10 Weird Ways to Blow Up Your Kubernetes

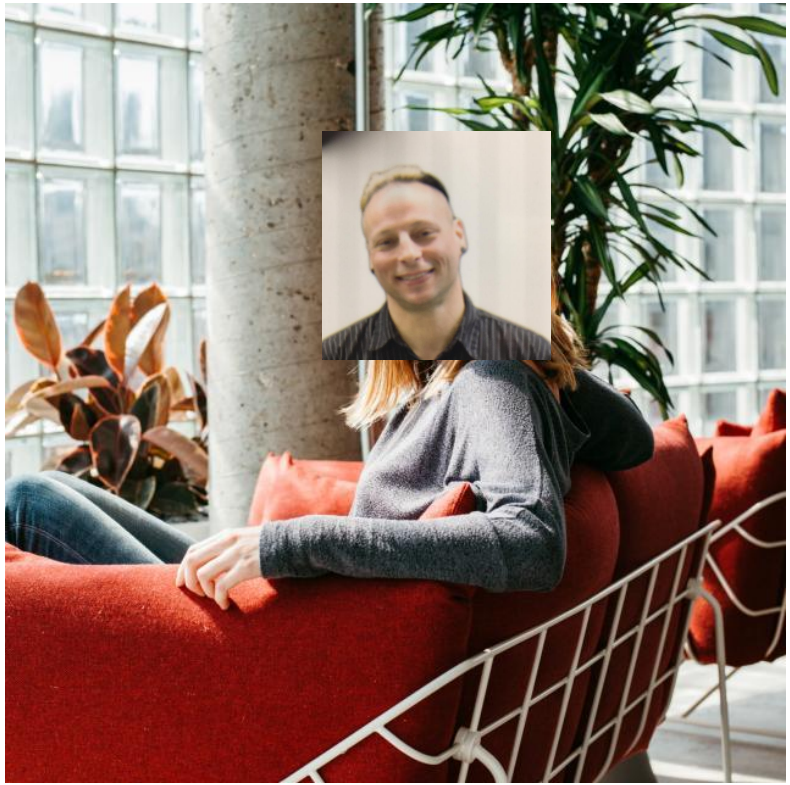
Melanie Cebula & Bruce Sherrod, KubeCon Nov 2019

Who are we?

Who are we?

Hi, I'm Melanie!

Hi, I'm Bruce!

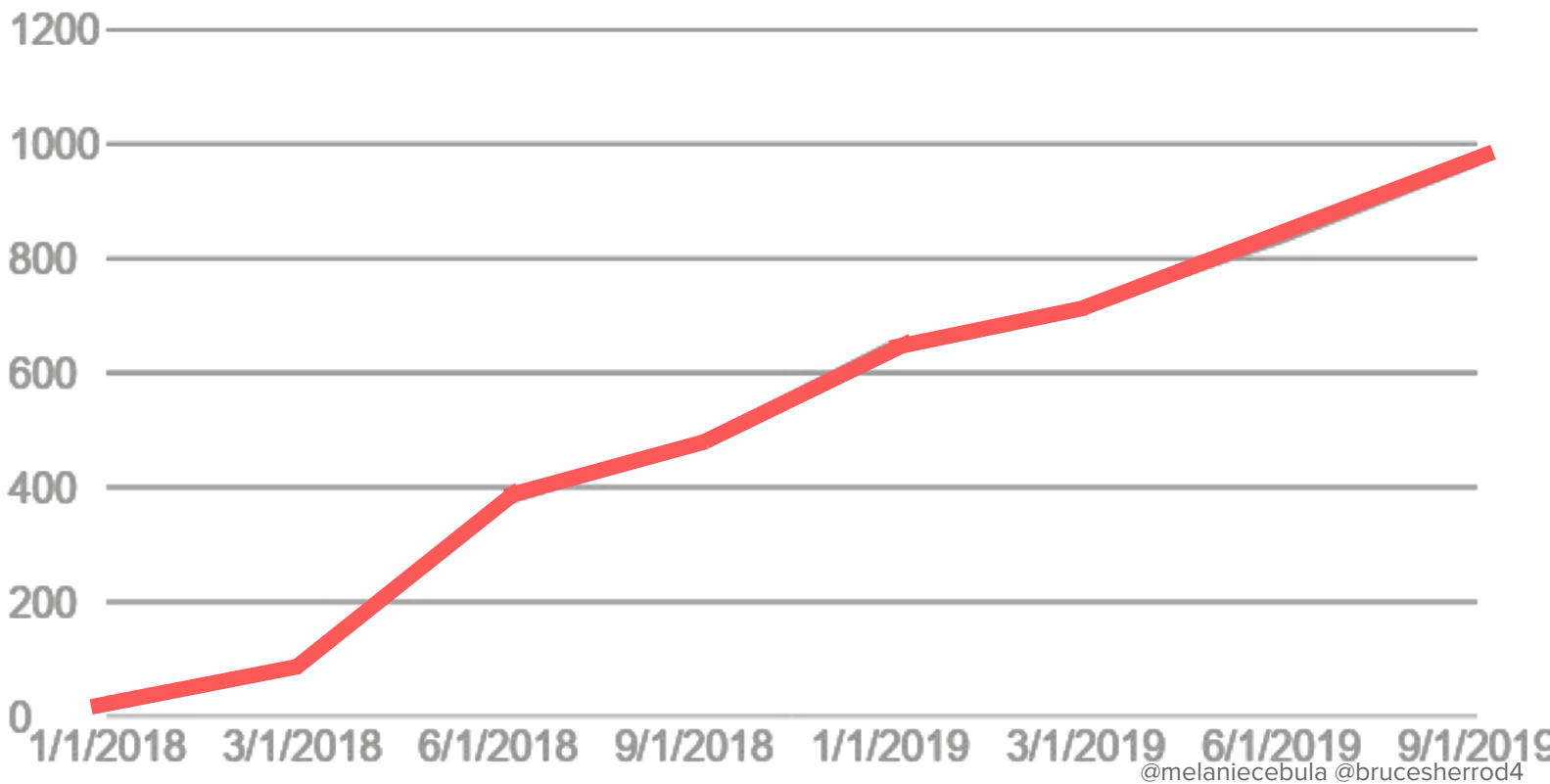


@melaniecebula @brucesherrod4

Kubernetes at Airbnb

Kubernetes @Airbnb

SERVICES

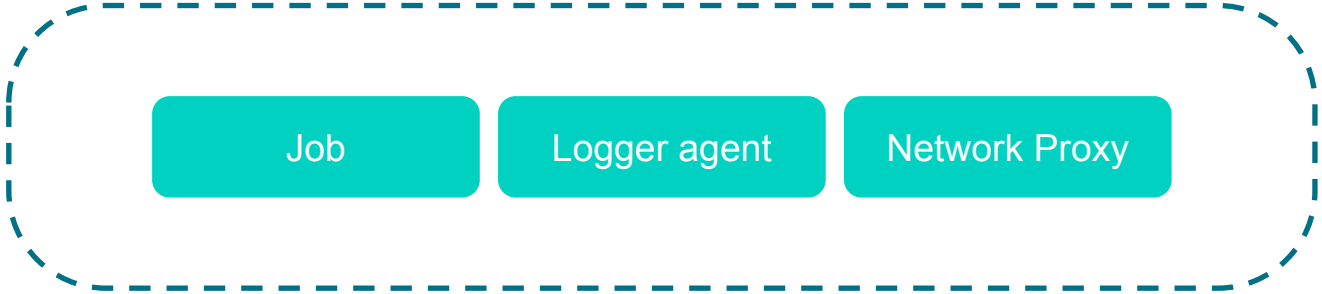


Lots of “Learnings” Along the Way

Zombie Jobs

Kubernetes Jobs and Cronjobs are great!

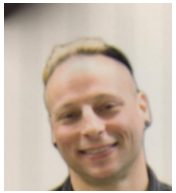
Except... when does it end?!



Try 1

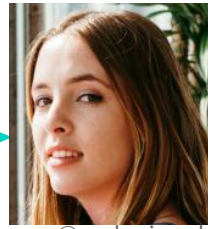
```
workload:
  cronJob:
    activeDeadlineSeconds: 86400
    concurrencyPolicy: Replace
    restartPolicy: OnFailure
```

Try 2



Simple! Just use touchfiles!

On it!



@melaniecebula @brucesherrod4

Try 2

Better support for sidecar containers in batch jobs #25908

Open **a-robinson** opened this issue on May 19, 2016 · 97 comments

a-robinson commented on May 19, 2016 • edited Member

Consider a Job with two containers in it -- one which does the work and then terminates, and another which isn't designed to ever explicitly exit but provides some sort of supporting functionality like log or metric collection.

What options exist for doing something like this? What options should exist?

Currently the Job will keep running as long as the second container keeps running, which means that the user has to modify the second container in some way to detect when the first one is done so that it can cleanly exit as well.

This question was asked on Stack Overflow a while ago with no better answer than to modify the second container to be more Kubernetes-aware, which isn't ideal. Another customer has recently brought this up to me as a pain point for them.

@kubernetes/goog-control-plane **@erictune**

226 26

Assignees

No one assigned

Labels

area/batch
area/workload-api/job
kind/feature
lifecycle/frozen
priority/important-lon
sig/apps
sig/node

Projects

Workloads

Backlog

@melaniecebula @brucesherrod4

Scroll down on the github issue...

jmillikin-stripe commented on Jun 14, 2017 • edited

Contributor

For reference, here's the bash madness I'm using to simulate desired sidecar behavior:

```yaml
containers:
  - name: main
    image: gcr.io/some/image:latest
    command: ["/bin/bash", "-c"]
    args:
      - |
        trap "touch /tmp/pod/main-terminated" EXIT
        /my-batch-job/bin/main --config=/config/my-job-config.yaml
    volumeMounts:
      - mountPath: /tmp/pod
        name: tmp-pod
  - name: envoy
    image: gcr.io/our-envoy-plus-bash-image:latest
    command: ["/bin/bash", "-c"]
    args:
      - |
        /usr/local/bin/envoy --config-path=/my-batch-job/etc/envoy.json &
        CHILD_PID=$!
        (while true; do if [[ -f "/tmp/pod/main-terminated" ]]; then kill $CHILD_PID; fi
        wait $CHILD_PID
        if [[ -f "/tmp/pod/main-terminated" ]]; then exit 0; fi
    volumeMounts:
      - mountPath: /tmp/pod
        name: tmp-pod
        readOnly: true
```

Let's introduce some bash? 🤔

Introduce a run-sidecar script

```
# main container
command:
  - bash
  - -c
  - |
    trap "touch /tmp/pod/main-terminated" EXIT
    [ do stuff ]
    touch /tmp/pod/main-terminated

# sidecar
command:
  - dumb-init
  - /scripts/run-sidecar.sh
  - [do stuff]
```

Main container writes touch file on exit

Sidecar runs a wrapper script that looks for touch file and exits

run-sidecar.sh: take 1

```
# run-sidecar.sh
#!/usr/bin/env bash

"$@" &
   CHILD_PID=$!
   (while true; do if [[ -f "/tmp/pod/main-terminated" ]]; then kill
$CHILD_PID; fi; sleep 1; done) &
   wait $CHILD_PID
   if [[ -f "/tmp/pod/main-terminated" ]]; then exit 0; fi
```

run-sidecar.sh: take 2

make dumb-init exit with the child process's exit code

```
CHILD_PID=0
_term () {
 if [[ "$CHILD_PID" -eq "0" ]]; then
   exit 0 # $CHILD_PID never started, quit
 fi

 if [[ "$$" -eq "1" ]]; then
   kill "$CHILD_PID" # I am running as PID 1, forward SIGTERM to child
 fi

 wait "$CHILD_PID"
 exit "$?" # wait for child to die and return its status code as my own
}

trap _term SIGINT SIGTERM
...
```

run-sidecar.sh: take ???

[run-sidecars] Better sidecar signal handling #190

Edit

Merged deployboard merged 3 commits into `master` from `dlow-better-sidecar-signal-handling` on Mar 15

Conv

[run-sidecars] Send sigkill to long running sidecars #194

Merged deployboard merged 7 commits into `master` from `dlow-sigkill-long-running-sidecar` on Mar 19

Conversation 11 | Commits 7 | Checks 0 | Files changed 1 | +62 –

[run-sidecar] Check if $CHILD_PID dies on each iteration. #203

Edit

Merged deployboard merged 1 commit into `master` from `dlow-run-sidecar-die-if-child-also-dies` on Mar 29

Conversation 2 | Commits 1 | Checks 0 | Files changed 1 | +16 –15

A “simple” bash script?

Whose PID is it anyway?!

```
$ wc -l run-sidecar.sh
     103 run-sidecar.sh
```

A “simple” bash script?

… let’s just solve it in k8s

Try 3: Solve it in K8s!!!

Sidecar Containers #753

Open Joseph-Irving opened this issue on Jan 29 · 44 comments

… but it's targeted for 1.17
… and it's an API change

Joseph-Irving commented on Jan 29 • edited Contributor ···

Enhancement Description

- One-line enhancement description: Containers can now be a marked as sidecars so that they startup before normal containers and shutdown after all other containers have terminated.

Type to enter a caption.

Sidecar KEP API implementation #919

Merged k8s-ci-robot merged 1 commit into `kubernetes:master` from `Joseph-Irving:sidecar_api` on Jun 19

Conversation 35 Commits 1 Checks 0 Files changed 1

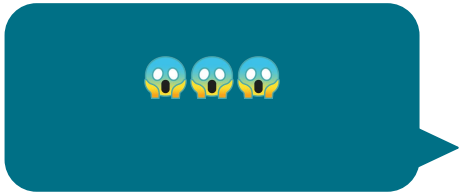
Type to enter a caption.

Try 4: Solve it in forked K8s!!!

Melanie Cebula
@MelanieCebula

Any recommendations for maintaining a forked Kubernetes? Mostly to get upstream enhancements or patches that we can't wait around for.

5:57 PM · May 23, 2019 · Twitter for Android



Try 5: Solve it in ~~forked~~ patched K8s!!!

There are times when a custom patch makes sense!

Lyft has a kubelet patch

- Enforce container start/shutdown ordering in kubelet based on annotations

lyft / kubernetes
forked from kubernetes/kubernetes

Watch 5 | Star 0 | Fork 21

Code | Pull requests 3 | Projects 0 | Security | Insights

sidecar: container ordered start/shutdown support

Browse files

add-test (#10) + containerrecovery + debug-stats (#8) + metrics-server-debug + patch-crio-assume-cadvisor + release-1.14.4-lyft + release-1.14.6-lyft + release-1.14.7-lyft + remove-terminated

tom

lyft / kubernetes
forked from kubernetes/kubernetes

Watch 5 | Star 0 | Fork 21.1k

Code | Pull requests 3 | Projects 0 | Security | Insights

pkg/kubelet: try restoring the container spec if its nil

Browse files

```
We're not guaranteed that the pod passed in has the ContainerSpec
we're looking for. With this, we check if the pod has the container
spec, and if it doesn't, we try to recover it one more time.
```

add-test (#10, #5) + containerrecovery (#5) + debug-stats (#8, #5) + release-1.14.4-lyft (#5) + release-1.14.6-lyft (#5) + release-1.14.7-lyft (#5) + remove-terminated (#5)

tomwans committed on Jul 29 — 1 parent 451bffc — commit 6e5d60f9d829f130f38e2b4ee60710b6447b7c59

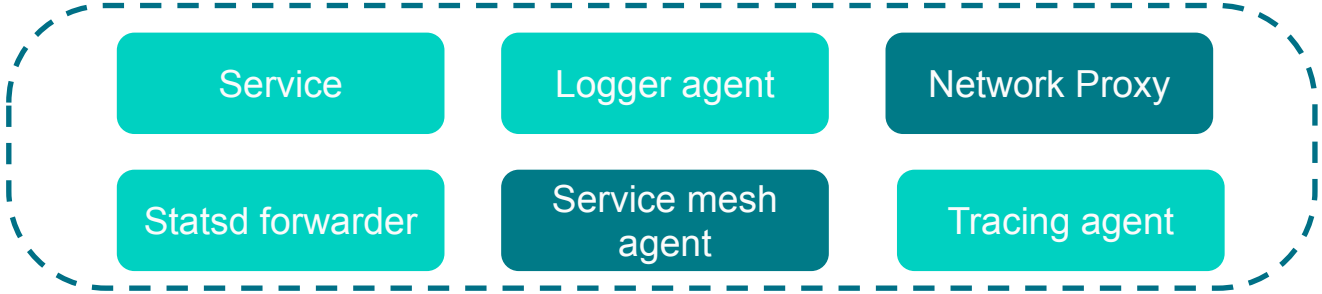
A Pod

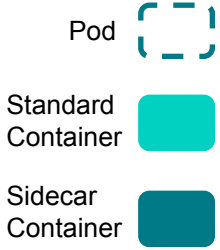

Pod
Container


Service
Logger agent
Network Proxy
Statsd forwarder
Service mesh agent
Tracing agent

Annotate containers as “Standard” or “Sidecar”

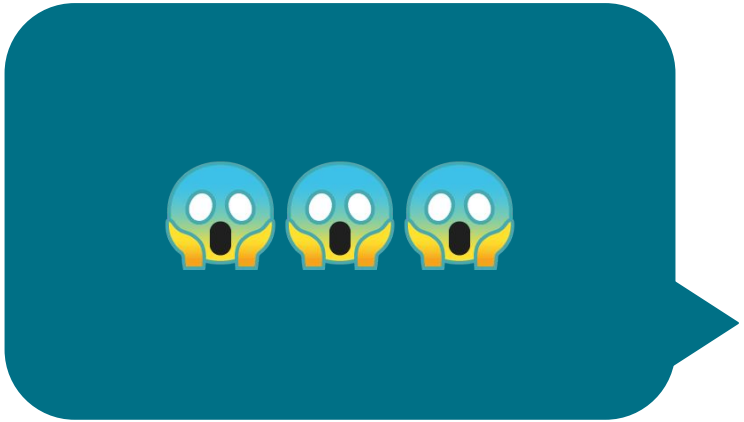
- “Sidecar” containers start before “Standard” containers and shutdown after them





Takeaway

Solve your problem at the appropriate abstraction level-- and that may be patching your behavior into kubernetes itself!



