DATADOG

100% Numérique

Telecom Valley

SophiaConf

Le cycle azurée de conférences et workshops Open Source

Autoscaling in Kubernetes: HPA, WPA, ClusterAutoscaler

Par **David BENQUE**

30 Juin | 18h20

Gratuit sur inscription **www.sophiaconf.fr**

Why do we want to scale?

Up: to cope with demand

Down: to save money

Why do we want to autoscale?

so you can sleep sometimes

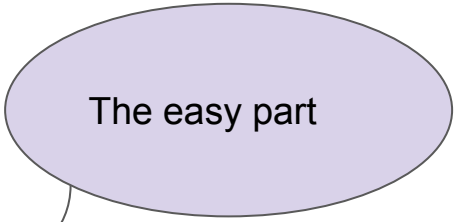
because we have fat fingers and we break things

Why do we want to scale?

Up: **to cope with demand**

Down: **to save money**

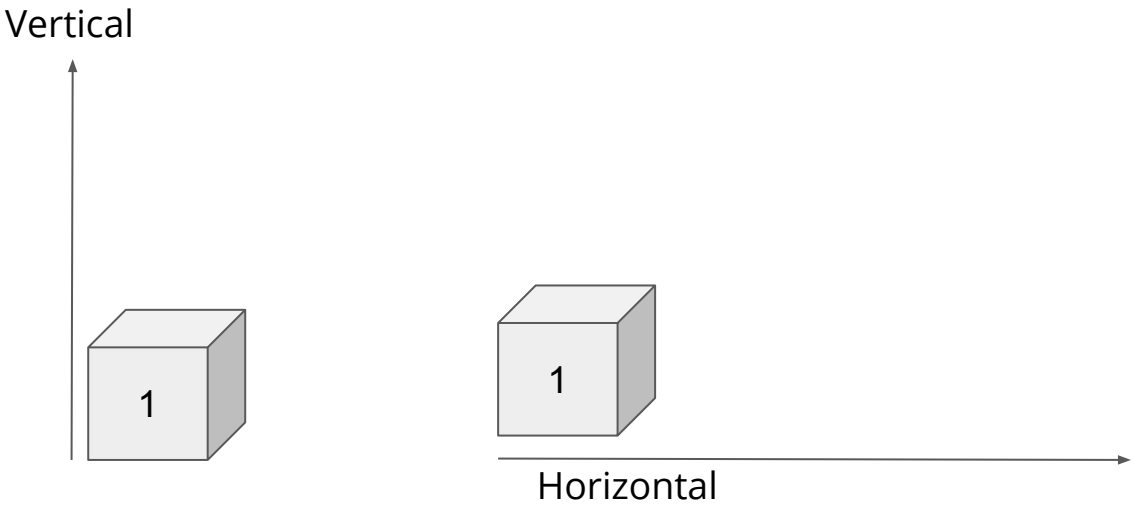
Why do we want to autoscale?



