



Do You Trust Your
PodDisruptionBudgets?
You Shouldn't!

About Me

Kārlis Akots Gribulis

- Joined Saxo Bank in 2024 as Senior Container Platform Engineer
- Previously build Kubernetes clusters for VELUX, Accenture
- Golden Kubestronaut



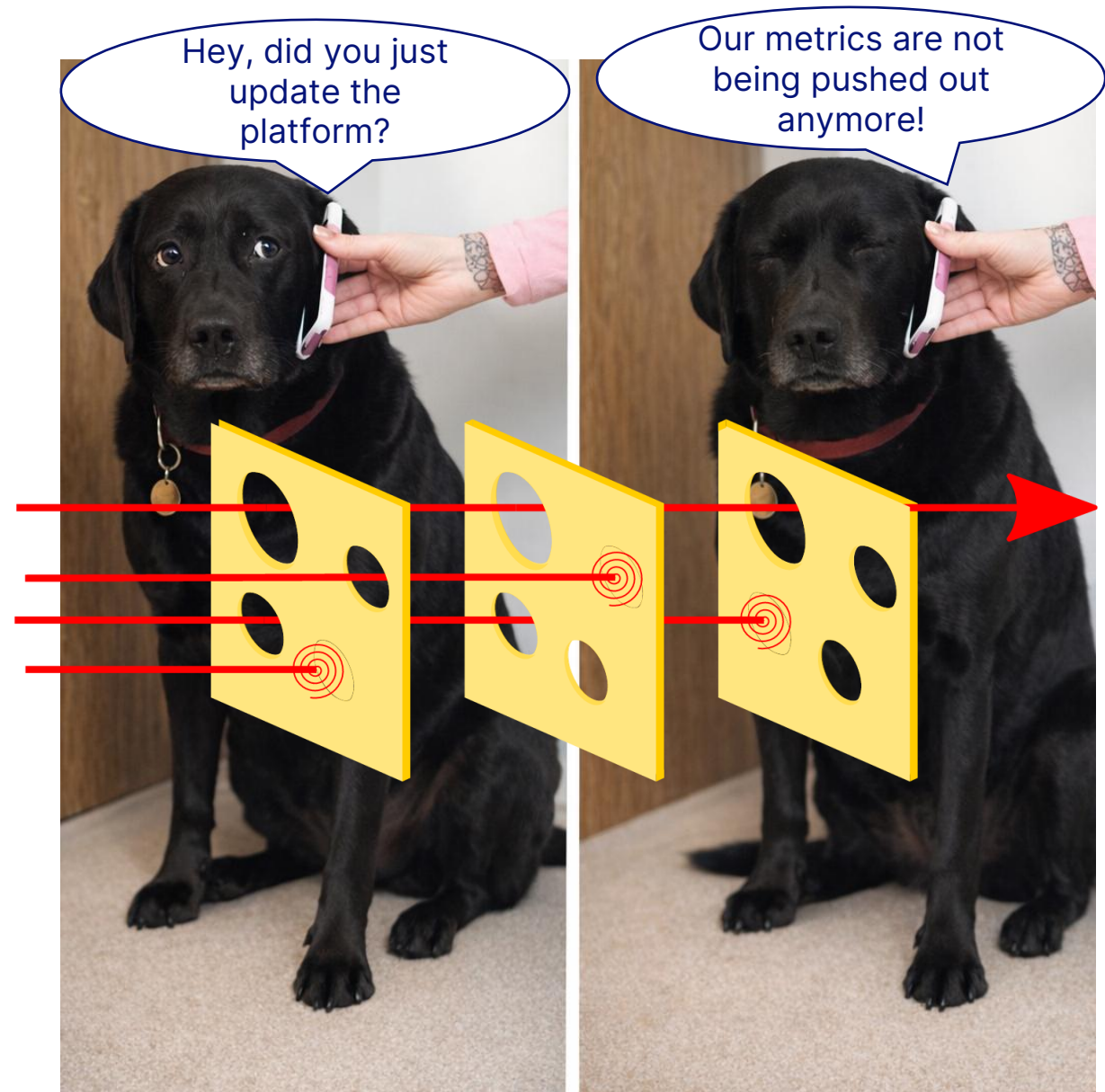
About Saxo Bank

- Founded 1992 in Copenhagen, Denmark
- +100 USD billion in client assets
- +1,5 mio clients
- +71.000 financial instruments
- Designated Systemically Important Financial Institution (SIFI)