Service Mesh Speeding Accidents

@melaniecebula @brucesherrod4

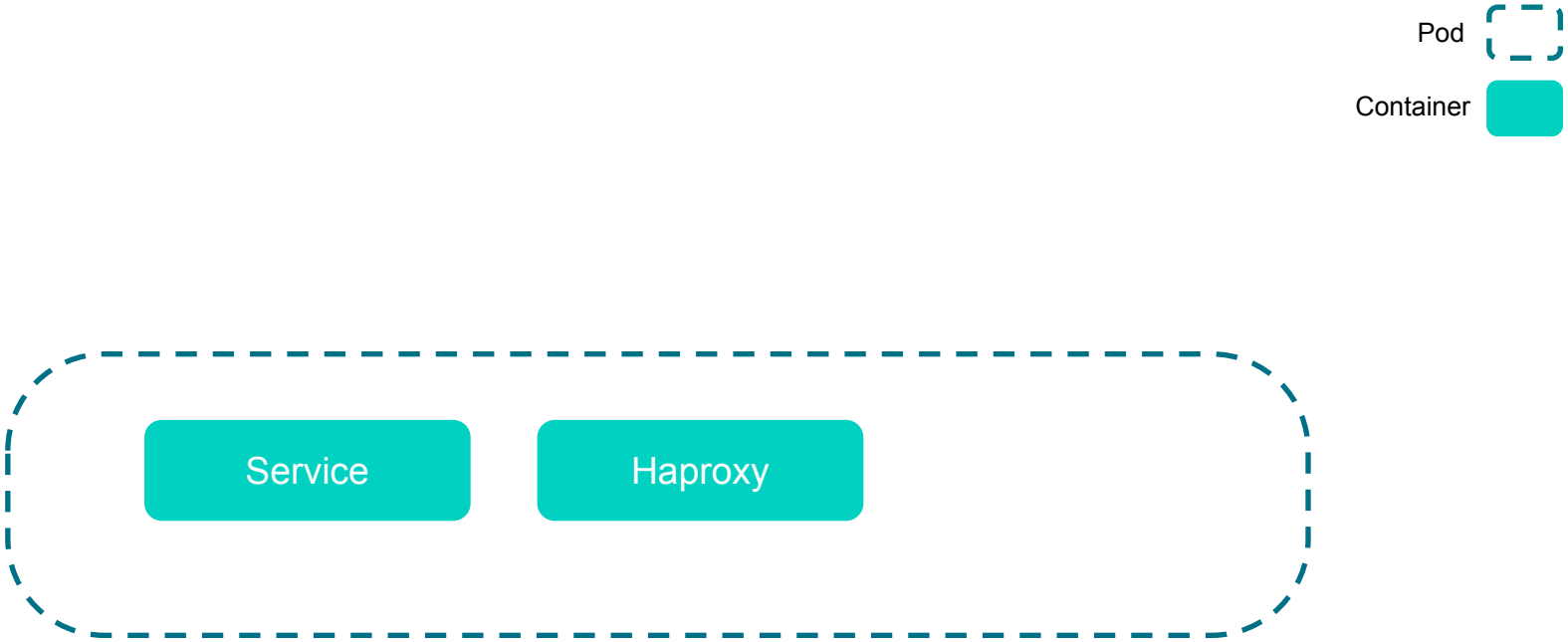
SmartStack: Service Discovery in the Cloud

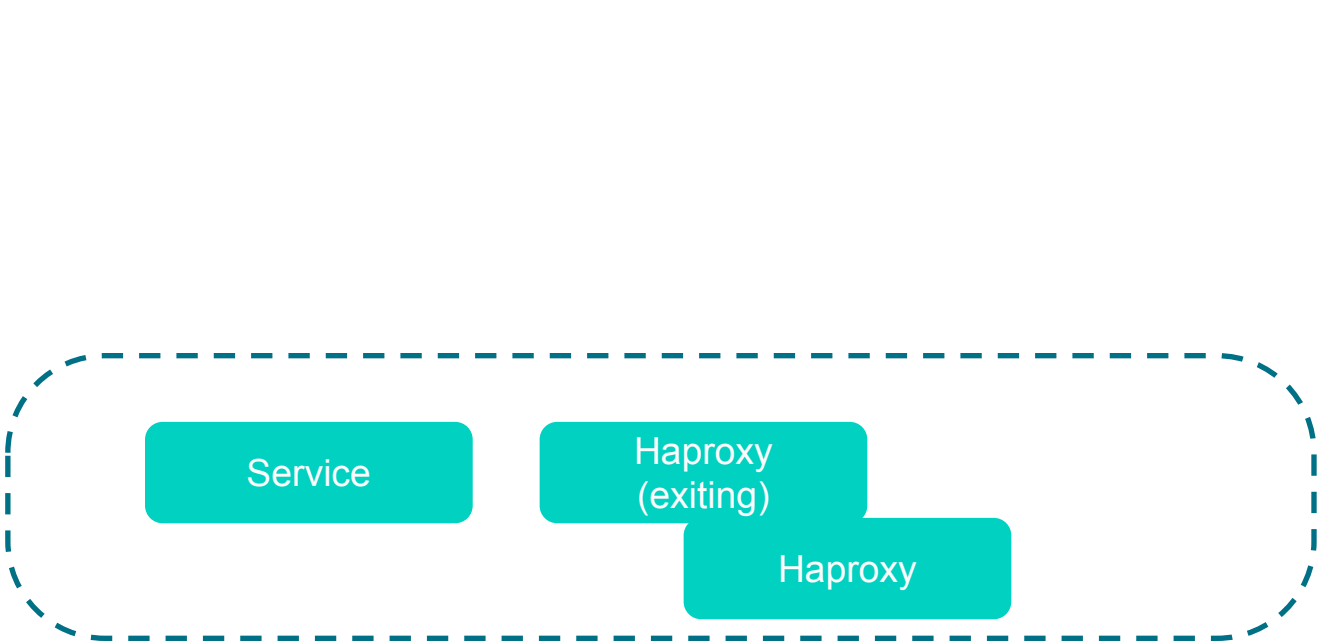
AirbnbEng Follow
Oct 23, 2013 · 13 min read

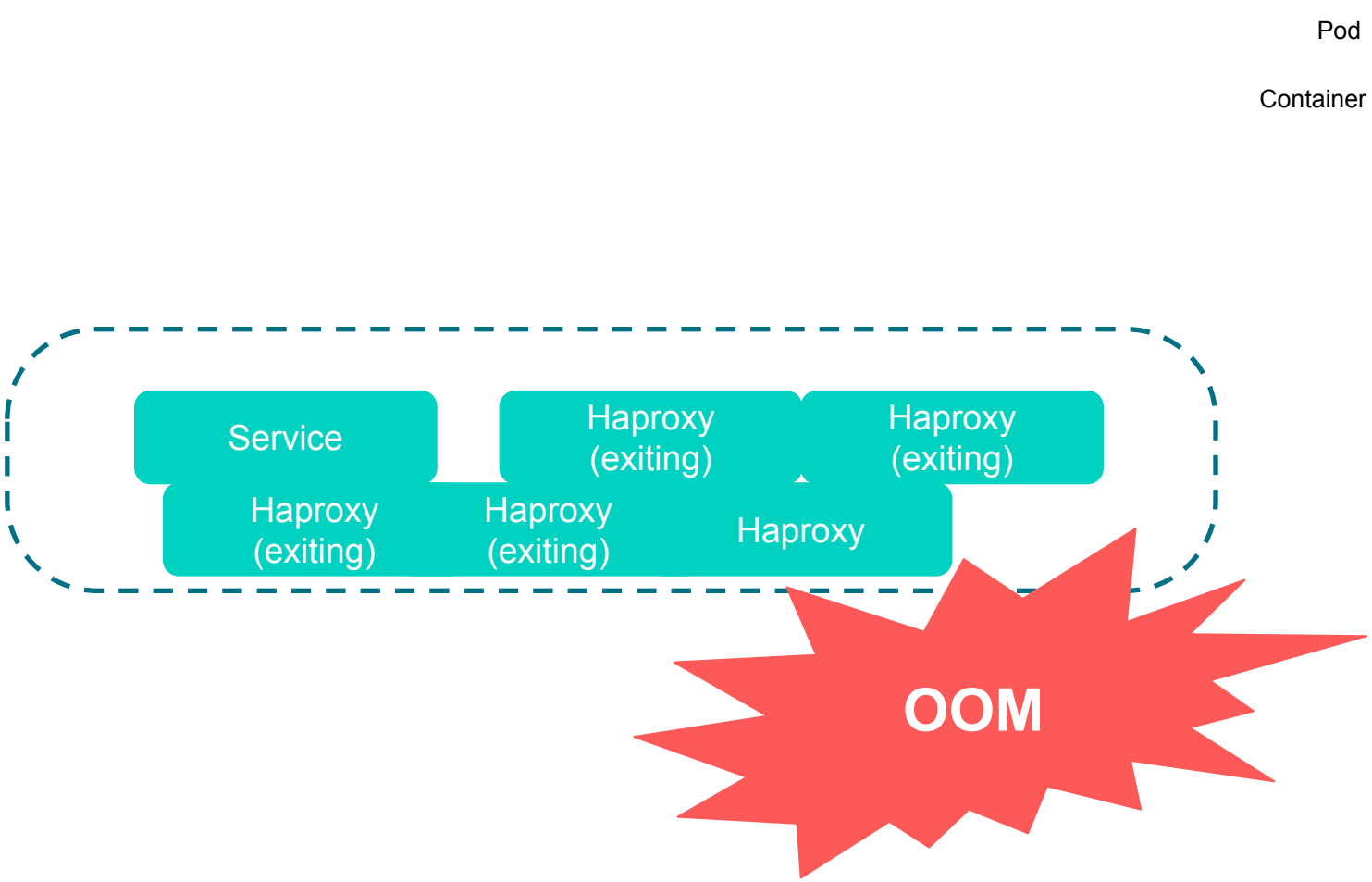
By Igor Serebryany & Martin Rhoads

What is SmartStack?

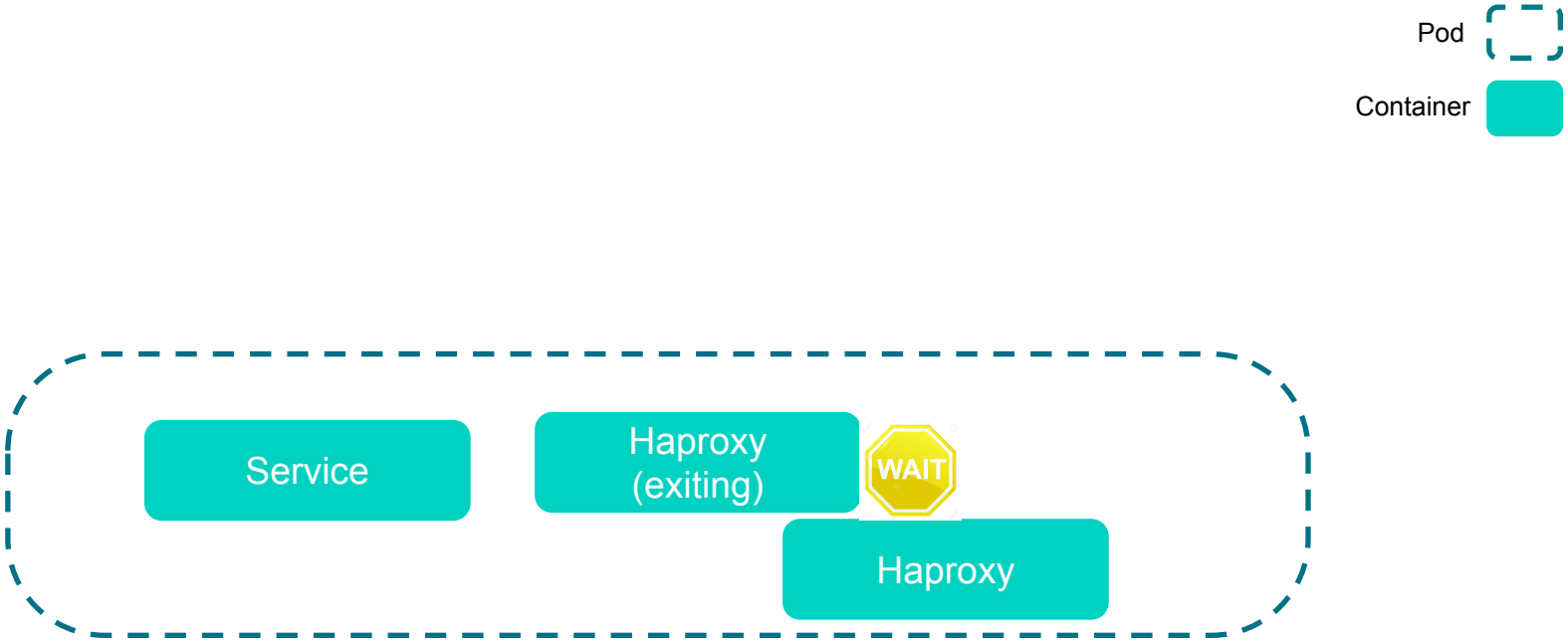
SmartStack is an automated service discovery and registration framework.


Pod
Container
Service
Haproxy


Pod
Container
Service
Haproxy
(exiting)
Haproxy

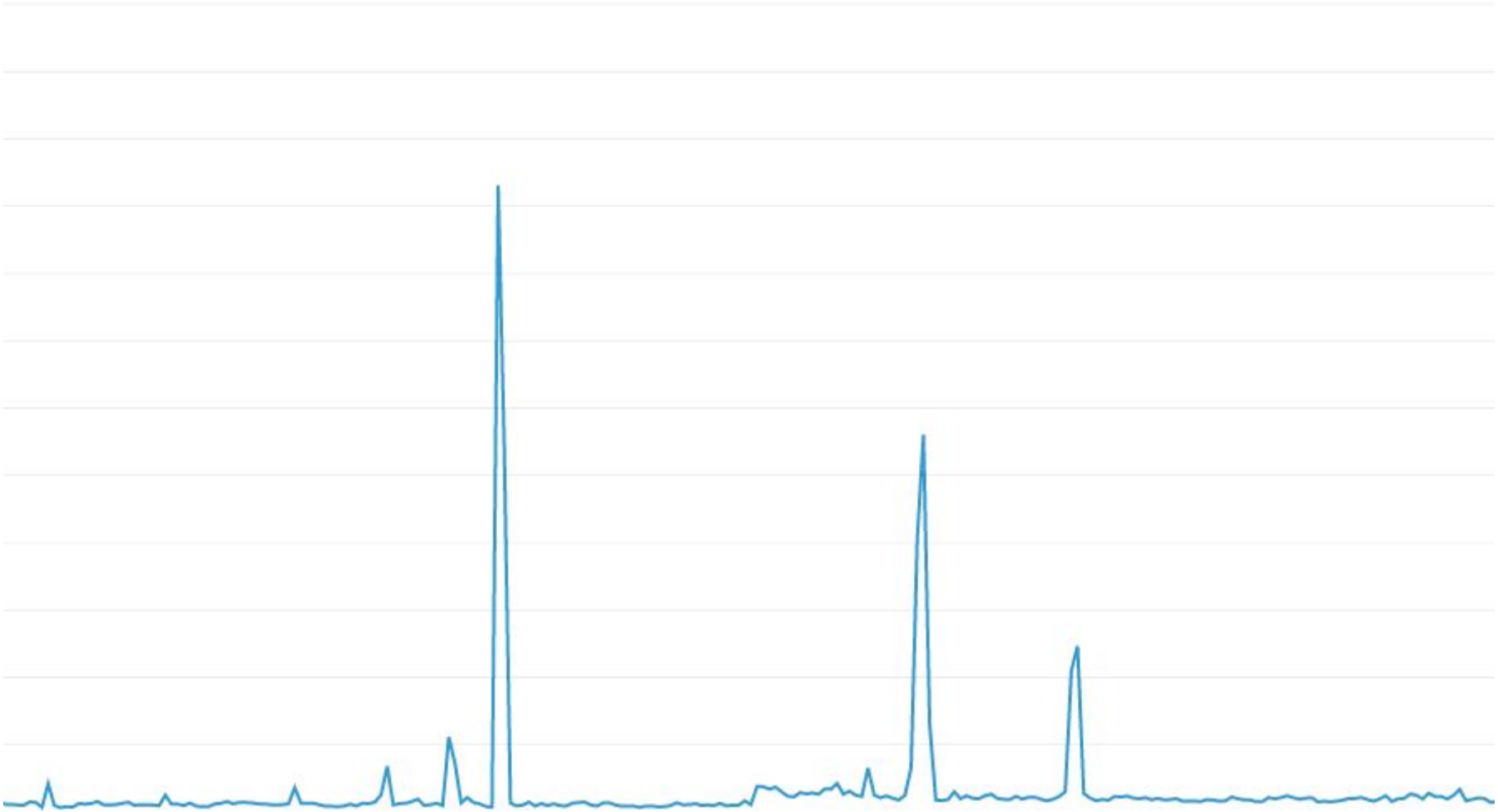


@melaniecebula @brucesherrod4



@melaniecebula @brucesherrod4

```
kind: Deployment
spec:
  strategy:
    rollingUpdate:
      maxSurge: 100%
```

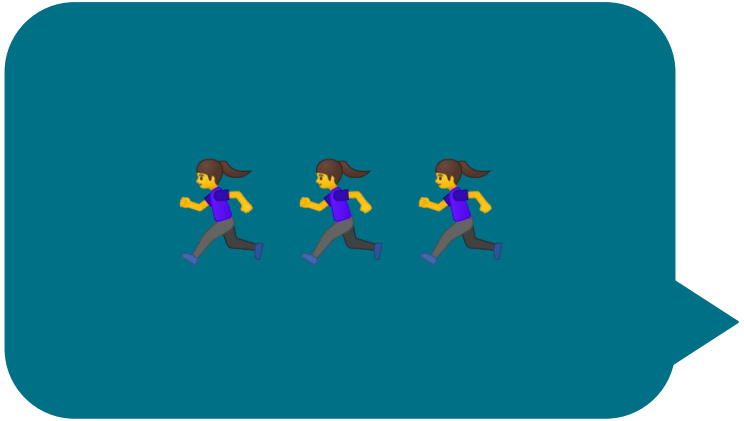


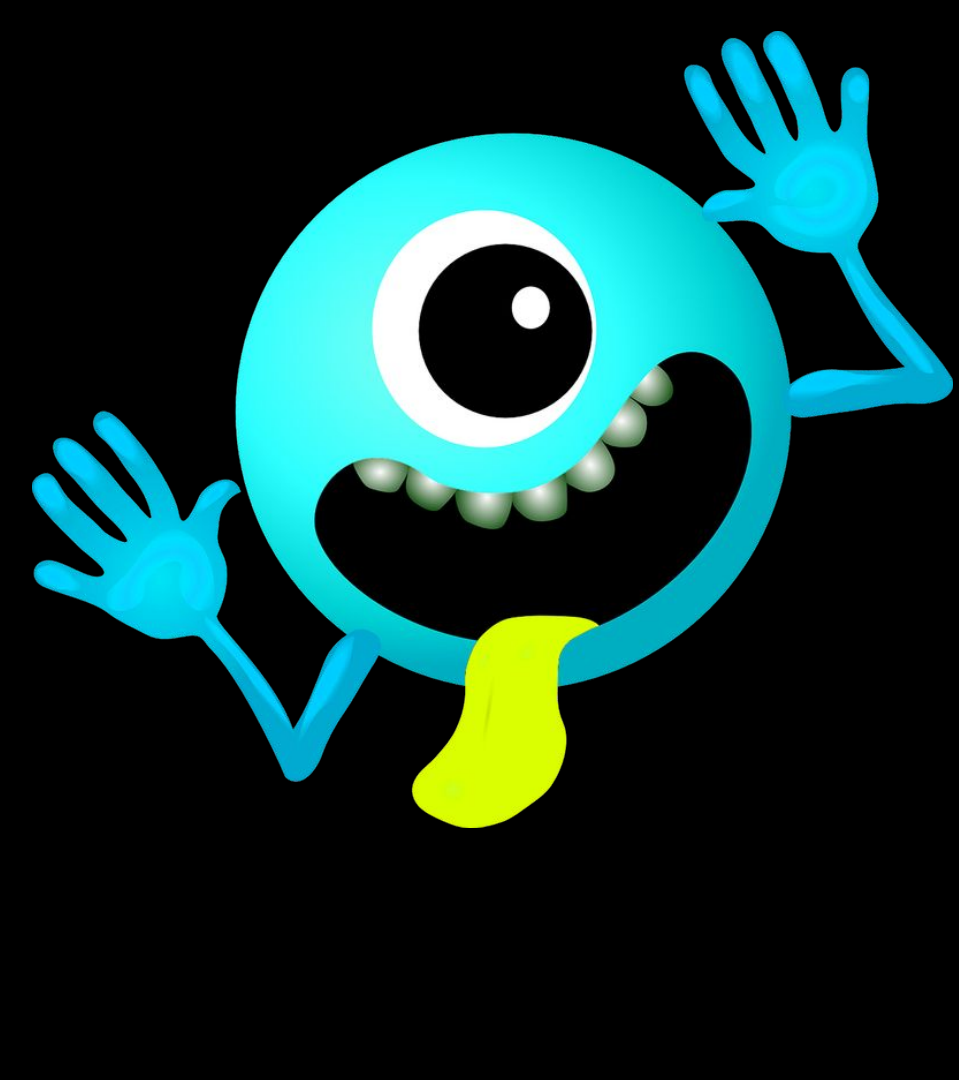
@melaniecebula @brucesherrod4

```
# Give our self 120 seconds to wait for smartstack
# to realize that this pod is gone and stop sending traffic
# to it, before termination.
lifecycle:
  preStop:
    exec:
      command:
        - /bin/sleep
        - "120"


containers:
  - name: main
    terminationGracePeriodSeconds: 180
```

Takeaway

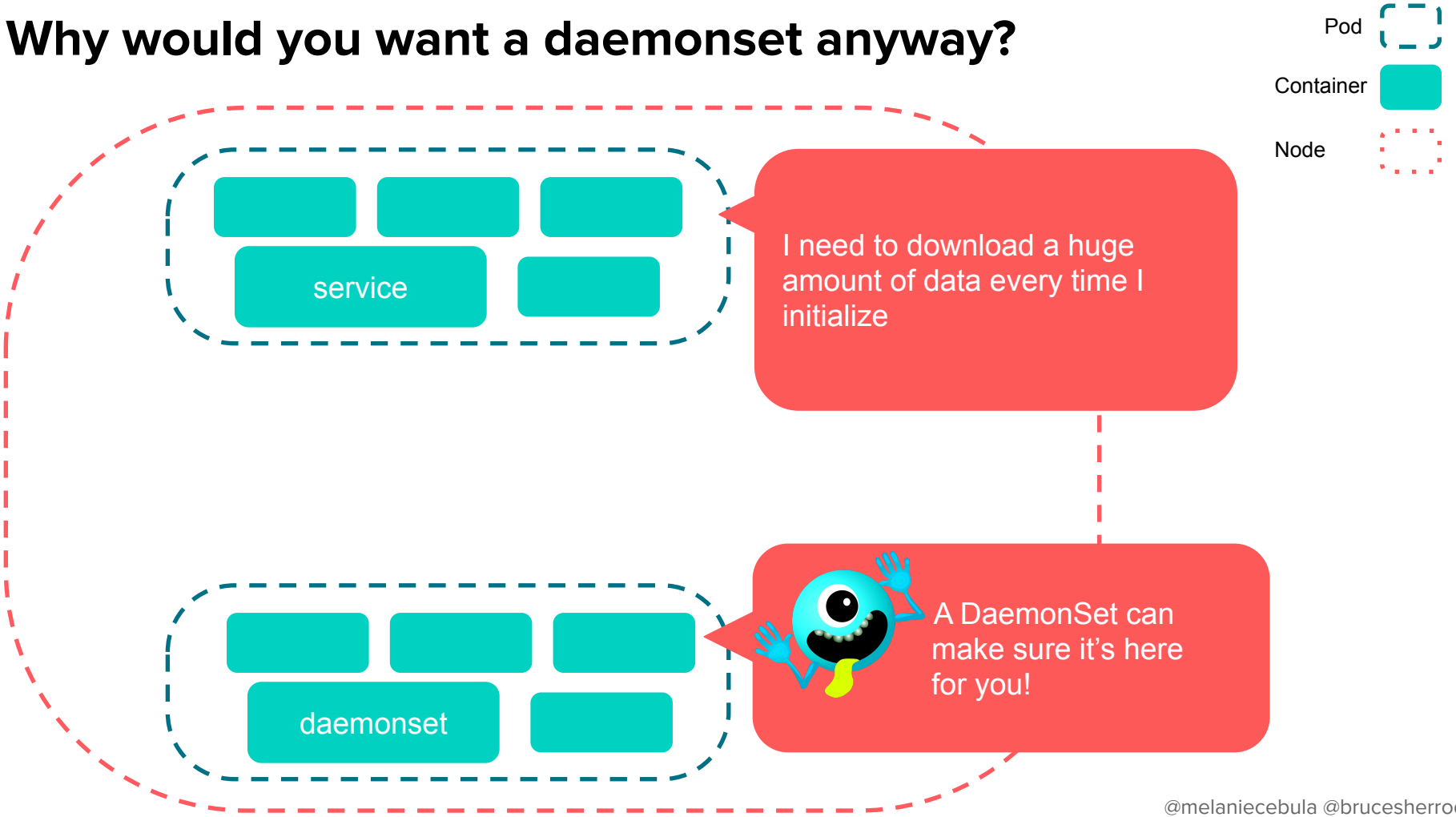
Kubernetes deploys can cycle pods super fast, whether the rest of your infrastructure can keep up or not!