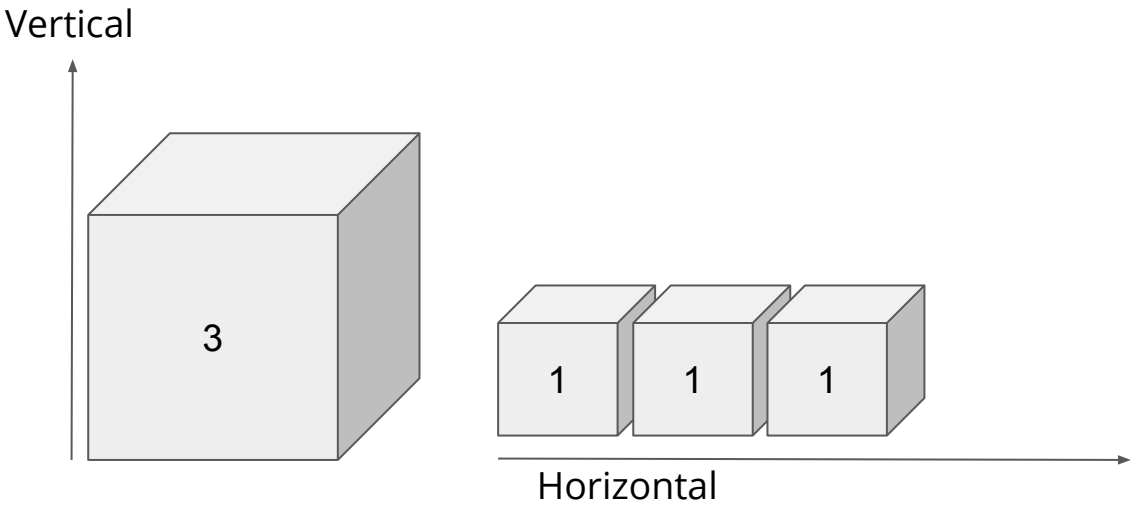
so you can sleep sometimes

because we have fat fingers and we break things

Scaling types



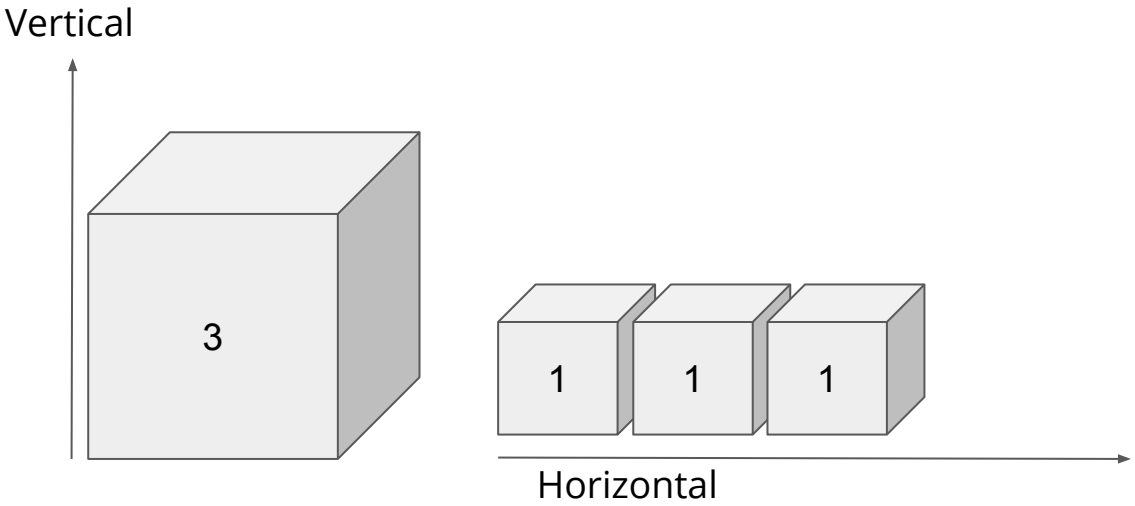
Scaling types



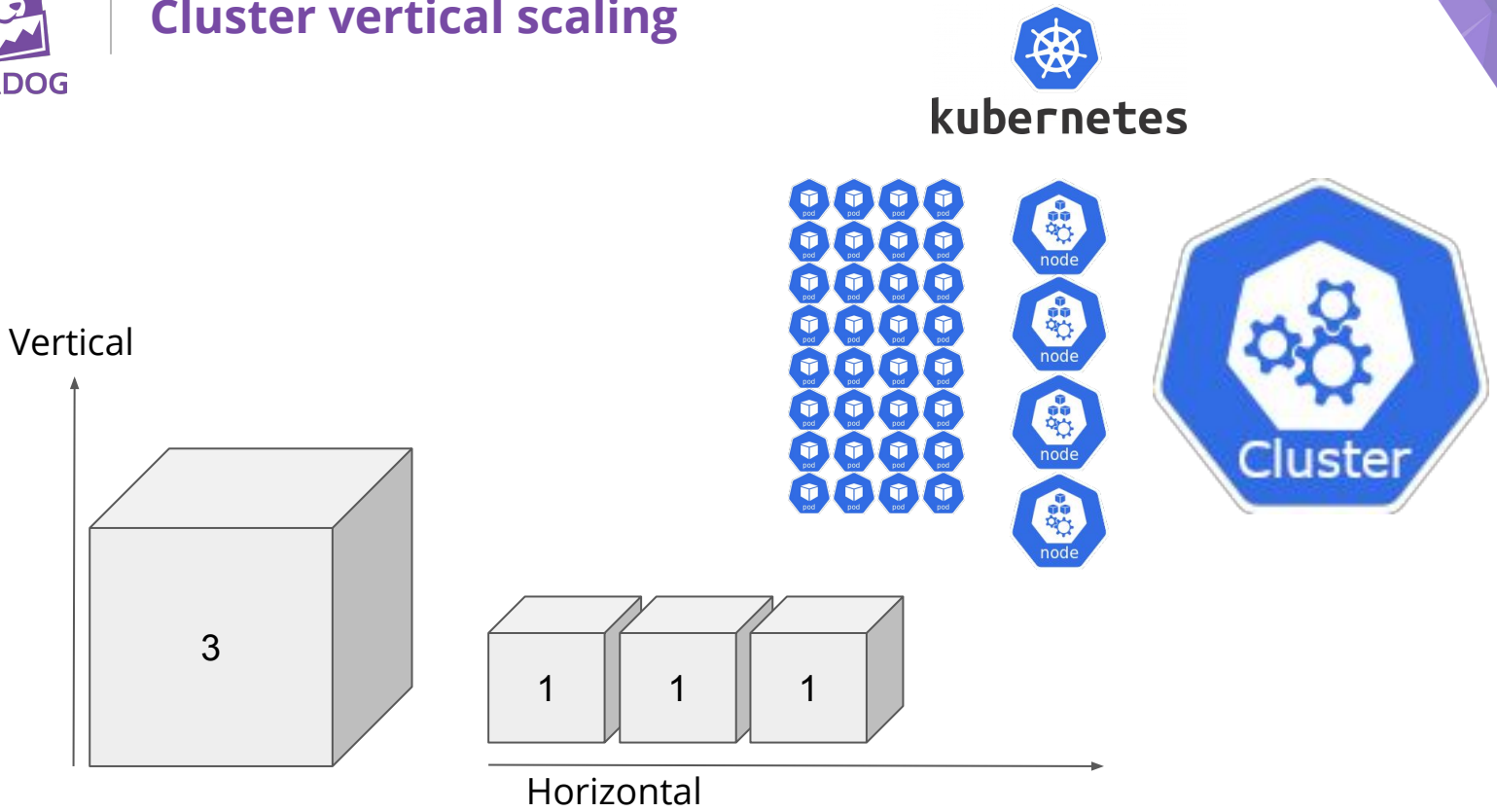
Scaling resources

kubernetes

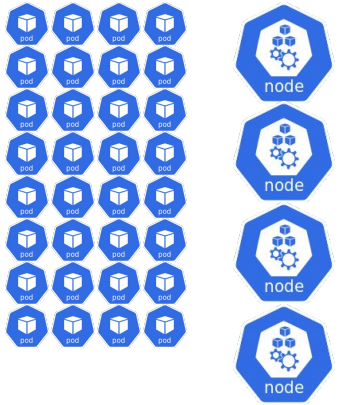
pod node Cluster



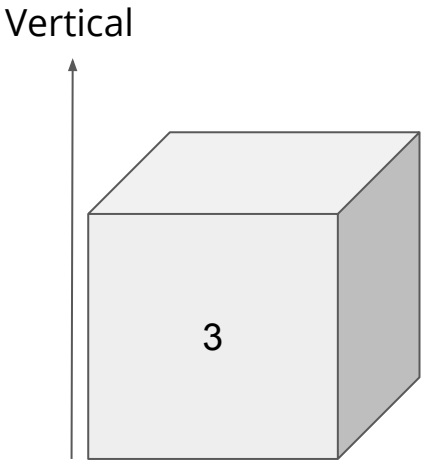
Cluster vertical scaling

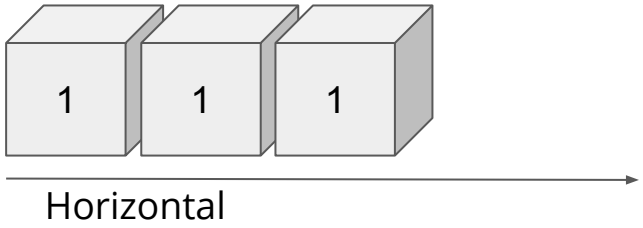


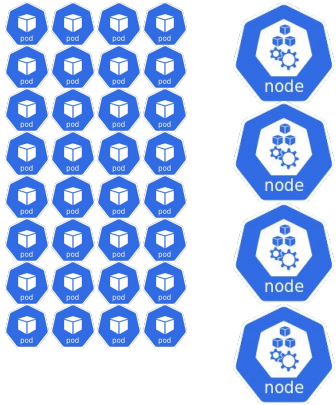
Cluster vertical scaling

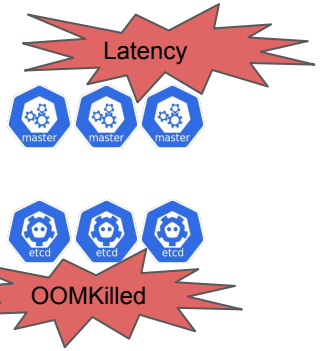


Cluster vertical scaling

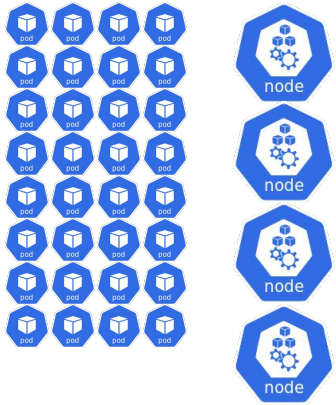


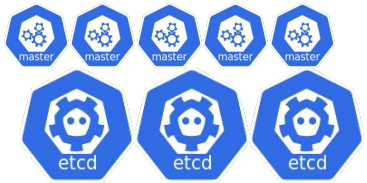




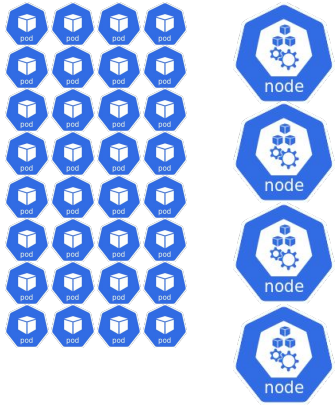


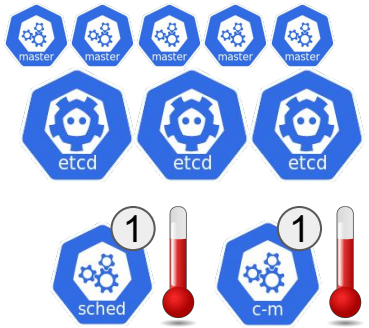
Cluster vertical scaling





Cluster vertical scaling





Vertical scaling

Vertical

change instance type:

c4.xlarge → c4.2xlarge

change resources request (and limits) in the pod definition:

```
resources:
  limits:
    cpu: 500m
    memory: 500Mi
  requests:
    cpu: 500m
    memory: 500Mi
```

→

```
resources:
  limits:
    cpu: 2
    memory: 2Gi
  requests:
    cpu: 2
    memory: Gi
```

Vertical scaling

Vertical

change instance type:

c4.xlarge → c4.2xlarge

apply only on new nodes/pods

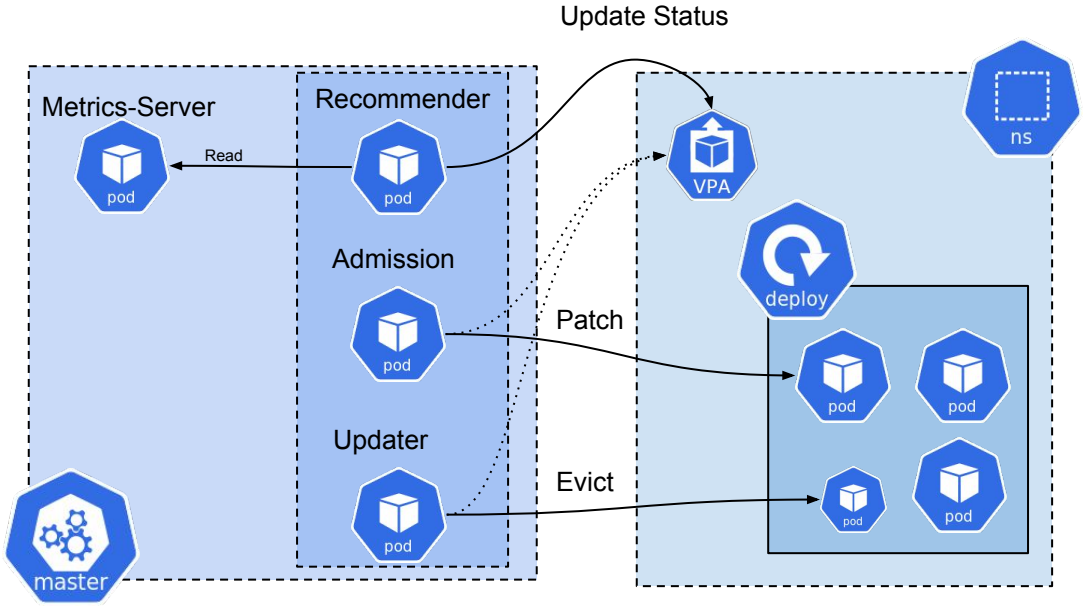
change resources request (and limits) in the pod definition:

```
resources:
  limits:
    cpu: 500m
    memory: 500Mi
  requests:
    cpu: 500m
    memory: 500Mi
```

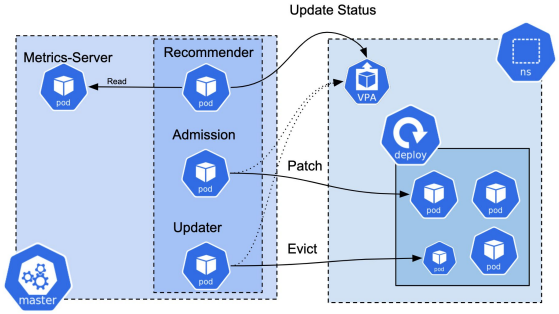
→

```
resources:
  limits:
    cpu: 2
    memory: 2Gi
  requests:
    cpu: 2
    memory: 2Gi
```

Vertical Pod Autoscaler (VPA)



Vertical Pod Autoscaler




```
Recommendation:
   Container Recommendations:
     Container Name:  hamster
     Lower Bound:
       Cpu:     550m
       Memory:  262144k
     Target:
       Cpu:     587m
       Memory:  262144k
     Uncapped Target:
       Cpu:     587m
       Memory:  262144k
     Upper Bound:
       Cpu:     21147m
       Memory:  387863636
```

Comments:

- kubernetes native support
- pod admission !!!
- per container recommendation
- historical data depth
- update frequency → disruption/eviction
- eviction
  → pod disruption budget (PDB)
  → not suitable for stateful application
  → pending pod
- maintain ratio requests/limits

Configuration:

- dedicated VPA resource
- target deployment
- min/max range
- dry-run mode:
  - `updatePolicy:`
  - `  updateMode: "Off"`

Vertical scaling on nodes, how ?

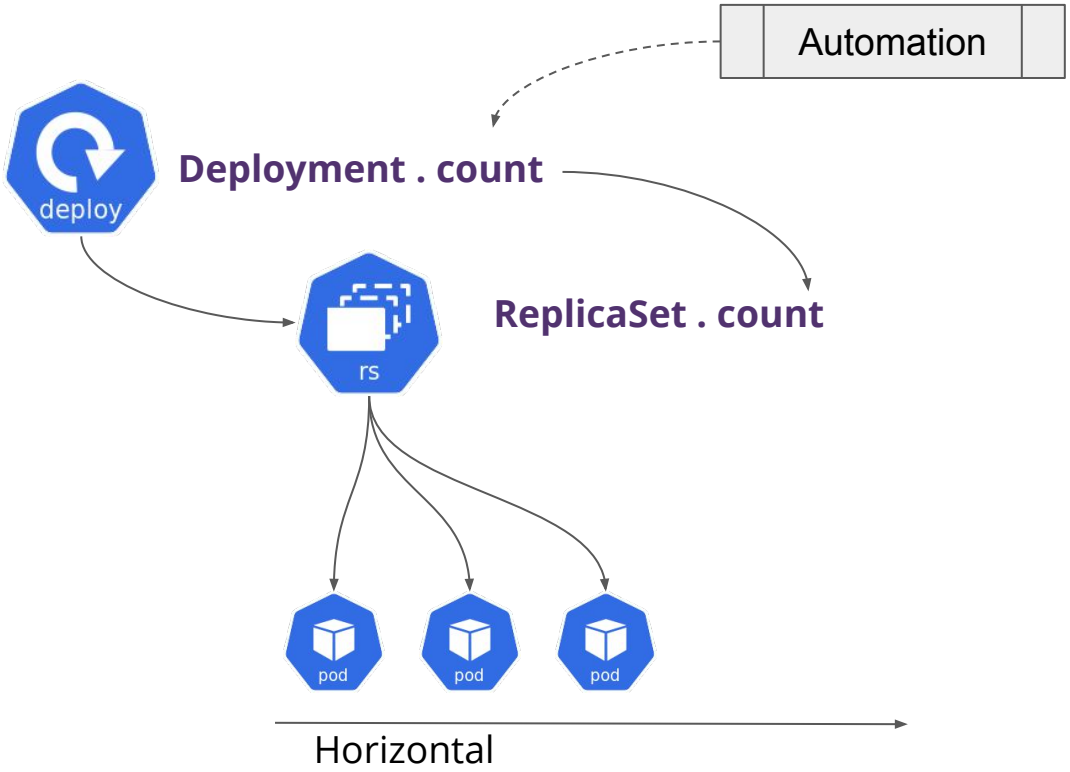
Vertical

change instance type:

c4.xlarge → c4.2xlarge

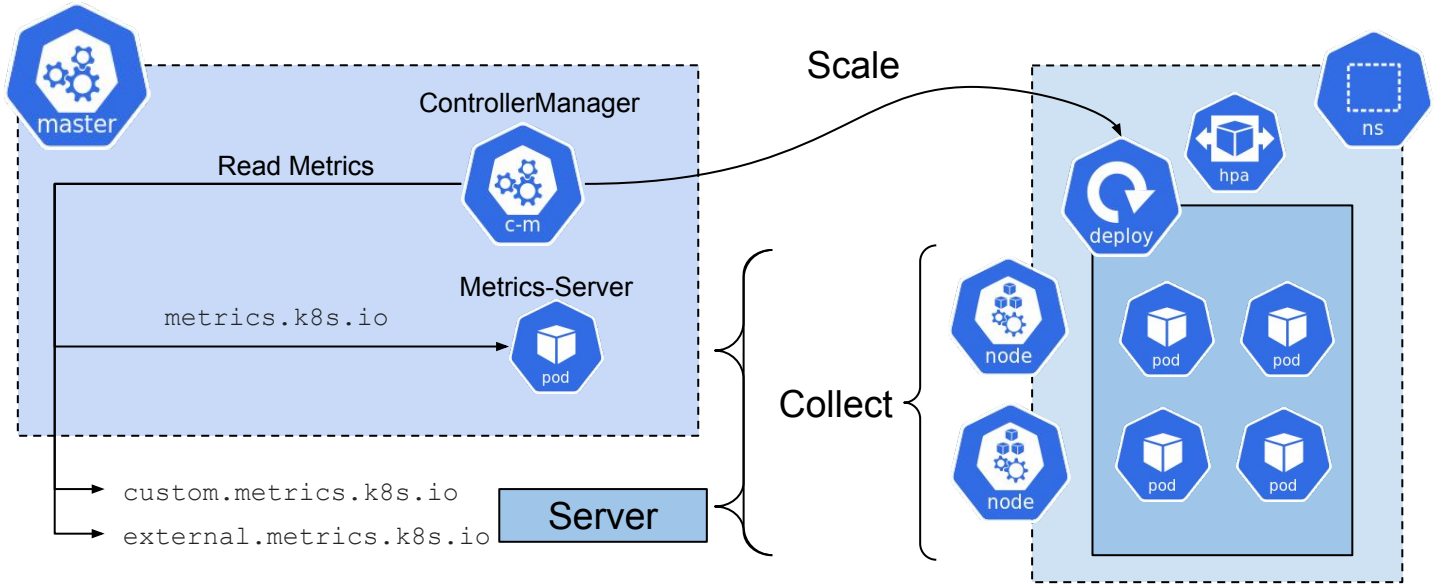
How do you provision your nodes ?

Horizontal scaling for pods

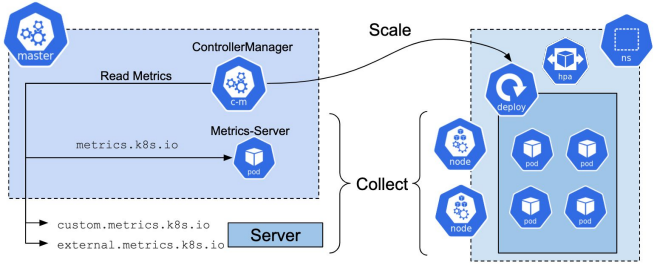


Horizontal Pod Autoscaler (HPA)

HPA runs inside ControllerManager



Horizontal Pod Autoscaler (HPA)



Comments:

- kubernetes native support

```
desiredReplicas = ceil[currentReplicas*(currentMetricValue/desiredMetricValue)]
```

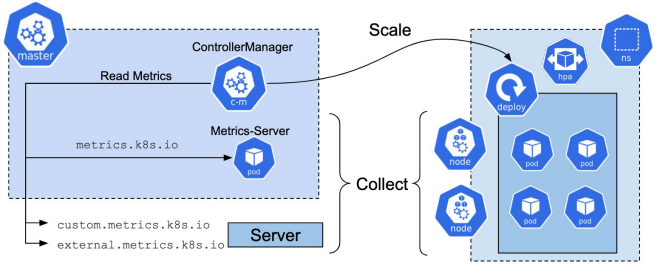
Configuration:

- dedicated HPA resource
- min/max range
- different source of metrics

- forbiddenWindow defined at controller level
  - Improved in 1.18 (1)
- tolerance defined at controller level (10%)

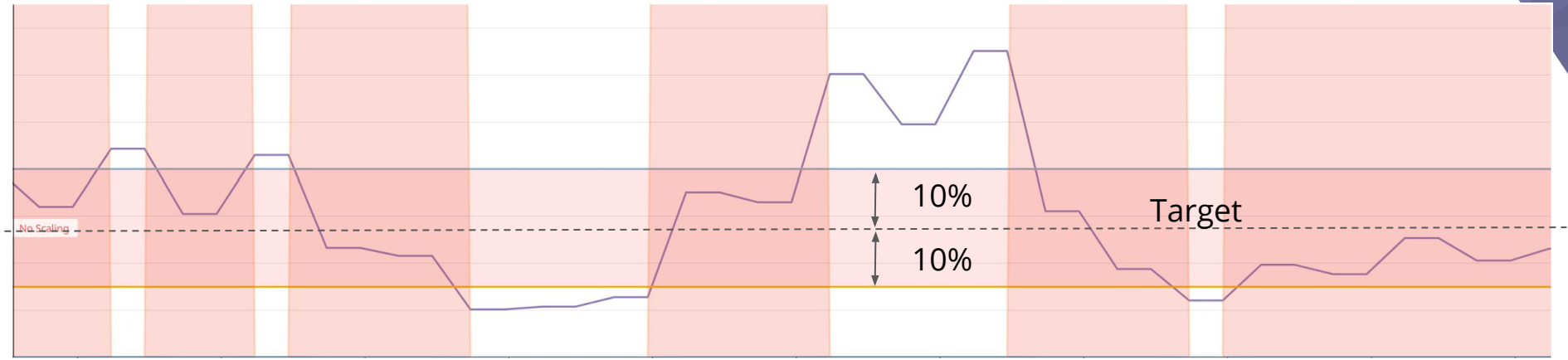
(1) https://github.com/kubernetes/enhancements/blob/master/keps/sig-autoscaling/20190307-configurable-scale-velocity-for-hpa.md

Horizontal Pod Autoscaler (HPA)



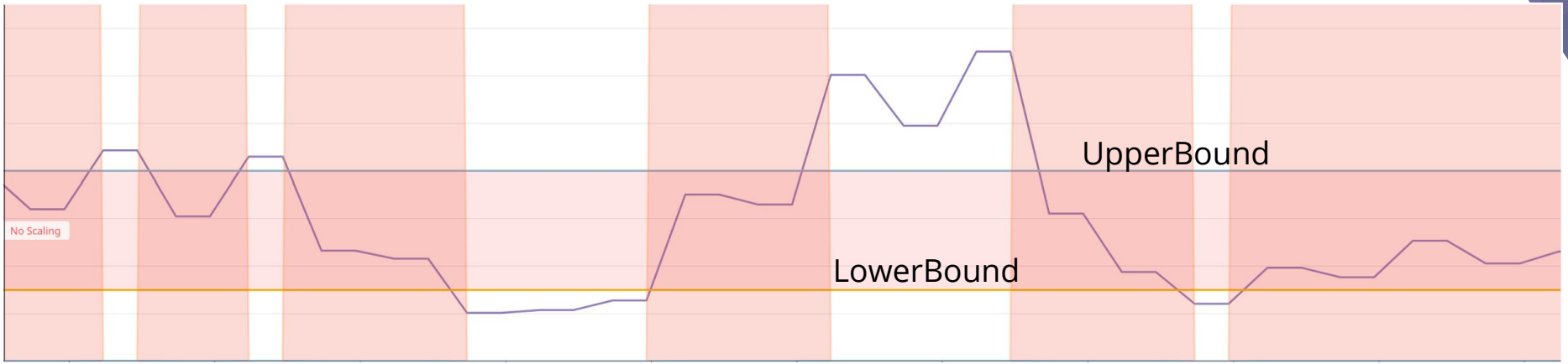
tolerance is 10% for all HPAs in the cluster

target is defined for each HPA



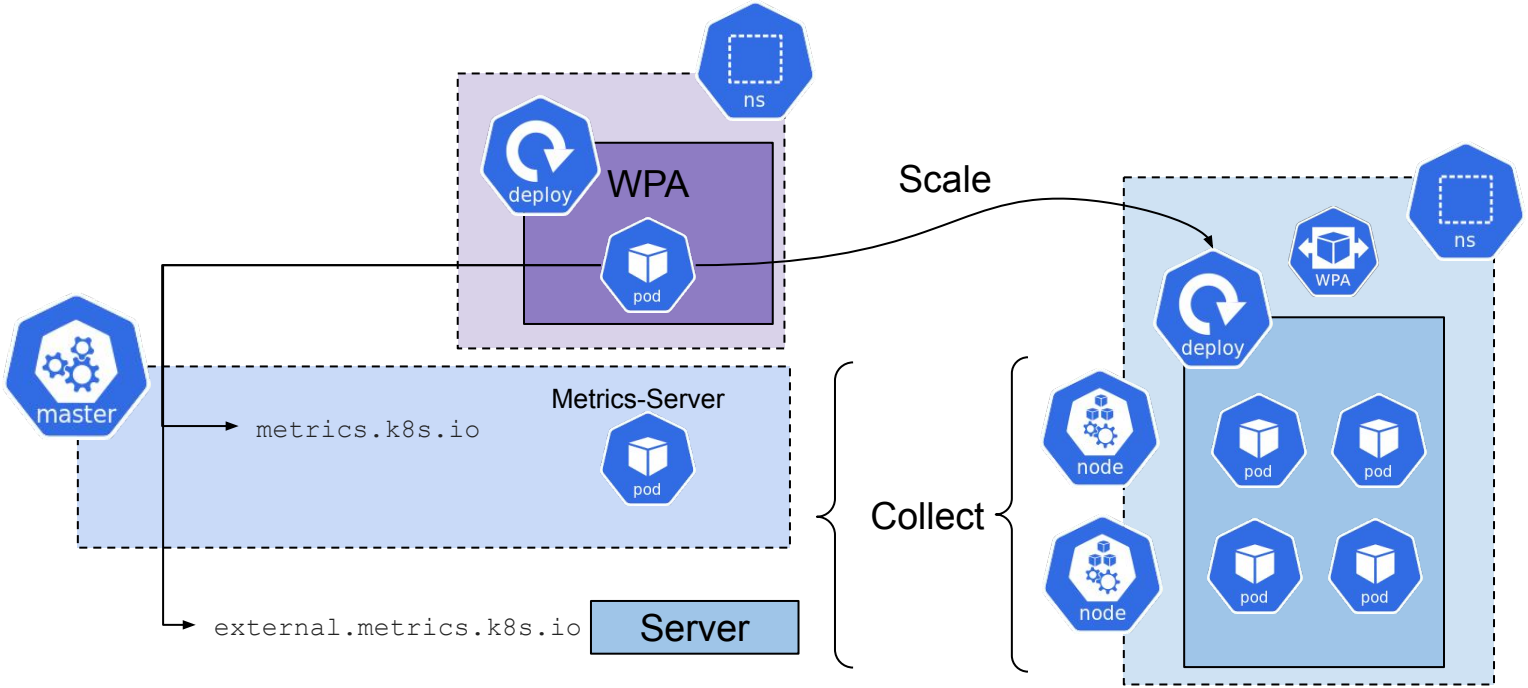
Watermark Pod Autoscaler (WPA) by DATADOG

bounds are defined for each WPA

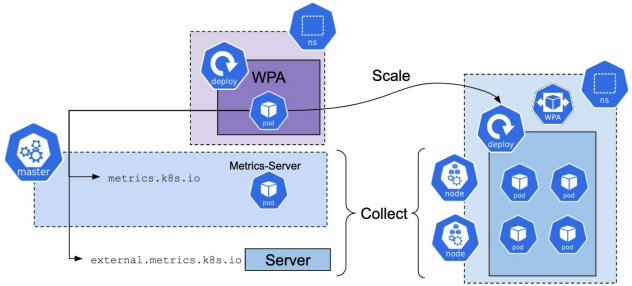


(1) https://github.com/kubernetes/enhancements/blob/master/keps/sig-autoscaling/20190307-configurable-scale-velocity-for-hpa.md