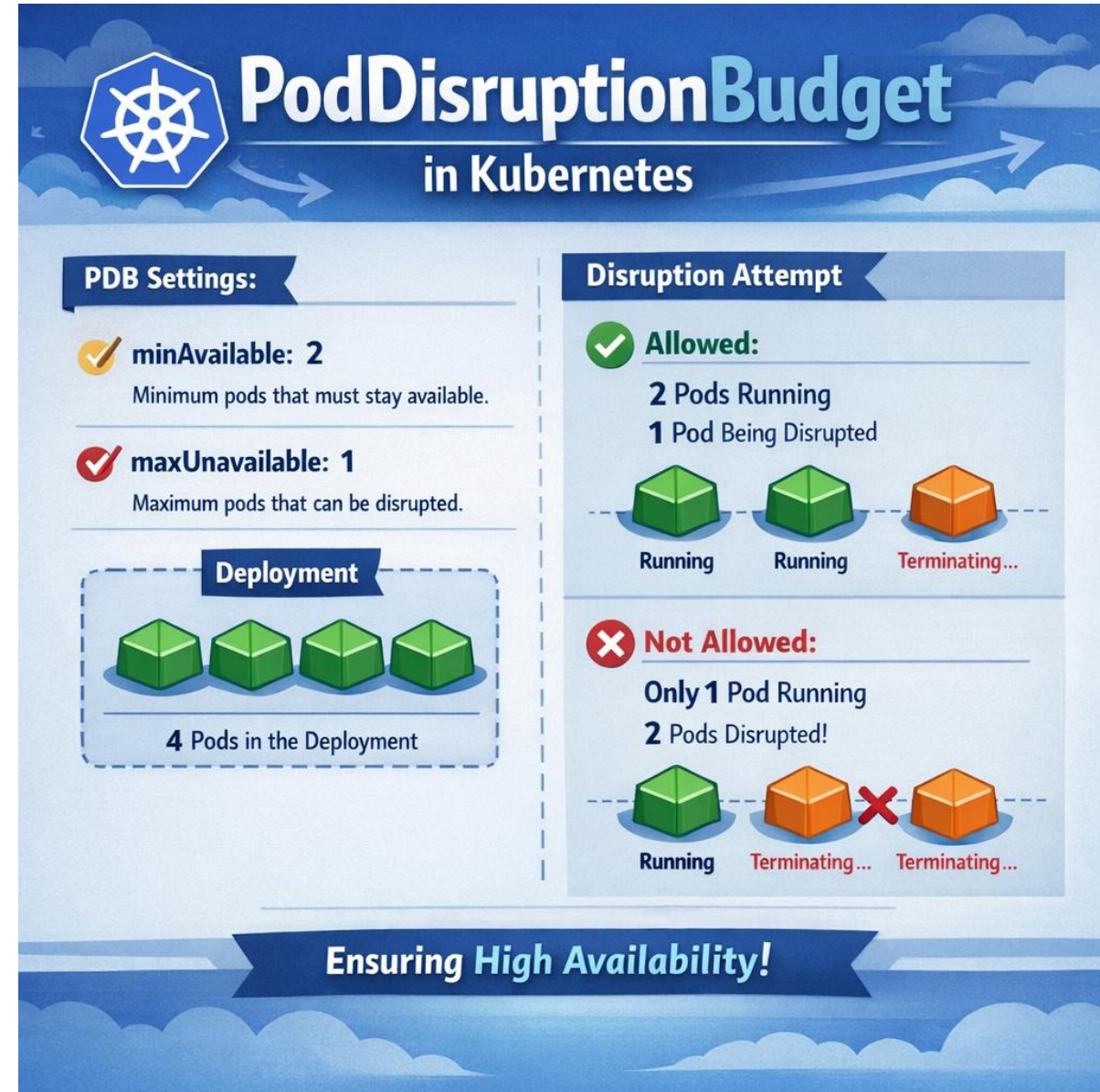
Assumed Total PodDisruptionBudget (PDB) Safety

