KubeCon | CloudNativeCon

Europe 2019

10 Ways to Shoot Yourself in the Foot with Kubernetes, #9 Will Surprise You!

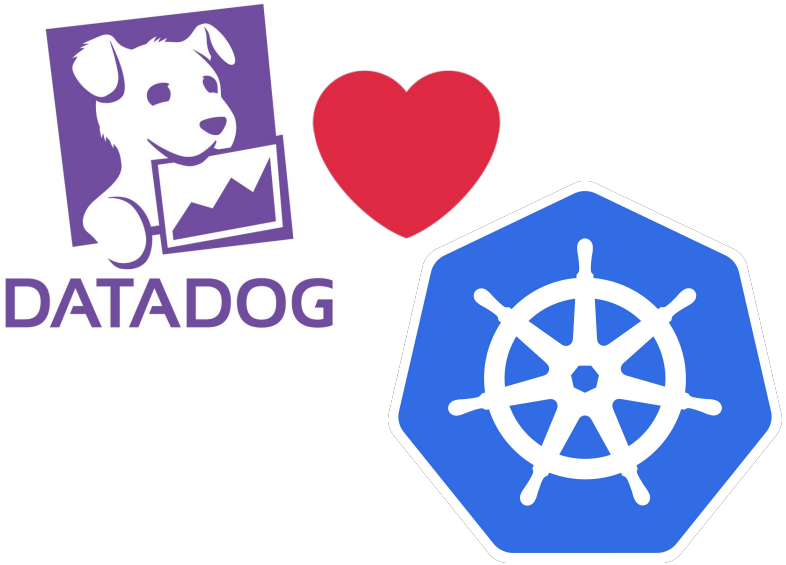
Laurent Bernaille & Rob Boll
Datadog Infrastructure Team

Who are we?

Datadog is a monitoring service; metrics, traces, logs, dashboards, alerts, etc.

Cloud Native service provider

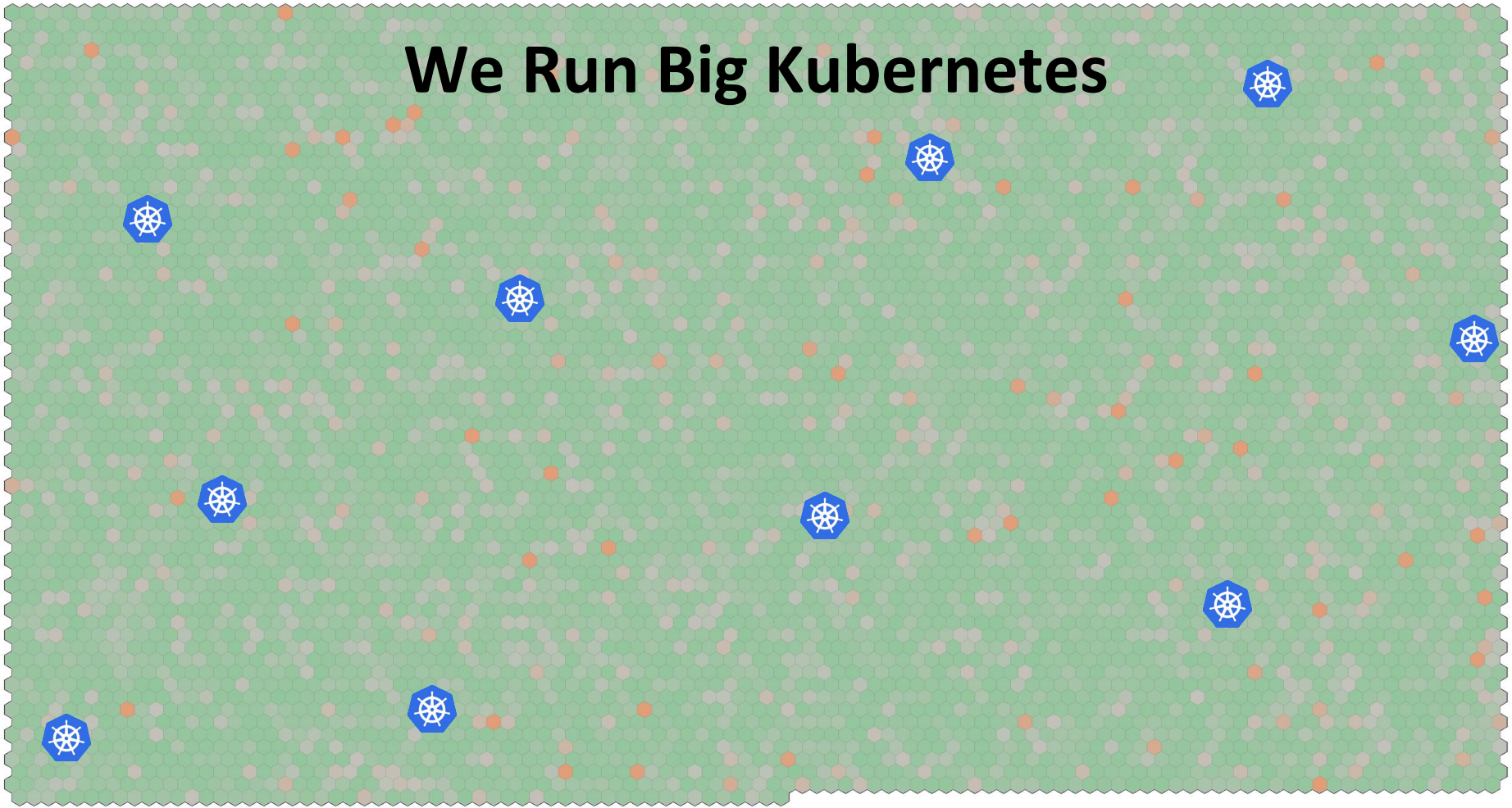
www.datadoghq.com



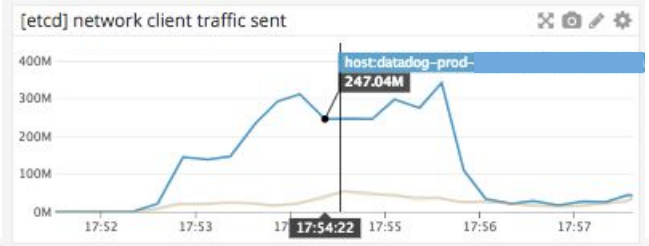
We run the cloud infrastructure that powers the product.

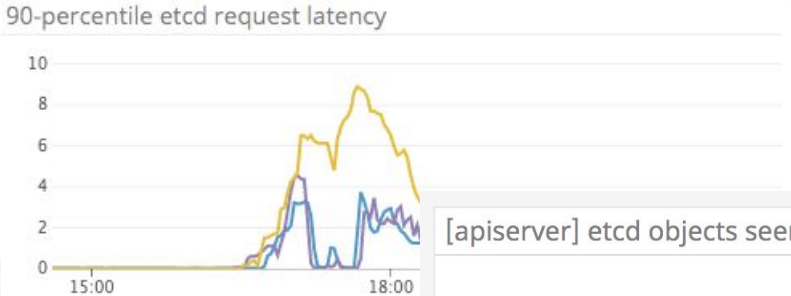
Cloud Native end user

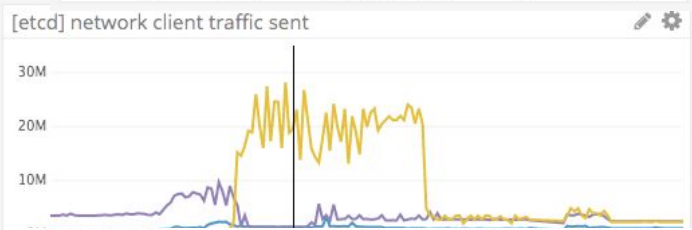
We Run Big Kubernetes

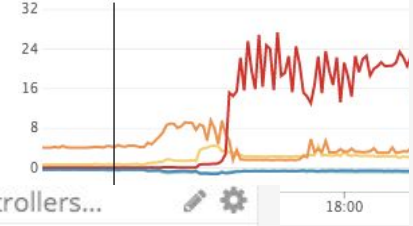


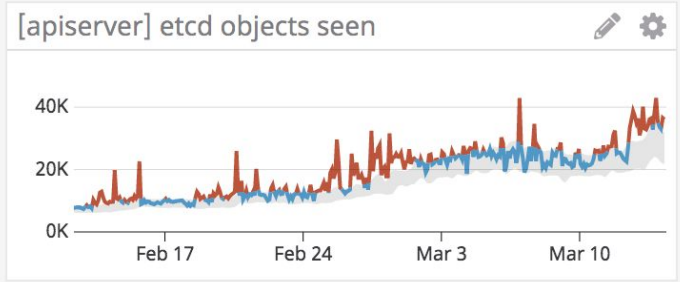
Sometimes it Breaks

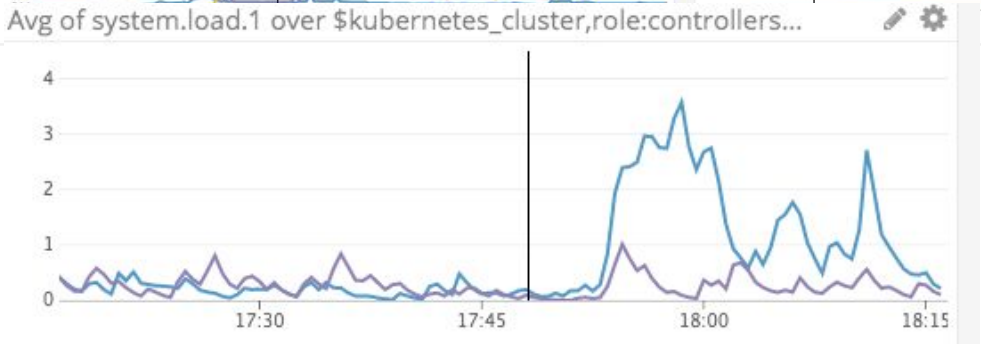


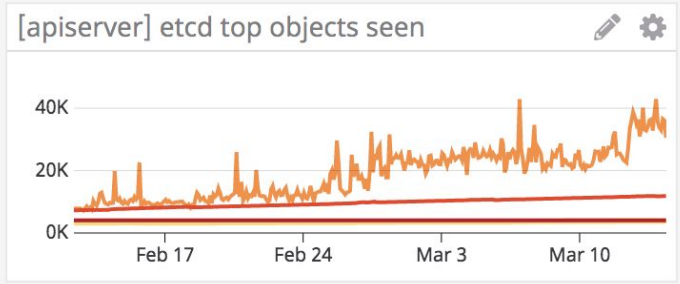












Why are we here?

Lessons learned from incidents with Kubernetes in production!

And especially the self inflicted ones.

#1

It's ~~never~~ always DNS.

#1.1

It's ~~never~~ always DNS.