Watermark Pod Autoscaler (WPA) by DATADOG



Watermark Pod Autoscaler (WPA) by DATADOG



Comments:

- opensource project by Datadog
- Can be used even without Datadog product

Configuration:

- dedicated WPA resource
- same as HPA with more:
  - tolerance replaced by upper/lower bound
  - velocity

c: current number of replicas
x: sum of external metrics provider
h,l: HighWaterMark, LowWaterMark

y: computed number of replicas

$$\forall x \in \mathbb{R}^{+}, \quad c \in \mathbb{N}^{*}, \quad \{l,h\} \in \mathbb{R}^{2}[l<h], \quad \exists! y \in \mathbb{N}^{*} /$$
$$\begin{cases} x < l < h \implies y = \lfloor \frac{x*c}{l} \rfloor \\ l < x < h \implies y = c \\ l < h < x \implies y = \lceil \frac{x*c}{h} \rceil \end{cases}$$

What about ?

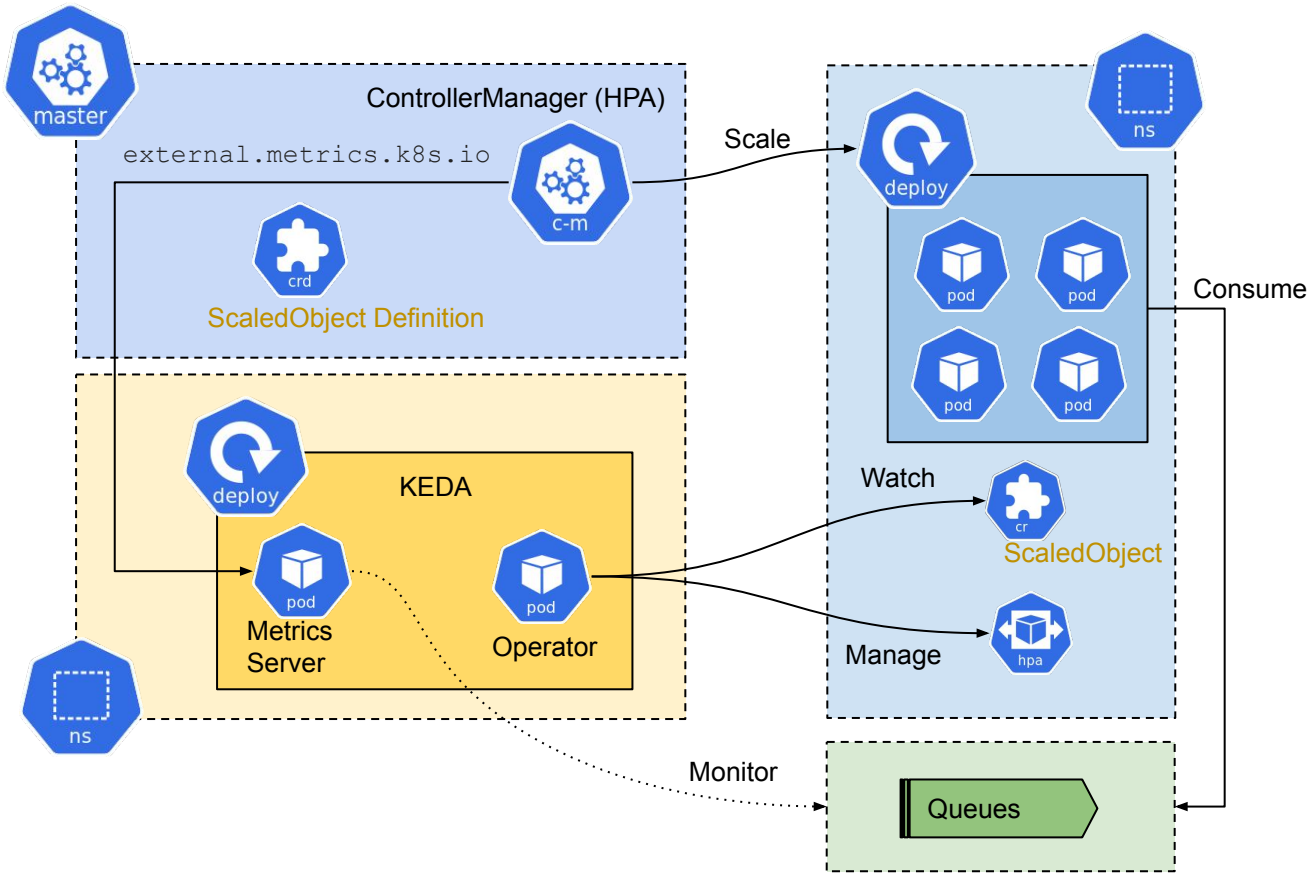
serveless...

scale-down to 0...

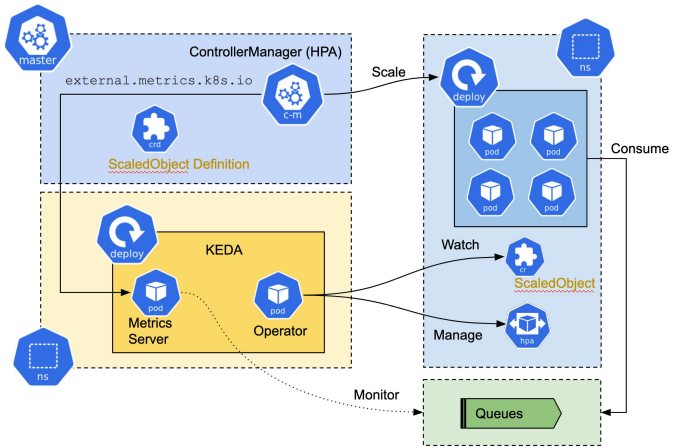
CPU when 0 instance ...

event driven architectures ...

Kubernetes-based Event Driven Autoscaling



Kubernetes-based Event Driven Autoscaling



Comments:

- opensource project: www.keda.sh
- can create jobs (long running scenario)

Configuration:

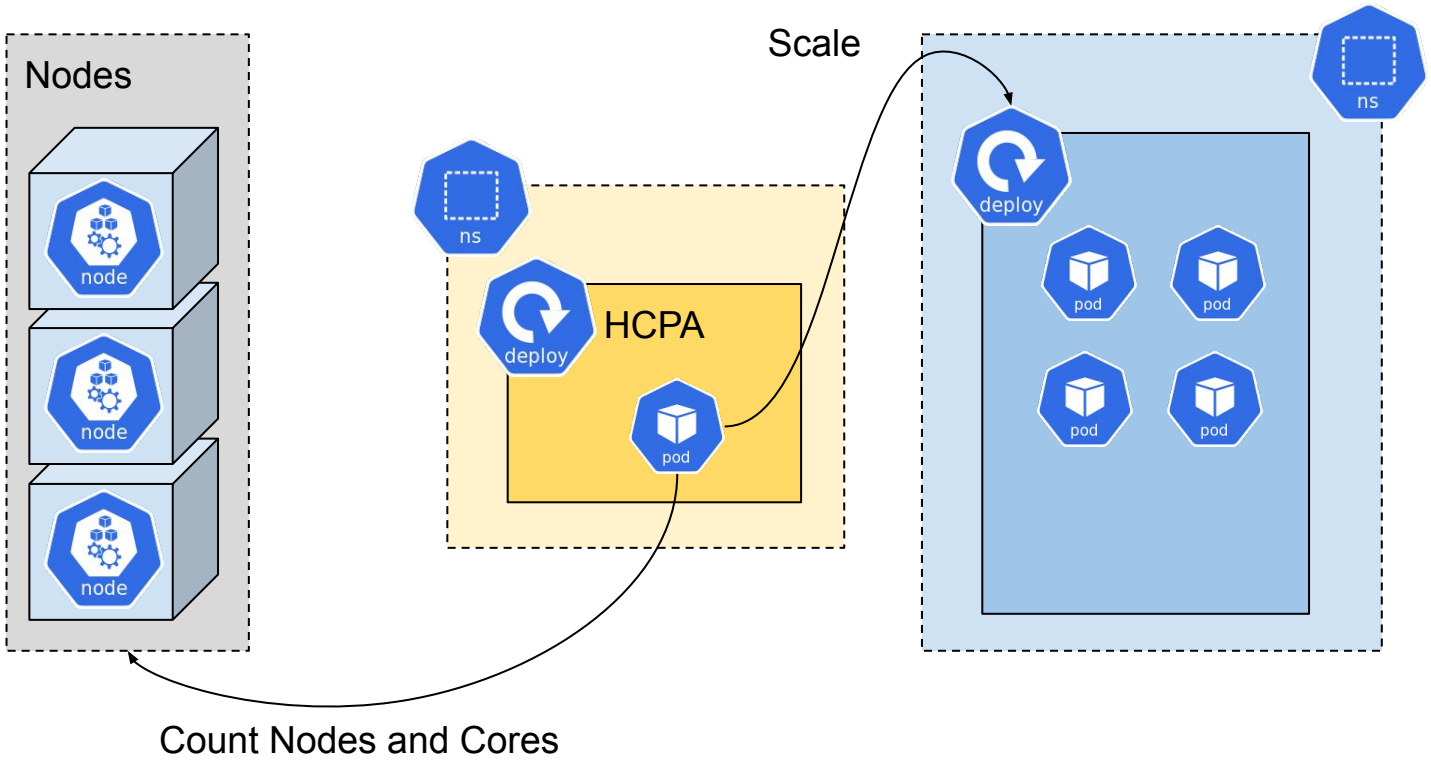
- dedicated ScaledObject resource
- 19 built-in scalers
  - kafka
  - redis
  - rabbitMQ
  - NATS
  - AWS (Kinesis Stream, SQS Queue, CloudWatch)
  - Azure (Event Hubs, Monitor, Storage Queue, Blob Storage ...)

