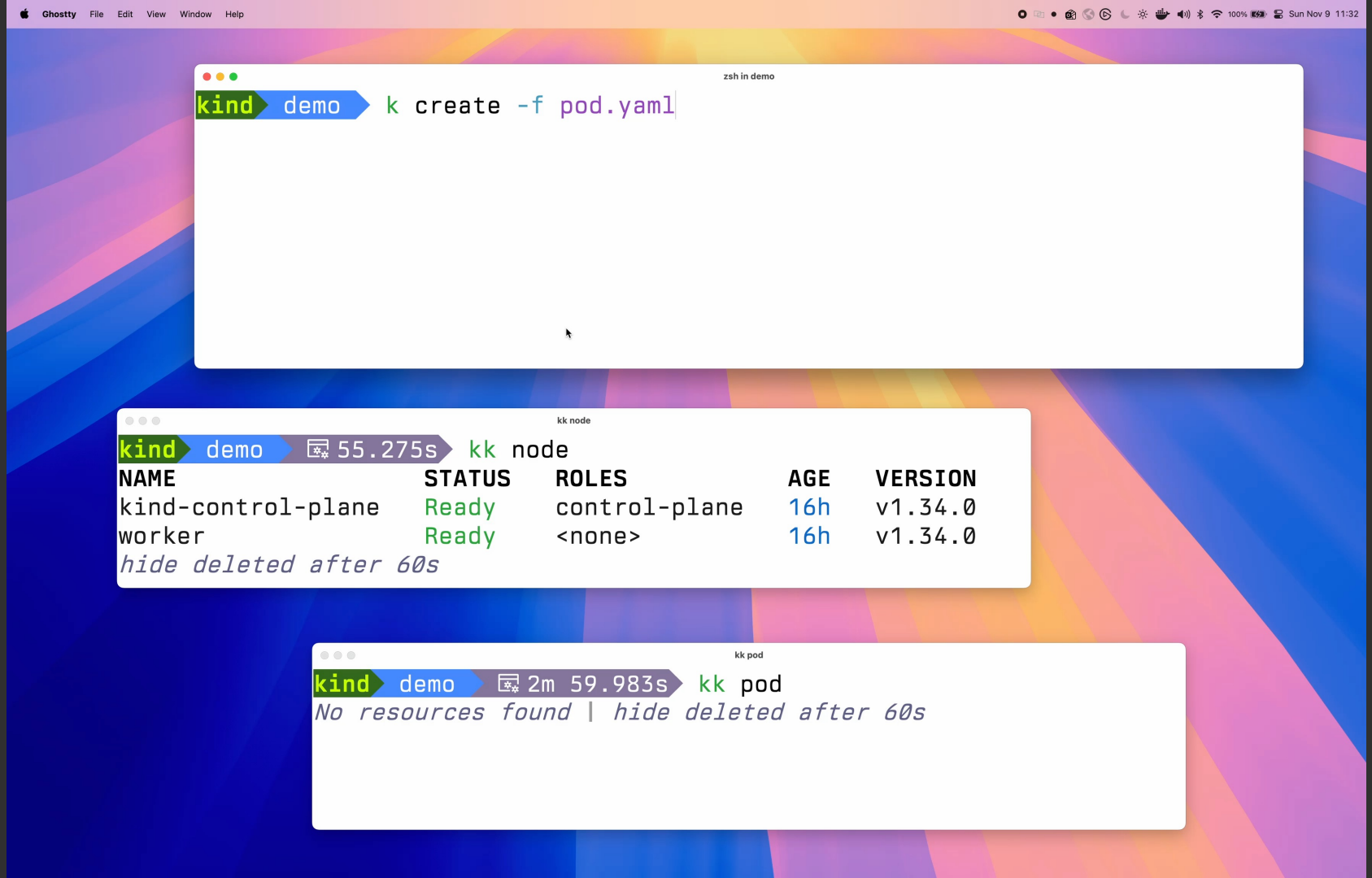
Nodes can get `NoExecute` taints for many reasons

- *unreachable* taint → lack of heartbeats
- *not-ready* taint → kubelet detected node faults (CRI/CNI…)

After the “toleration period”, the pod is gracefully terminated.