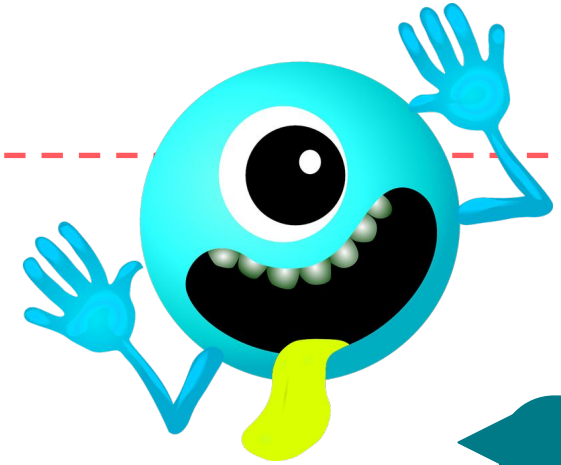


Monster Daemonsets

Why would you want a daemonset anyway?



Reality



Pod

Container

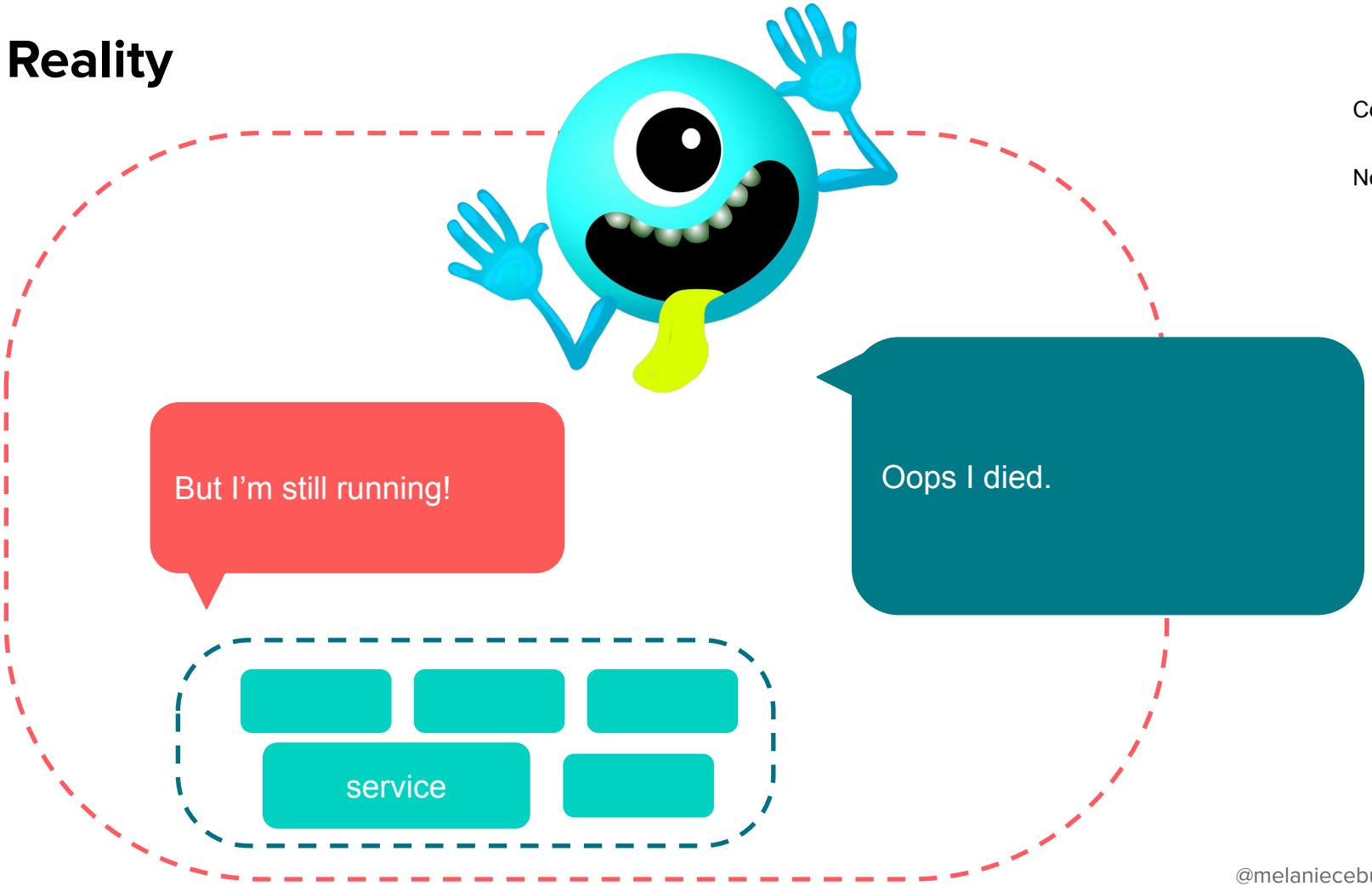
Node

But I'm still running!

I'm gonna step out for a bit while I deploy.

service

Reality



@melaniecebula @brucesherrod4

Soln: Change Daemonset to Deployment

```
# service that depends on deployment
spec:
affinity:
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
     -  labelSelector:
          matchExpressions:
              -  key: app
                 operator: In
                 Values:
                 - { DEPLOYMENT NAMESPACE }
         topologyKey: “kubernetes.io/hostname”
         namespaces:
         - { DEPLOYMENT NAMESPACE }
```

Soln: Dynamic tainter based on pod readiness

Taints and Tolerations

Node affinity, described here, is a property of *pods* that *attracts* them to a set of nodes (either as a preference or a hard requirement). Taints are the opposite – they allow a *node* to *repel* a set of pods.

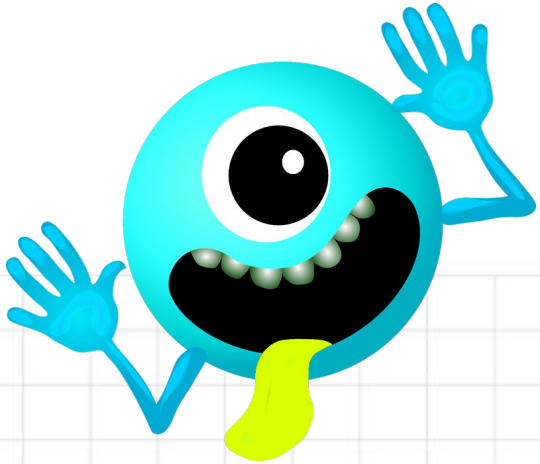
Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node; this marks that the node should not accept any pods that do not tolerate the taints. Tolerations are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints.

Reality

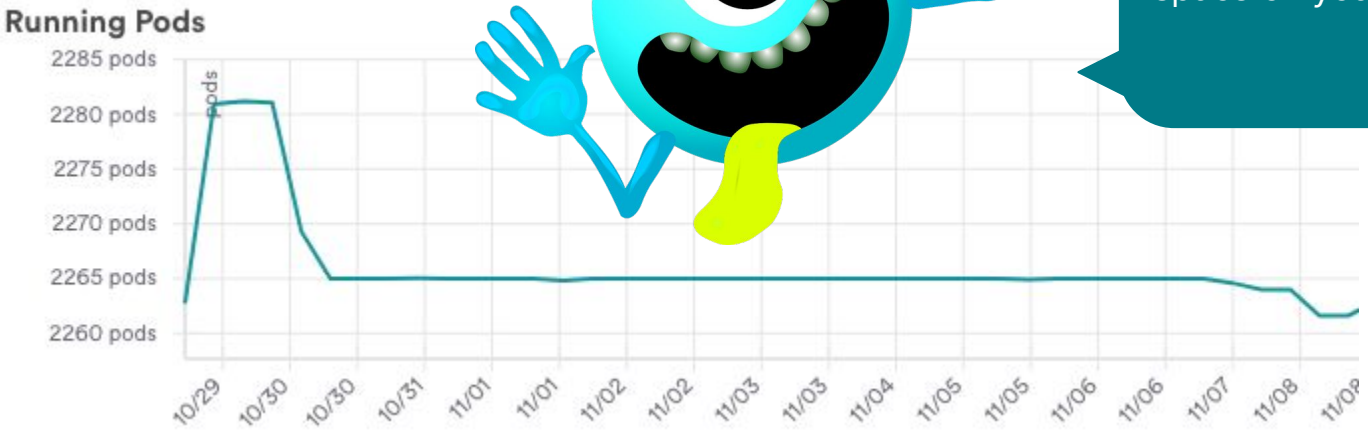
CPU 6h Average Total Cores Requested: 2273.3 cores

Memory 6h Average Total Bytes Requested: 12.37 TB

Reality



I take up a HUGE amount of space on your cluster!



CPU 6h Average Total Cores Requested: 2273.3 cores

Memory 6h Average Total Bytes Requested: 12.37 TB

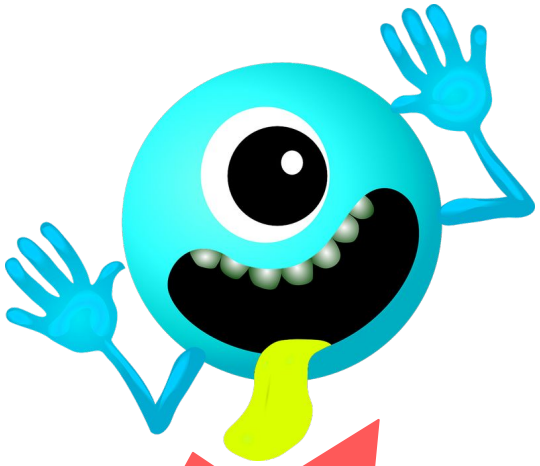
Reality

I'm going to create and deploy a daemonset workload

The cluster can't hold that many pods! Now we have 1000s of pods getting OOM-killed!

Now etcd is filling up with all these events!

Reality



Wow, it's so easy for me to take down your entire cluster! Weeee~

OOM

OOM

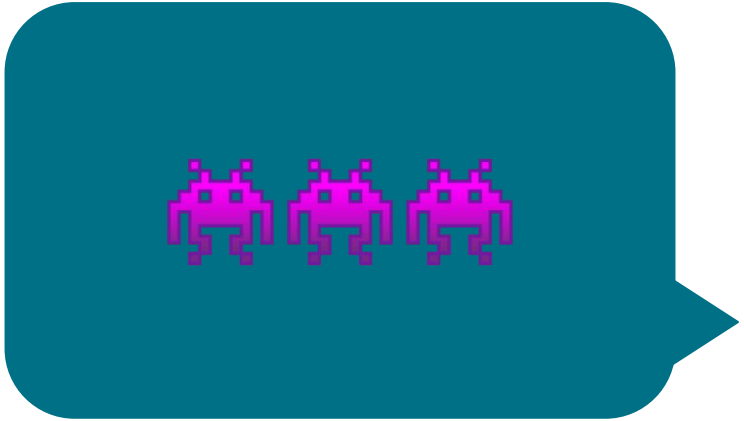
OOM

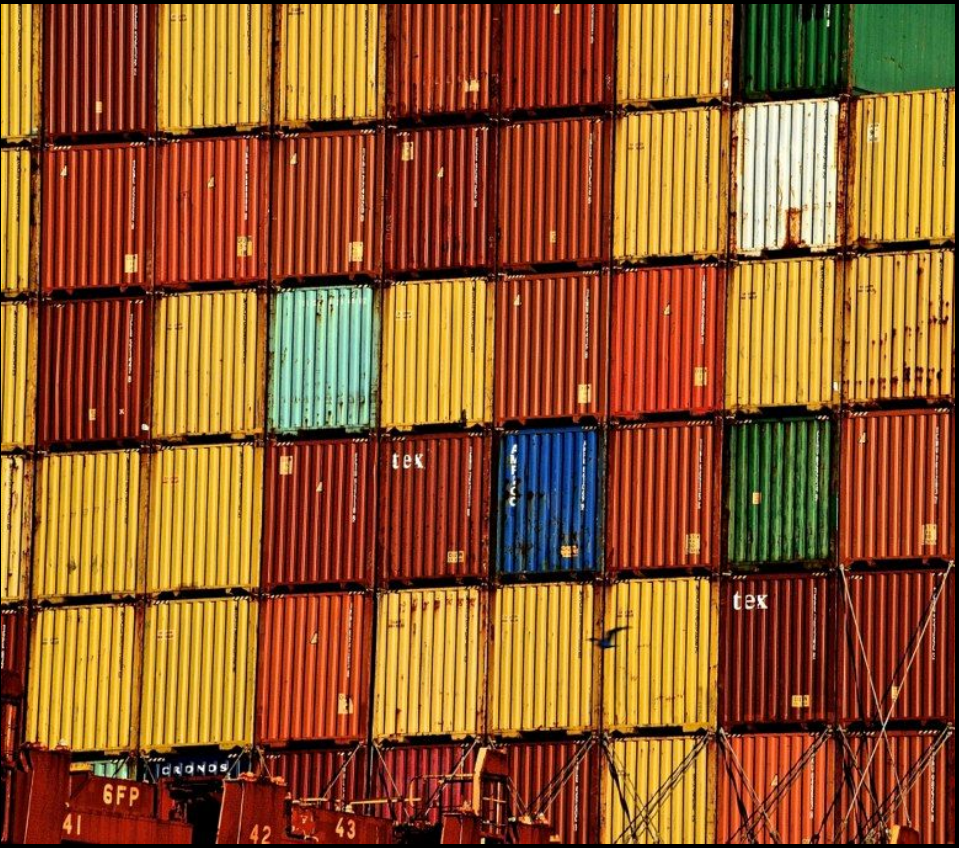
Soln

- … Give up and migrate back to sidecar model
- Use admission controller to enforce that new daemonsets go through a strict review process
- Dedicated clusters for services that need daemonsets or other special node behavior

Takeaway

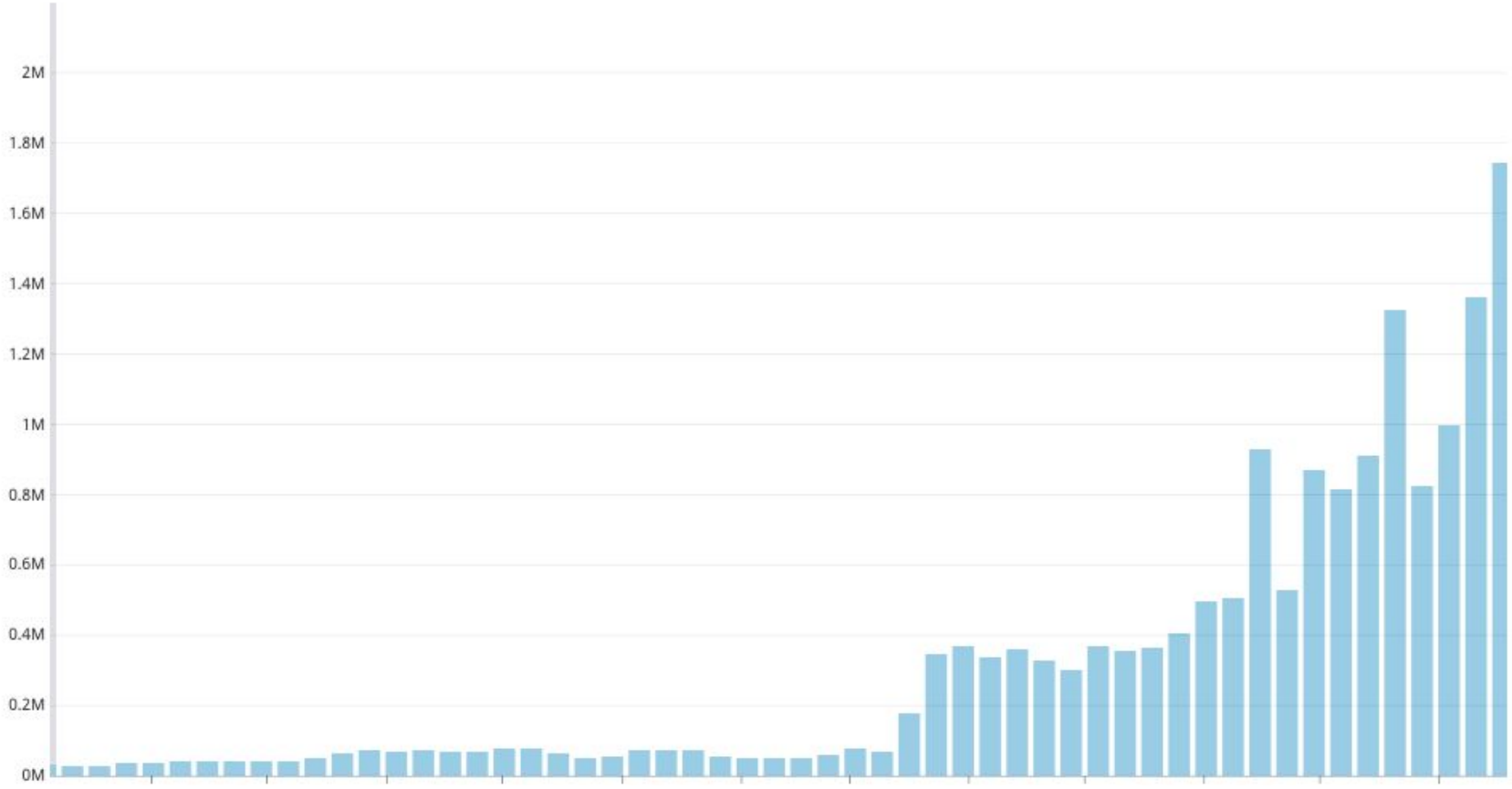
Daemonsets can take down a cluster in a way that other workloads can only can aspire to





Where’s my docker image?!