What about ?

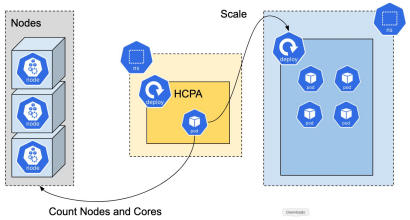
cluster size...

growing with number of nodes...

Horizontal cluster-proportional-autoscaler



Horizontal cluster-proportional-autoscaler




```
replicas = max(
      ceil( cores * 1/coresPerReplica ) ,
      ceil( nodes * 1/nodesPerReplica ) )

replicas = min(replicas, max)
replicas = max(replicas, min)
```

Comments:

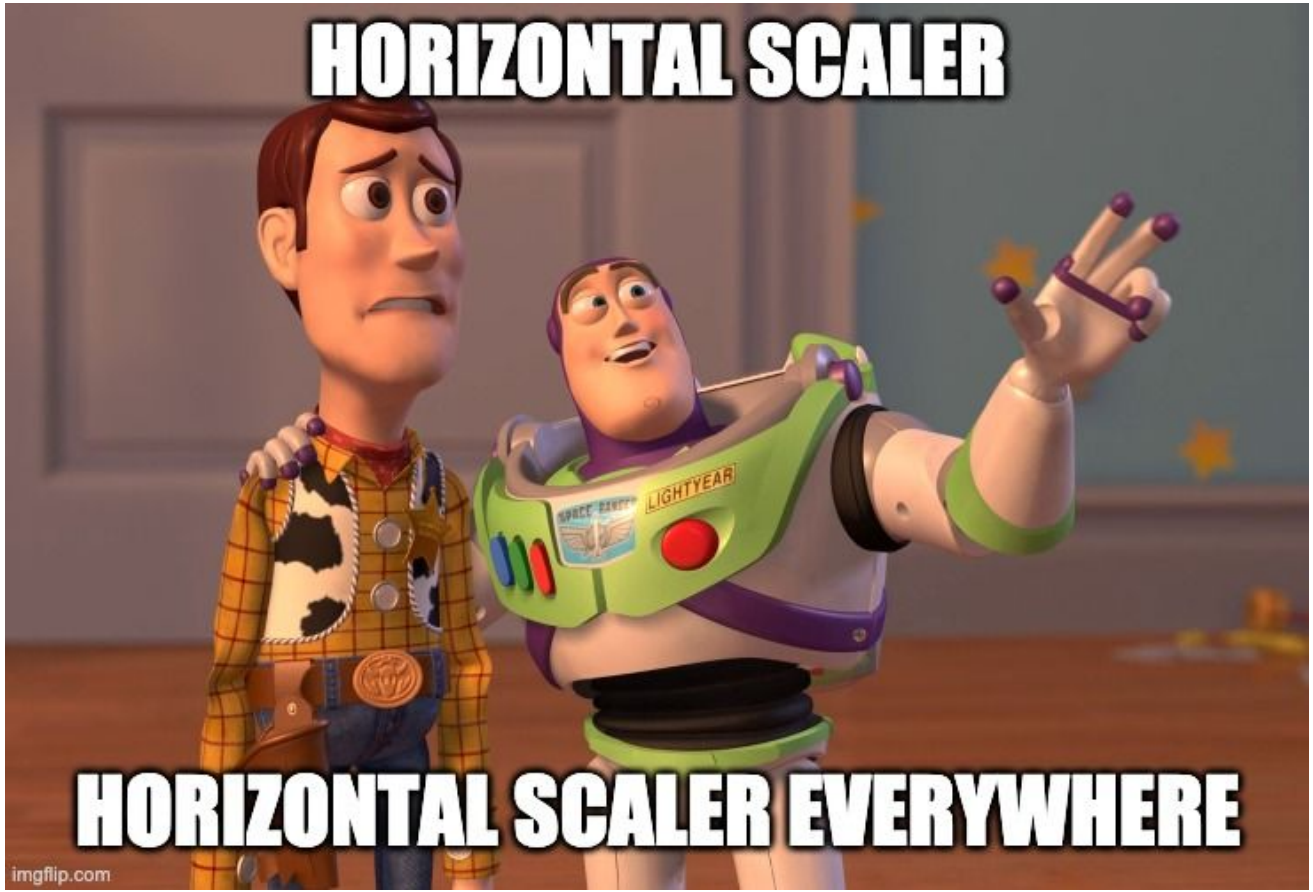
- one HCPA per target deployment
- nodes scoped by labels
- not suitable for general applications
- suitable for cluster application (coredns)

Configuration:

- ConfigMap
- linear (node or core)
- ladder (node or core)
- no timing definition
- `includeUnschedulableNodes` **recommended**

DATADOG

Any other horizontal scaler for Pods?



Don’t, don’t *

Vertical Pod Autoscaler

+

Horizontal Pod Autoscaler

=

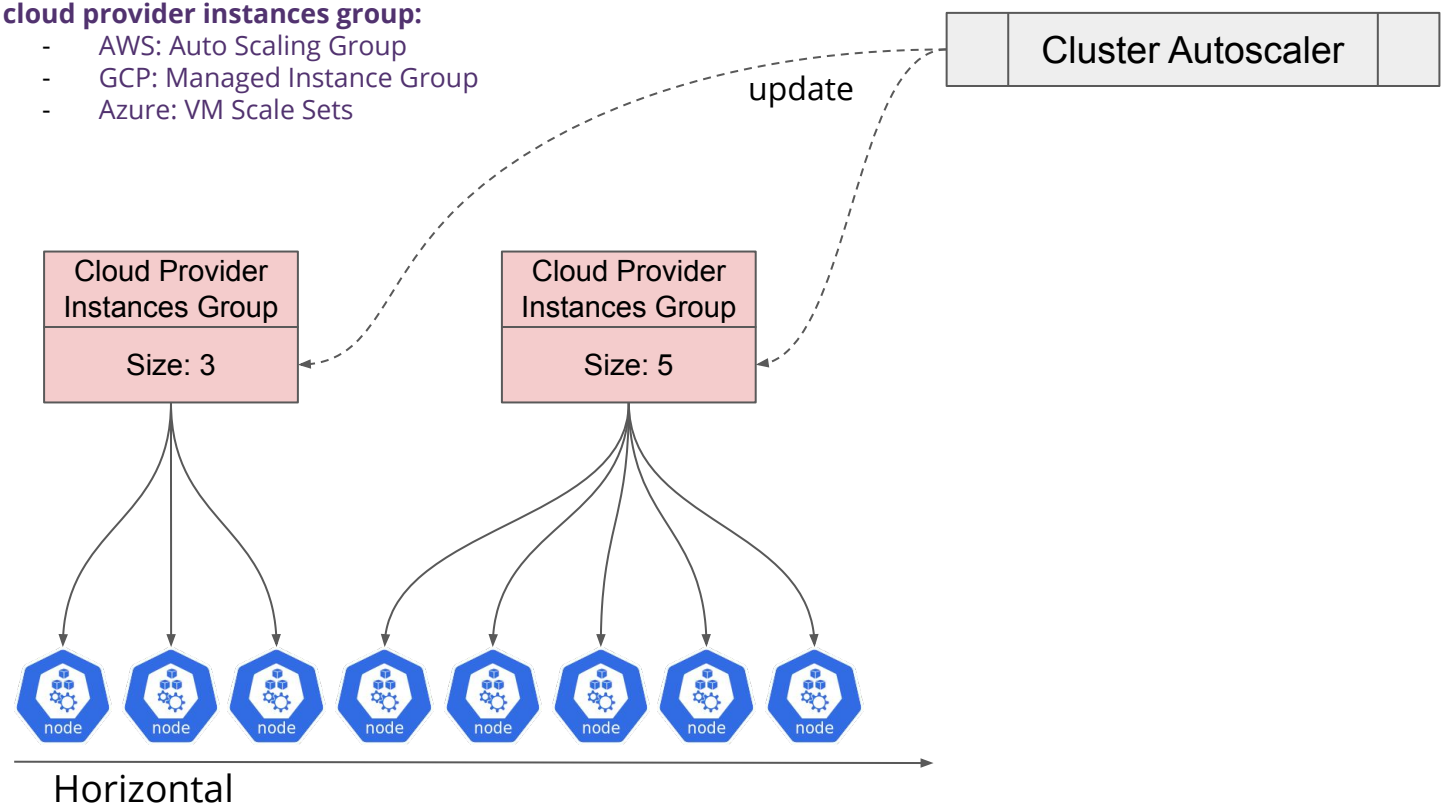
FUN

***at least not on the same metrics**

Horizontal scaling for nodes

cloud provider instances group:

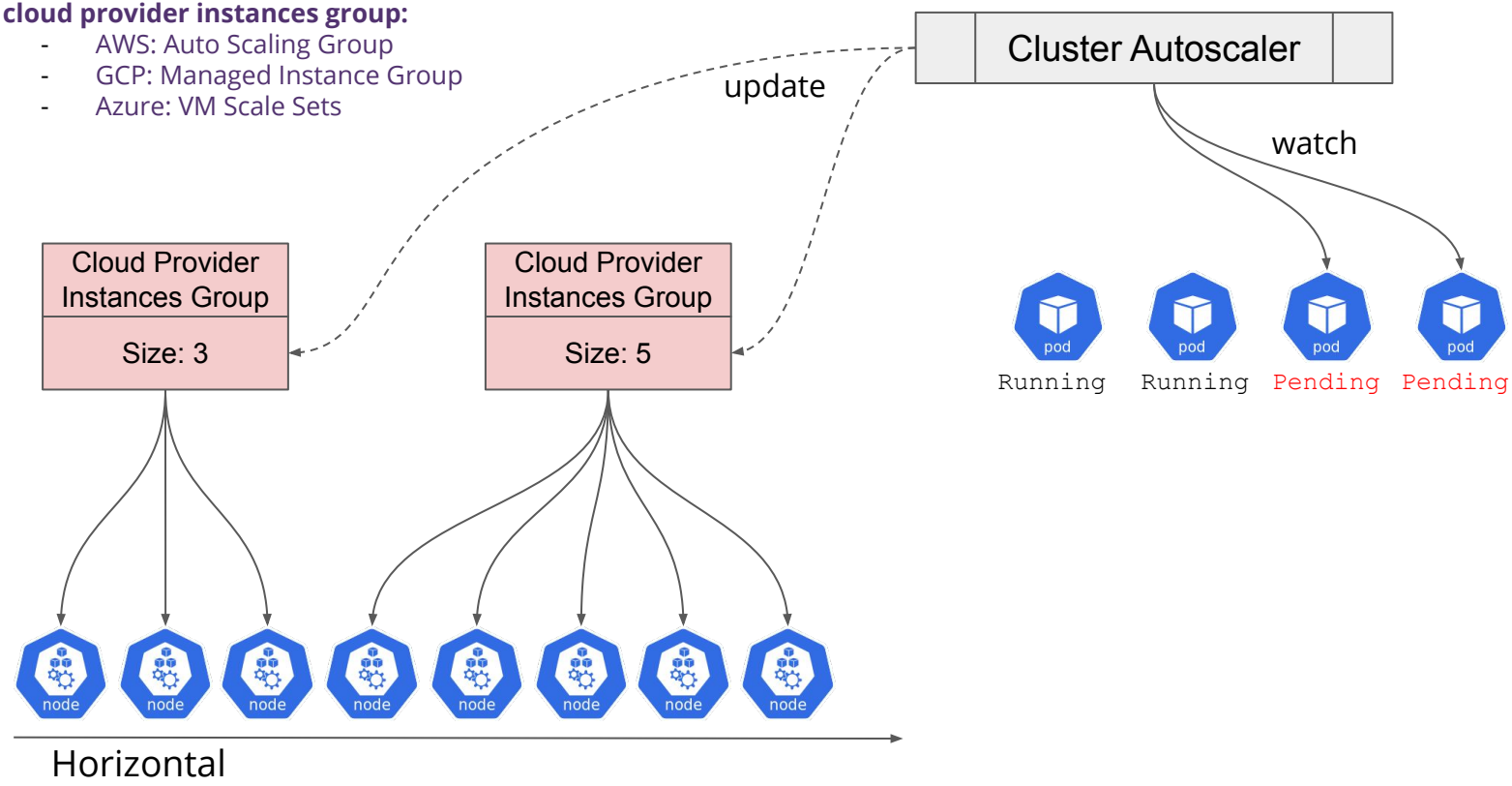
- AWS: Auto Scaling Group
- GCP: Managed Instance Group
- Azure: VM Scale Sets