Classic Kubernetes DNS

resolv.conf

```
search     <namespace>.svc.cluster.local
           svc.cluster.local
           cluster.local
           ec2.internal

options    ndots:5
```

3+ search domains
ndots : 5

Classic Kubernetes DNS

resolv.conf

```
search     <namespace>.svc.cluster.local
           svc.cluster.local
           cluster.local
           ec2.internal

options    ndots:5
```

3+ search domains
ndots : 5

www.google.com?

1: www.google.com.<namespace>.svc.cluster.local	**A? / AAAA?**	**NXDOMAIN**
2: www.google.com.svc.cluster.local	**A? / AAAA?**	**NXDOMAIN**
3: www.google.com.cluster.local	**A? / AAAA?**	**NXDOMAIN**
4: www.google.com.google.internal	**A? / AAAA?**	**NXDOMAIN**
5: www.google.com	**A? / AAAA?**	**NOERROR**

Coredns autopath

Query

www.google.com.<namespace>.svc.cluster.local **A? / AAAA?**

Coredns with autopath option

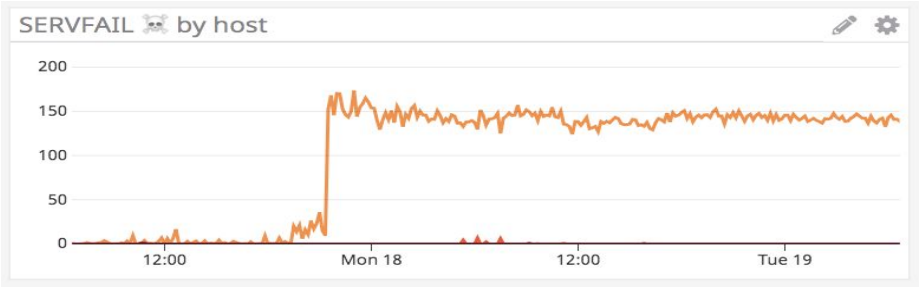
Remove “<namespace>.svc.cluster.local “
Try to find the proper answer

Response

CNAME www.google.com, A: X.X.X.X

=> One DNS query instead of 5

DNS is broken



Symptoms

DNS failure for some applications

Cause

- Rate limited by the upstream

DNS is broken

Symptoms

DNS failure for some applications

Cause

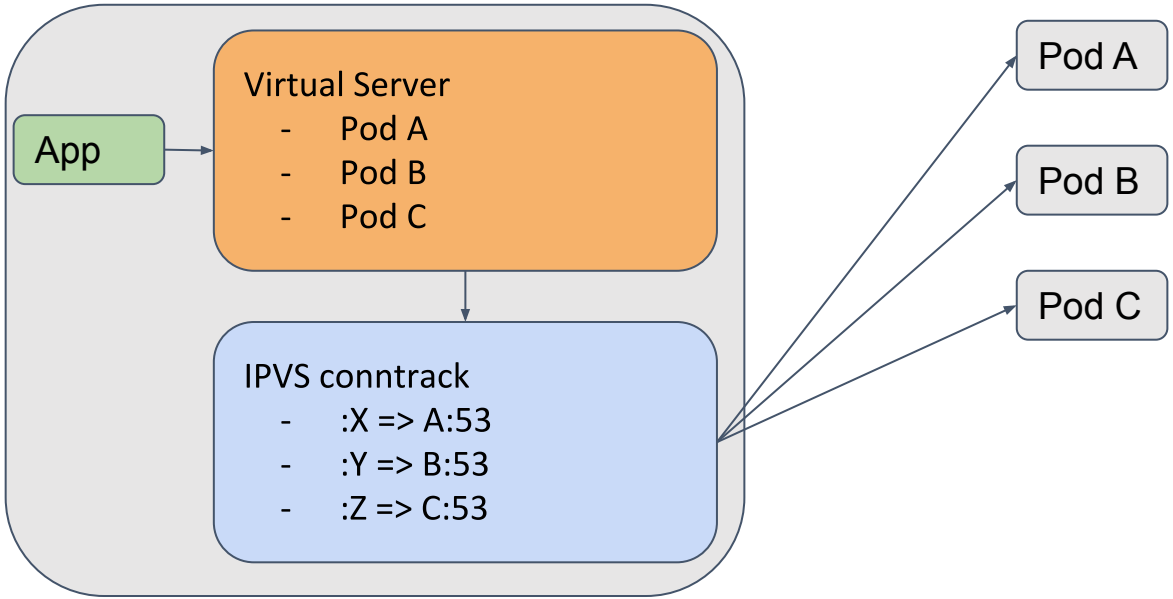
- Rate limited by the upstream
- Autopath enabled
- Sudden upstream queries increase
- Autopath prevents caching

#1.2

It's ~~never~~ always DNS.

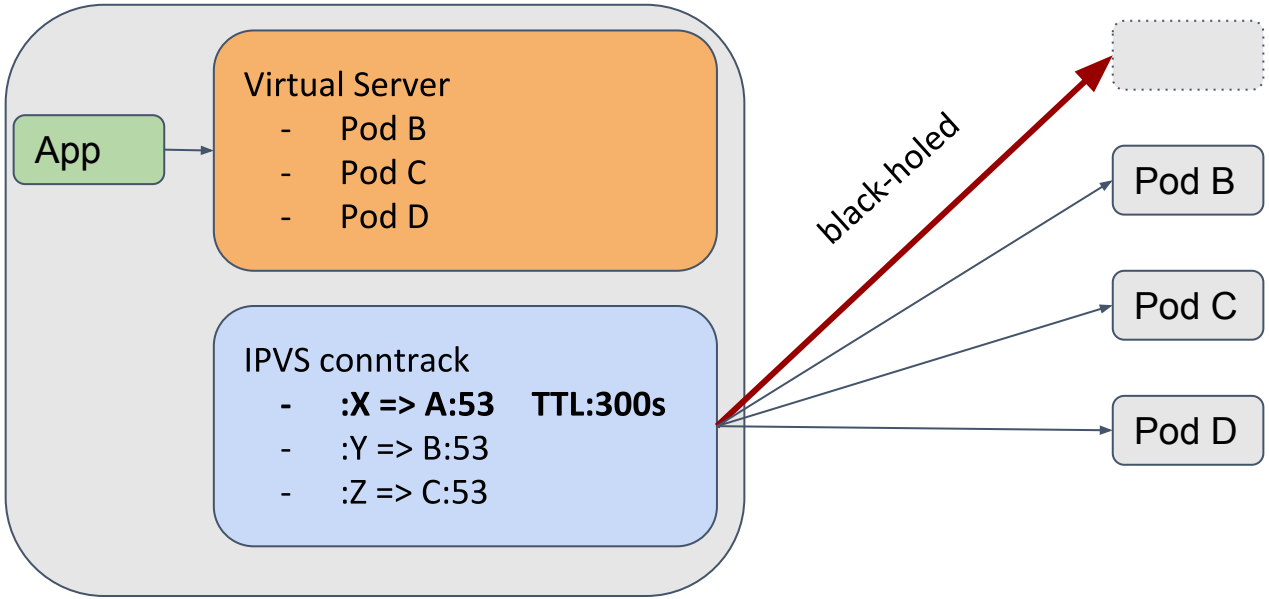
DNS is broken #2

Coredns rolling update and IPVS



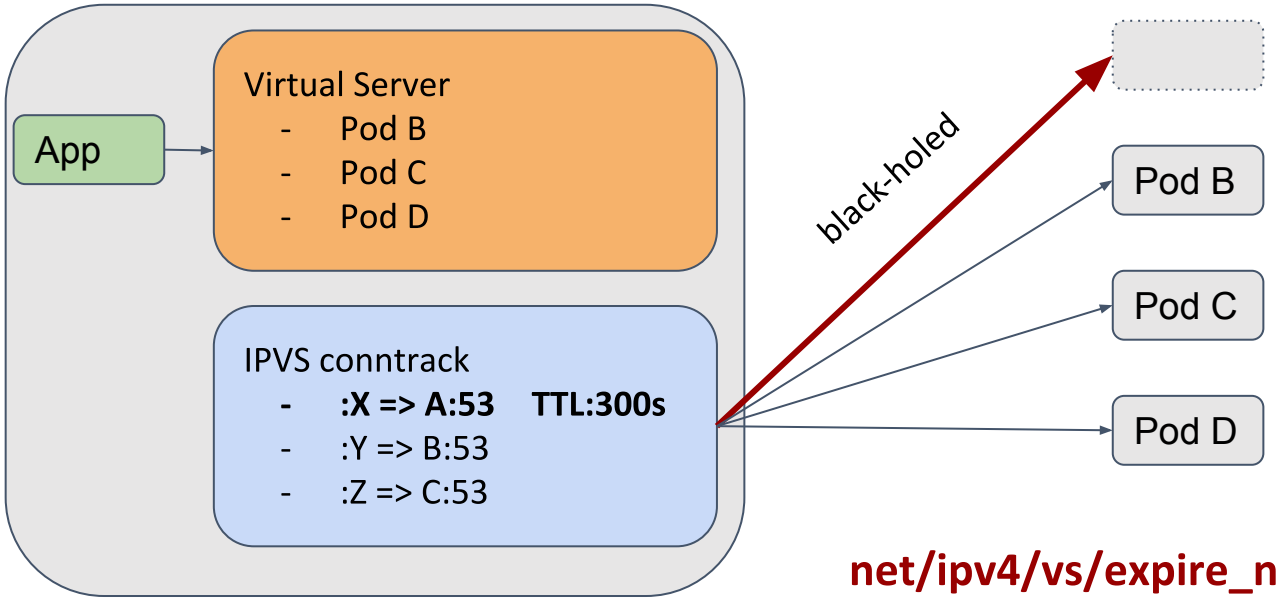
DNS is broken #2

Under high load ports are reused faster than they expire



DNS is broken #2

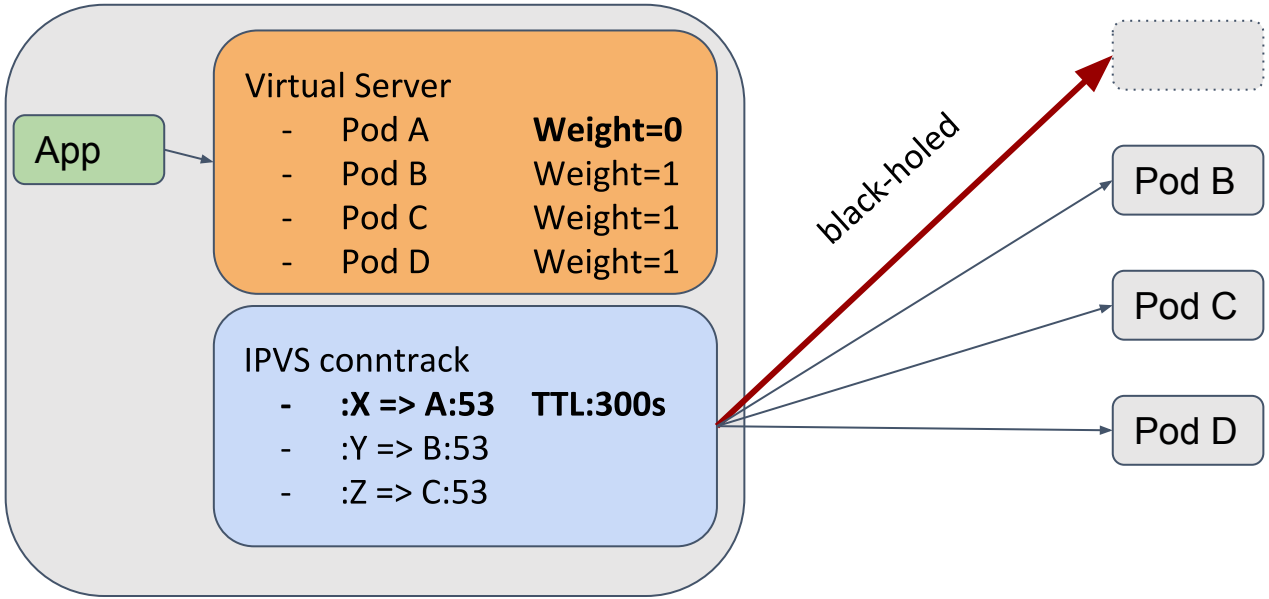
Under high load ports are reused faster than they expire



net/ipv4/vs/expire_nodest_conn

DNS is broken #3

Graceful termination?



