10 More Ways to Blow Up Your Kubernetes

Jian Cheung & Joseph Kim • KubeCon Nov 2020

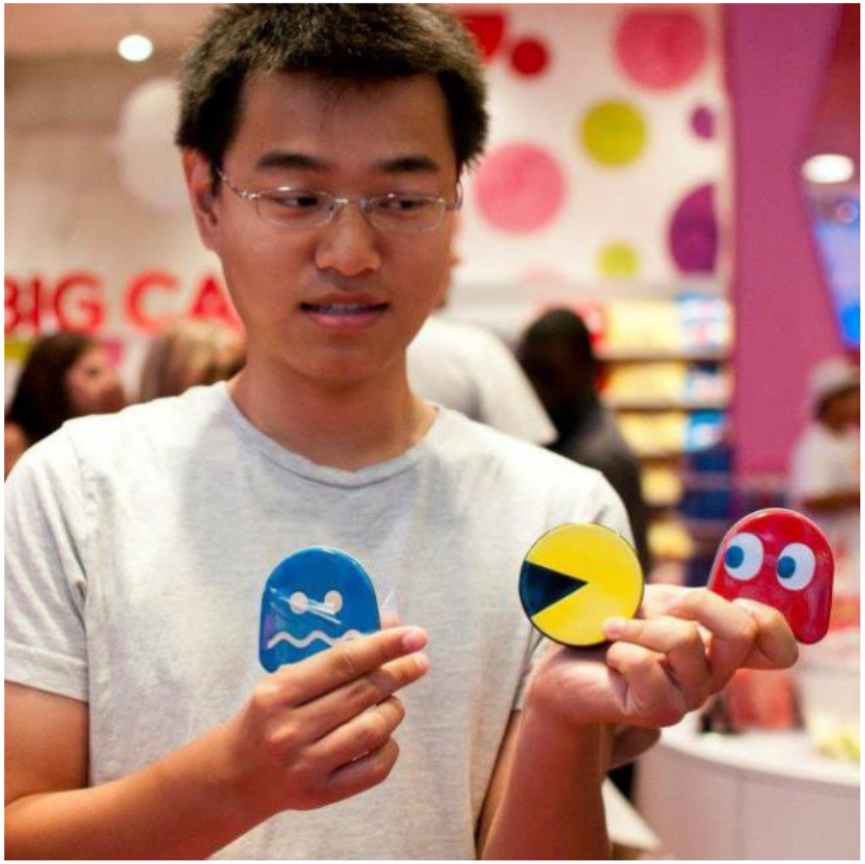
We found more ways since last year's talk

@jiancheung & Joseph Kim

Who are we?

Who are we?

Hi, I'm Jian!



Hi, I'm Joseph!