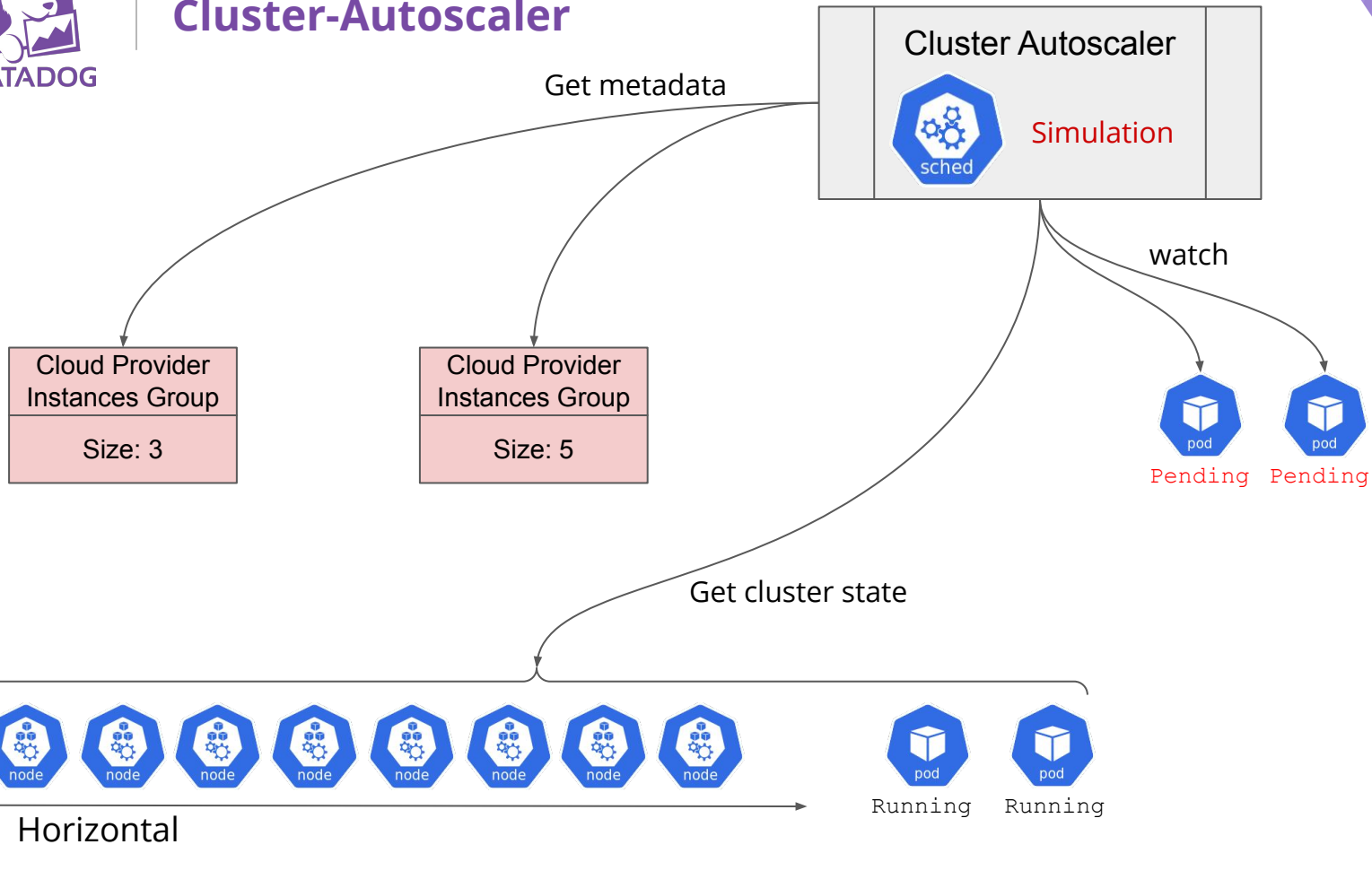
Horizontal scaling for nodes

cloud provider instances group:

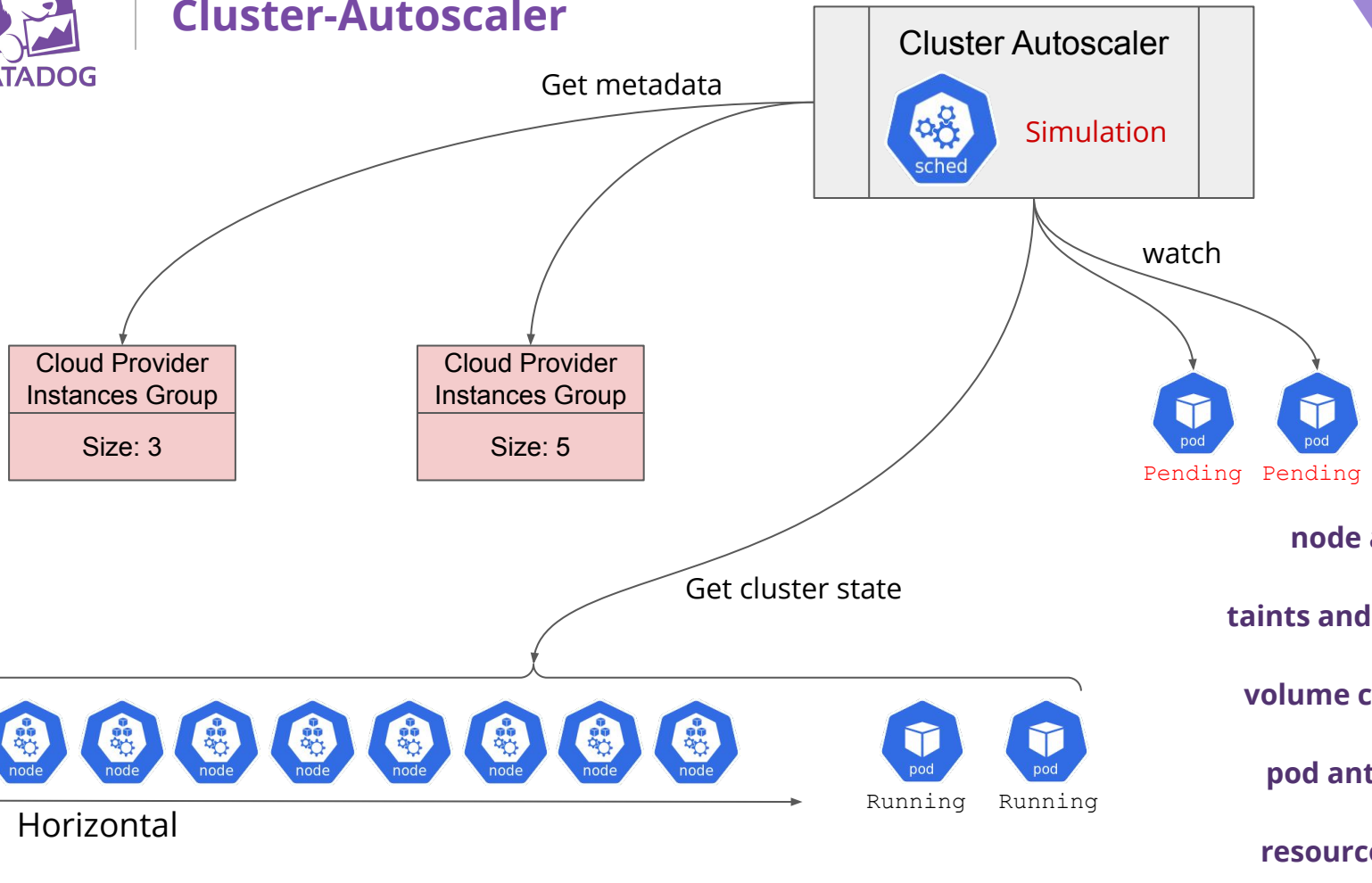
- AWS: Auto Scaling Group
- GCP: Managed Instance Group
- Azure: VM Scale Sets