DNS is broken #2

- Before graceful termination
  - Backend removed
  - Traffic reusing port is blackholed

- After graceful termination
  - Under high-load port are reused fast
  - The backend pod is never removed
  - When the pod terminates, black-holed
  - Fix: https://github.com/kubernetes/kubernetes/pull/77802

#2

Jobs are not starting, image pulls fail.

```
     52 Completed
      1 ErrImagePull
    155 Evicted
      7 ImagePullBackOff
      7 Init:ErrImagePull
      4 Init:Error
     49 Init:ImagePullBackOff
     16 Pending
   1267 Running
      2 Terminating
```

```
     52 Completed
      1 ErrImagePull
    155 Evicted
      7 ImagePullBackOff
      7 Init:ErrImagePull
      4 Init:Error
     49 Init:ImagePullBackOff
     16 Pending
   1267 Running
      2 Terminating
```

Image Stampede

Number of image pulls sharply increases, sustained for several hours.

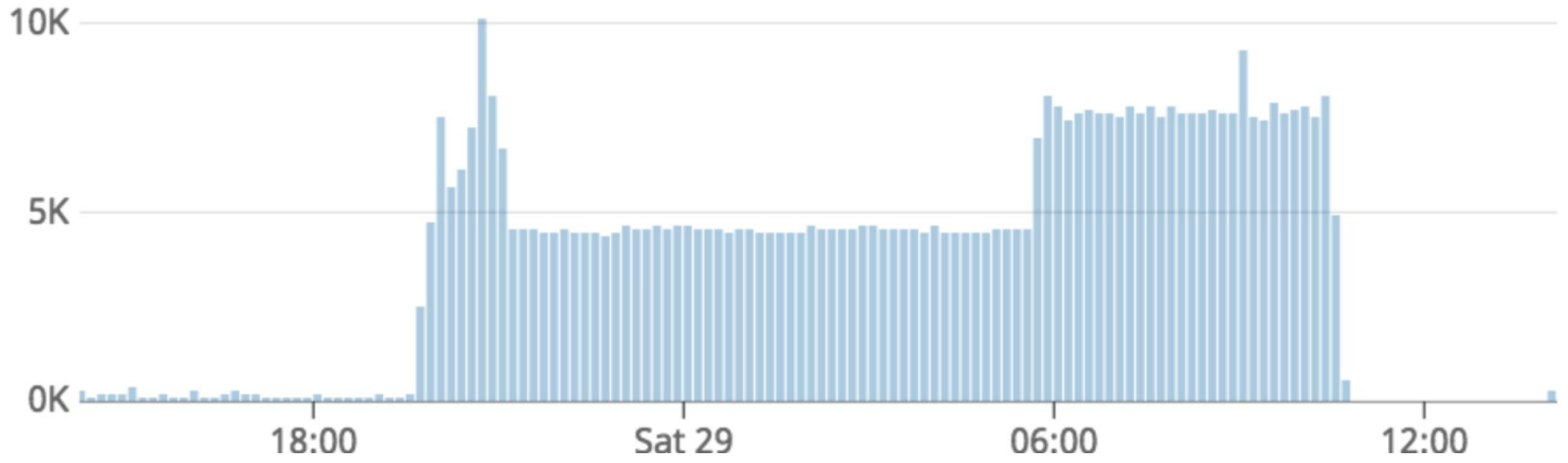


Image Stampede

Number of image pulls sharply increases, sustained for several hours.

Then a sudden change.

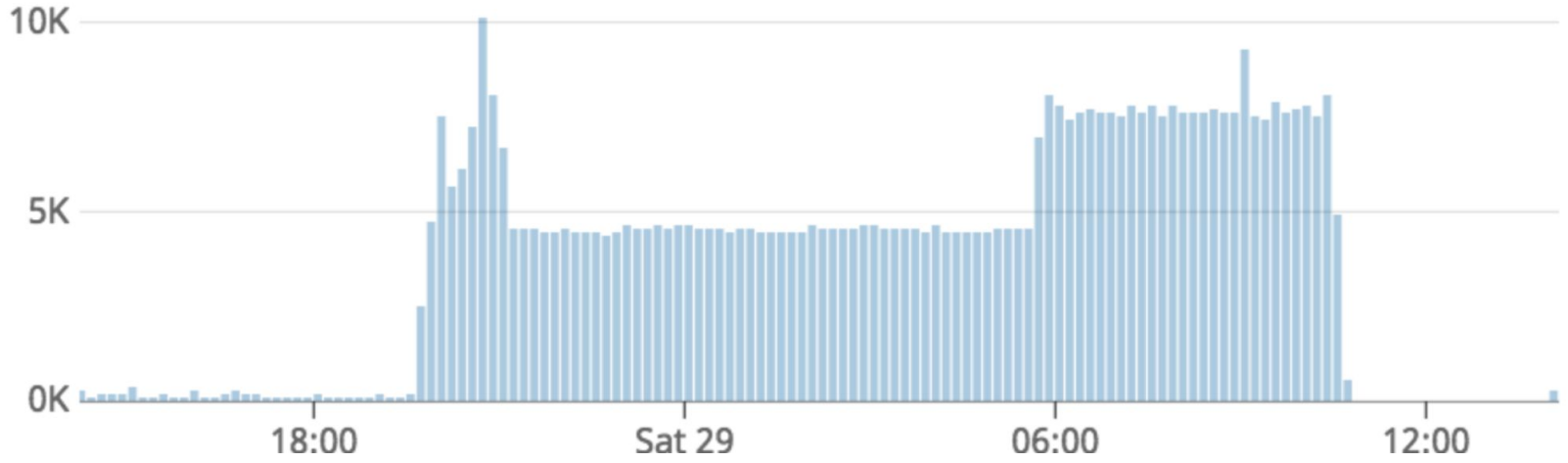


Image Stampede

Number of image pulls sharply increases, sustained for several hours.

Then a sudden change.

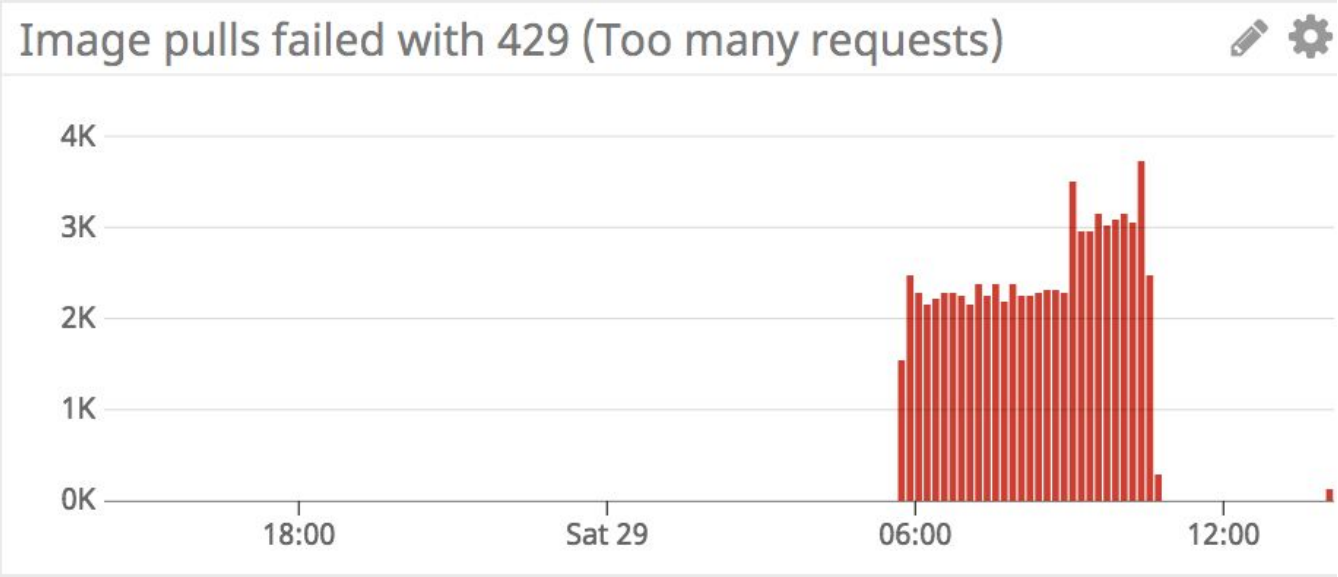


Image Stampede

- Permission change on bucket
- DaemonSet in CrashLoopBackoff

Image Stampede

- Permission change on bucket
- DaemonSet in CrashLoopBackoff
- **imagePullPolicy: Always**

Image Stampede

- Permission change on bucket
- DaemonSet in CrashLoopBackoff
- **imagePullPolicy: Always**
- ~1000 pods pulling through 3 NAT instances
- Daily quota reached for NAT IPs
- All NAT instances are impacted

Image Stampede, follow-up #1

- Replaced the impacted NAT instances

Image Stampede, follow-up #1

- Replaced the impacted NAT instances
- Apps couldn’t connect to CloudSQL anymore...