- Regular platform maintenance evening
- PDB felt like a Swiss cheese



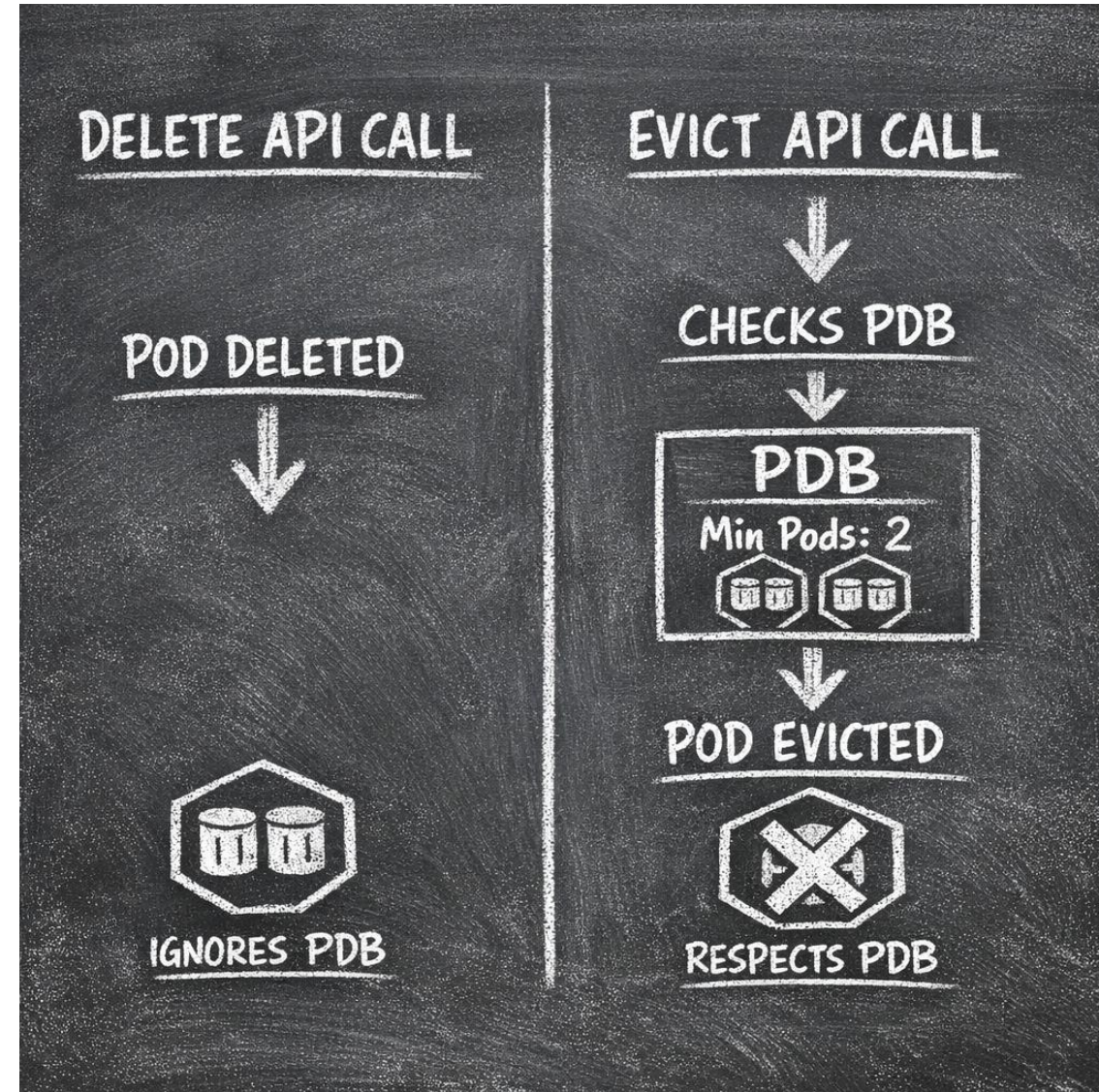
What is PDB?