@jiancheung & Joseph Kim

Outline

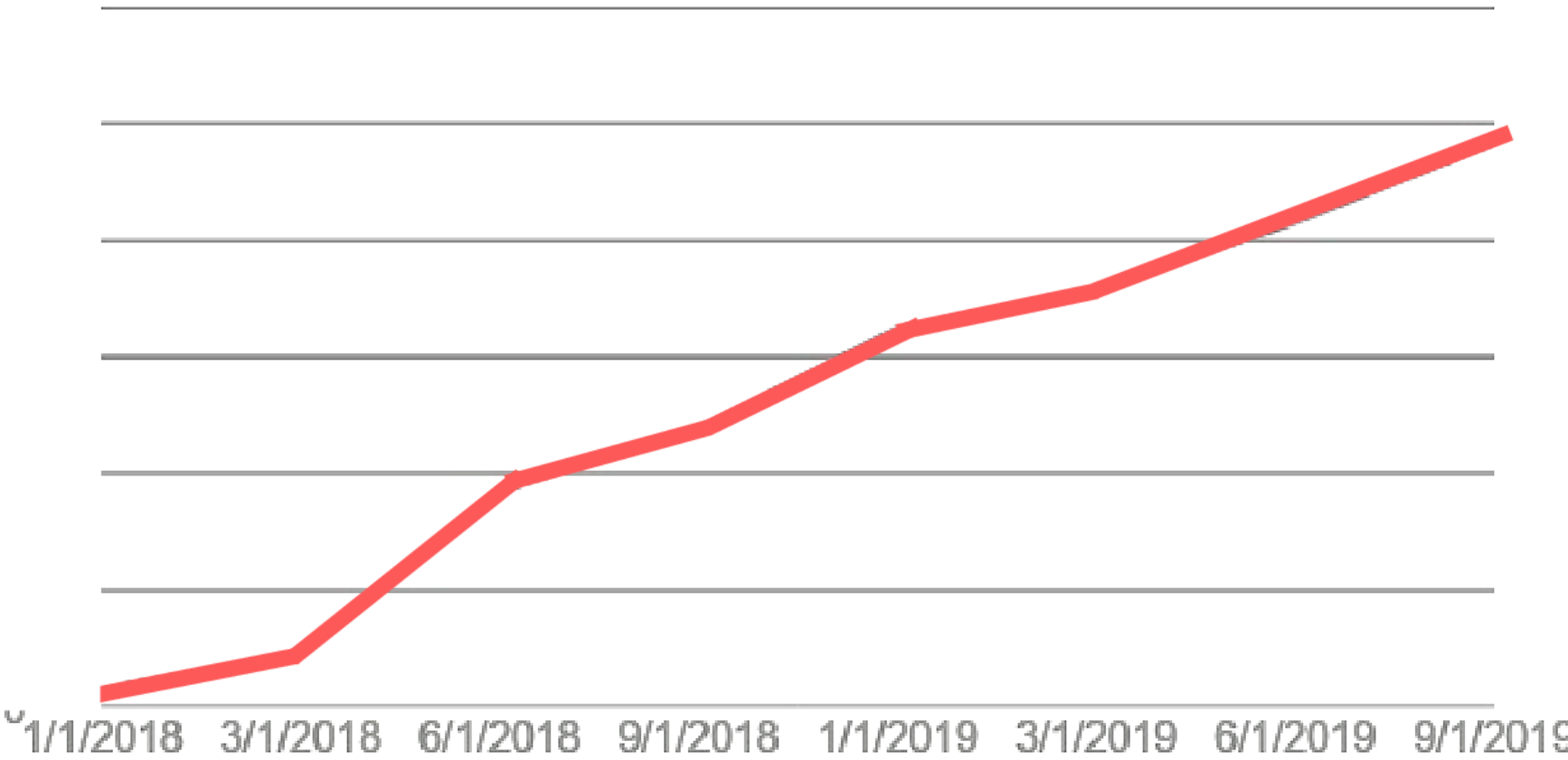
- Brief intro of Kubernetes at Airbnb
- Dive in to 10 cases
- Recap

Kubernetes And Containers at Airbnb

@jiancheung & Joseph Kim

Kubernetes @Airbnb

SERVICES



@jiancheung & Joseph Kim

Airbnb Kubernetes Environment

- Amazon Linux 2
- Ubuntu Images
- Canal (Calico + Flannel) CNI plugin
- Nodeport services/Smartstack
- Many languages (ruby, java, python, go, etc)

Now let’s dive into the real fun stuff...

Replicating ReplicaSets

Context

- We have a job that regularly scrapes our clusters
  - kubectl get pods --all-namespaces -o yaml
- It starts OOMing in one specific cluster (test-mc-e). So let's check it out...

```
$ kubectl --context=prod-h  get rs --all-namespaces | wc -l
    807
$ kubectl --context=prod-a  get rs --all-namespaces | wc -l
    496
$ kubectl --context=test-mc-a  get rs --all-namespaces | wc -l
   1530
$ kubectl --context=test-mc-e  get rs --all-namespaces | wc -l
```

Any guesses first?