Image Stampede, follow-up #2

- Admission webhook denying “latest” tag

Image Stampede, follow-up #2

- Admission webhook denying “latest” tag
- Applied to
  - Deployments
  - StatefulSets
  - DaemonSets
  - Jobs
  - Pods

Image Stampede, follow-up #2

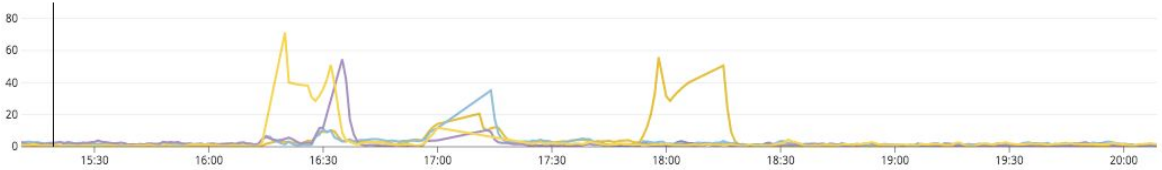
- Admission webhook denying “latest” tag
- Applied to
  - Deployments
  - StatefulSets
  - DaemonSets
  - Jobs
  - Pods
- Controllers can’t create pods for existing workloads

3

I can't kubectl

API server DDOS

Load on apiservers

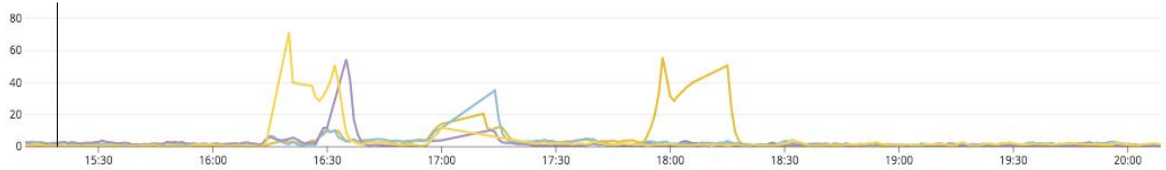


Symptoms: apiservers unresponsive

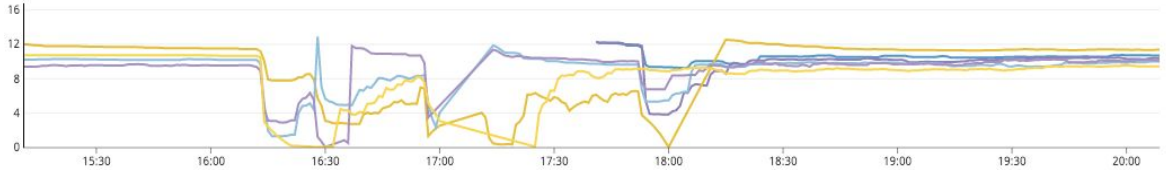
> load => 50 (on 8 core nodes)

API server DDOS

Load on apiservers



Usable memory



Symptoms: apiservers unresponsive

> load => 50 (on 8 core nodes)

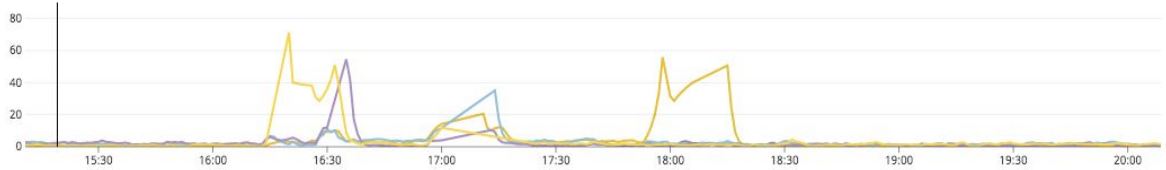
> Free memory => 0

> apiservers OOM killed

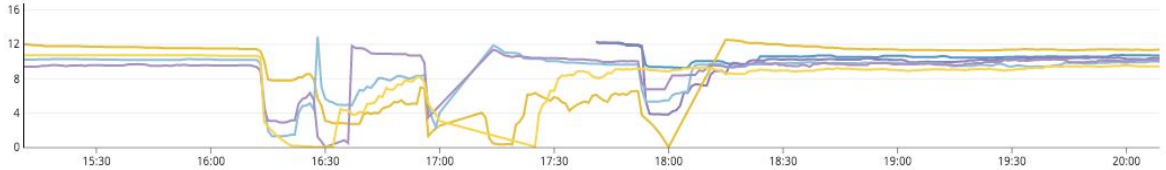
API server DDOS

KubeCon CloudNativeCon Europe 2019

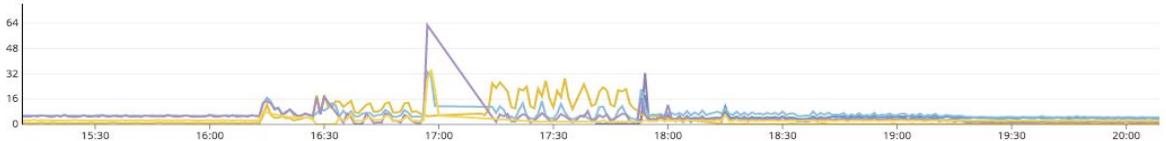
Load on apiservers



Usable memory



Outgoing traffic

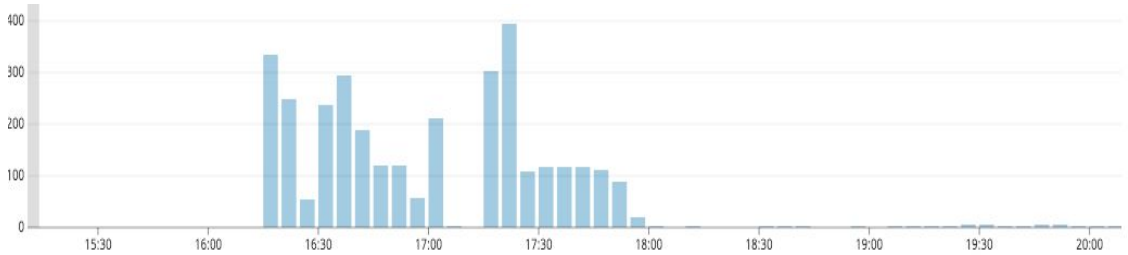


Symptoms: apiservers unresponsive

> load => 50 (on 8 core nodes)
> Free memory => 0
> apiservers OOM killed
> network traffic much higher

API server DDOS

kube2iam pod restarts

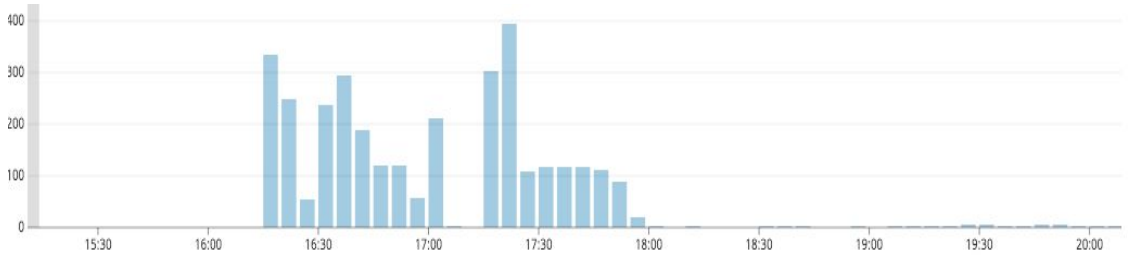


Seemed related to kube2iam update

> Lots of restarts

API server DDOS

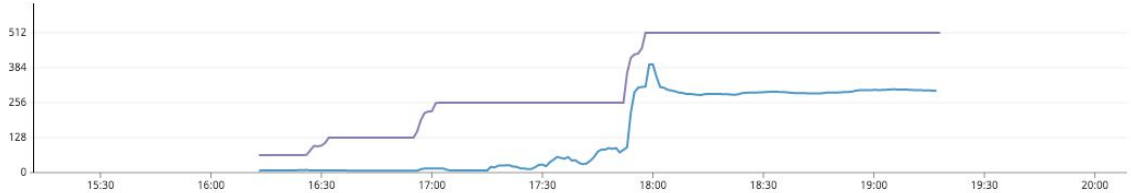
kube2iam pod restarts



Seemed related to kube2iam update

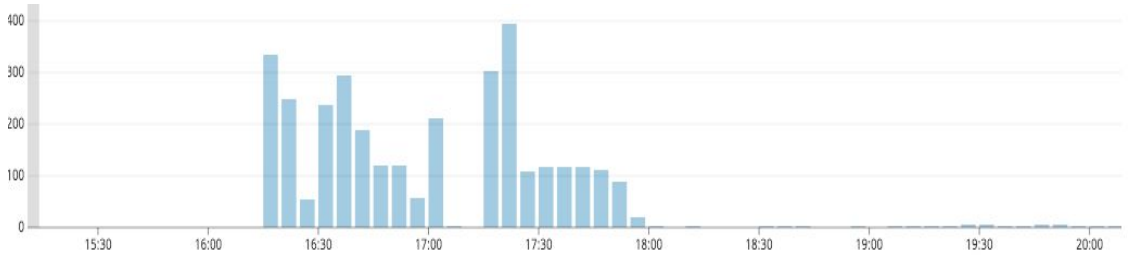
> Lots of restarts
> cgroup OOM-killer
> Memory usage much higher
> Increase limit

kube2iam memory usage / limit

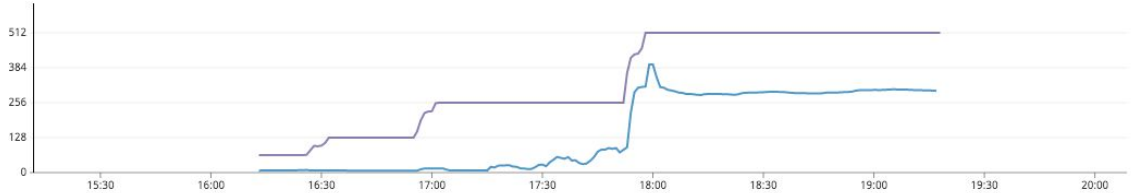


API server DDOS

kube2iam pod restarts



kube2iam memory usage / limit



Seemed related to kube2iam update

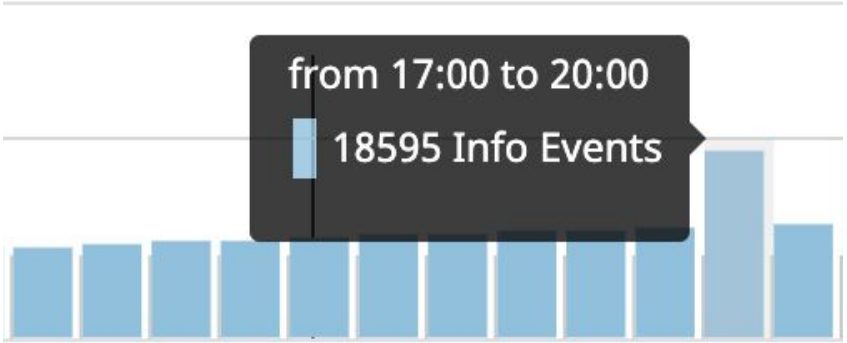
> Lots of restarts
> cgroup OOM-killer
> Memory usage much higher
> Increase limit

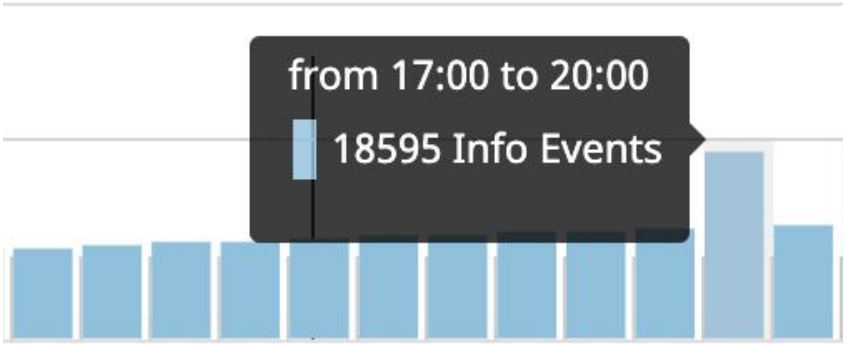
Cause?