Risk: False positives/bugs risk your service availability

- degraded node state might be ok for some workloads
- you may roll your own taints for execution


Ghostty File Edit View Window Help
Sun Nov 9 11:32
zsh in demo
kind demo k create -f pod.yaml
kk node
kind demo 55.275s kk node
NAME STATUS ROLES AGE VERSION
kind-control-plane Ready control-plane 16h v1.34.0
worker Ready <none> 16h v1.34.0
hide deleted after 60s
kk pod
kind demo 2m 59.983s kk pod
No resources found | hide deleted after 60s

7. Taint-based eviction

What can you do about it?

KEP 3902: Separates taintmanager controller (which adds the unreachable/not-ready taints) from taint-based eviction controller.

(beta in 1.29+, GA 1.34+, thanks Apple!)

8. Pod GC controller

Ever wondered what happens if you `kubectl delete node --all` ?

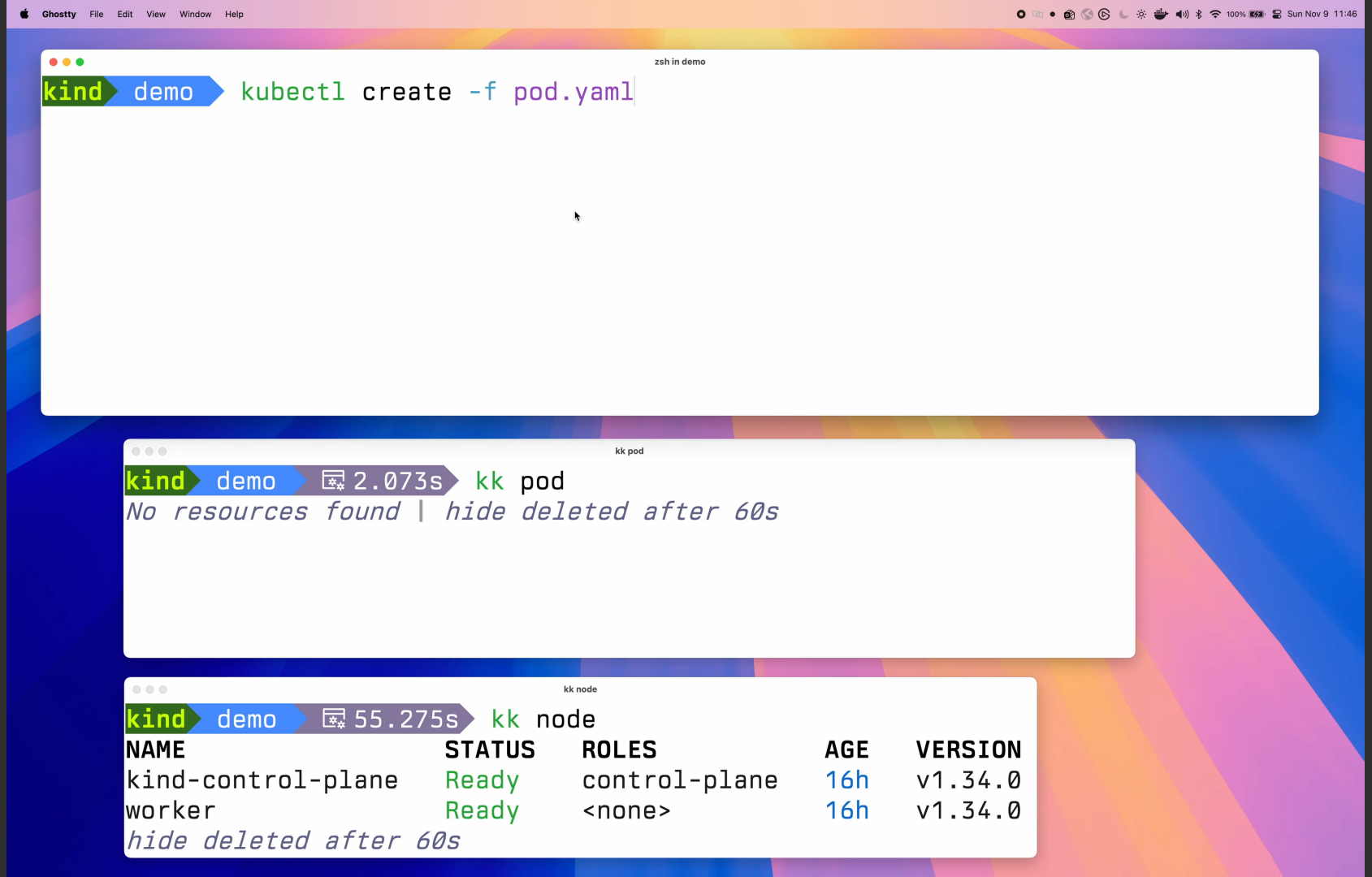
8. Pod GC controller

Ever wondered what happens if you `kubectl delete node --all`?

Mass descheduling event.

PodGC controller forcibly deletes any orphan Pod that has no corresponding Node object in ~1 min.

If you manage node lifecycle yourself: Implement a lot of guardrails for automated paths that lead to Node deletion.


Ghostty File Edit View Window Help
Sun Nov 9 11:46
zsh in demo
kind demo kubectl create -f pod.yaml
kk pod
kind demo 2.073s kk pod
No resources found | hide deleted after 60s
kk node
kind demo 55.275s kk node
NAME STATUS ROLES AGE VERSION
kind-control-plane Ready control-plane 16h v1.34.0
worker Ready <none> 16h v1.34.0
hide deleted after 60s

what can you do about these?

Eviction interception