Cluster-Autoscaler



Cluster-Autoscaler



node affinity

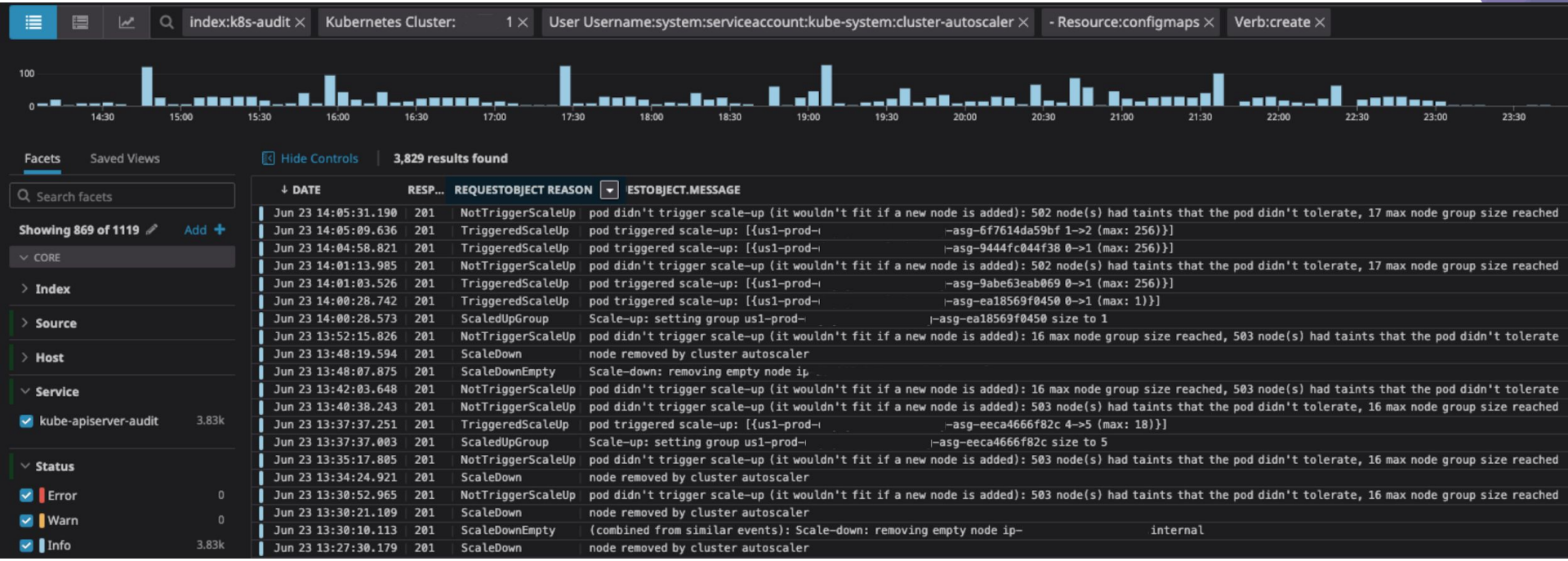
taints and tolerations

volume constraints

pod anti-affinity

resource request

Cluster-Autoscaler events



Cluster-Autoscaler NotTriggerScaleUp

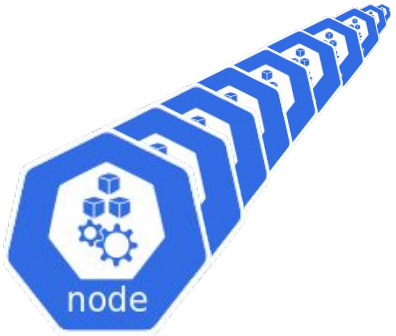
The usual suspects:

- **max group size reached**
- **PVC bound to existing node**
- **Resources does not match** ***(*** 

I am looking at you)

- **user definition issue:**
  - **tolerations**
  - **node selectors**

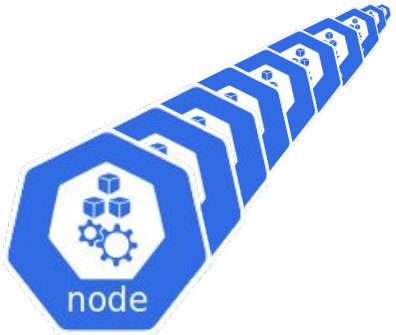
Cluster-Autoscaler Infinite ScaleUp



N nodes scaled-up just with 1 pod

Pending

Cluster-Autoscaler Infinite ScaleUp

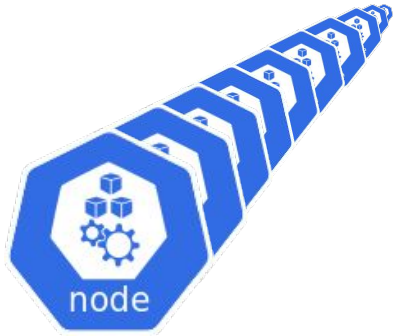


N nodes scaled-up just with 1 pod

Pending

Simulation **!=** **Real**

Cluster-Autoscaler Infinite ScaleUp



N nodes scaled-up just with 1 pod

Pending

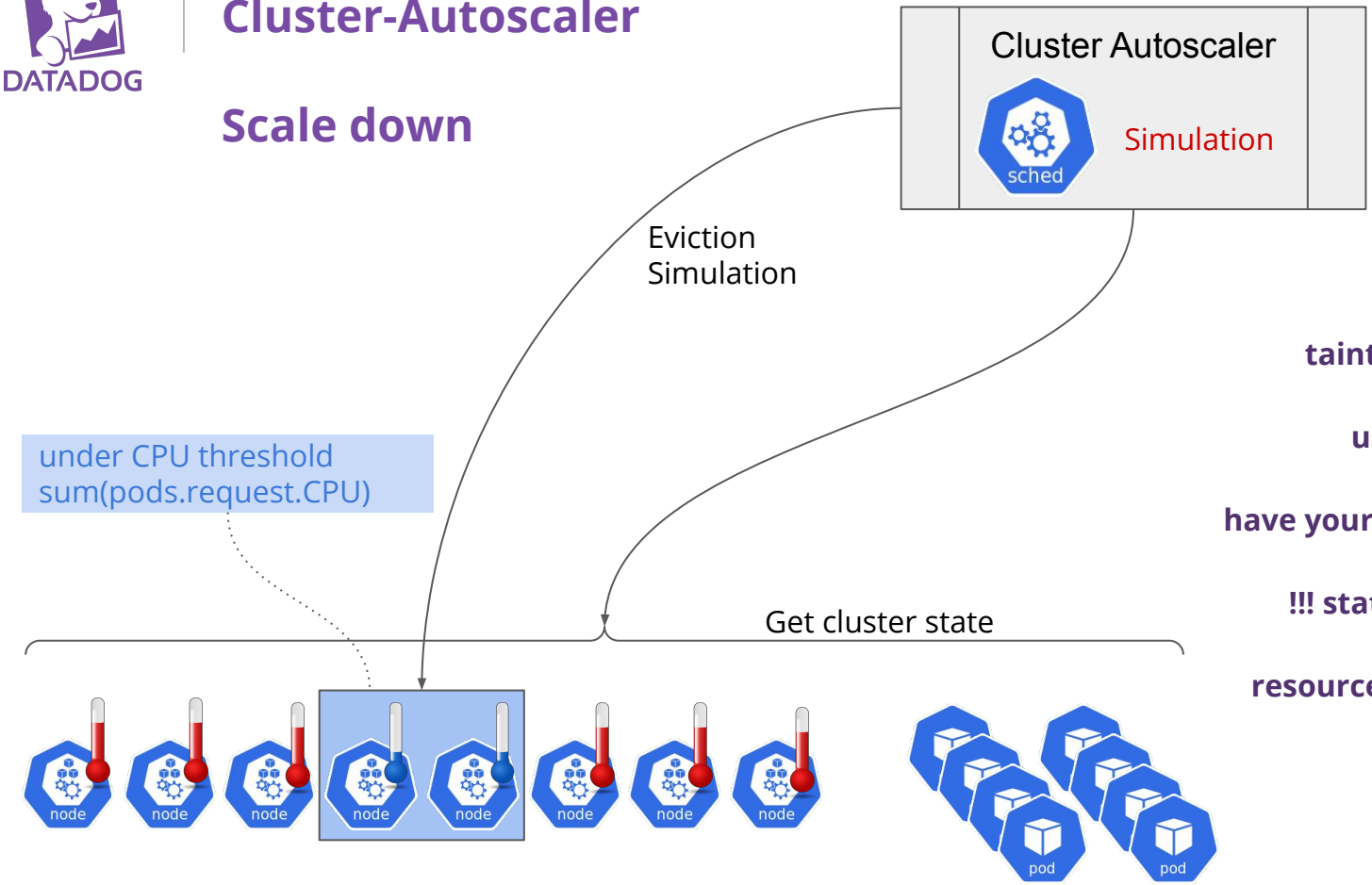
Simulation != Real

node capacity != node allocatable

metadata cache != live metadata **(aws at least)**

Cluster-Autoscaler

Scale down



taints unneeded nodes

uses eviction API

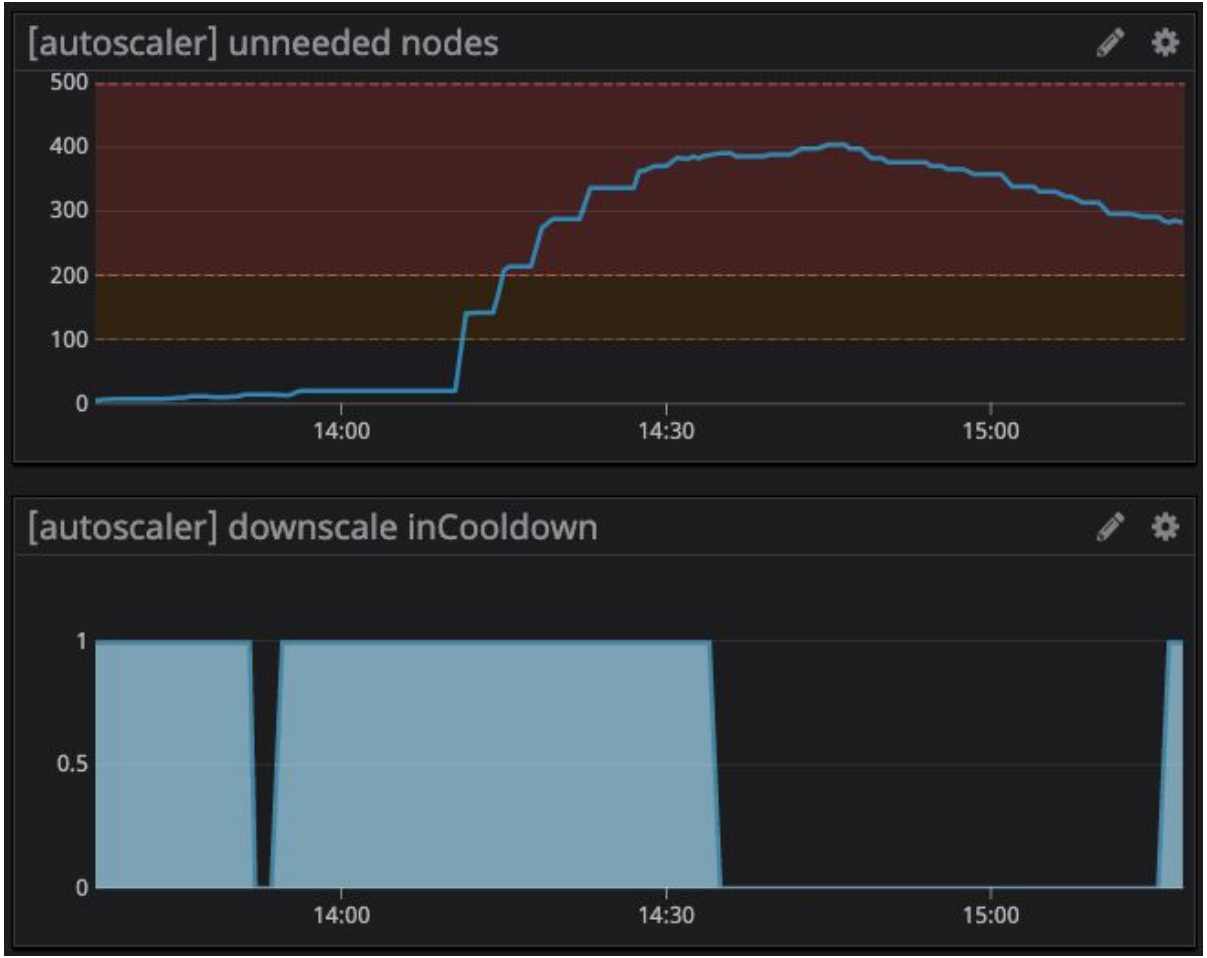
have your pod disruption budgets

!!! stateful applications !!!