> Patch in kube2iam
> Small typo
> kube2iam pods syncing all pods
> Broke apiservers + kube2iam

#4

New nodes aren't scheduling application pods.

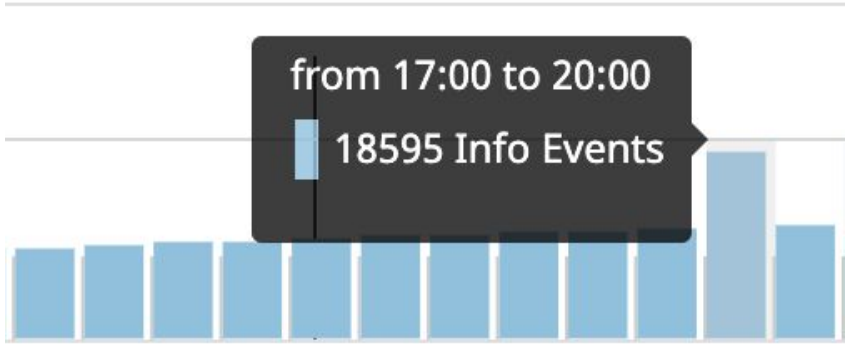

from 17:00 to 20:00
18595 Info Events



Events from the [redacted]276hh Pod

186 **FailedScheduling**: 0/2125 nodes are available: 11 node(s) were unschedulable, 1263 Insufficient memory, 1354 Insufficient cpu, 2111 node(s) didn't match node selector.

Events emitted by the default-scheduler seen at 2019-05-20 11:58:01 +0000 UTC



Events from the [redacted]276hh Pod

186 **FailedScheduling**: 0/2125 nodes are available: 11 node(s) were unschedulable, 1263 Insufficient memory, 1354 Insufficient cpu, 2111 node(s) didn't match node selector.

Events emitted by the default-scheduler seen at 2019-05-20 11:58:01 +0000 UTC

Events from the [redacted]-276hh Pod

10 **NotTriggerScaleUp**: pod didn't trigger scale-up (it wouldn't fit if a new node is added): 146 node(s) didn't match node selector, 65 Insufficient cpu, 54 Insufficient memory

Events emitted by the cluster-autoscaler seen at 2019-05-20 11:59:21 +0000 UTC

Scheduling Faux Pas

Scheduling at Datadog:

- Single tenancy on a node.
- Resources match node type.
- Minus DaemonSet reserved.

```
nodegroup.yaml
---
instanceType: c5.4xlarge
```

```
deployment.yaml
---
resources:
  requests:
    cpu: 15
    memory: 30Gi
  limits:
    cpu: 15
    memory: 30Gi
```

Scheduling Faux Pas

- Added a DaemonSet with resource requests.
- Scheduler couldn’t fit applications on nodes.

Scheduling Faux Pas

- Added a DaemonSet with resource requests.
- Scheduler couldn’t fit applications on nodes.
- **Even worse**: the DaemonSet had a critical PodPriority.

Scheduling Faux Pas

- Added a DaemonSet with resource requests.
- Scheduler couldn’t fit applications on nodes.
- **Even worse**: the DaemonSet had a critical PodPriority.
- **Lucky**: the cluster was running k8s 1.10.

Scheduling Faux Pas

- Added a DaemonSet with resource requests.
- Scheduler couldn't fit applications on nodes.
- **Even worse**: the DaemonSet had a critical PodPriority.
- **Lucky**: the cluster was running k8s 1.10.

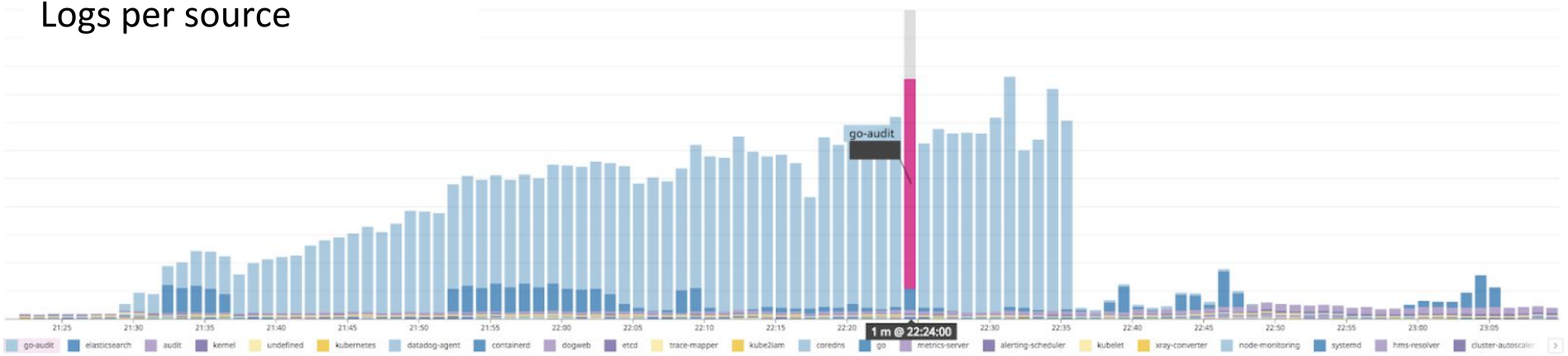
On newer clusters, applications would have been evicted!

#5

Log intake volume just increased by 10x.

#5: Kernel Audit DDOS

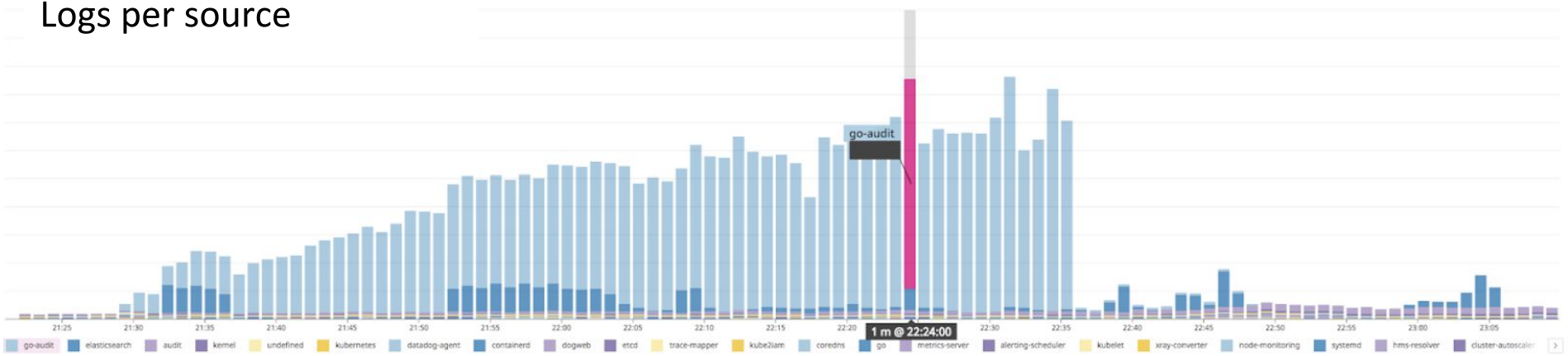
Logs per source



For this account, log volume did x20 in 30mn

#5: Kernel Audit DDOS

Logs per source



For this account, log volume did x20 in 30mn

> Enabled audit with a daemonset
> Nodes running Kubernetes generate a huge amount of audit logs (exec, clones, iptables…)

#6

Where did my pods go?

HorizontalPodAutoscaler


```
apiVersion: apps/v1
kind: Deployment
spec:
  replicas: 60
```

HorizontalPodAutoscaler


```
apiVersion: apps/v1
kind: Deployment
spec:
  replicas: 60
```

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: myapp
  minReplicas: 1
  maxReplicas: 10
targetCPUUtilizationPercentage: 50
```

HorizontalPodAutoscaler


```
apiVersion: apps/v1
kind: Deployment
spec:
  replicas: 60
```

Controller manages replica count of deployment based on the value of a metric.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: myapp
  minReplicas: 1
  maxReplicas: 10
targetCPUUtilizationPercentage: 50
```

HorizontalPodAutoscaler

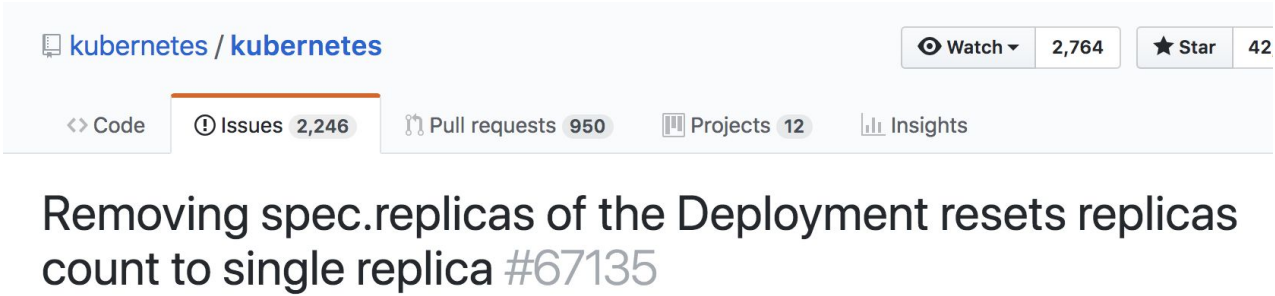

```
apiVersion: apps/v1
kind: Deployment
spec:
  replicas: 60
```

Controller manages replica count of deployment based on the value of a metric.

Must remove explicit replica count!