We make a lot of docker images



@melaniecebula @brucesherrod4

ECR lifecycle is a start but ...

```
pod-5f969c7554-hz4bg   12/12   Init:ImagePullBackoff
pod-5f969c7554-jfth4   12/12   Init:ImagePullBackoff
pod-5f969c7554-jtwz2   12/12   Init:ImagePullBackoff
pod-5f969c7554-kfd7r   12/12   Init:ImagePullBackoff
pod-5f969c7554-kwfqz   12/12   Init:ImagePullBackoff
pod-5f969c7554-ltc6g   12/12   Init:ImagePullBackoff
pod-5f969c7554-m2m2h   12/12   Init:ImagePullBackoff
pod-5f969c7554-ncdmn   12/12   Init:ImagePullBackoff
pod-5f969c7554-nnc4f   12/12   Init:ImagePullBackoff
pod-5f969c7554-ntnkd   12/12   Init:ImagePullBackoff
pod-5f969c7554-pcn75   12/12   Init:ImagePullBackoff
pod-5f969c7554-pq29h   12/12   Init:ImagePullBackoff
pod-5f969c7554-rgfws   12/12   Init:ImagePullBackoff
pod-5f969c7554-rjhch   12/12   Init:ImagePullBackoff
pod-5f969c7554-s2zz5   12/12   Init:ImagePullBackoff
pod-5f969c7554-v2ns4   12/12   Init:ImagePullBackoff
pod-5f969c7554-vb7kt   12/12   Init:ImagePullBackoff
pod-5f969c7554-xp6cn   12/12   Init:ImagePullBackoff
pod-5f969c7554-xxfl5   12/12   Init:ImagePullBackoff
```

ECR Cleaner

```
cronJob:
  schedule: "@daily”
```

```
find_all_images_in_use()

for repo in ecr_repos:
  delete_old_images(except in use)
```

Oops #1

CI fails due to docker errors

Oops #1

```
cronJob:
  schedule: "@daily”
```

```
find_all_images_in_use()

for repo in ecr_repos:
  delete_old_images(except in use)
```

Hacky solution #1

```
cronJob:
  schedule: "@hourly”
```

```
find_all_images_in_use()

for repo in ecr_repos:
  delete_old_images(except in use)
```

Oops #2

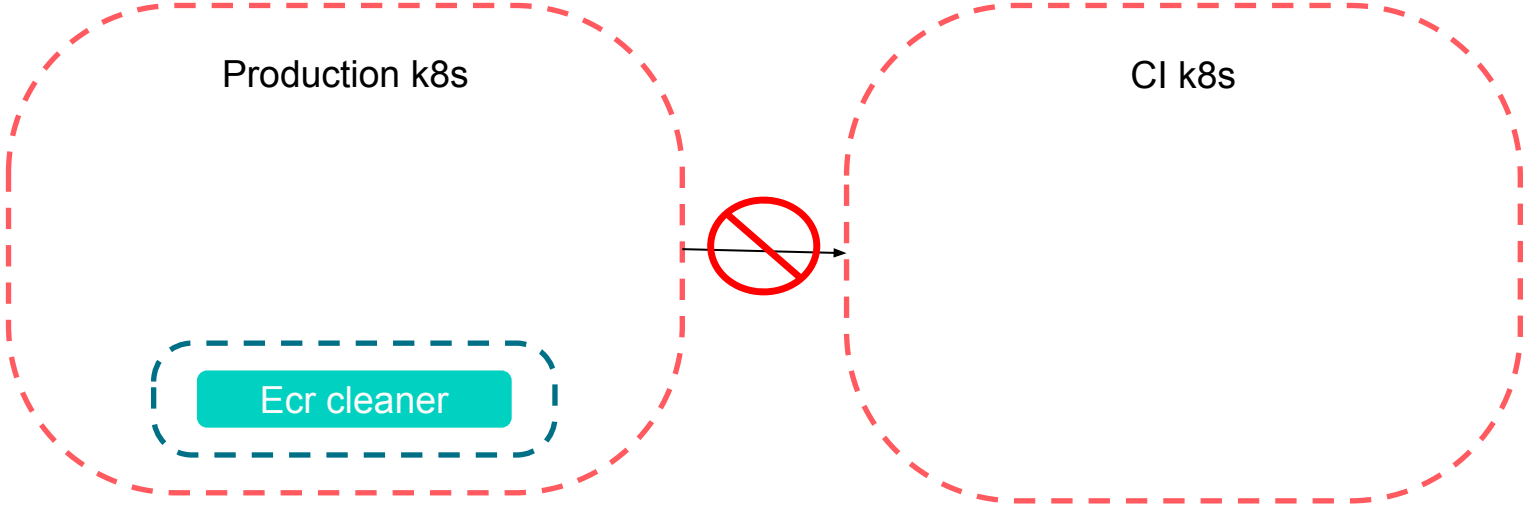
Rotating kubernetes minion node takes down services

Oops #2

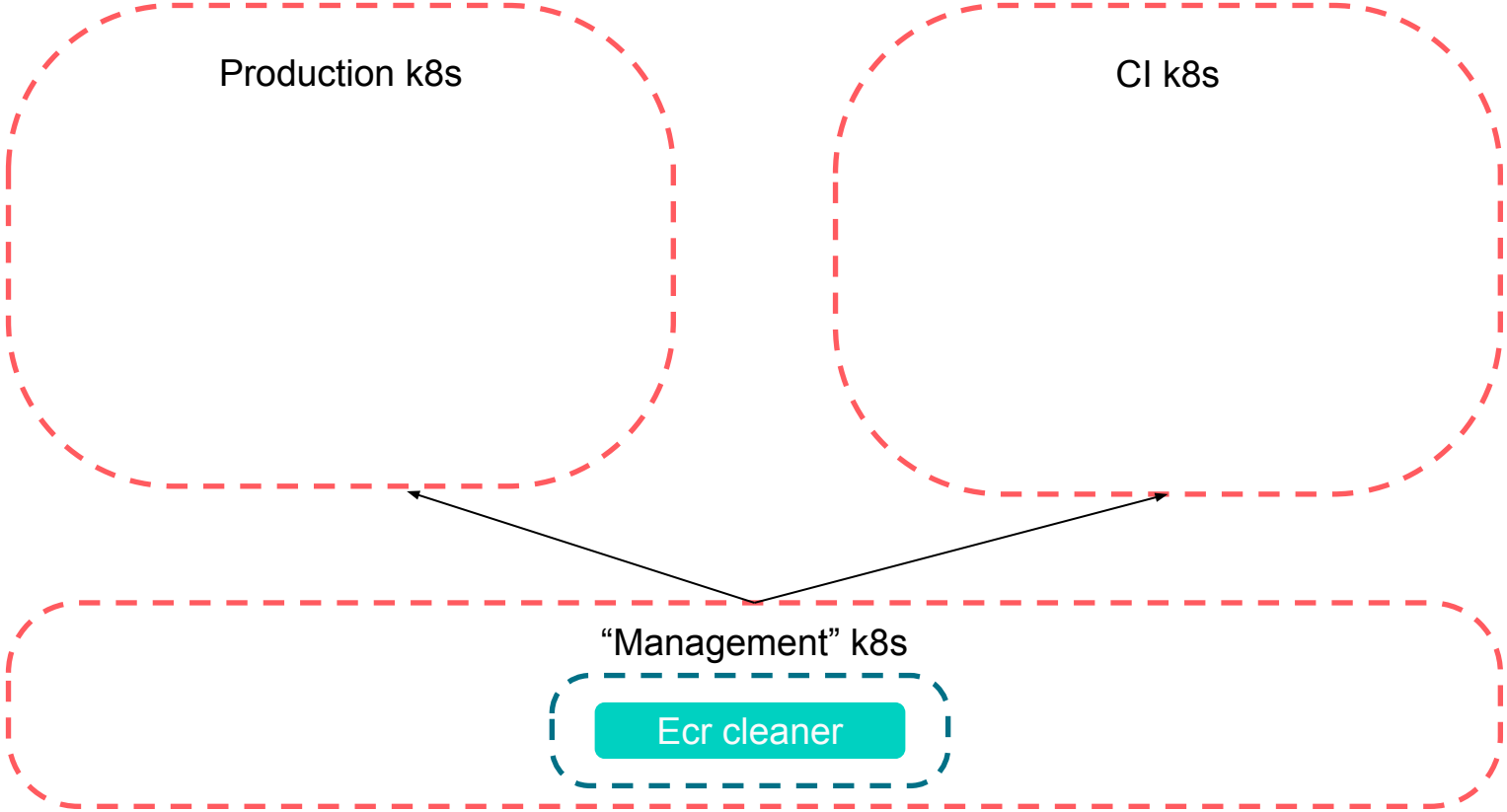
```
cronJob:
  schedule: "@hourly”
```

```
find_all_images_in_use()

for repo in ecr_repos:
  delete_old_images(except in use)
```

What about dev/prod isolation?

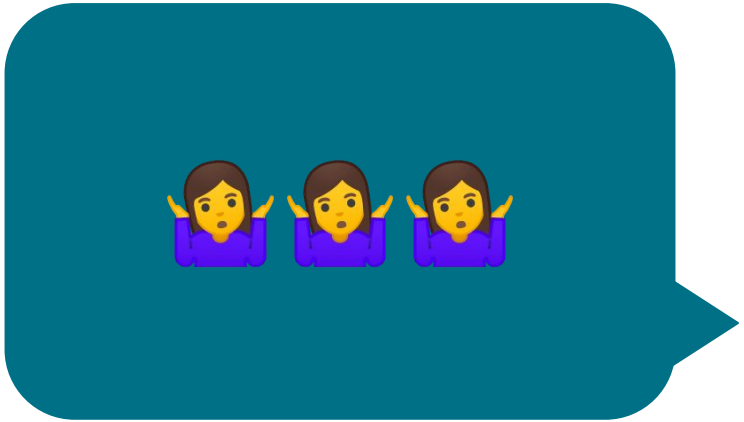


Solution: more k8s clusters!



Takeaway

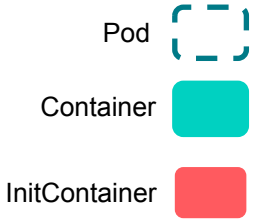
Why can’t I hold all these docker images? Make sure to keep track of them!



To Init or not to init... that is the question

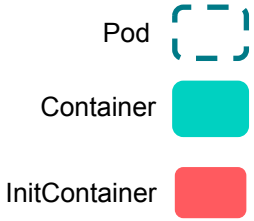
@melaniecebula @brucesherrod4

Typical Pod, Right?



Runtime config service propagates value changes to applications within seconds

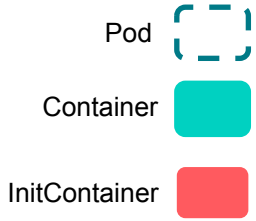
Problem



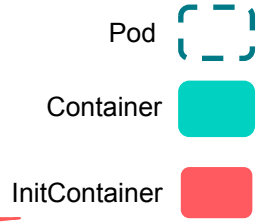
Runtime config agent is a container in our pod that polls Runtime config service. But our pod needs that data before it starts!

Try 1

Split update logic into init container



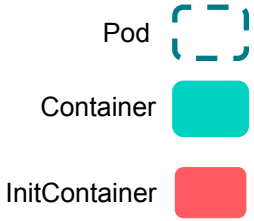
Try 1


Pod
Container
InitContainer


Failed to connect to runtime:
Connection refused

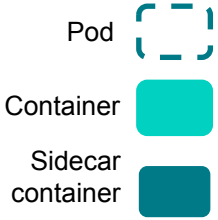

Init Runtime Config
Service
Runtime config agent
Network Proxy

Try 2



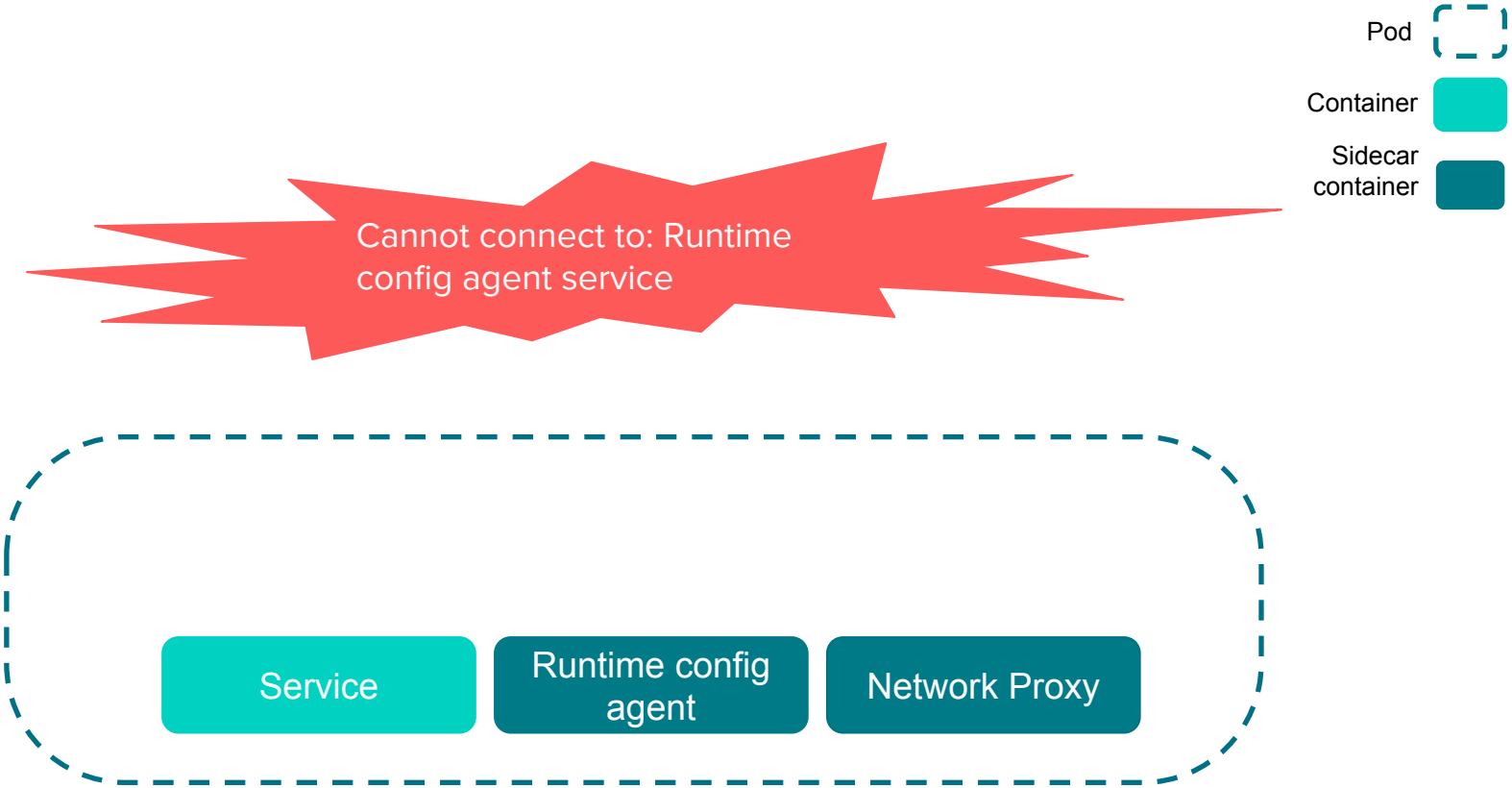
Init container still needs to connect to Runtime config service and therefore needs to start up and stop its own network proxy

Try 3

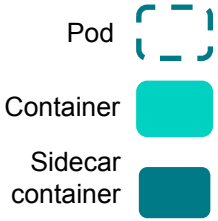


With sidecar ordering support, we can make Runtime config agent and Network Proxy sidecars

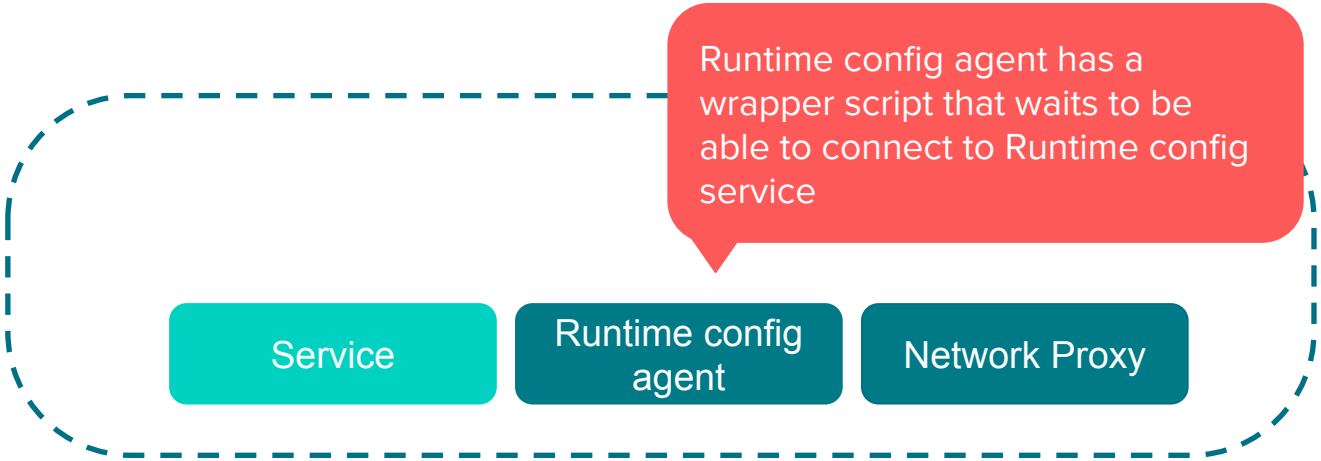
Try 3


Pod
Container
Sidecar container
Cannot connect to: Runtime config agent service
Service
Runtime config agent
Network Proxy

Try 4

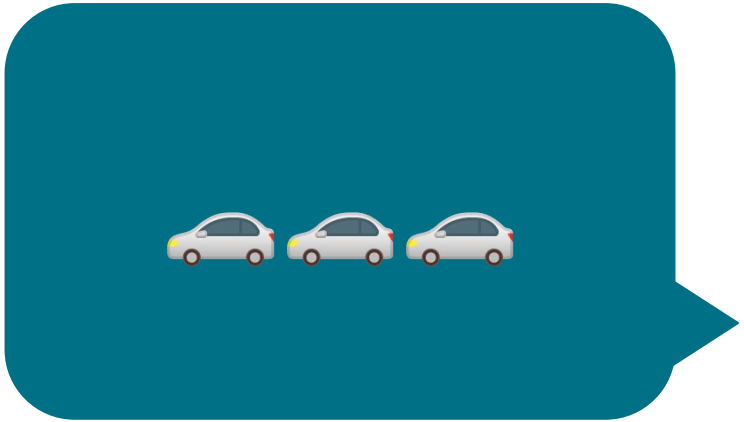


We add logic to Runtime config agent sidecar to wait for Network Proxy sidecar to be ready



Takeaway

You might need to support ordering between sidecars too!



Where’s my Custom Resource ?

Custom Resources are great!

```
apiVersion: airbnb.com/v1
spec:
  ...
```

When is a custom resource deploy complete?

```
apiVersion: airbnb.com/v1
spec:
  …
status:
  status: Pending
```

kubectl apply
sets the resource:
status.status=Pending

```
apiVersion: airbnb.com/v1
spec:
  …
status:
  status: Ready
  message: synced at 2019-10-30 14:25:31 +0000
```

Controller updates the resource:
status.status=[Ready,Error]
status.message=what happened

The problem

Kubernetes < 1.12

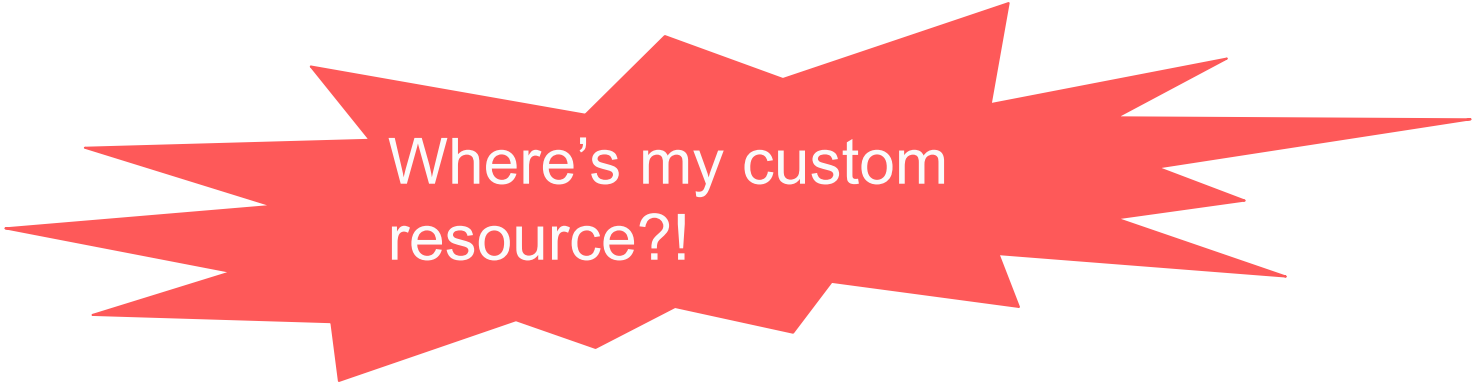
- Works great!

Kubernetes >= 1.12

- Only controllers can modify `.status`

The problem

- New Custom Resources work fine: Pending->Ready
- Changes to existing Custom Resources “succeeds” but does not apply changes



Kubernetes >= 1.12

```
apiVersion: airbnb.com/v1
spec:
  metadata:
     generation: 1
status:
  status: Ready
```

```
apiVersion: airbnb.com/v1
spec:
  metadata:
     generation: 2
status:
  status: Ready
```

When you `kubectl apply`, changes to `status` are ignored.

How do we know if we already acted on generation 2?

When is a custom resource deploy complete?

Halting problem

From Wikipedia, the free encyclopedia

This article includes a list of references, b… …as **insufficie**
this article by introducing more precise cita… …o remove this te

Seems like a pretty hard problem...

In computability theory, the **halting problem** is the problem of determining, from a description of an arbitrary computer program an
run forever.

Alan Turing proved in 1936 that a general algorithm to solve the halting problem for all possible program-input pairs cannot exist. F
"pathological" program *g* called with an input can pass its own source and its input to *f* and then specifically do the opposite of what
part of the proof was a mathematical definition of a computer and program, which became known as a Turing machine; the halting
one of the first cases of decision problems to be concluded. The theoretical conclusion that it is not solvable is significant to practica
programming invention can possibly perform perfectly.

@melaniecebula @brucesherrod4

Airbnb “Solution”

```
apiVersion: airbnb.com/v1
spec:
  metadata:
     generation: 2
status:
  observedGeneration: 1
```

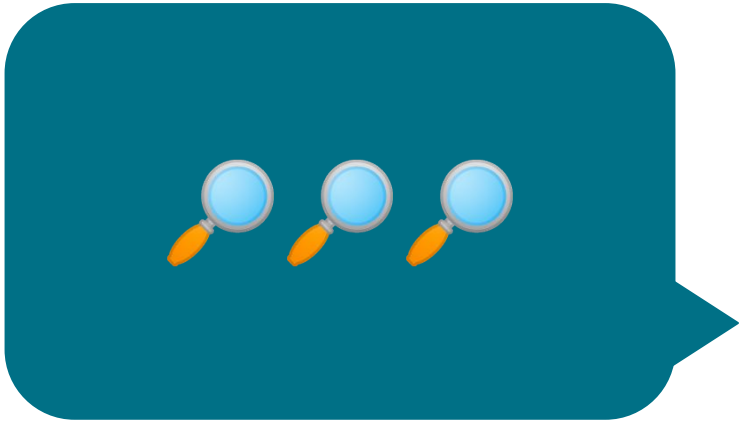
Controller adds `observedGeneration` to indicate what is done.