- Feature that helps run Highly Available (HA) applications in disruptive environments
- Limits the number of Pods of a replicated application that are down simultaneously from voluntary disruptions



When Does It Apply?

- When calls are made against Eviction API (POST /eviction)
- API DELETE bypasses PDB



When Is PDB *Not* Respected?

Involuntary disruption

- A hardware failure of the physical machine backing the node
- Cloud provider or hypervisor failure makes VM disappear
- Cluster administrator deletes VM (instance) by mistake
- A kernel panic

```
user@cluster:~$ kubectl delete node worker-node1
node "worker-node1" deleted
```

```
user@cluster:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE
master	Ready	control-plane	30d
worker-node1	NotReady	<none>	42d

```
user@cluster:~$ kubectl config current-context
```

```
production
```

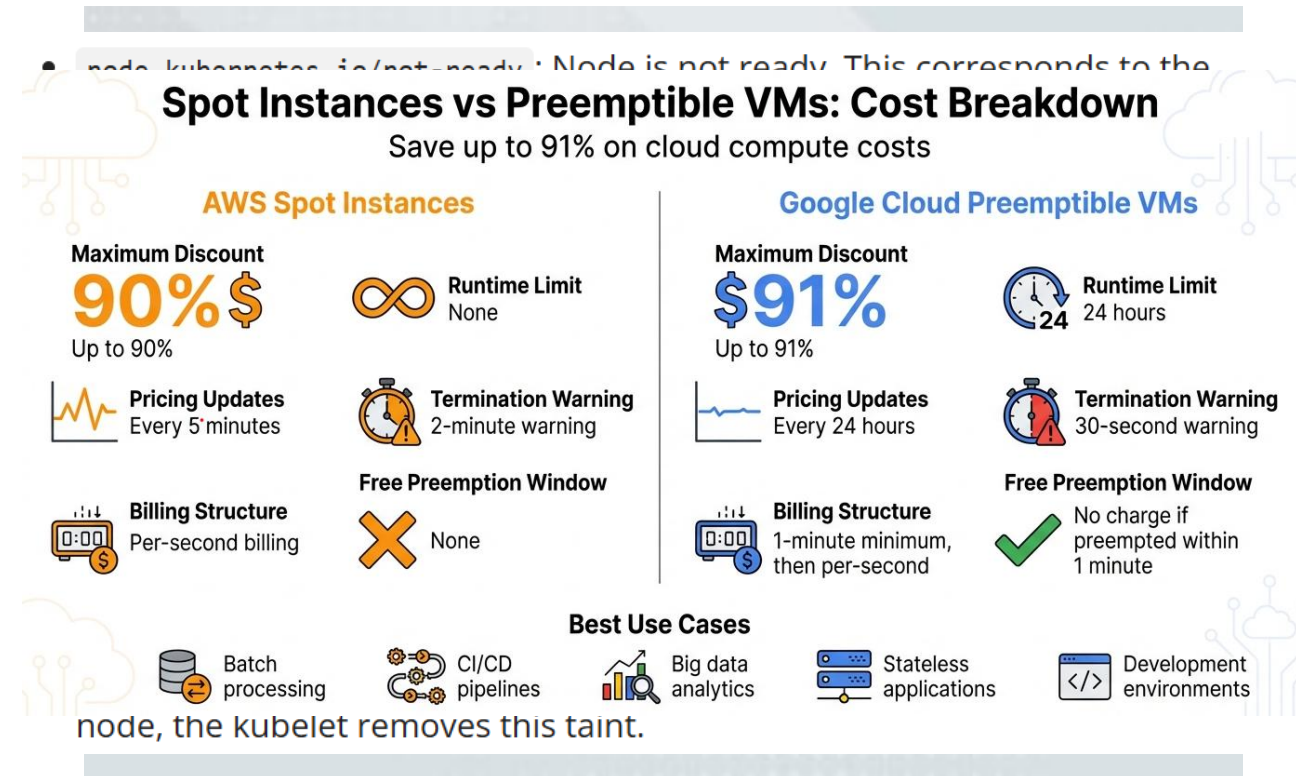
```
user@cluster:~$ ...oh no...
```

When Is PDB *Not* Respected?

Involuntary disruption

- The node disappears from the cluster due to cluster network partition
- Node being out-of-resources (DiskPressure, MemoryPressure, PIDPressure)
- Spot nodes

Pods removed involuntary still count against the budget!

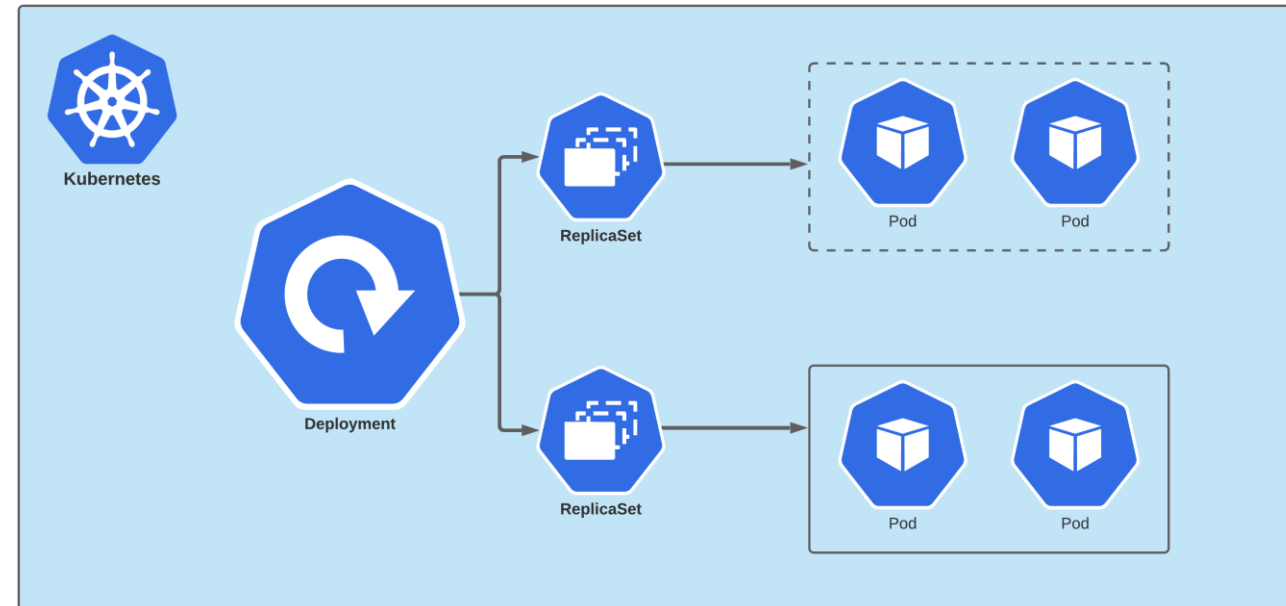


When Is PDB Respected?

Voluntary disruption

From application owner's perspective:

- Updating a deployment's pod template causing a restart
- Deleting the deployment or other controller that manages the pod
- Directly deleting a pod

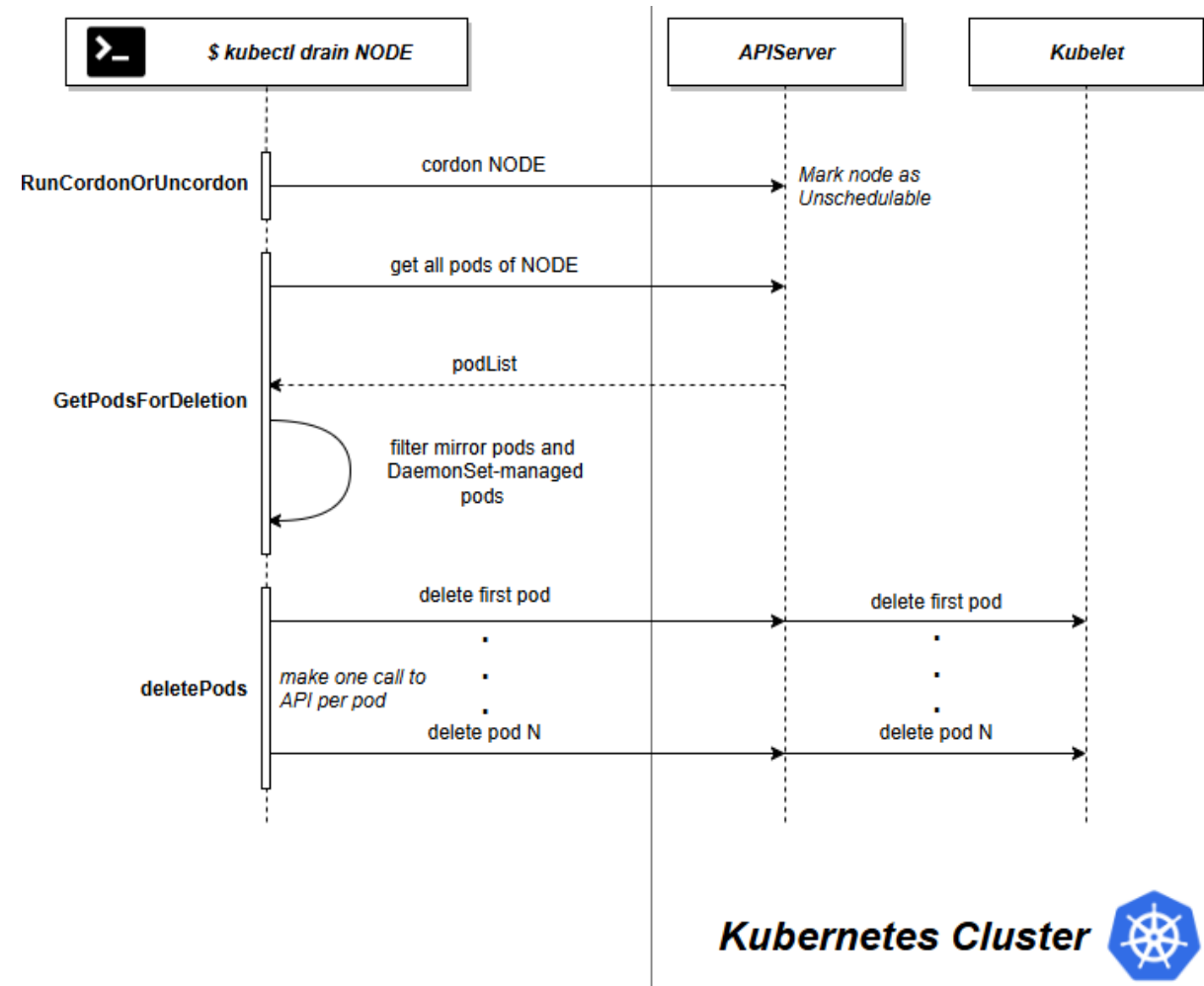


When Is PDB Respected?

Voluntary disruption

From cluster administrator's perspective:

- Draining a node for repair or upgrade
- Draining a node from a cluster to scale the cluster down
- Removing a pod from a node to permit something else to fit on that node



How PDB's Can Hurt Maintenance

- PDB with:
 - maxUnavailable: 0
 - minAvailable: 100%
- This blocks *all* voluntary evictions deadlock cluster operations (drain hangs forever)
- Be careful of bad developers or controllers managing PDB's

