Avoid overspending on Kubernetes

Webb Brown
Niko Kovacevic

About us

Niko is a Founding Engineer on the Kubecost team and a maintainer of the Kubecost open source project.

Webb is CEO of Kubecost. He is a former Google PM where he led teams building monitoring & performance tools.

Kubernetes is a powerful tool that presents great opportunities to reduce costs... but it *can* make it easier to overspend on cloud too. Why?

1. Enables more decentralized deployments
2. Promotes faster release cycles
3. Powerful abstractions for programmatically provisioning resources

Reducing spend will always touch one of these inputs...

$$\underbrace{\overset{1}{\{\text{ time provisioned }\}} \quad * \quad \overset{2}{\{\text{ quantity of resource }\}}}_{\{\text{ resource efficiency }\}} \quad * \quad \overset{3}{\{\text{ price of resource}\}}$$

The price of anything
is the amount of life
you exchange for it.

Henry David Thoreau

5 anti-patterns for overspending

Orphaned resources

Externally provisioned cloud resources without an active owner.

Orphaned resources

Common examples:

- Persistent volumes
- Elastic IPs
- Load Balancers
- Databases

Oftentimes caused during application teardown or by oversight.

Impact: 10% of spend

Difficulty: **Easy**

Orphaned resource solutions

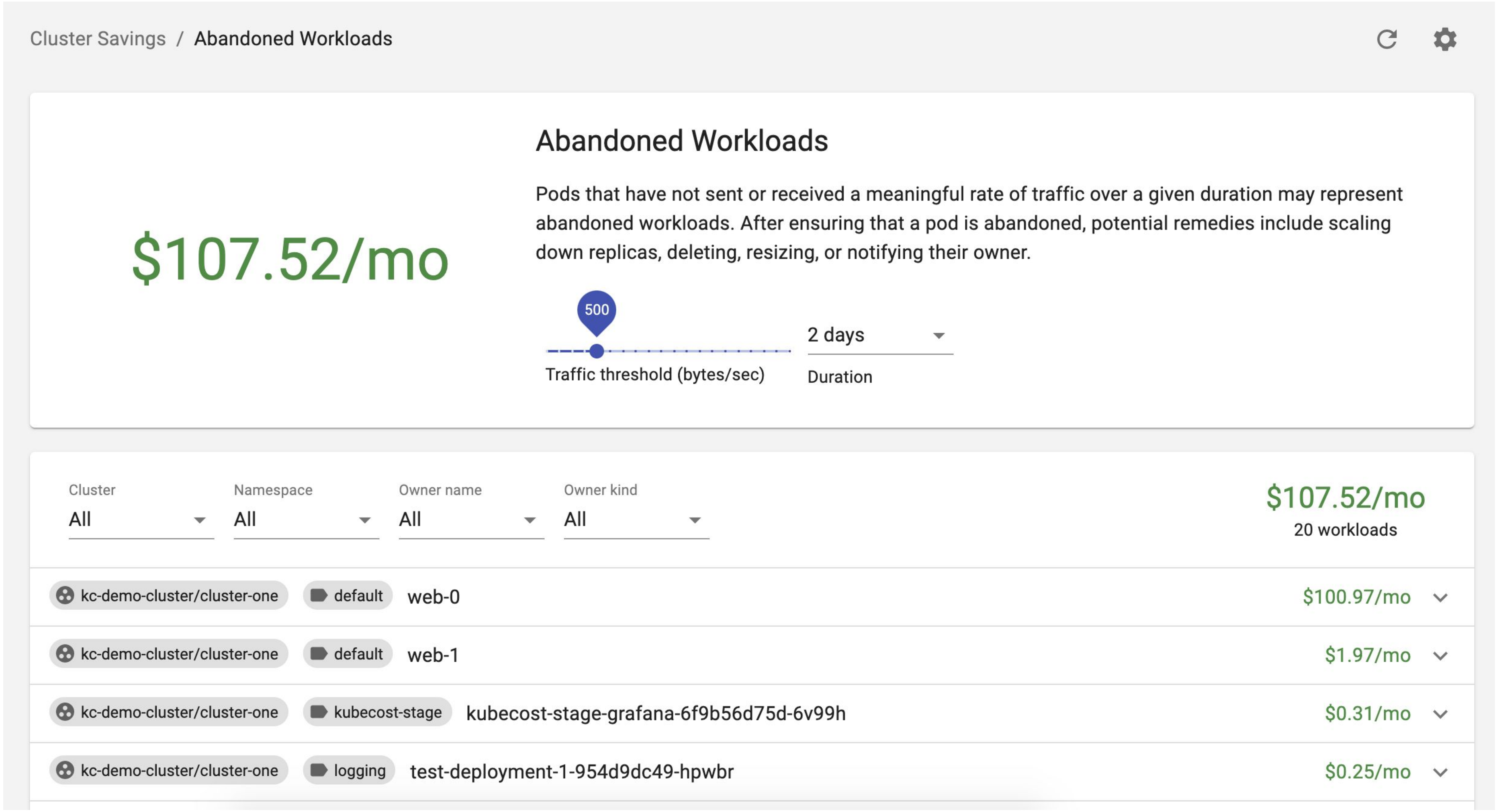
- Some mechanism to detect when orphaned resources cross a certain dollar threshold
  - Implementation can be alerting rules, dashboards, or mgmt platform