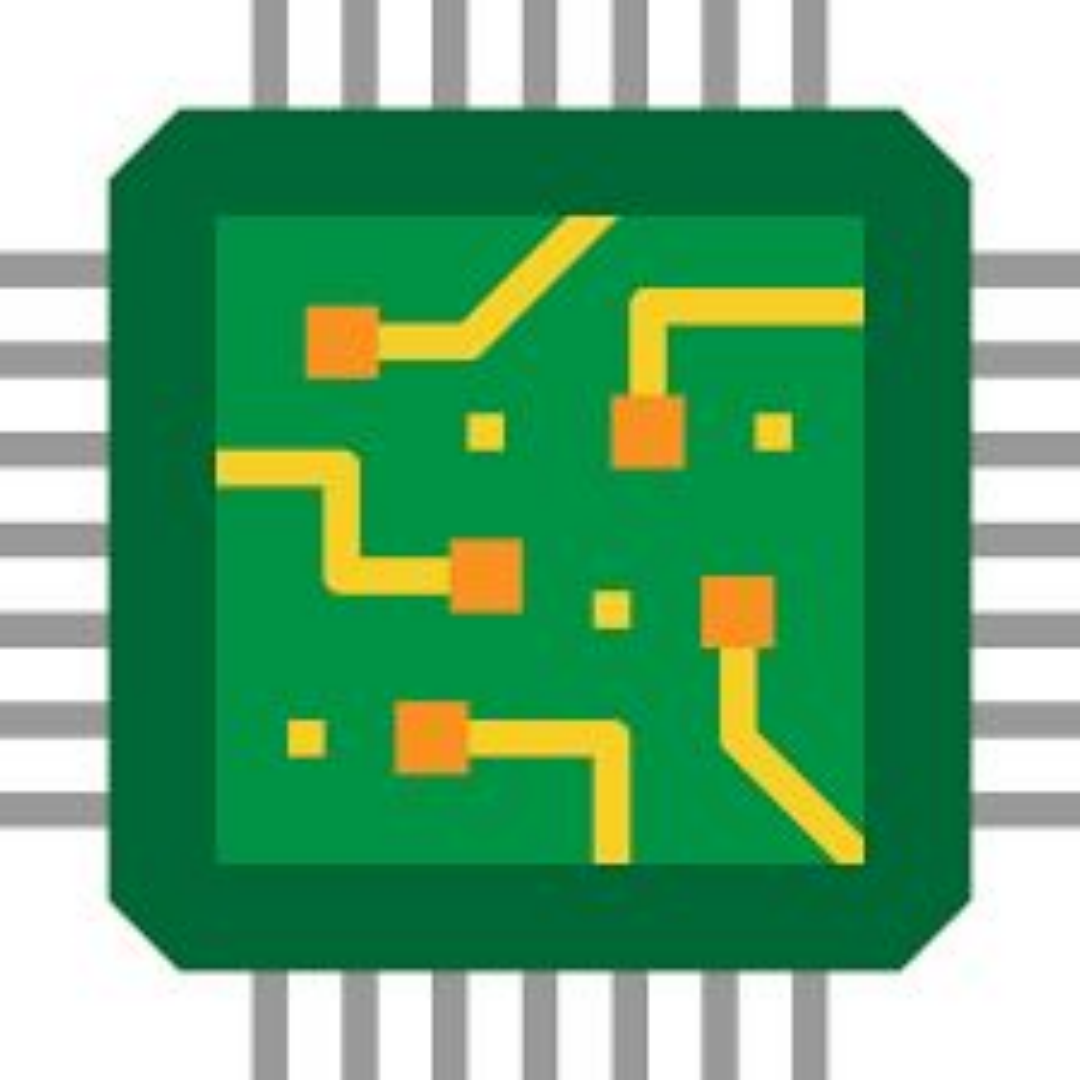
```
apiVersion: airbnb.com/v1
spec:
  metadata:
     generation: 2
status:
  observedGeneration: 2
  status: [Ready, Error]
```

Deployer (and users) check that `spec.metadata.generation == status.observedGeneration`

Takeaway

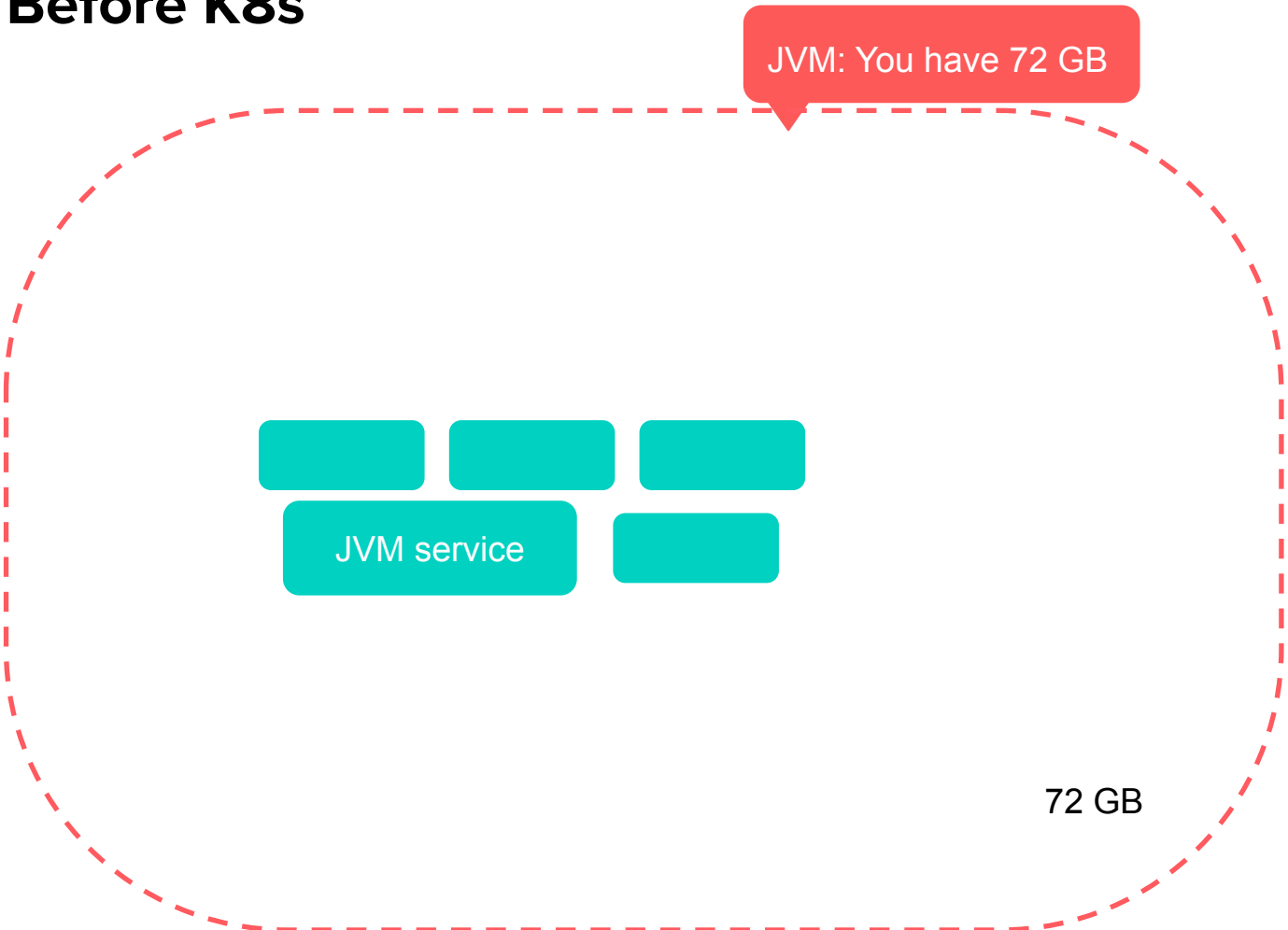
Knowing when a deploy is complete and if is succeeded or not is tricky