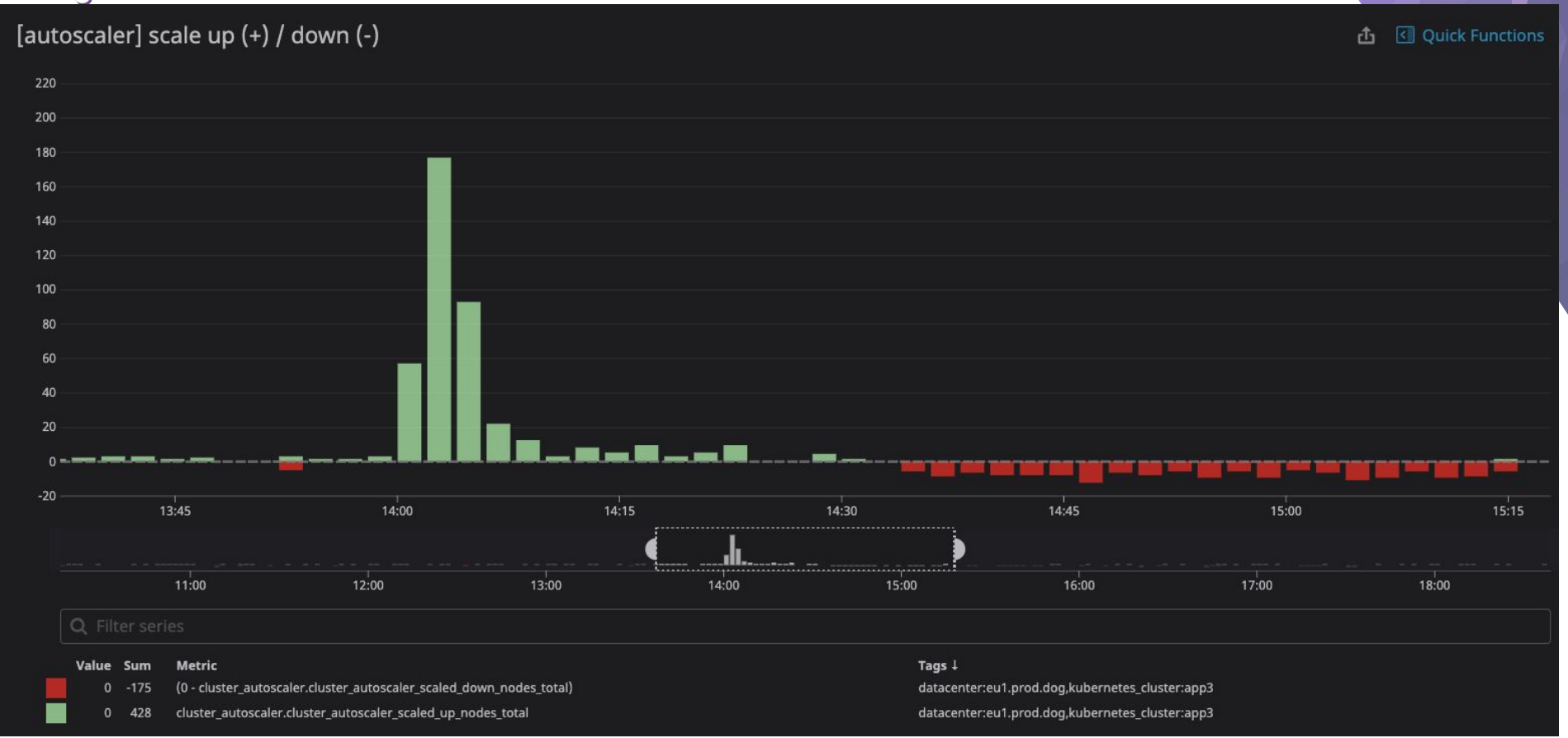
resource request MUST BE SET

DATADOG

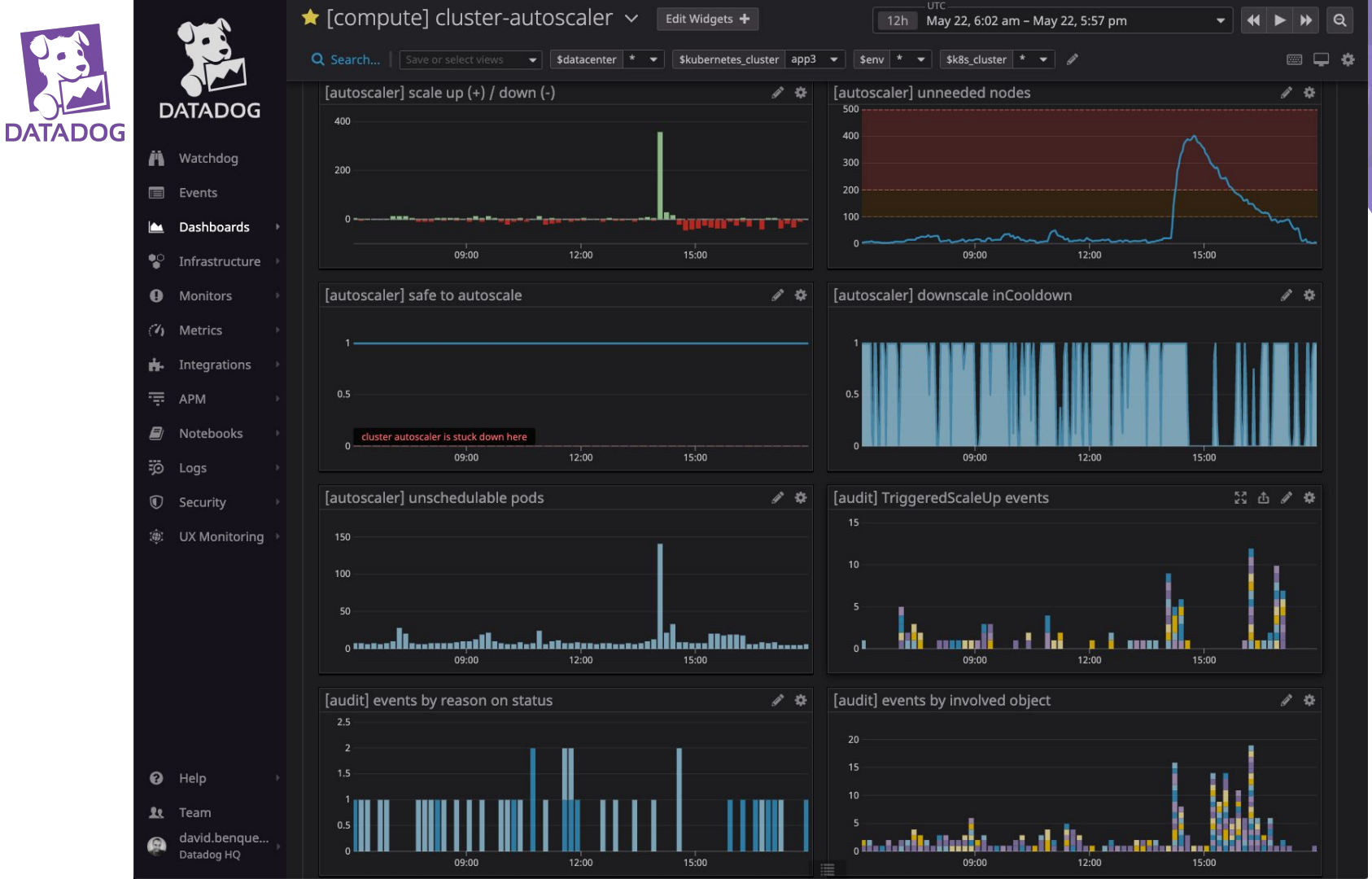
CoolDown



Cluster Autoscaler - velocity



Infinite scale-up


DATADOG
[compute] cluster-autoscaler
Edit Widgets
12h
UTC
May 22, 6:02 am – May 22, 5:57 pm
Search...
Save or select views
$datacenter *
$kubernetes_cluster app3
$env *
$k8s_cluster *
Watchdog
Events
Dashboards
Infrastructure
Monitors
Metrics
Integrations
APM
Notebooks
Logs
Security
UX Monitoring
Help
Team
david.benque...
Datadog HQ
[autoscaler] scale up (+) / down (-)
[autoscaler] unneeded nodes
[autoscaler] safe to autoscale
cluster autoscaler is stuck down here
[autoscaler] downscale inCooldown
[autoscaler] unschedulable pods
[audit] TriggeredScaleUp events
[audit] events by reason on status
[audit] events by involved object
09:00
12:00
15:00

Cluster-Autoscaler Scaleup to ?

You can set your limit!

Official test ~1000 nodes, ~30 pods per node
Loop iteration set to 10s
... it remains under 30s

The complexity is O(nodes x pods)

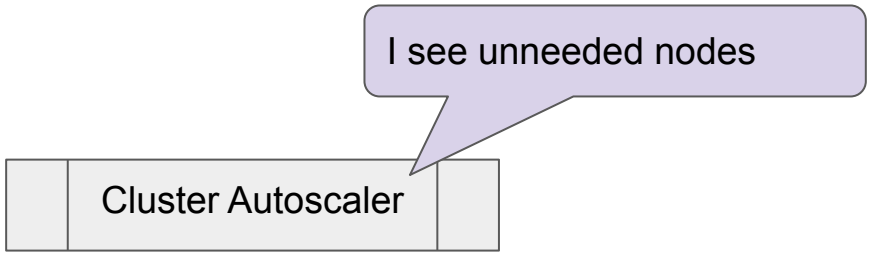
so with less pods on each nodes, >1000 nodes is ok
some feature (podAntiAffinity increase complexity)

Cluster-Autoscaler, the case of jobs ...

Cluster autoscaler is doing a loop every 10 seconds (parameter)

The job may last only few seconds ....

If you have groups of nodes dedicated to jobs:



solution: create a node holder deployment

Combining solutions

cluster-autoscaler
+
any horizontal pod autoscaler
=
fine

High Frequency

use VPA alone or to get recommandations

apply the recommendations manually

Low Frequency

Combining solutions

cluster-autoscaler
+
any horizontal pod autoscaler
=
fine

high frequency

so you can sleep and save money… in theory

use VPA alone or to get recommandations

apply the recommendations manually

low frequency

Questions

https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler
https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler
https://github.com/kubernetes-sigs/cluster-proportional-autoscaler

https://github.com/kedacore/keda

https://github.com/DataDog/watermarkpodautoscaler

https://github.com/kubernetes/enhancements/tree/master/keps/sig-autoscaling