- Part two: have a mechanism to identify an owner, e.g. labels or external system of record

#4

Abandoned workloads

Cluster workloads that do not provide real business value.


Cluster Savings / Abandoned Workloads

$107.52/mo

Abandoned Workloads

Pods that have not sent or received a meaningful rate of traffic over a given duration may represent abandoned workloads. After ensuring that a pod is abandoned, potential remedies include scaling down replicas, deleting, resizing, or notifying their owner.

Traffic threshold (bytes/sec): 500

Duration: 2 days

Cluster: All
Namespace: All
Owner name: All
Owner kind: All

$107.52/mo
20 workloads

Cluster	Namespace	Workload	Cost
kc-demo-cluster/cluster-one	default	web-0	$100.97/mo
kc-demo-cluster/cluster-one	default	web-1	$1.97/mo
kc-demo-cluster/cluster-one	kubecost-stage	kubecost-stage-grafana-6f9b56d75d-6v99h	$0.31/mo
kc-demo-cluster/cluster-one	logging	test-deployment-1-954d9dc49-hpwbr	$0.25/mo

Abandoned resources

Common examples are:

- Deprecated deployments
- Outdated deployment configurations
- Dev environments on nights and weekends

Oftentimes caused by lack of awareness or organizational changes.

Impact: 10-20% of spend

Difficulty: **Medium**

Abandoned resources solutions

A mechanism to detect when abandoned resources cross a certain threshold. Signals for abandonment can include a combination of:

- Little/no pod network traffic
- Low cpu usage
- No recent upgrades

Solution implementation can be alerting rule, dashboard, management platform, or automation to address abandonment.

Part two: have a scalable mechanism to identify an owner

#3

Rogue deployments

Kubernetes workloads gone crazy.

Rogue deployments

Common causes:

- Mis-configured deployments (e.g. 100s of replicas instead of 10s)
- Application bugs combined with improperly configured autoscaling
- Malware (e.g. cryptomining malware)

Impact: 20% of spend

Difficulty: **Medium**

Rogue deployment solutions

- Alerts to detect unexpected or sharp increases in spend, i.e. spend anomalies
- Guardrails to prevent egregious errors from having major impact.

#2

Incorrect usage type

Selecting a compute type (e.g. on-demand) inconsistent with application needs.

Incorrect usage type

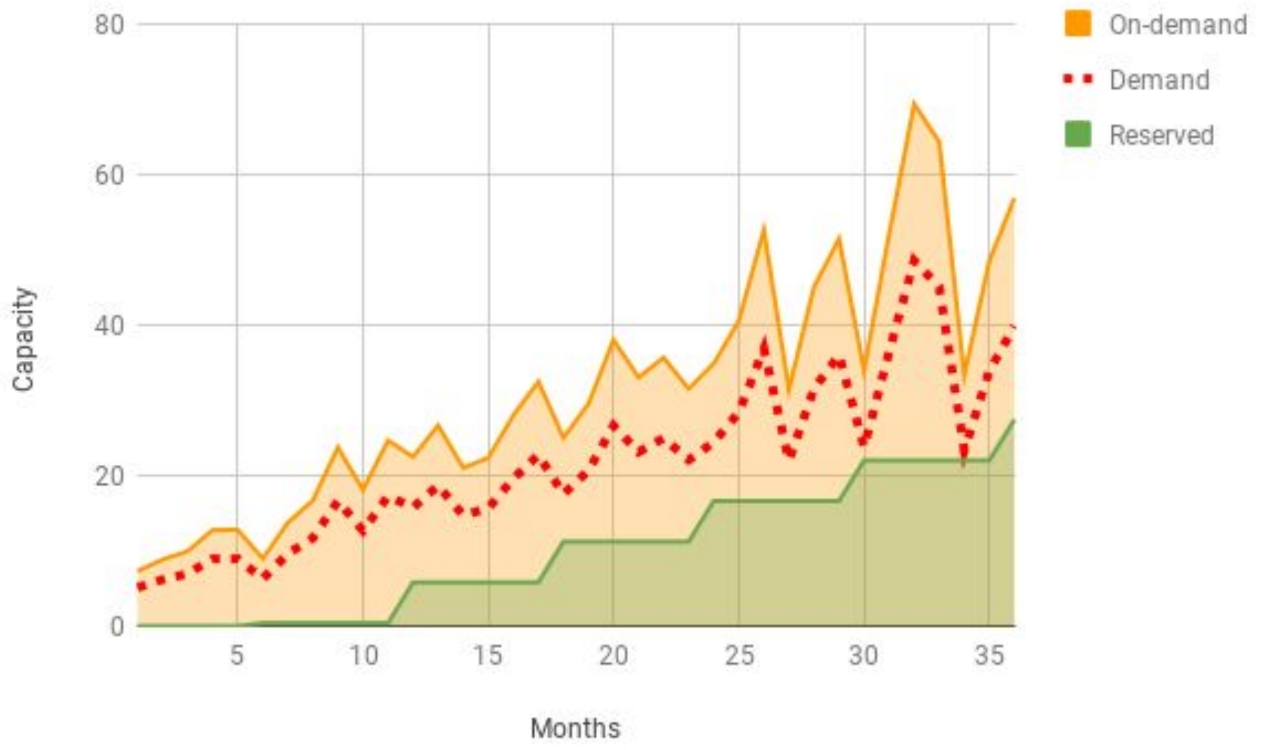
Most teams default to on-demand, when selecting between these categories:

- On-demand
- Reserved
- Spot/Preemptible

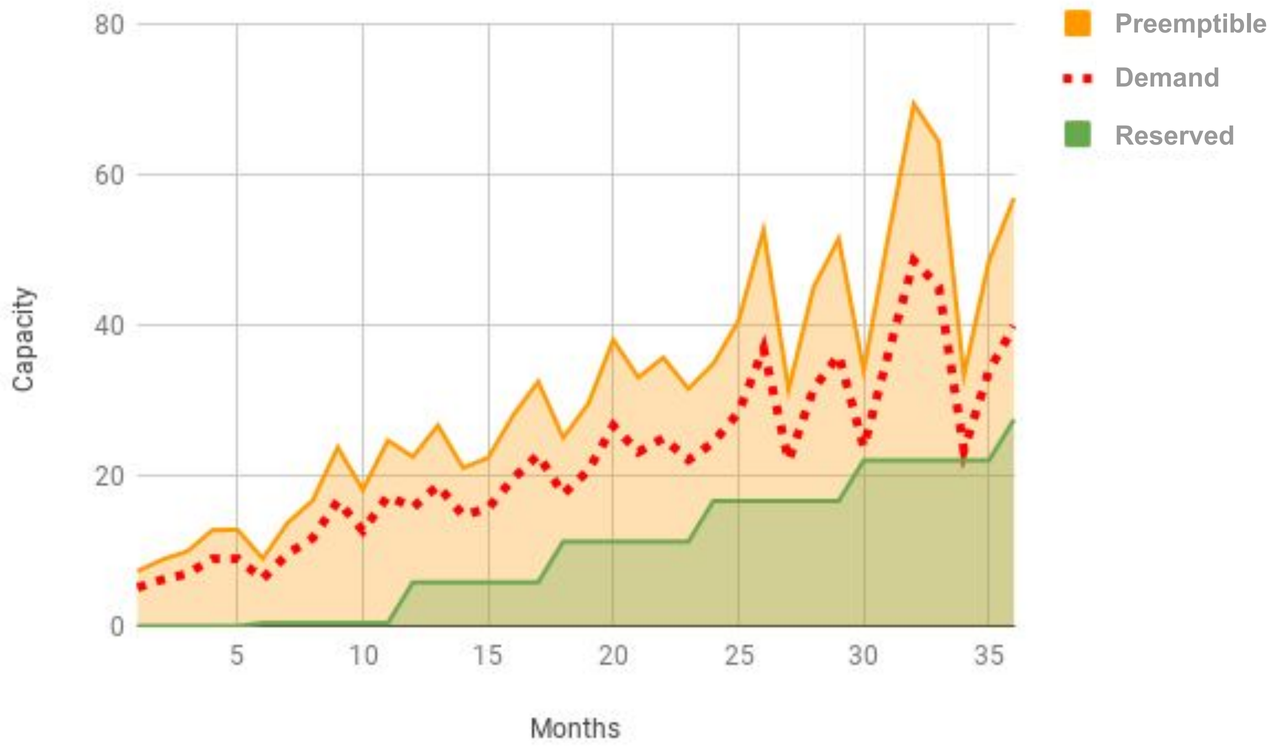
Impact: 70% of spend

Difficulty: **Medium/Hard**

Autoscaling environment



Spot or Preemptible environment



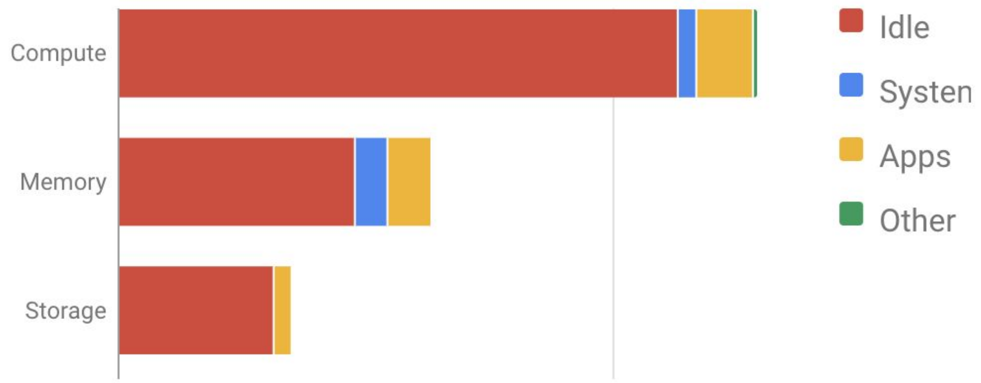
#1

Overprovisioning

Allocating excess compute capacity far in excess of application needs

Overprovisioning

Most teams run with 60-80% idle capacity, when they typically need half this amount.



Impact: 50-60% of spend

Difficulty: **Medium**

Proactively avoiding these five patterns can...

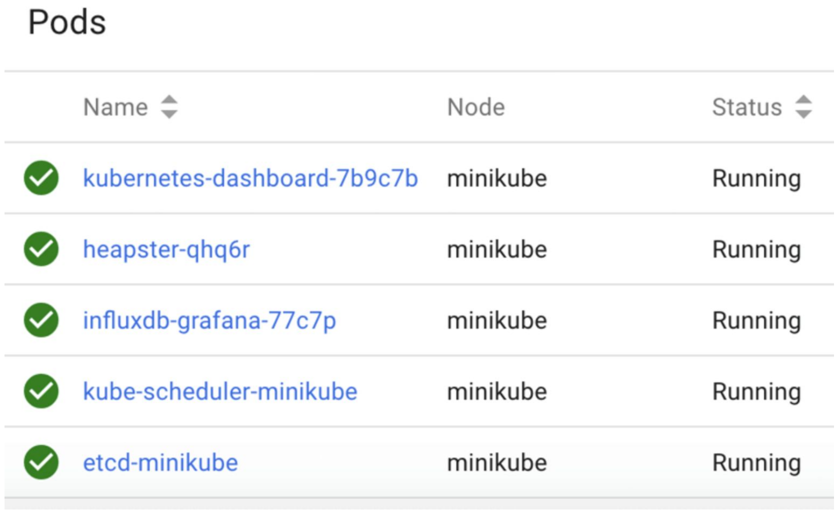
- Reduce Kubernetes spend by 80%+ with no impact to reliability
- Help remove security or privacy risks in the process
- Not make you spend your whole live investigating spend overruns