You can write webhooks for `CREATE pods/eviction` or `DELETE pods` requests

Use cases:

- PDBs are insufficient
- Custom eviction policy/guardrail
- Use Eviction request as a ‘signal’ to prep

(Caveat: make sure you don’t intercept all eviction requests, objectSelectors don’t work out of the box.)

Coming Soon: EvictionRequest API

https://kep.k8s.io/4563 (alpha 1.35) WG Node Lifecycle

You can explicitly register eviction interceptors to pods:

```
Pod.spec.evictionInterceptors = [a,b,c]
```

Interceptors must either evict the pod, or pass on to the next interceptor.

Observe

Pod Disruption Conditions (GA 1.31+)

```
reason = PreemptionByScheduler, DeletionByTaintManager,
EvictionByEvictionAPI, DeletionByPodGC, TerminationByKubelet…
```

(Caveat: It’s not consistently used in all eviction paths)

Collect API Audit Logs & Event Logs

Understand

Eviction Mode	Initiated by	Uses PDBs?	Graceful?
Pod delete API	kubectl delete Workload Controllers	❌	✅*
Eviction API	kubectl drain Cloud providers, CAPI	✅	✅
Node pressure	kubelet	❌	❌ hard ✅ soft
Local storage	kubelet	❌	✅
Kubelet admission	kubelet	❌	❌
Pod Preemption	kube-scheduler	✅*	✅
NoExecute Taint	controller-manager	❌	✅
Node deletion	controller-manager	❌	❌

Act

Look into your kubelet eviction thresholds settings.

Disaster recovery drills (e.g. take down your control plane)

Evaluate tolerations for your stateful apps.

Consider admission controls for evictions if PDBs aren’t enough.

Understand what happens when a Pod fails (who cleans it up)

thanks!

Find me on bsky: @ahmet.dev
x/github: @ahmetb



← We are hiring!