I can’t believe I have all the node’s resources

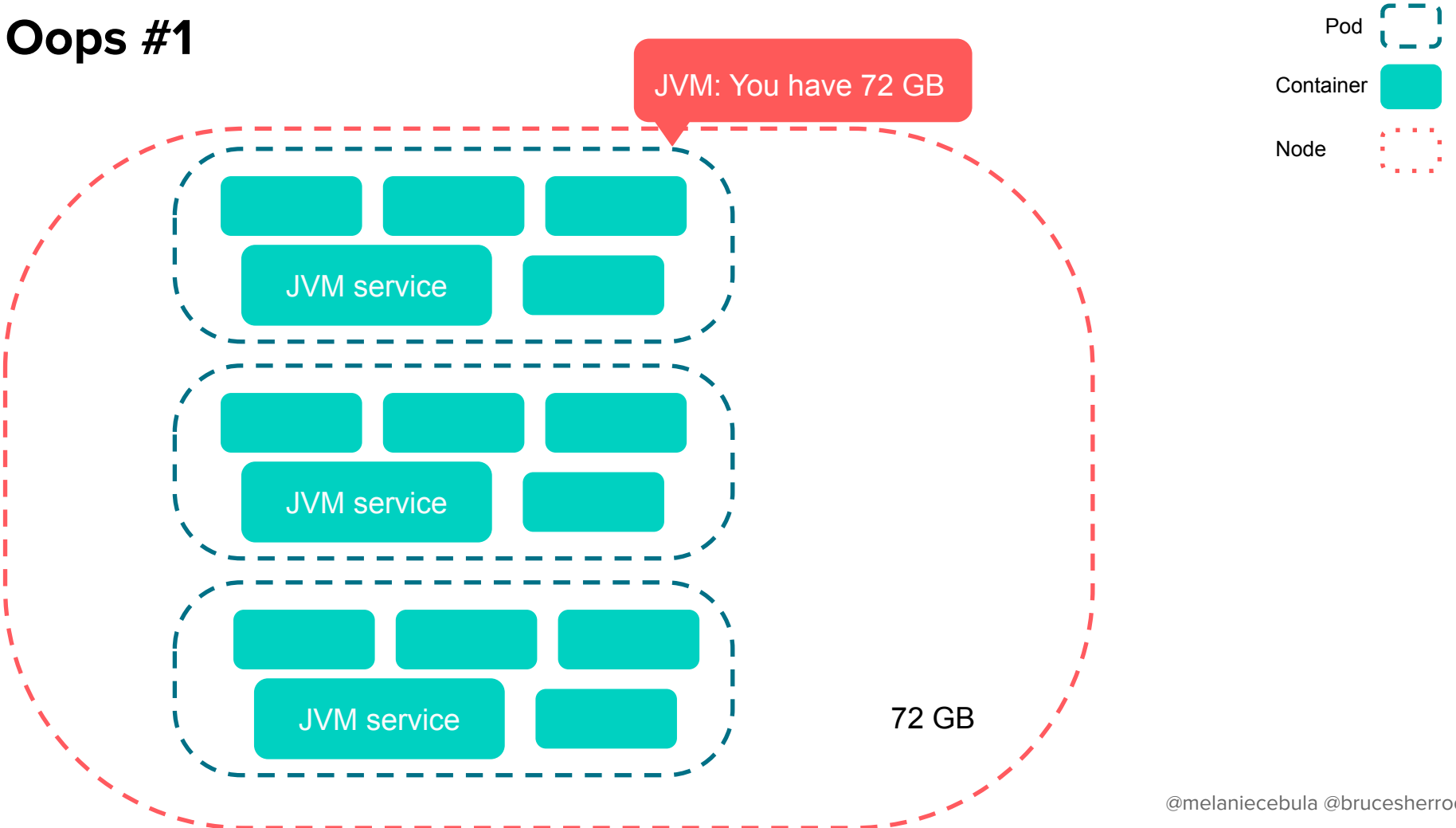
Before K8s



Runit processes

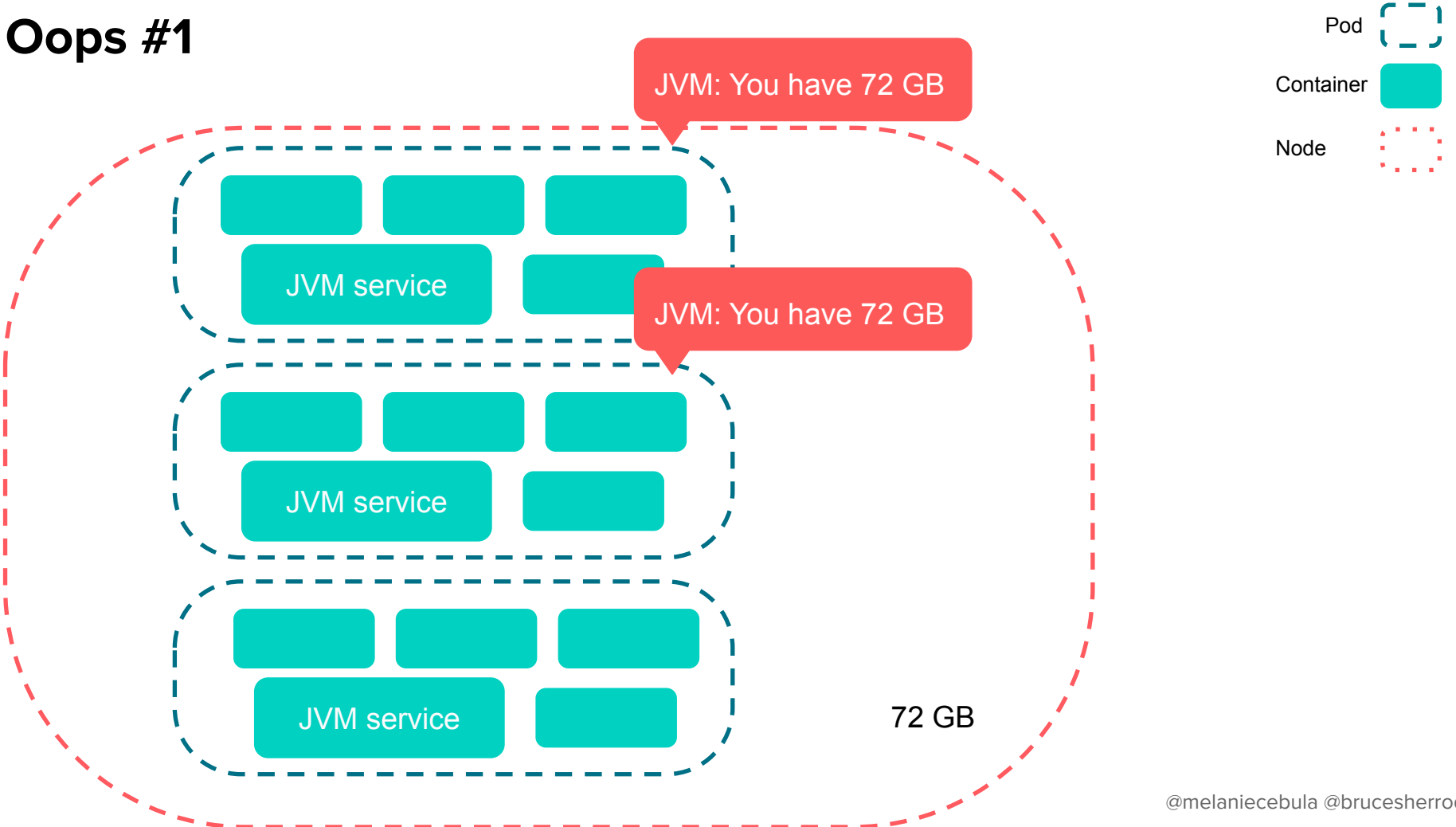
Instance

Oops #1



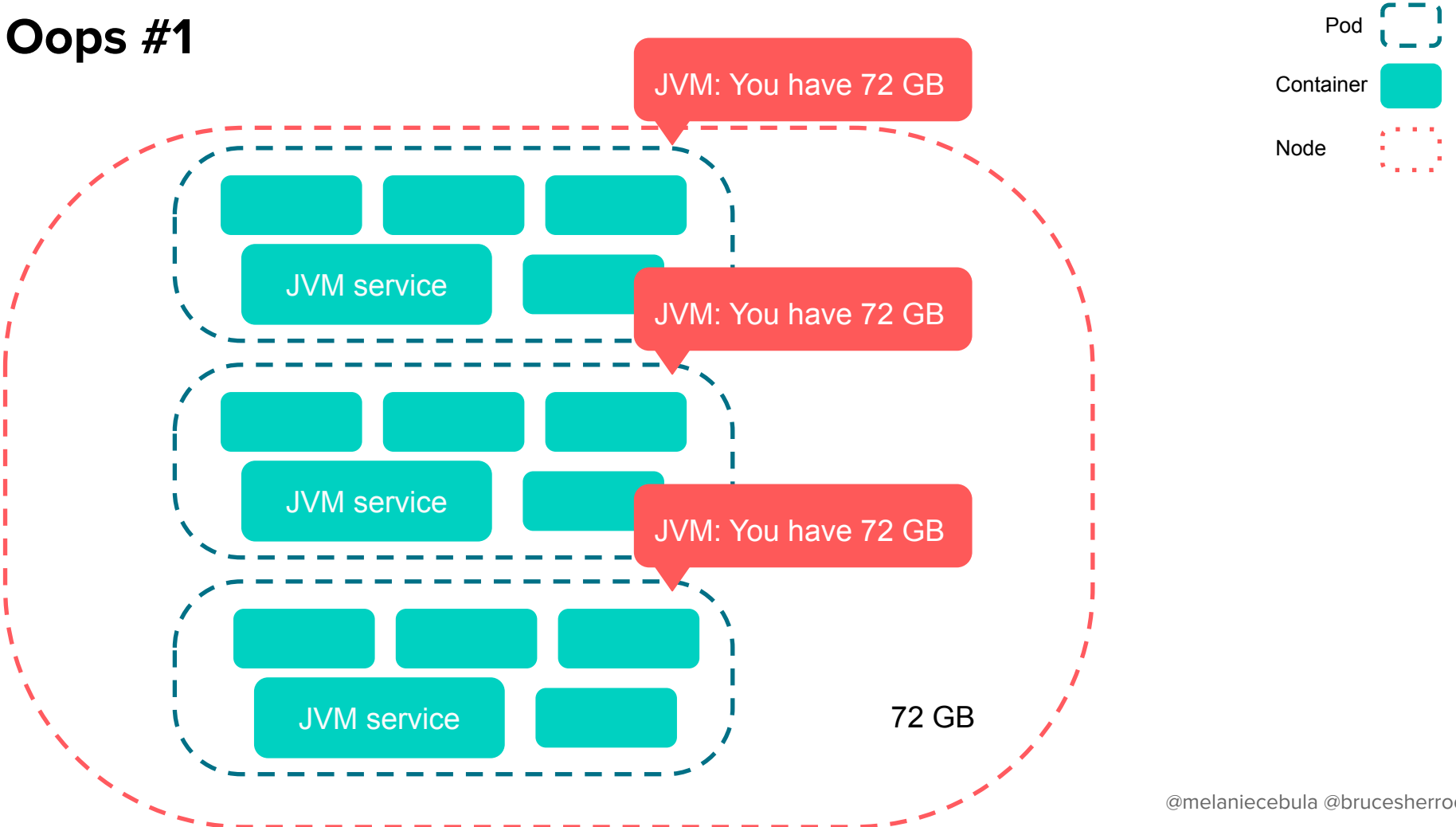
@melaniecebula @brucesherrod4

Oops #1



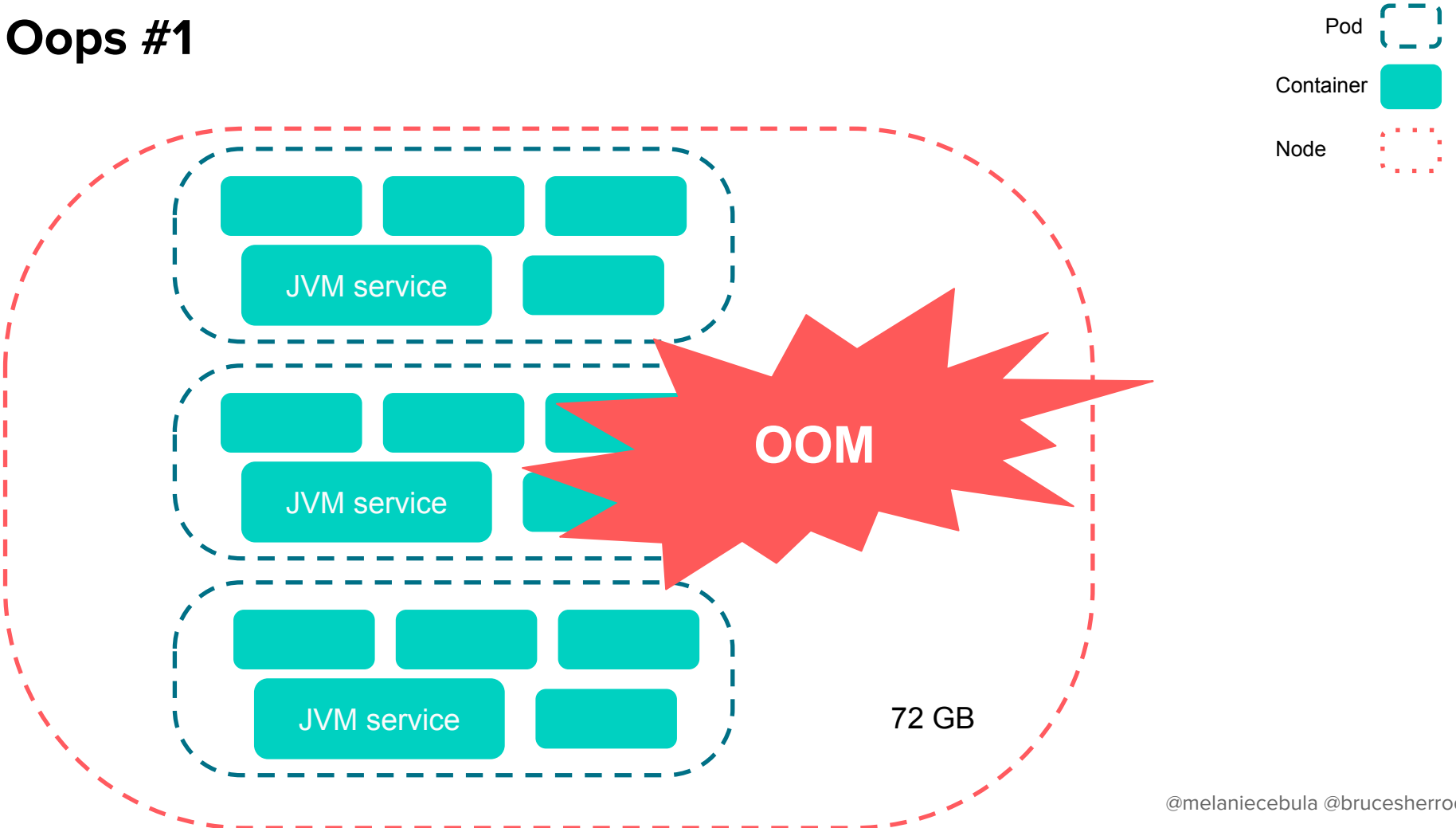
@melaniecebula @brucesherrod4

Oops #1



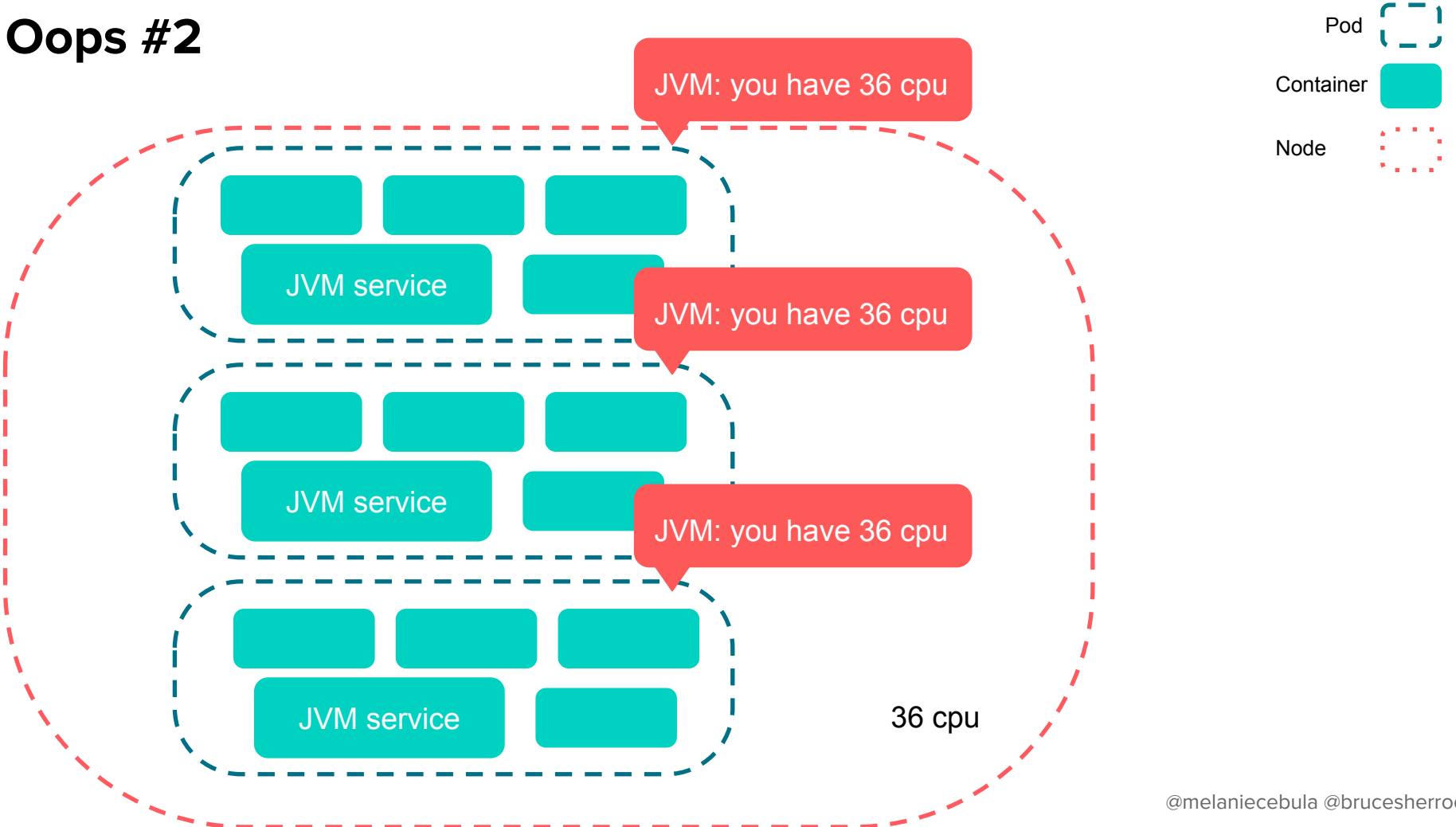
@melaniecebula @brucesherrod4

Oops #1



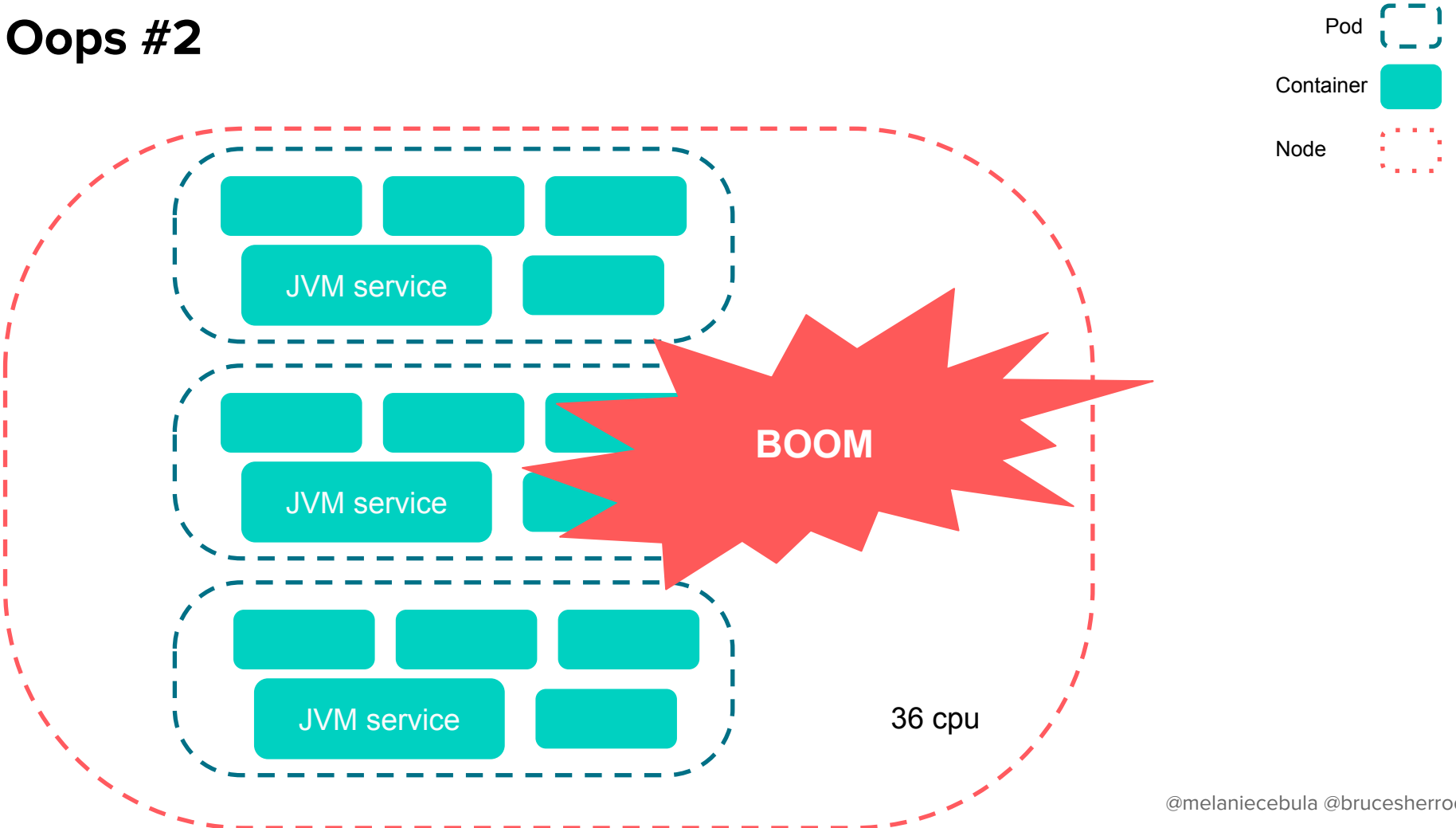
@melaniecebula @brucesherrod4

Oops #2



@melaniecebula @brucesherrod4

Oops #2



@melaniecebula @brucesherrod4

Soln

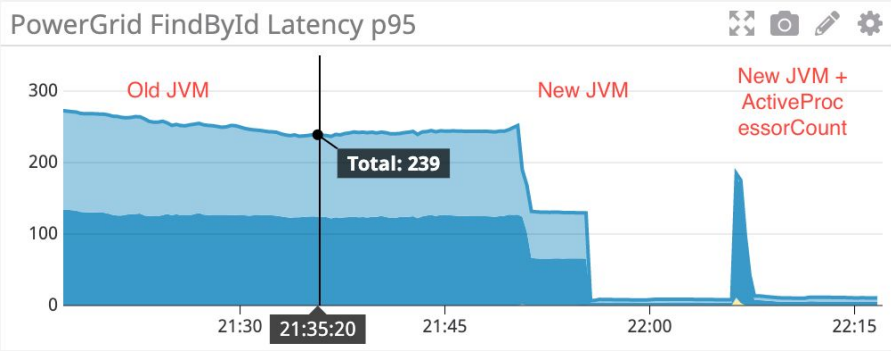
Older versions of the JDK allocate resources based on the underlying node and is unaware of cgroups

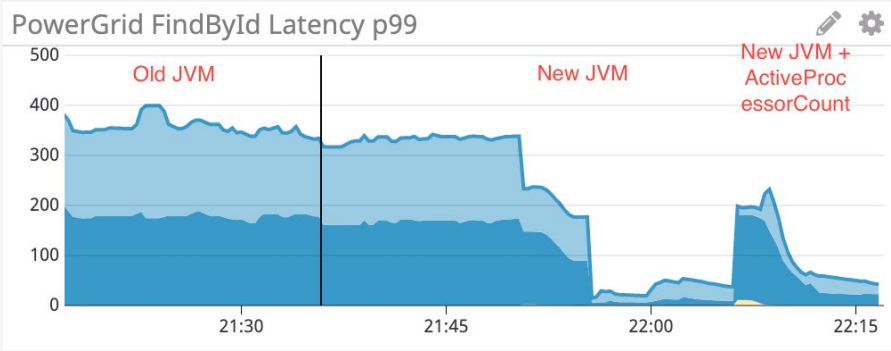
Soln

Older versions of the JDK allocate resources based on the underlying node and is unaware of cgroups

Let's upgrade the JDK. Then resources will be allocated based on the container!

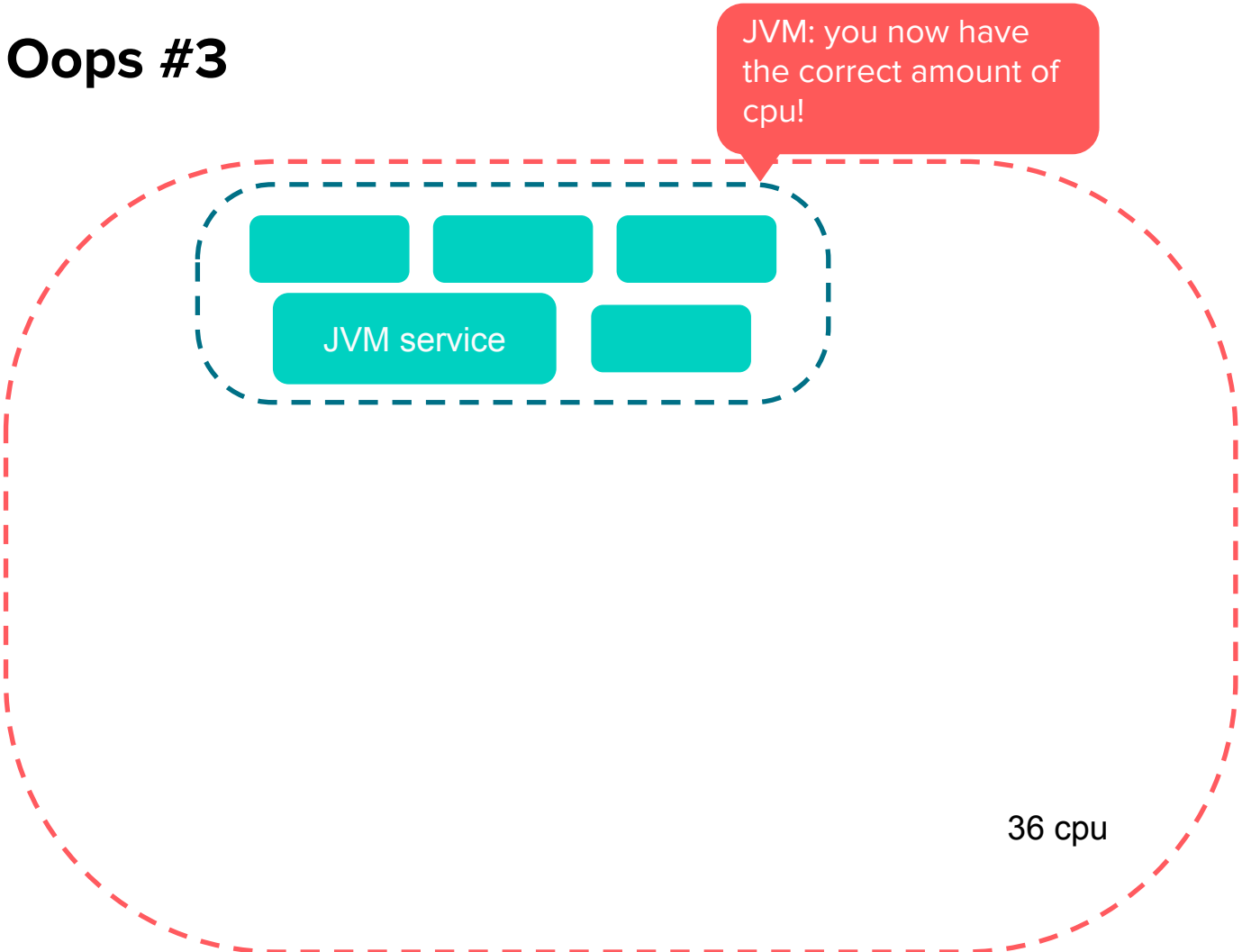
Soln

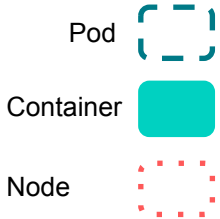




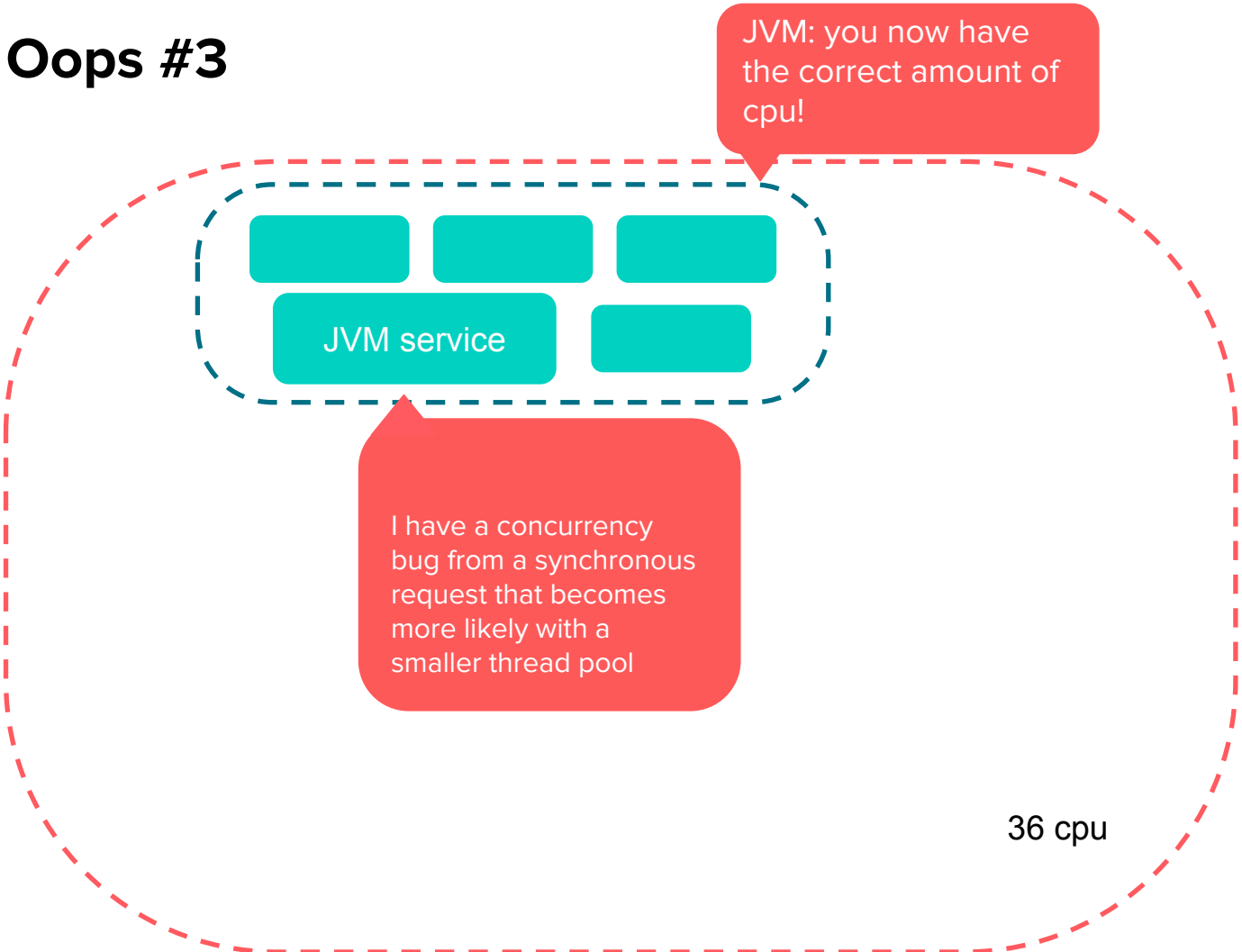
Also tried playing around with `-XX:ActiveProcessorCount` but it didn't have much of an effect