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: myapp
  minReplicas: 1
  maxReplicas: 10
targetCPUUtilizationPercentage: 50
```

Scale to One


kubernetes / kubernetes
Watch 2,764
Star 42
Code
Issues 2,246
Pull requests 950
Projects 12
Insights
Removing spec.replicas of the Deployment resets replicas count to single replica #67135

#7

There's ghosts in the Cassandra cluster.

#7: Cassandra broke

- 100+ nodes Cassandra cluster
- Deployed fine
- Broken the following morning

#7: Cassandra broke

- 100+ nodes Cassandra cluster
- Deployed fine
- Broken the following morning

> 25% pods pending: “Volume affinity issue”

#7: Cassandra broke

- 100+ nodes Cassandra cluster
- Deployed fine
- Broken the following morning

> 25% pods pending: “Volume affinity issue”

> 25% nodes had been deleted + local volumes bound to deleted nodes

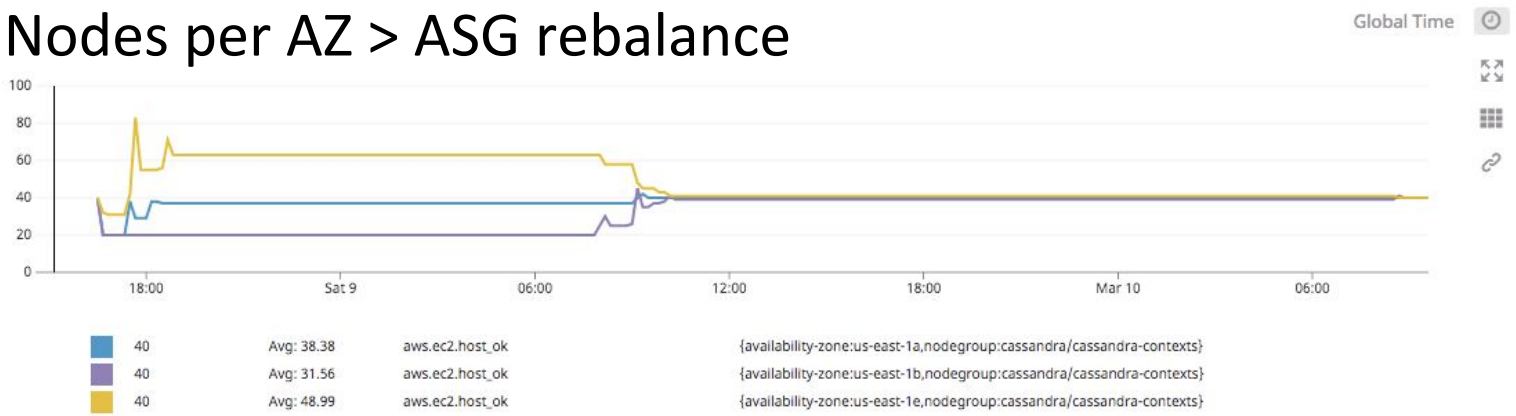
#7: Cassandra broke

- 100+ nodes Cassandra cluster
- Deployed fine
- Broken the following morning

> 25% pods pending: “Volume affinity issue”

> 25% nodes had been deleted + local volumes bound to deleted nodes

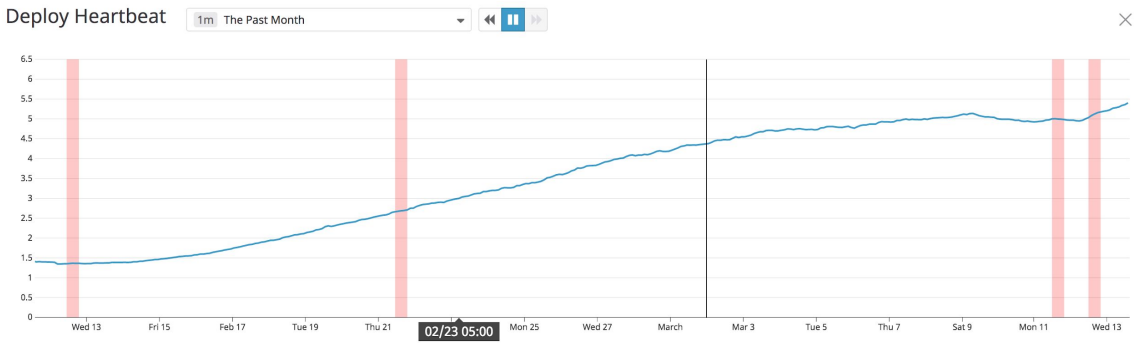
Nodes per AZ > ASG rebalance



8

Slow deploy heartbeat

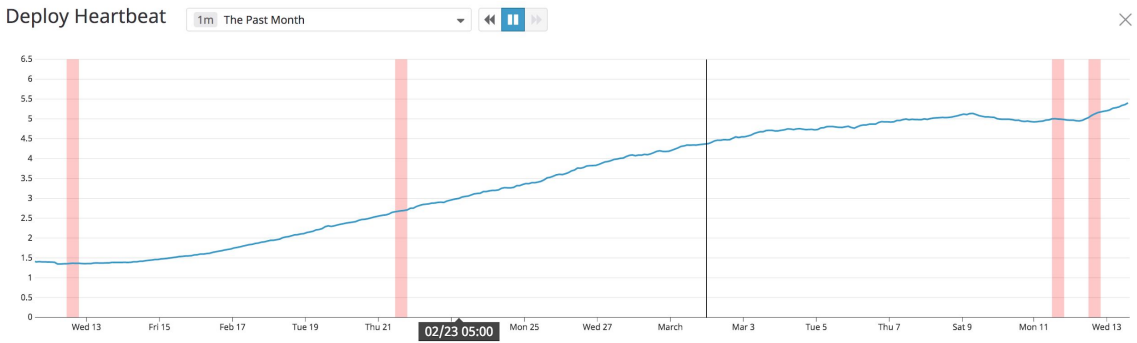
Slow deployments

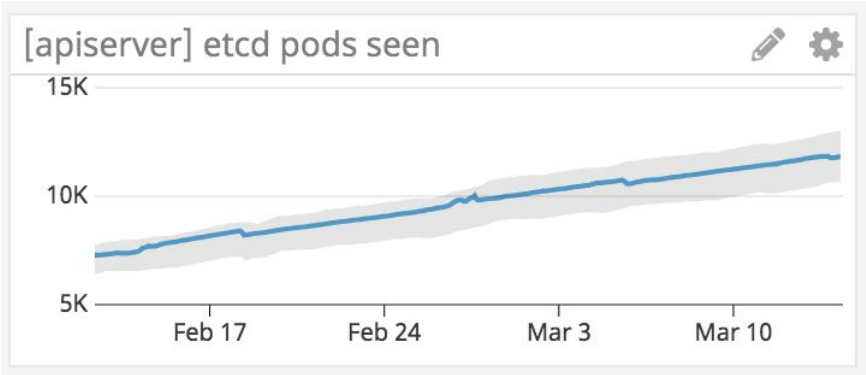


Symptoms

Deployments getting slower

Slow deployments





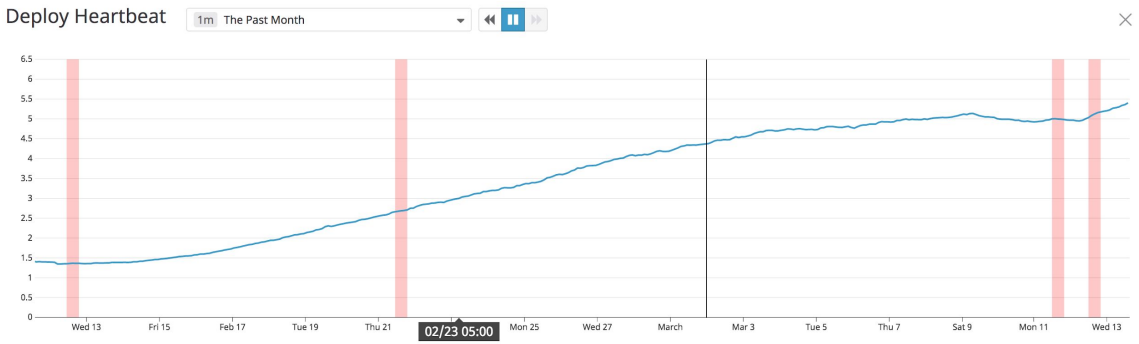
Symptoms

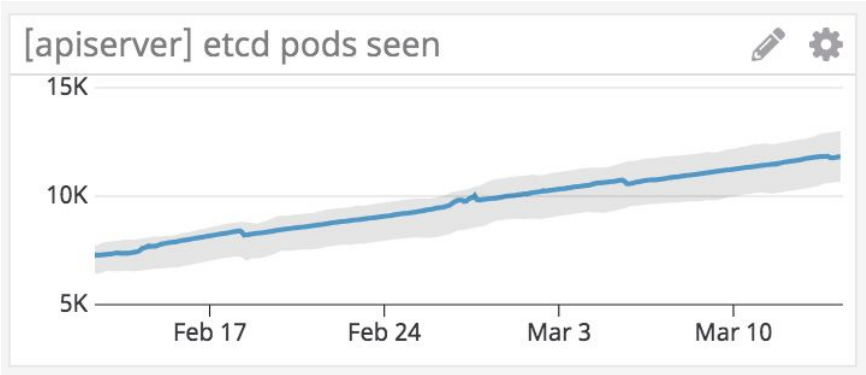
Deployments getting slower

Cause

4000 pending pods

Slow deployments





Symptoms

Deployments getting slower

Cause

4000 pending pods
cronjob (*/10) with wrong toleration
Scheduling loop having a hard time

#9

“Contained”

9.1

“Contained”

Broken runtime

Broken runtime #1 : Zombies


```
root      8502  0.7  0.0  11032  6200 ?        Sl   16:39   0:01  \_ containerd-shim -namespace k8s.io -workdir
/var/lib/containerd/io.containerd.runtime.v1.linux/k8s.io/0eacd7463b319a9f8423f927
root      8520  0.4  0.0  46396  5768 ?        Ssl  16:39   0:00      \_ redis-server *:6379
root     10791  0.0  0.0      0     0 ?        Z    16:39   0:00      |   \_ [server_readines] <defunct>
root     11632  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     12222  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [server_readines] <defunct>
root     13102  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     14115  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     14500  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     14893  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [server_readines] <defunct>
root     15309  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     16232  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     16895  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [redis-cli] <defunct>
root     17248  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     17876  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     18512  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     21932  0.0  0.0      0     0 ?        Z    16:42   0:00      |   \_ [server_readines] <defunct>
root     22648  8.5  0.0  22320  5756 ?        Rs   16:42   0:00      \_ /bin/bash /usr/local/bin/server_readiness_probe.sh
```

Broken runtime #1 : Zombies


```
root      8502  0.7  0.0  11032  6200 ?        Sl   16:39   0:01  \_ containerd-shim -namespace k8s.io -workdir
/var/lib/containerd/io.containerd.runtime.v1.linux/k8s.io/0eacd7463b319a9f8423f927
root      8520  0.4  0.0  46396  5768 ?        Ssl  16:39   0:00      \_ redis-server *:6379
root     10791  0.0  0.0      0     0 ?        Z    16:39   0:00      |   \_ [server_readines] <defunct>
root     11632  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     12222  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [server_readines] <defunct>
root     13102  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     14115  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     14500  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     14893  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [server_readines] <defunct>
root     15309  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     16232  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     16895  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [redis-cli] <defunct>
root     17248  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     17876  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     18512  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     21932  0.0  0.0      0     0 ?        Z    16:42   0:00      |   \_ [server_readines] <defunct>
root     22648  8.5  0.0  22320  5756 ?        Rs   16:42   0:00      \_ /bin/bash /usr/local/bin/server_readiness_probe.sh
```