```
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: test-app
  namespace: test-ns
spec:
  minAvailable: 100%
  selector:
    matchLabels:
      app.kubernetes.io/component: something
      app.kubernetes.io/name: test-app
---
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: test-app
  namespace: test-ns
spec:
  maxUnavailable: 0
  selector:
    matchLabels:
      app.kubernetes.io/component: something
      app.kubernetes.io/name: test-app
```

Can You Ignore It During Maintenance?

Node Drain

- `--disable-eviction`
 - Bypasses the Eviction API entirely
- `--force`
 - Forces deletion of unmanaged Pods (not owned by a controller)
- `--grace-period`

```
--force=false:  
Continue even if there are pods that do not declare a controller.
```

Pod Priority and Preemption

- Priority indicates the importance of a Pod relative to other Pods
- If a Pod cannot be scheduled, the scheduler tries to preempt (evict) lower priority Pods
- Pods with no specified priority have priority value 0
- Avoid preemption with “preemptionPolicy: Never”

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority-nonpreempting
  value: 1000000
  preemptionPolicy: Never
  globalDefault: false
  description: "This priority class will not cause other pods to be preempted."
```

Demo

```
$ ./demo.sh
$ # Setup
$ kubectl get pdb -n pdb-demo
```

NAME	MIN	AVAILABLE	MAX	UNAVAILABLE	ALLOWED	DISRUPTIONS	AGE
high-prio-has-pdb	25		N/A		0		42m
high-prio-has-pdb-besteffort	25		N/A		0		42m
high-prio-has-pdb-burstable	25		N/A		0		42m
high-prio-has-pdb-guaranteed	25		N/A		0		42m
low-prio-has-pdb	30		N/A		0		42m

Dealing With Disruptions

- Ensure your pod **requests** the resources it needs
- Use Pod replication
 - topologySpreadConstraints
 - Pod anti-affinity
 - Node affinity
 - Update strategy
- PriorityClasses to avoid preemption
- Quality of Service to avoid disruption during node resource pressure
- Use PodDisruptionBudget's

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

- matchExpressions:

- key: topology.kubernetes.io/zone

operator: In

values:

- antarctica-east1

- antarctica-west1

preferredDuringSchedulingIgnoredDuringExecution:

- weight: 1

preference:

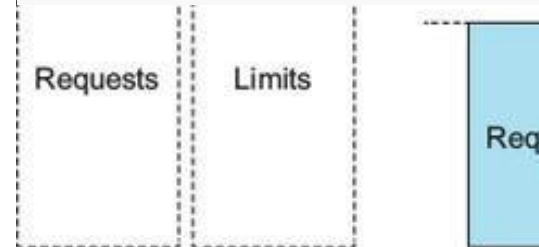
matchExpressions:

- key: another-node-label-key

operator: In

values:

- another-node-label-value



BestEffort

topologySpreadConstraints:

- maxSkew: 1

topologyKey: kubernetes.io/hostname

whenUnsatisfiable: DoNotSchedule

labelSelector:

matchLabels:

app: foo

matchLabelKeys:

- pod-template-hash

strategy:

type: RollingUpdate

rollingUpdate:

maxUnavailable: 1

icy/v1

Requests
equal limits

podAntiAffinity:

preferredDuringSchedulingIgnoredDuringExecution:

- weight: 100

podAffinityTerm:

labelSelector:

matchExpressions:

- key: security

operator: In

values:

- S2

topologyKey: topology.kubernetes.io/zone

keeper

Should You *Blindly* Trust Your
PodDisruptionBudgets?