Oops #3


JVM: you now have the correct amount of cpu!
JVM service
36 cpu


Pod
Container
Node

Oops #3

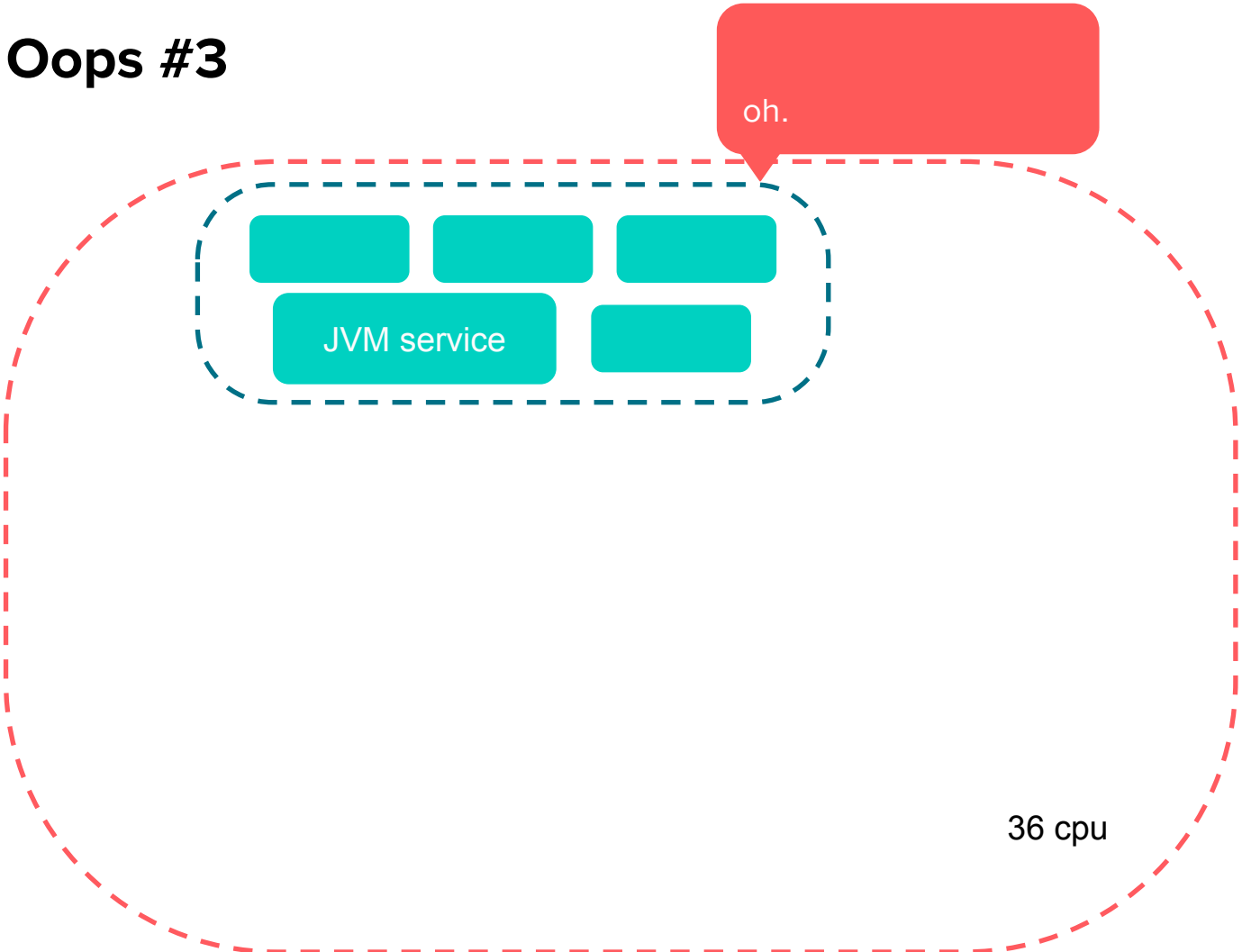


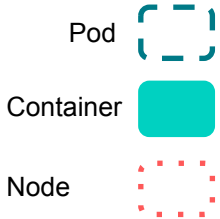
Pod

Container

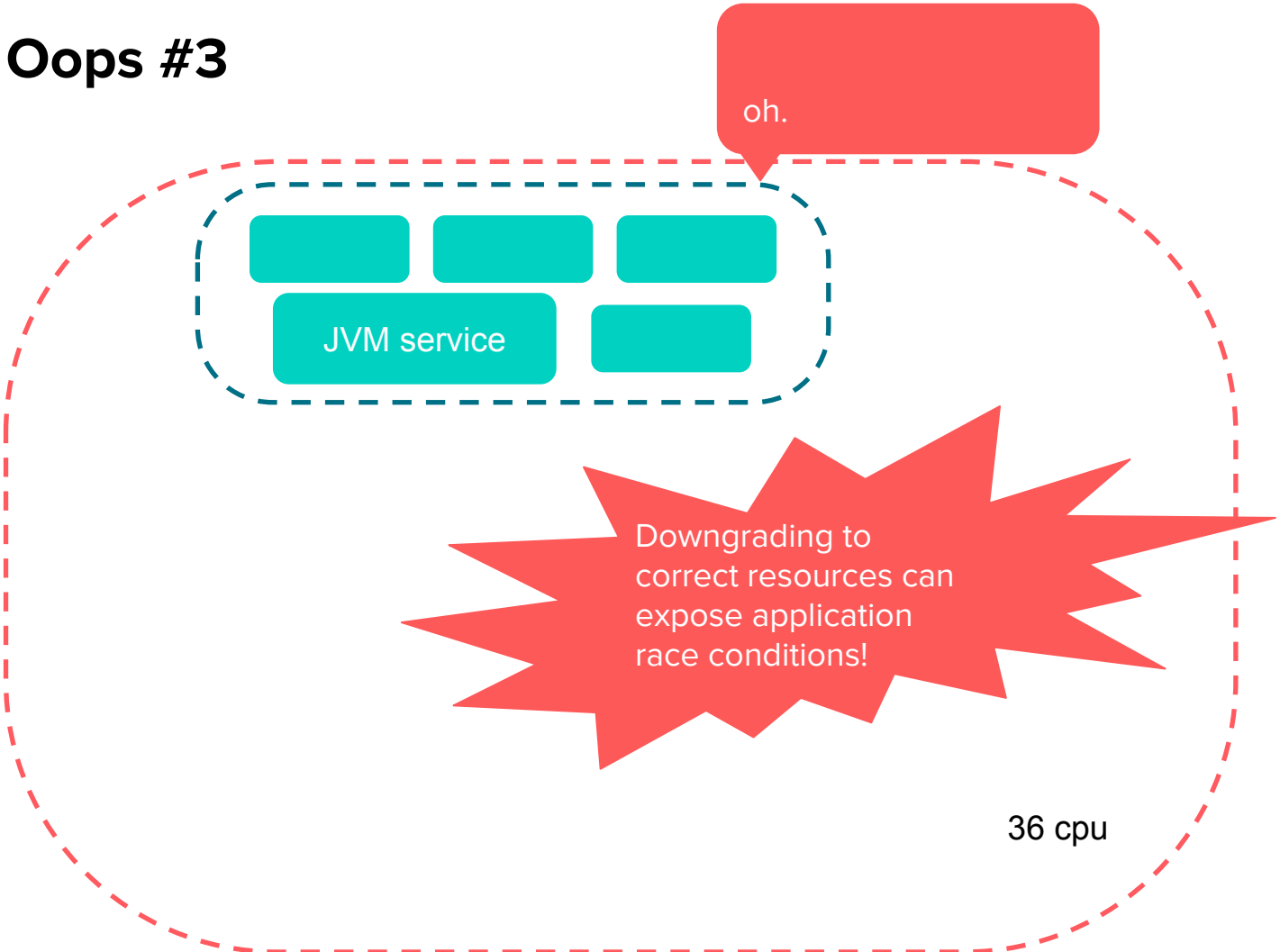
Node

Oops #3





Oops #3

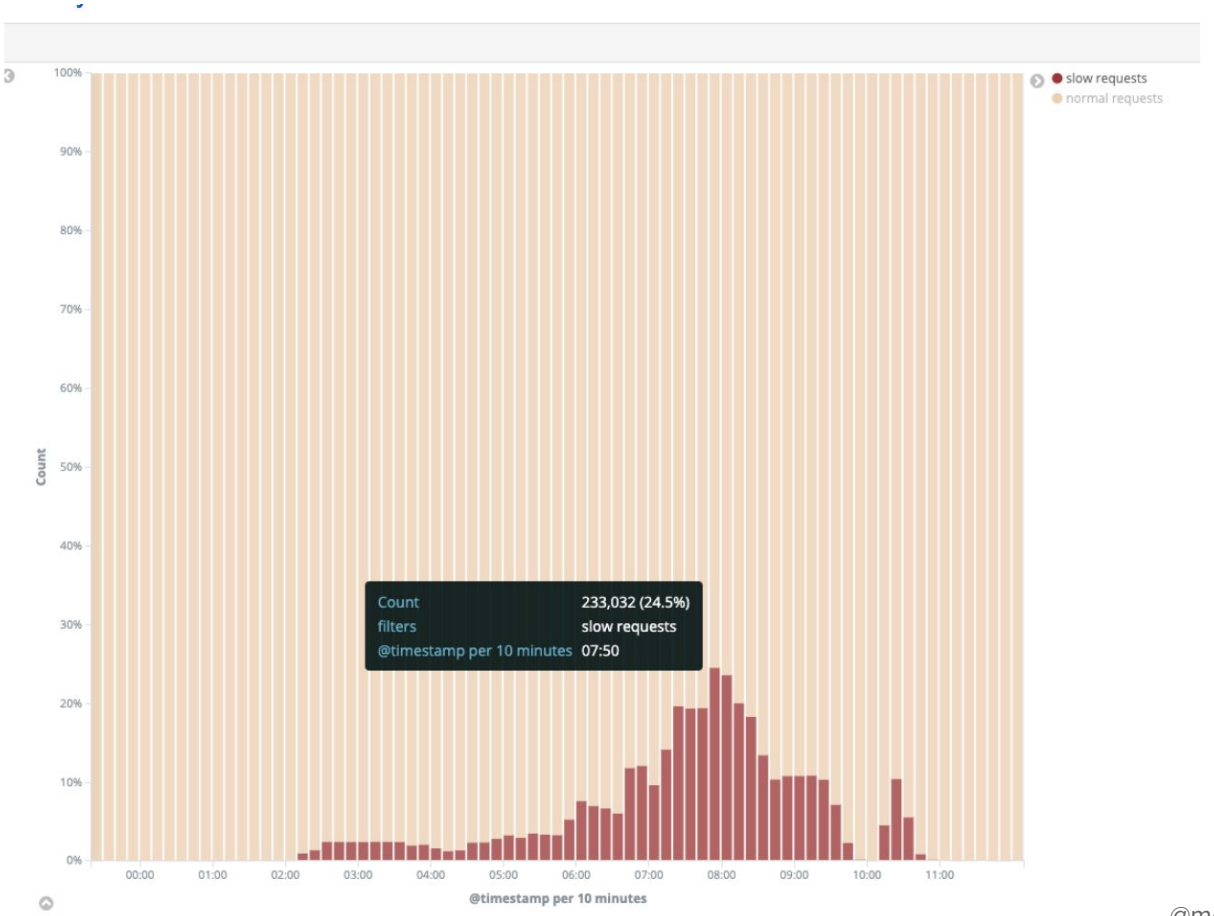


Pod

Container

Node

Oops #3



@melaniecebula @brucesherrod4

Soln

When upgrading the JDK, check for correct thread pool usage, synchronous thread-blocking calls, and other concurrency bugs with your multithreaded programs!

Soln

When upgrading the JDK, check for correct thread pool usage, synchronous thread-blocking calls, and other concurrency bugs with your multithreaded programs!

👌

This is not specific to Java services!

- “Yeah yeah, everyone knows older JVMs are not aware of cgroups”
- Similar pitfalls can happen in other language frameworks and sidecars

Envoy sets concurrency to # cpus on the underlying host by default

This is not specific to Java services!

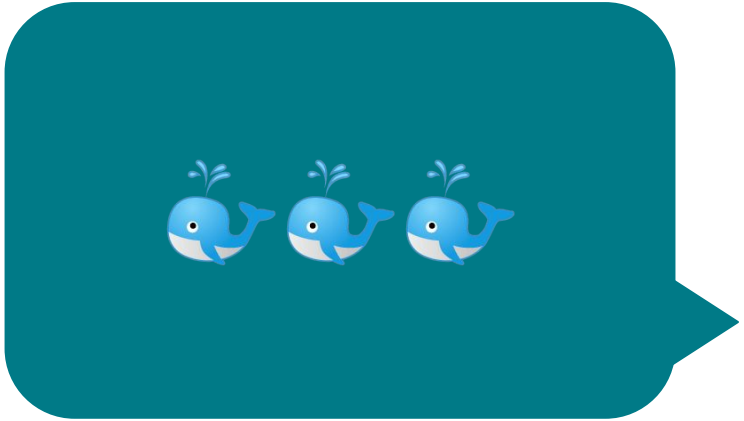
- “Yeah yeah, everyone knows older JVMs are not aware of cgroups”
- Similar pitfalls can happen in other language frameworks and sidecars

Envoy sets concurrency to # cpus on the underlying host by default

That’s going to cause contention on the host!

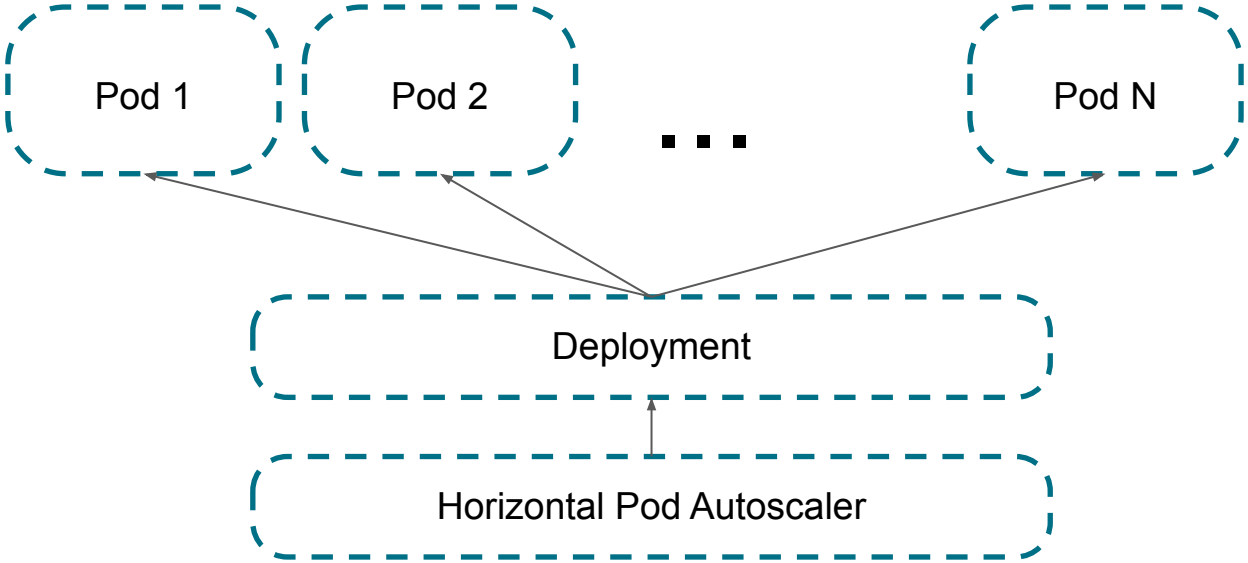
Takeaway

Beware of languages frameworks and sidecars that are not aware of container abstractions

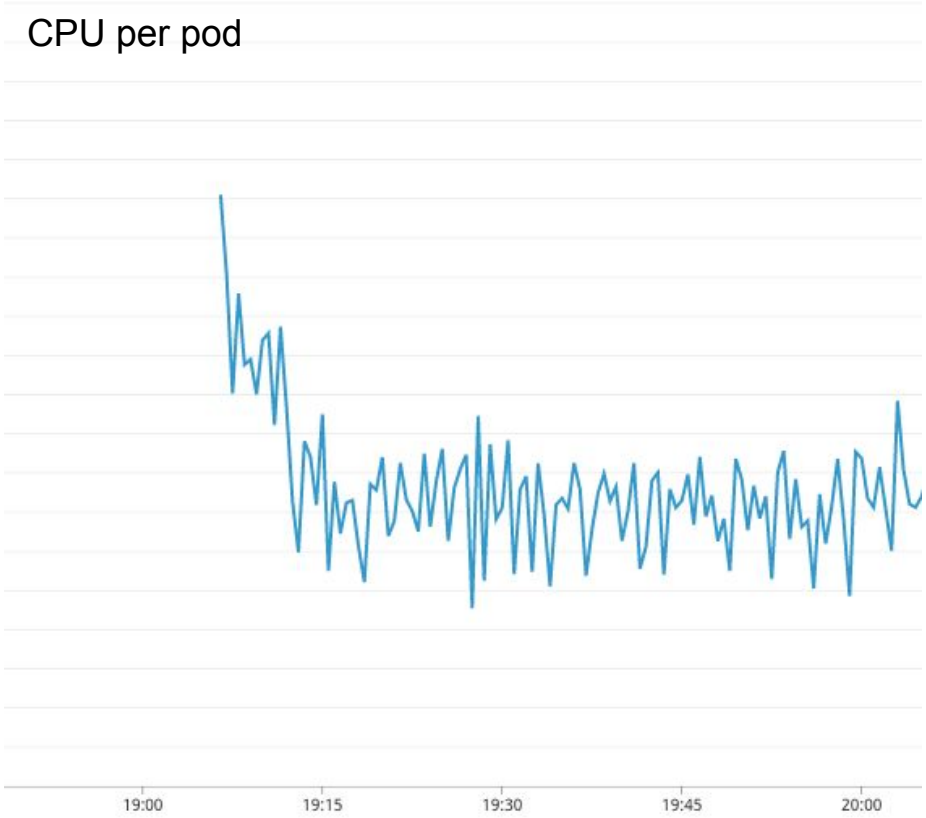


Autoscale -ocalypse

Autoscaling is Great!



Suppose you have this..



Autoscaling not so great

How do you fix this?

HPA Configuration? Nope!

"Starting from v1.12, a new algorithmic update removes the need for the upscale delay."

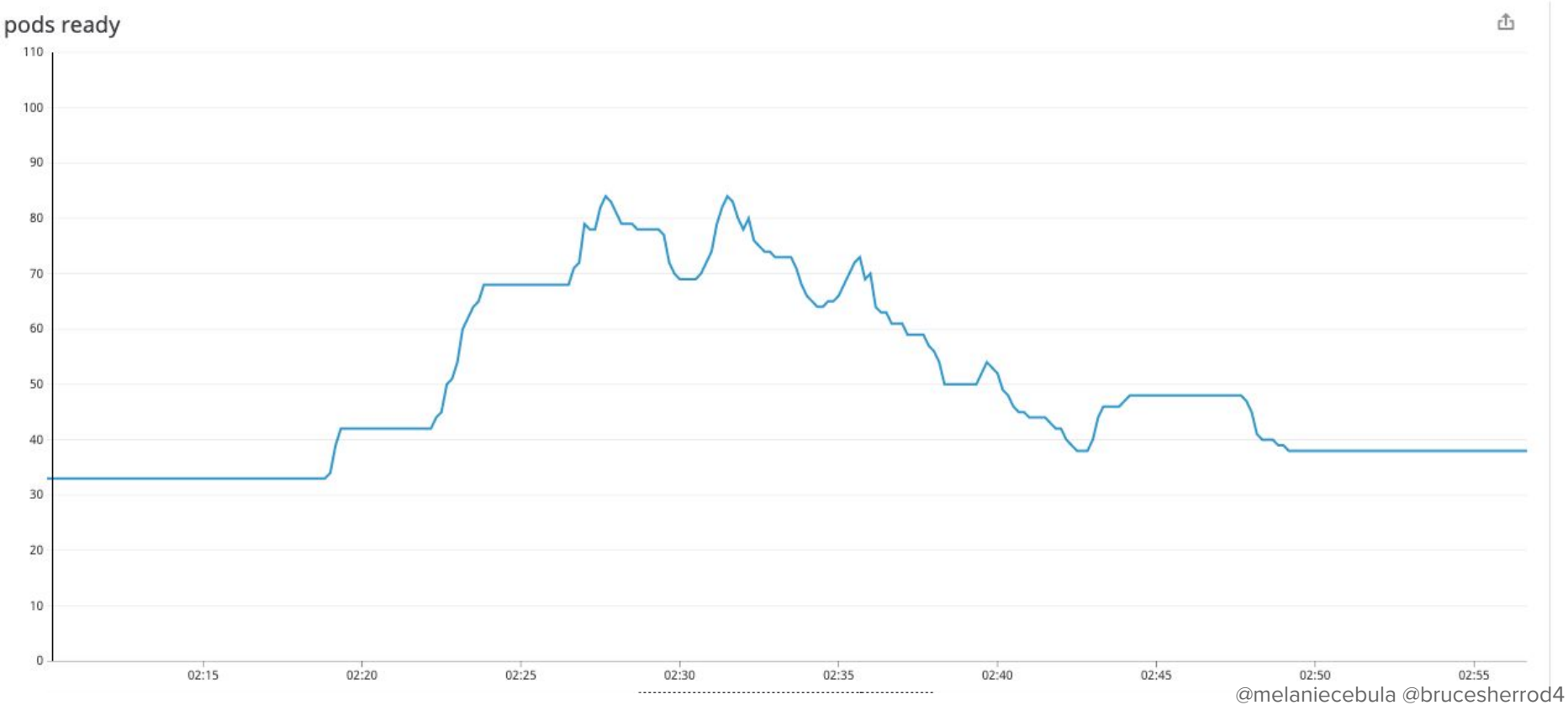
Disagree!