```
ps auxf | grep -c defunct
16018
```

Broken runtime #1 : Zombies


```
root      8502  0.7  0.0  11032  6200 ?        Sl   16:39   0:01  \_ containerd-shim -namespace k8s.io -workdir
/var/lib/containerd/io.containerd.runtime.v1.linux/k8s.io/0eacd7463b319a9f8423f927
root      8520  0.4  0.0  46396  5768 ?        Ssl  16:39   0:00      \_ redis-server *:6379
root     10791  0.0  0.0      0     0 ?        Z    16:39   0:00      |   \_ [server_readines] <defunct>
root     11632  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     12222  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [server_readines] <defunct>
root     13102  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     14115  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     14500  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     14893  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [server_readines] <defunct>
root     15309  0.0  0.0      0     0 ?        Z    16:40   0:00      |   \_ [redis-cli] <defunct>
root     16232  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     16895  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [redis-cli] <defunct>
root     17248  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     17876  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     18512  0.0  0.0      0     0 ?        Z    16:41   0:00      |   \_ [server_readines] <defunct>
root     21932  0.0  0.0      0     0 ?        Z    16:42   0:00      |   \_ [server_readines] <defunct>
root     22648  8.5  0.0  22320  5756 ?        Rs   16:42   0:00      \_ /bin/bash /usr/local/bin/server_readiness_probe.sh
```

```
ps auxf | grep -c defunct
16018
```

```
readinessProbe:
  exec:
  command: [server_readiness_probe.sh]
  timeoutSeconds: 1
```

Broken runtime #2

Laurent Bernaille @lbernail · Sep 26

Some days you know things are going to be weird^Winteresting:

cat /proc/28019/wchan

__refrigerator

Broken runtime #2

Laurent Bernaille @lbernail · Sep 26

Some days you know things are going to be weird^Winteresting:

cat /proc/28019/wchan

__refrigerator

- Blocked io (nvme issue)
- Deletion starts with cgroup freeze
- Freezing was hanging

#9.2

“Contained”

Performance issues

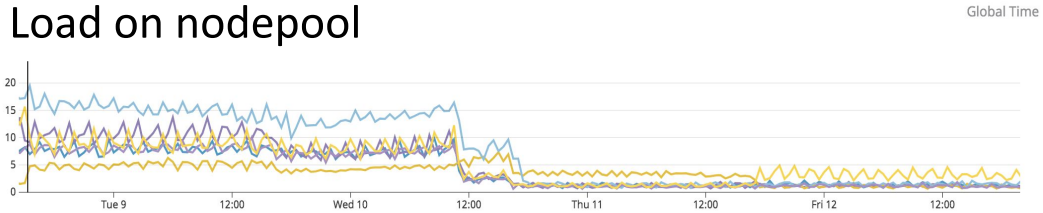
Perf issues: slow deployments

Symptoms

- Deployment slow
- Pod took 1+mn to start

Perf issues: slow deployments

Load on nodepool



Symptoms

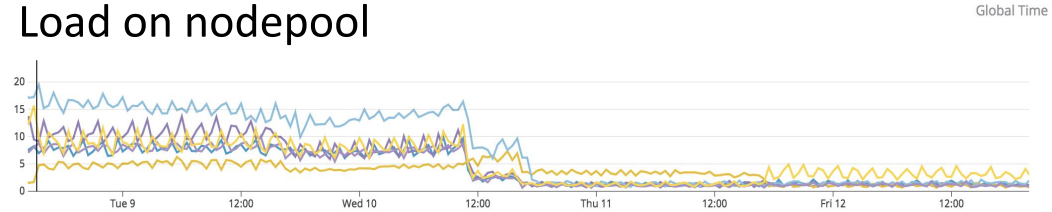
- Deployment slow
- Pod took 1+mn to start

Cause

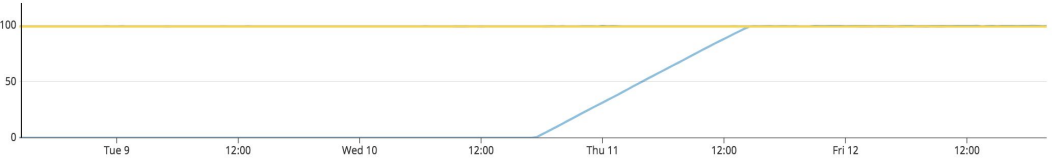
- Load on nodes running deployment high

Perf issues: slow deployments

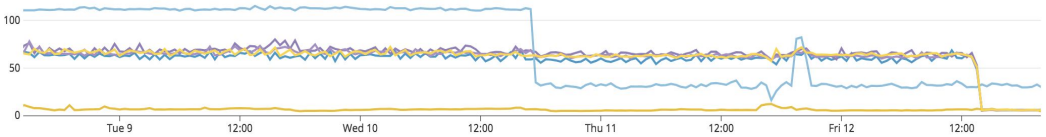
Load on nodepool



EBS burst balance



Write iops



Symptoms

- Deployment slow
- Pod took 1+mn to start

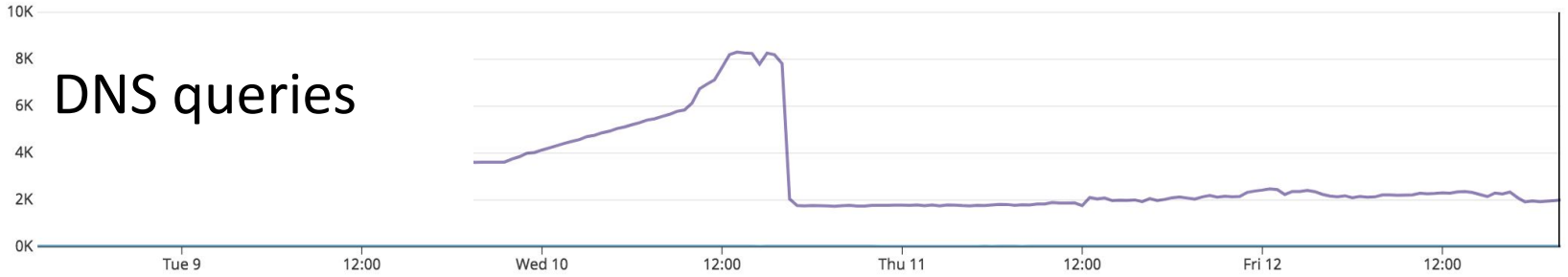
Cause

- Load on nodes running deployment high
- Because IOs are being rate limited

Why??


```
sum:coredns.request_count{kubernetes_cluster:chinook,nodegroup:datadog-system_system} by {proto}.as_rate()
```

Global Time

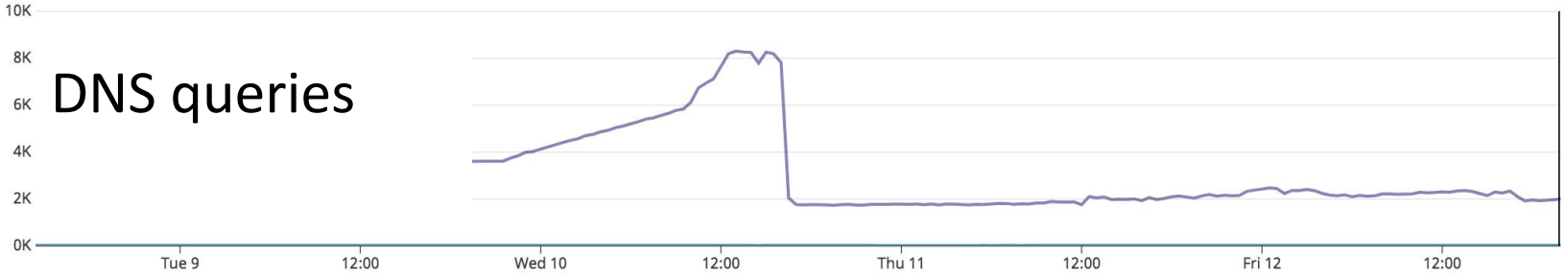


DNS queries

Why??


```
sum:coredns.request_count{kubernetes_cluster:chinook,nodegroup:datadog-system_system} by {proto}.as_rate()
```

Global Time

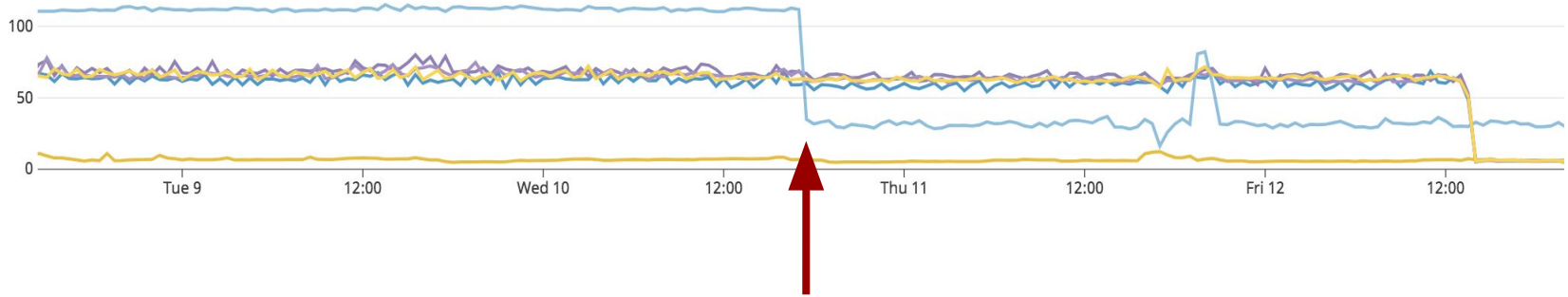


- Nodes were running coredns pods
- An app started doing 5000+ dns queries per second
- coredns logs filled the disk

Fix #1


```
sum:system.io.w_s{kubernetes_cluster:chinook,nodegroup:datadog-system_system,device:nvme0n1p1} by {host}
```

Global Time

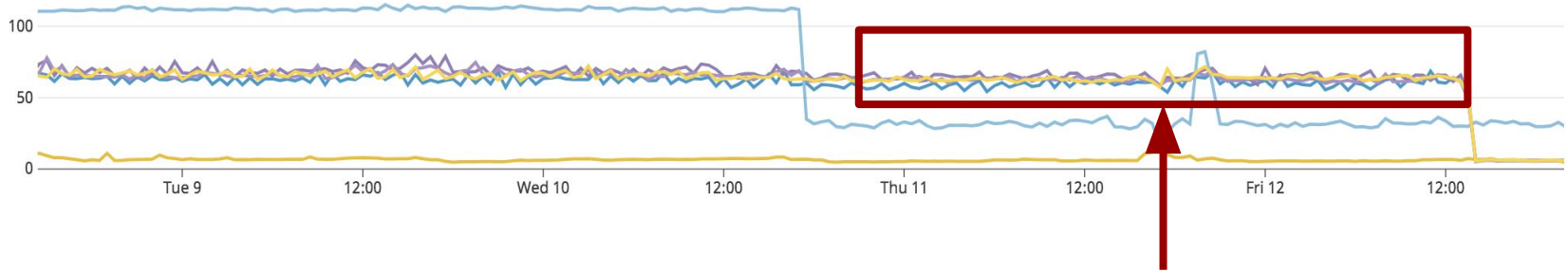


Change app to use local DNS cache

IOs still too high