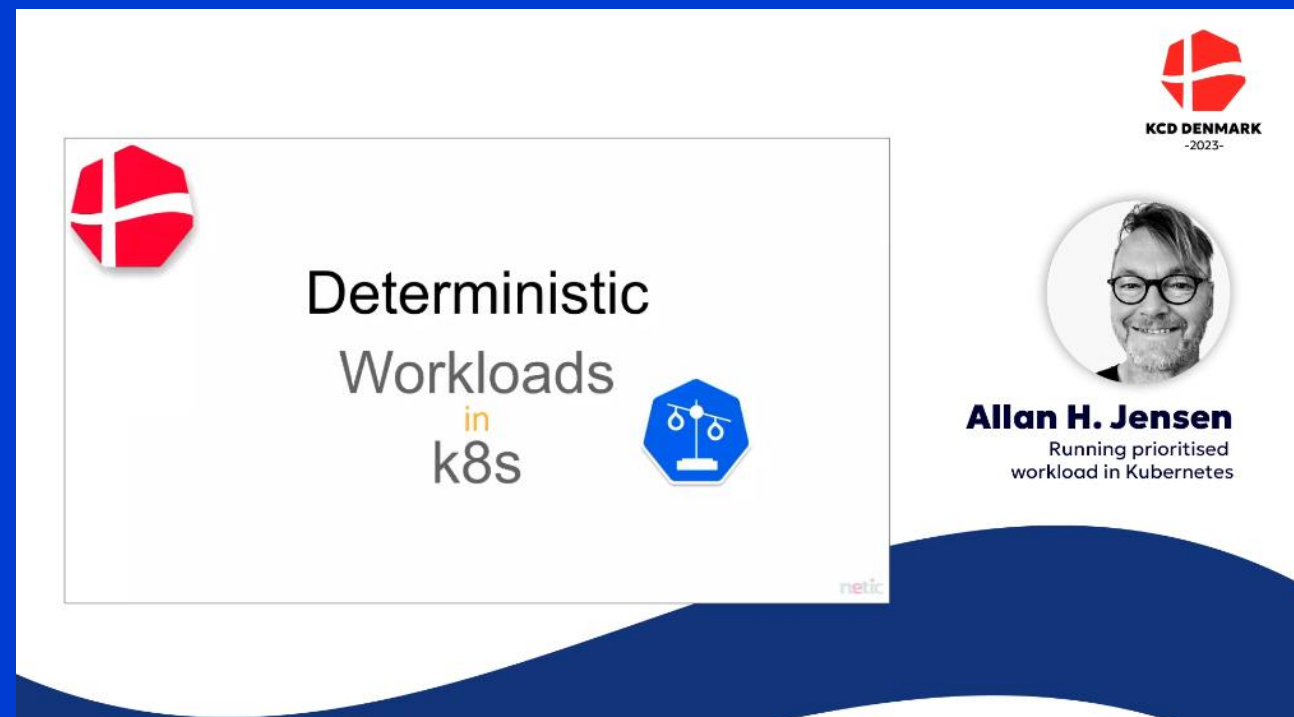
Should You ~~Blindly~~ Trust Your
PodDisruptionBudgets?

Want To Learn More?

KCD Denmark 2023

Allan Højgaard Jensen - Running prioritised workload in Kubernetes

https://www.youtube.com/watch?v=MCXSRQoo_RE&list=PLO9s8ZaIKQe85ilzlixAOQEH_oFMoMKtM&index=2



Thank you & QA

Rate the session