Please reach out (team@kubecost.com) if we can help in any way!

Questions?

Determining the cost of a resource...

1 { time provisioned } * 2 { quantity of resource } * 3 { price of resource}



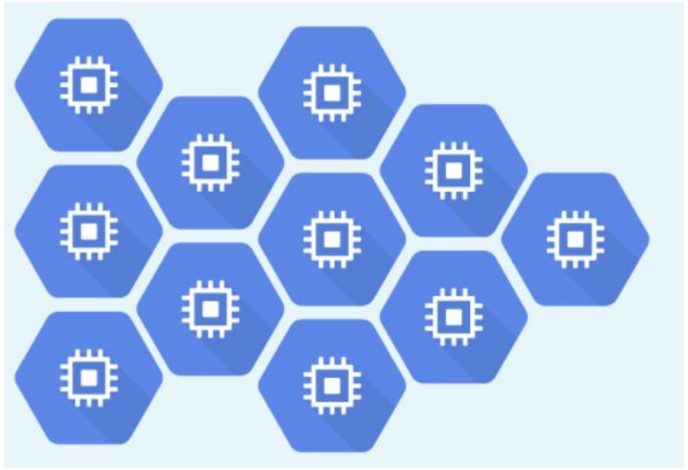
Pods

	Name	Node	Status
✓	kubernetes-dashboard-7b9c7b	minikube	Running
✓	heapster-qhq6r	minikube	Running
✓	influxdb-grafana-77c7p	minikube	Running
✓	kube-scheduler-minikube	minikube	Running
✓	etcd-minikube	minikube	Running

Determining the cost of a resource...

1 2 3

{ time provisioned } * { quantity of resource } * { price of resource}



Determining the cost of a resource...

1 { time provisioned } * 2 { quantity of resource } * 3 { price of resource}