```
sum:system.io.w_s{kubernetes_cluster:chinook,nodegroup:datadog-system_system,device:nvme0n1p1} by {host}
```

Global Time

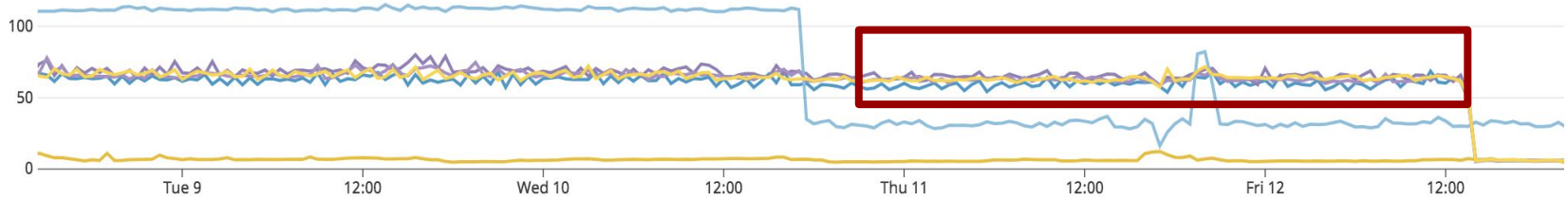


What is this???

IOs still too high


```
sum:system.io.w_s{kubernetes_cluster:chinook,nodegroup:datadog-system_system,device:nvme0n1p1} by {host}
```

Global Time



Remember audit?

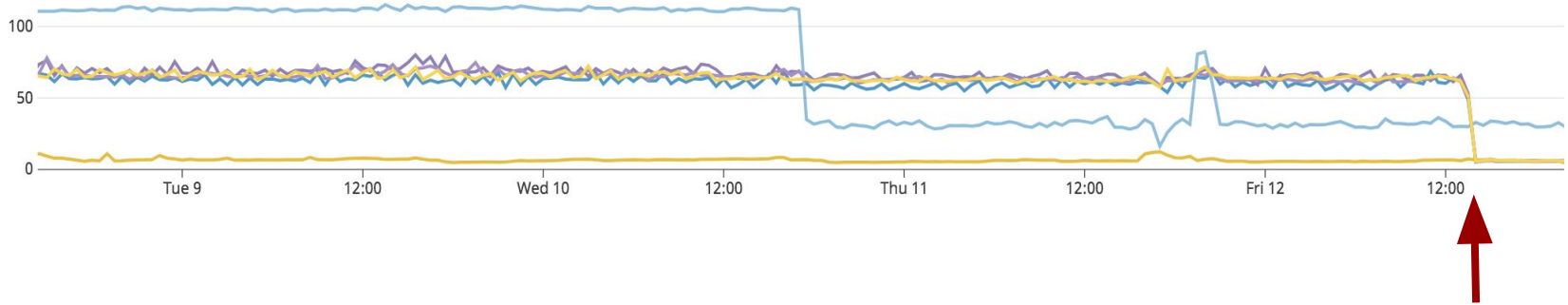
Daemonset had been removed, but audit was still enabled

- We dropped traffic at log intake
- But we were still writing to disk...

Fix #2


```
sum:system.io.w_s{kubernetes_cluster:chinook,nodegroup:datadog-system_system,device:nvme0n1p1} by {host}
```

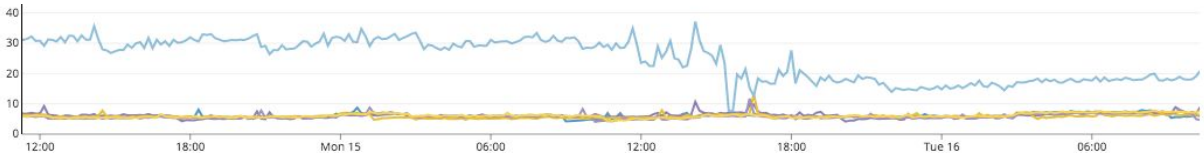
Global Time



Disable audit **for real**

Weird outlier

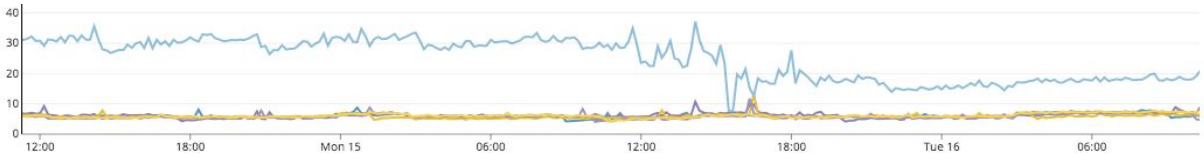
iops by node in the group



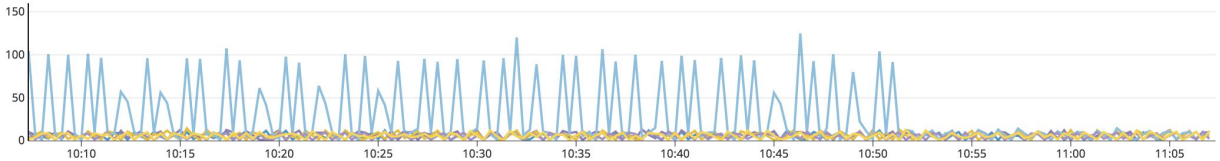
One node still had more iops

Weird outlier

iops by node in the group



iops by node in the group (zoomed)

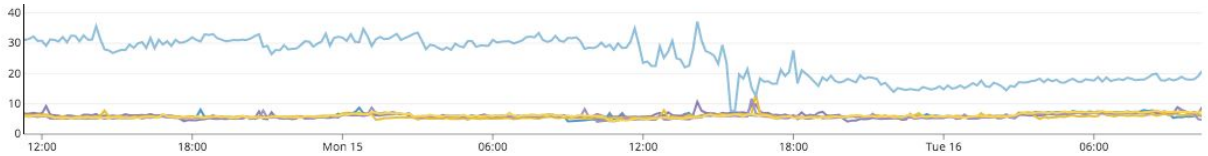


One node still had more iops

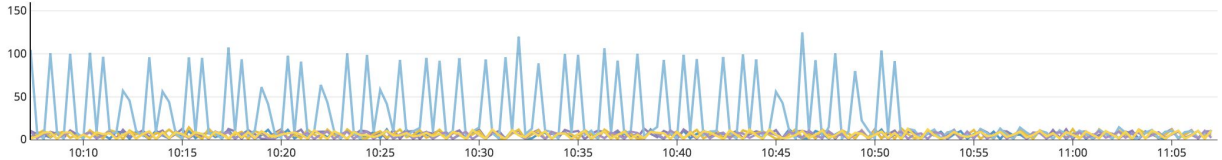
> io spikes every minute

Weird outlier

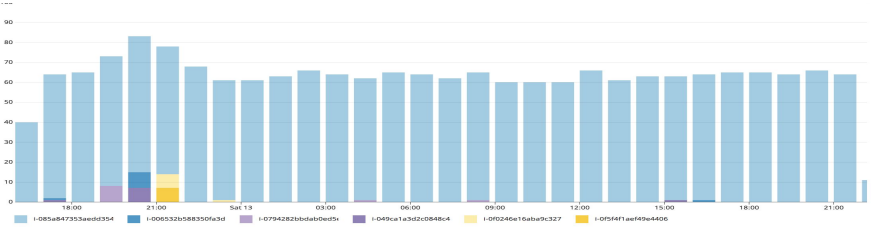
iops by node in the group



iops by node in the group (zoomed)



containers created by node



One node still had more iops

> io spikes every minute
> Single host get all pod for a cronjob
> Suspending it => 0
> Changing the job to sleep => 0

Why is the job using so much IOs???

Job

```
ips = $(aws ec2 describe-instances --filter=consul)
kubectl update endpoints consul $ips
```

Why is the job using so much IOs???

Job

```
ips = $(aws ec2 describe-instances --filter=consul)
kubectl update endpoints consul $ips
```

$ kubectl get componentstatuses
$ find $HOME/.kube/ |wc -l
163

$ kubectl -v=8 get componentstatuses | grep "GET https" -c
45

#10

“Graceful” Termination

“Graceful” Termination

- Queue consumer autoscaled on queue depth
- On scale down, job must finish (hours)

“Graceful” Termination

- Queue consumer autoscaled on queue depth
- On scale down, job must finish (hours)
  - Pod enters Terminating state

“Graceful” Termination

- Queue consumer autoscaled on queue depth
- On scale down, job must finish (hours)
  - Pod enters Terminating state
    - kube2iam refuses to refresh credentials

“Graceful” Termination

- Queue consumer autoscaled on queue depth
- On scale down, job must finish (hours)
  - Pod enters Terminating state
    - kube2iam refuses to refresh credentials
      - ✔️ Fixed upstream

“Graceful” Termination

- Queue consumer autoscaled on queue depth
- On scale down, job must finish (hours)
  - Pod enters Terminating state
    - kube2iam refuses to refresh credentials
      - ✔️ Fixed upstream
    - Kubelet restart cancels context

“Graceful” Termination

- Queue consumer autoscaled on queue depth
- On scale down, job must finish (hours)
  - Pod enters Terminating state
    - kube2iam refuses to refresh credentials
      - ✔️Fixed upstream
    - Kubelet restart cancels context
      - Application gets SIGKILL’ed

“Graceful” Termination

- Queue consumer autoscaled on queue depth
- On scale down, job must finish (hours)
  - Pod enters Terminating state
    - kube2iam refuses to refresh credentials
      - ✔️Fixed upstream
    - Kubelet restart cancels context
      - Application gets SIGKILL’ed
      - Must restart kubelet for cert rotation, so very likely over 48h timeout

“Graceful” Termination

- Queue consumer autoscaled on queue depth
- On scale down, job must finish (hours)
  - Pod enters Terminating state
    - kube2iam refuses to refresh credentials
      - ✔️Fixed upstream
    - Kubelet restart cancels context
      - Application gets SIGKILL’ed
      - Must restart kubelet for cert rotation, so very likely over 48h timeout
      - ✔️Fixed upstream

Key take-aways

1. Careful with Daemonsets
2. DNS is hard
3. Cloud infra not transparent
4. Containers not really contained

KubeCon | CloudNativeCon

We're hiring!

Come join us (and break k8s!)

https://www.datadoghq.com/careers/

@lbernail laurent@datadoghq.com
@roboll_ roboll@datadoghq.com