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#support-for-cooldown-delay

Embarrassing Hack

```
25      # update (Tue Jun 18 16:42:14 PDT 2019): added horizontal-pod-autoscaler-sync-period to force slower scaleups
26      - --horizontal-pod-autoscaler-initial-readiness-delay=300s
27      - --horizontal-pod-autoscaler-sync-period=300s
28      - --leader-elect=true
29      - --master=http://127.0.0.1:80
```

Better but not fixed



Takeaway

Autoscaling does not work well for services that burn CPU on start

Hey, my scheduled operation took down all services

FYI, we are running an operation to upgrade Statsd Agent in Kubernetes in a few minutes. There may be a few minutes where metrics are not available.

Uhh.. I think all Kubernetes services are down right now...

Kubernetes “Health Checks”

Pod

Container

Readiness Probe: “Don’t send traffic to me”

Liveness Probe: “Replace me”

Service

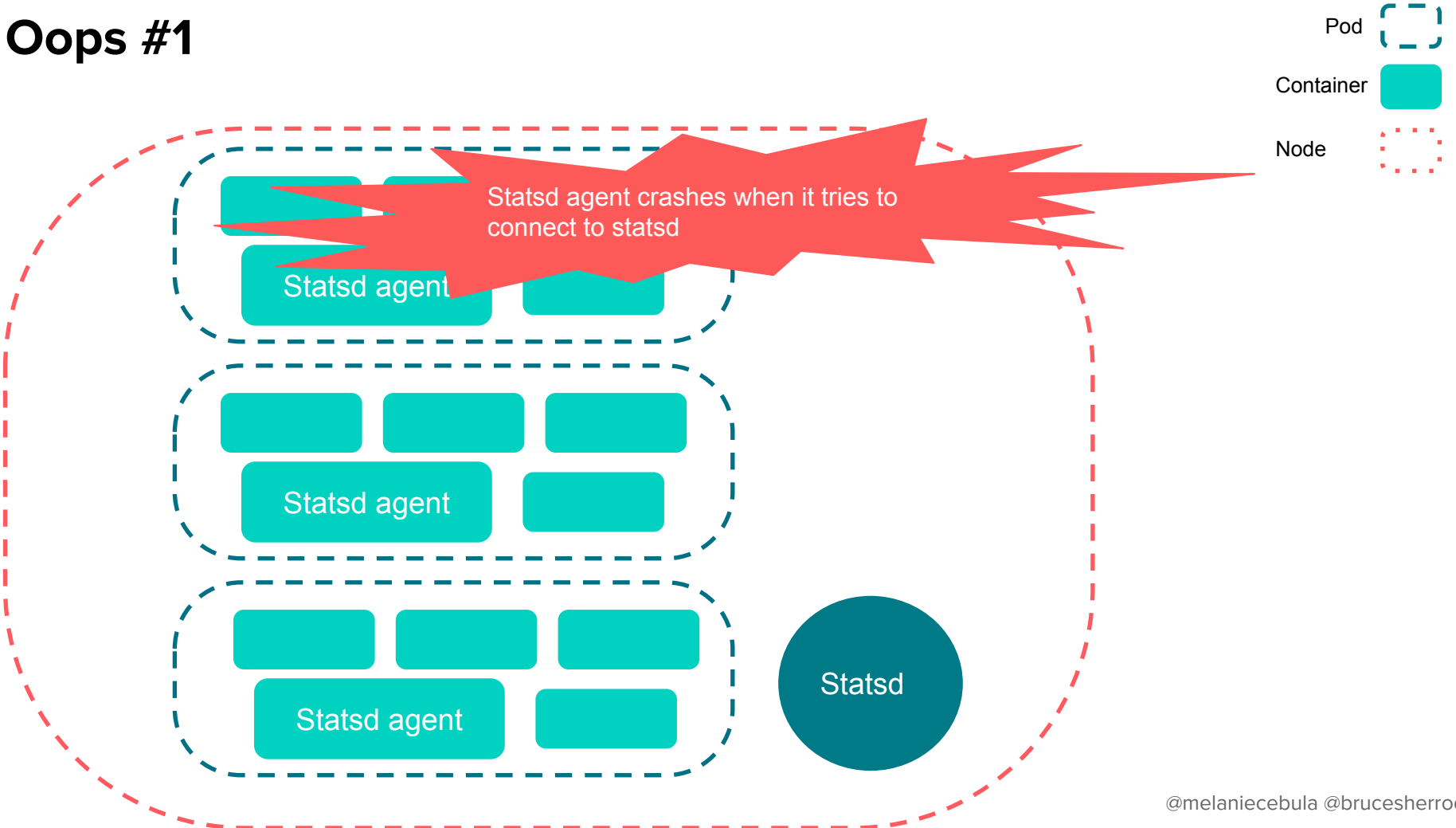
/health

Health checking is per container

Try 1: Don’t set probes for non-critical containers

- Non-critical containers such as statsd-agent should not affect service health
- Easy, just don’t set readiness probes for these containers!

Oops #1



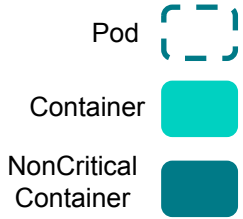
@melaniecebula @brucesherrod4

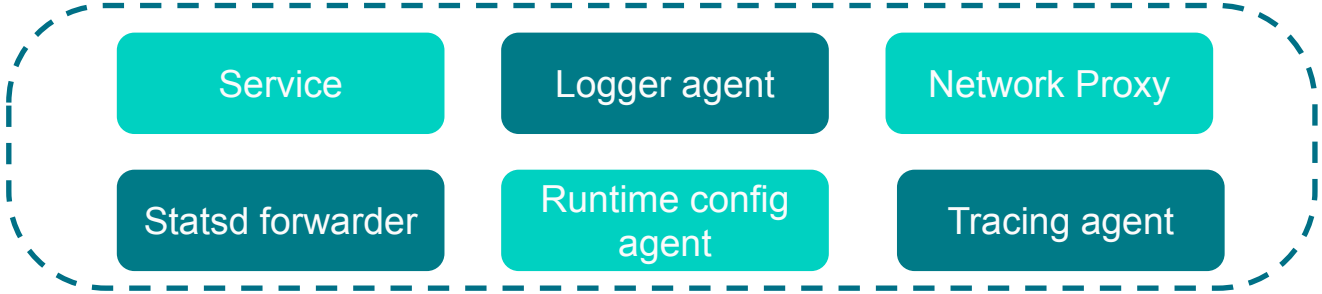
Oops #2

```
isPodReady := true

for each containerStatus in pod:
  if !containerStatus.Ready:
    isPodReady = false

if isPodReady:
  publishPodIsReady()
```

Pod only receives traffic if all containers are Ready

Soln: Some containers are more equal than others



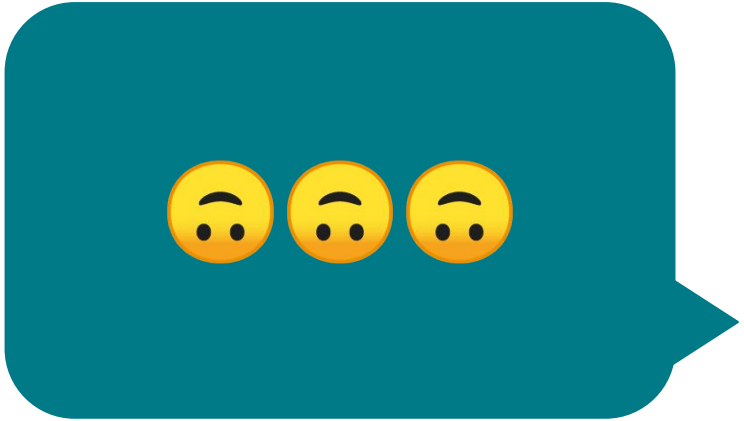


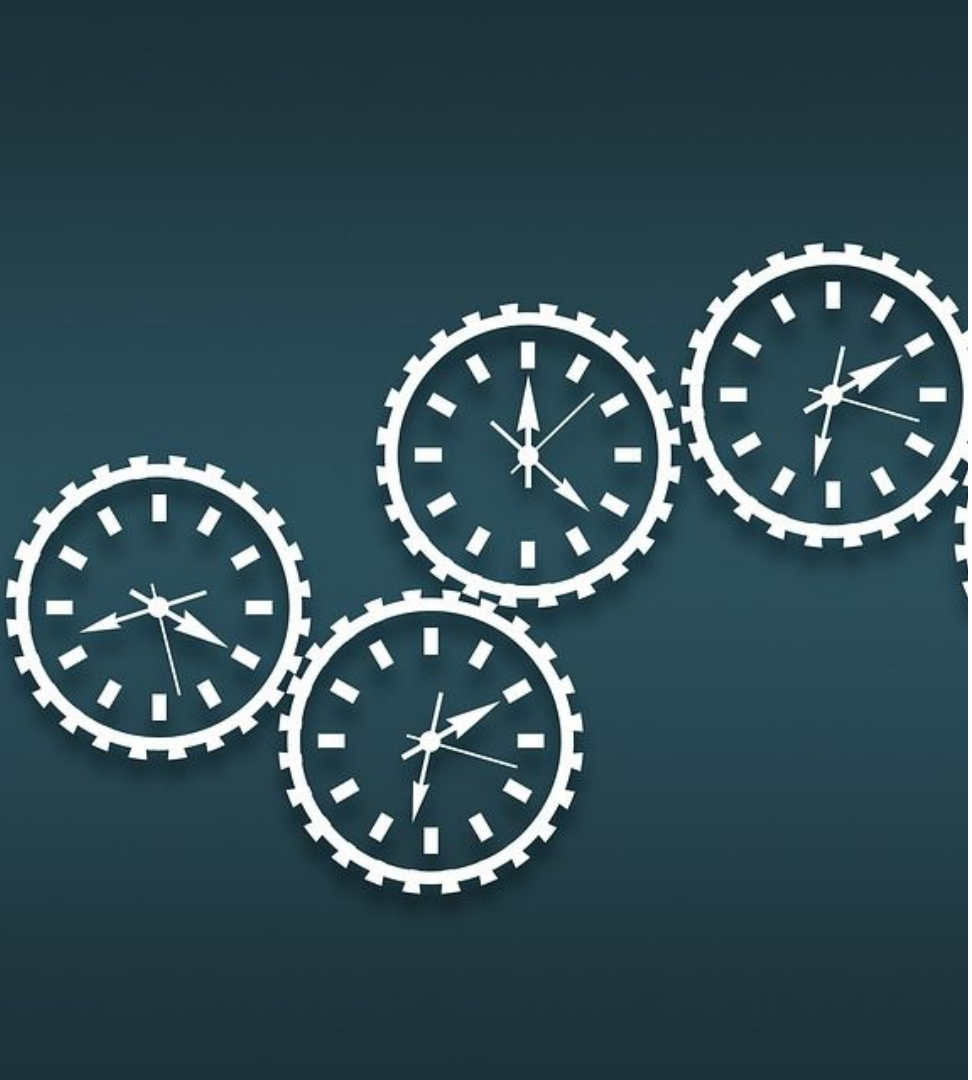
Soln: Some containers are more equal than others

```
isPodReady := true

for each containerStatus in pod:
  continue if container.isNonCritical()
  if !containerStatus.Ready:
    isPodReady = false

if isPodReady:
  publishPodIsReady()
```

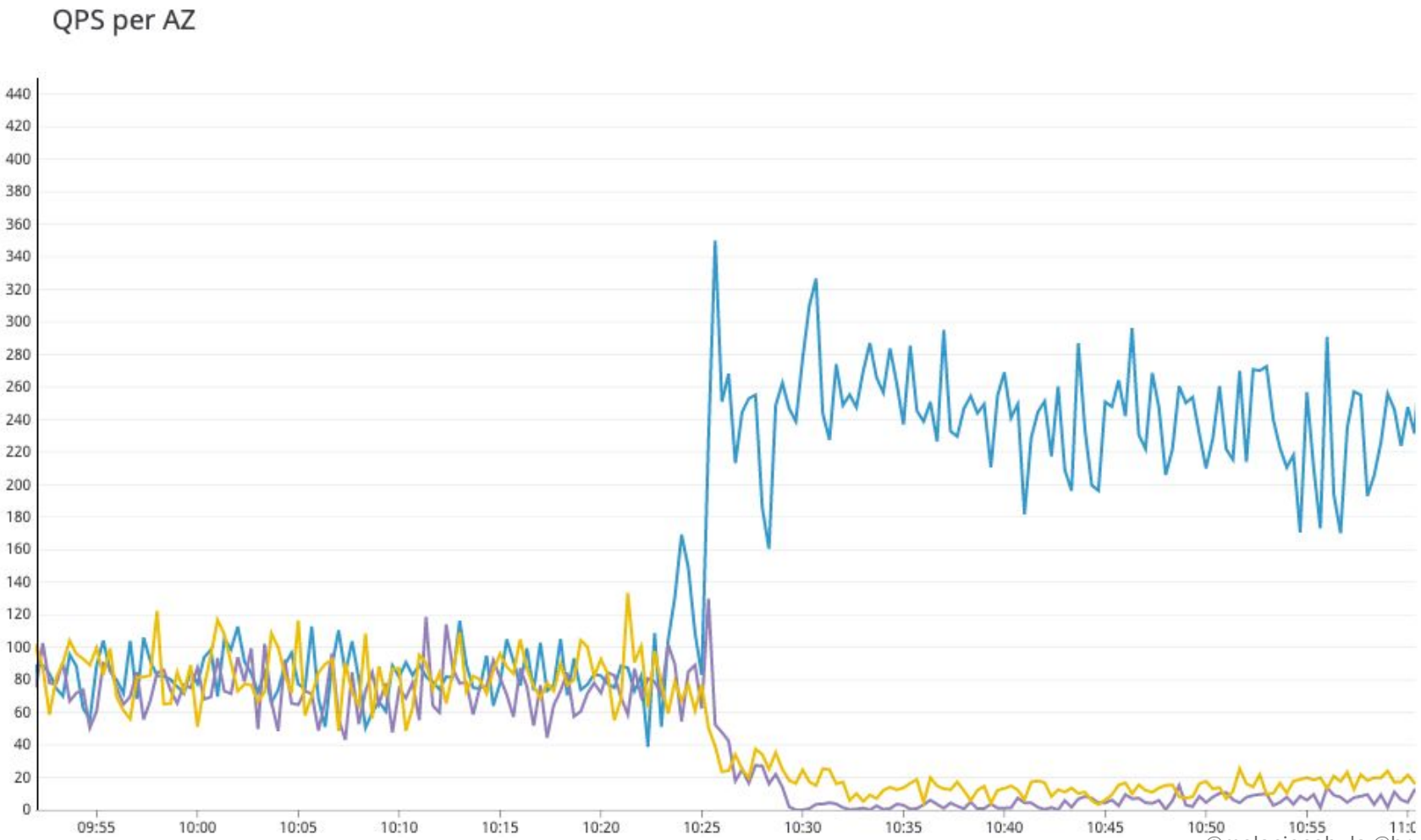
Takeaway

Be careful when configuring health checks for your pods, especially with crashing sidecar containers



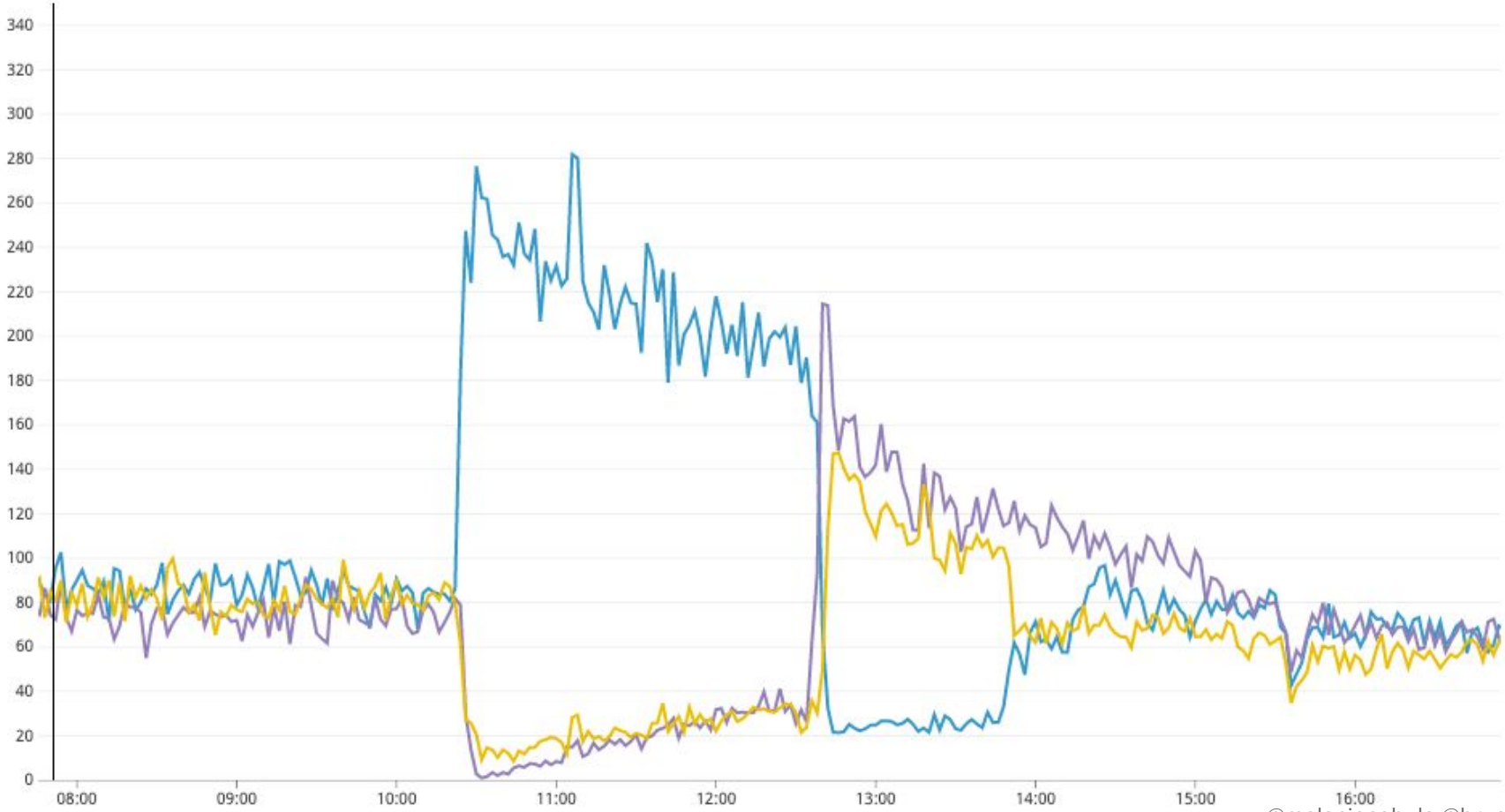


Scheduling is easy and fun



@melaniecebula @brucesherrod4

Deployment “Pruner” Controller - Better but not enough better



@melaniecebula @brucesherrod4

Solve it in K8s!!

kubernetes / kubernetes

Watch 3.1k Sta

Code | Issues 2,275 | Pull requests 1,100 | Projects 9 | Security | Insights

Disable matching on few selectors. Remove duplicates. #72801

Merged k8s

Conversatio

kubernetes / kubernetes

Watch 3.1k Star

Code | Issues 2,275 | Pull requests 1,098 | Projects 9 | Security | Insights

In SelectorSpreadPriority, consider all pods when scoring for zone #73711

Open Ramyak wants to merge 1 commit into `kubernetes:master` from `Ramyak:ramya/deleted-pods-selector-spread`

Conversation 24 | Commits 1 | Checks 0 | Files changed 8

@melaniecebula @brucesherrod4

Whew! No forked k8s necessary

```
metadata:
  name: kube-scheduler
  namespace: kube-system
spec:
  containers:
  - name: kube-scheduler
    image: our-custom-scheduler
```

Takeaway

You may need to make fixes to the Kubernetes scheduler, but you can easily upload them as a custom image!

10 takeaways

1. Solve your problem at the appropriate abstraction level-- and that may be patching your behavior into kubernetes itself! 😱
2. Kubernetes deploys cycles hardware super fast, whether the rest of your infrastructure can keep up or not! 🏃‍♀️
3. Daemonsets can take down a cluster in a way that other workloads can only can aspire to 👾
4. Why can't I hold all these docker images? Make sure to keep track of them! 💁‍♀️
5. You might need to support ordering between sidecars too! 🚗
6. Knowing when a deploy is complete and if is succeeded or not is tricky 🔍
7. Beware of languages frameworks and sidecars that are not aware of container abstractions 🐳
8. Autoscaling does not work well for services that burn CPU on start 🔥
9. Be careful when configuring health checks for your pods, especially with crashing sidecar containers 🙃
10. You may need to make fixes to the Kubernetes scheduler, but you can easily upload them as a custom image! 👌

@melaniecebula @brucesherrod4

Thanks!

- Learn more @ medium.com/airbnb-engineering
- Jobs @ airbnb.com/careers
- Contact us @melaniecebula @brucesherrod4

Tuesday, November 19 • 4:25pm - 5:00pm

Scaling Kubernetes to Thousands of Nodes Across Multiple Clusters, Calmly - Ben Hughes, Airbnb

Wednesday, November 20 • 11:50am - 12:25pm

Did Kubernetes Make My p95s Worse? - Jian Cheung & Stephen Chan, Airbnb