Context

- We have a job that regularly scrapes our clusters
  - kubectl get pods --all-namespaces -o yaml
- It starts OOMing in one specific cluster (test-mc-e). So let's check it out...

```
$ kubectl --context=prod-h  get rs --all-namespaces | wc -l
    807
$ kubectl --context=prod-a  get rs --all-namespaces | wc -l
    496
$ kubectl --context=test-mc-a  get rs --all-namespaces | wc -l
   1530
$ kubectl --context=test-mc-e  get rs --all-namespaces | wc -l
   56292   # all of these are actually under 1 deployment / 1 namespace
```

What else is up with this cluster

```
$ kubectl --context=test-mc-e  get rs --all-namespaces | wc -l

   56292   # all of these are actually under 1 deployment / 1 namespace
```

29 days ago

At the time of this writing

```
kubectl --context=test-mc-e get deployments --namespace=autoscaler-overprovisioning overprovisioned-pause-pod-c5d9xl -o
json | jq .status.collisionCount

102295
```

Likely that the Deployment object is creating ReplicaSets that aren't getting "adopted" by it. So it keeps creating more.

Let’s do more diff’ing

```
<       topologySpreadConstraints:
<       - labelSelector:
<           matchLabels:
<             run: overprovisioned-pause-pod-
<         maxSkew: 1
<         topologyKey: failure-domain.beta.kubernetes.io/zone
<         whenUnsatisfiable: DoNotSchedule
```

This specific cluster was testing out topologySpreadConstraints.

These specs were **not** getting picked up by the ReplicaSets.

The Fix

@jiancheung & Joseph Kim

Takeaway

- Do test new features in test clusters

Mutating Time Bomb

Mutating Time Bomb

178 replies

at 2:10:54 AM

6 months ago

banana is experiencing high error rate. The service cannot be redeployed / scaled up due to K8s error: `Error creating: Internal error occurred: Internal error occurred: Unable to access invalid index: 5`

Mutating Time Bomb

```
01:43:23]  INFO Run: kubectl --context=prod-f get pods --namespace=
NAME                                    READY   STATUS    RESTARTS   AGE
banana-production                       11/11   Running   0          10d
banana-production                       11/11   Running   0          6d8h
```

```
[01:45:41.984] banana (master) $ kubectl --context=prod-f --namespace=  get hpa
NAME                          REFERENCE                        TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
Banana-production-autoscaler  Deployment/ Banana-production    234%/50%   3         24        9          91d
```

@jiancheung & Joseph Kim

1 deployment, 3 replicasets??

```
[01:52:49.460] banana (master) $ kubectl --context=prod-f --namespace= banana-production get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
banana-production     2/10    0            1           91d
[01:52:54.010] banana (master) $ kubectl --context=prod-f --namespace= banana-production get rs
NAME                            DESIRED   CURRENT   READY   AGE
banana-production               8         2         2       16d
banana-production               4         0         0       53m
banana-production               0         0         0       35d
```

```
Conditions:
  Type             Status  Reason
  ----             ------  ------
  Available        False   MinimumReplicasUnavailable
  ReplicaFailure   True    FailedCreate
  Progressing      True    ReplicaSetUpdated
OldReplicaSets:    banana-production (2/9 replicas created), banana-production (0/3
replicas created)
NewReplicaSet:     <none>
Events:
  Type    Reason             Age                   From                   Message
  ----    ------             ----                  ----                   -------
  Normal  ScalingReplicaSet  3h20m (x5 over 10h)   deployment-controller  Scaled down replica set
banana-production to 6
  Normal  ScalingReplicaSet  150m                  deployment-controller  Scaled up replica set
banana-production to 6
  Normal  ScalingReplicaSet  140m (x2 over 155m)   deployment-controller  Scaled down replica set
banana-production to 5
  Normal  ScalingReplicaSet  105m (x49 over 39h)   deployment-controller  Scaled down replica set
banana-production to 8
```

One hour later...

What have we tried?

- Deleting the HPA and manually scaling
- Deleting the old (presumably bad replica sets) **but not the one that has our last 2 pods running**
- We tried creating a new Deployment object
- Maybe just recreating the namespace? **But that would definitely delete the last 2 pods running**

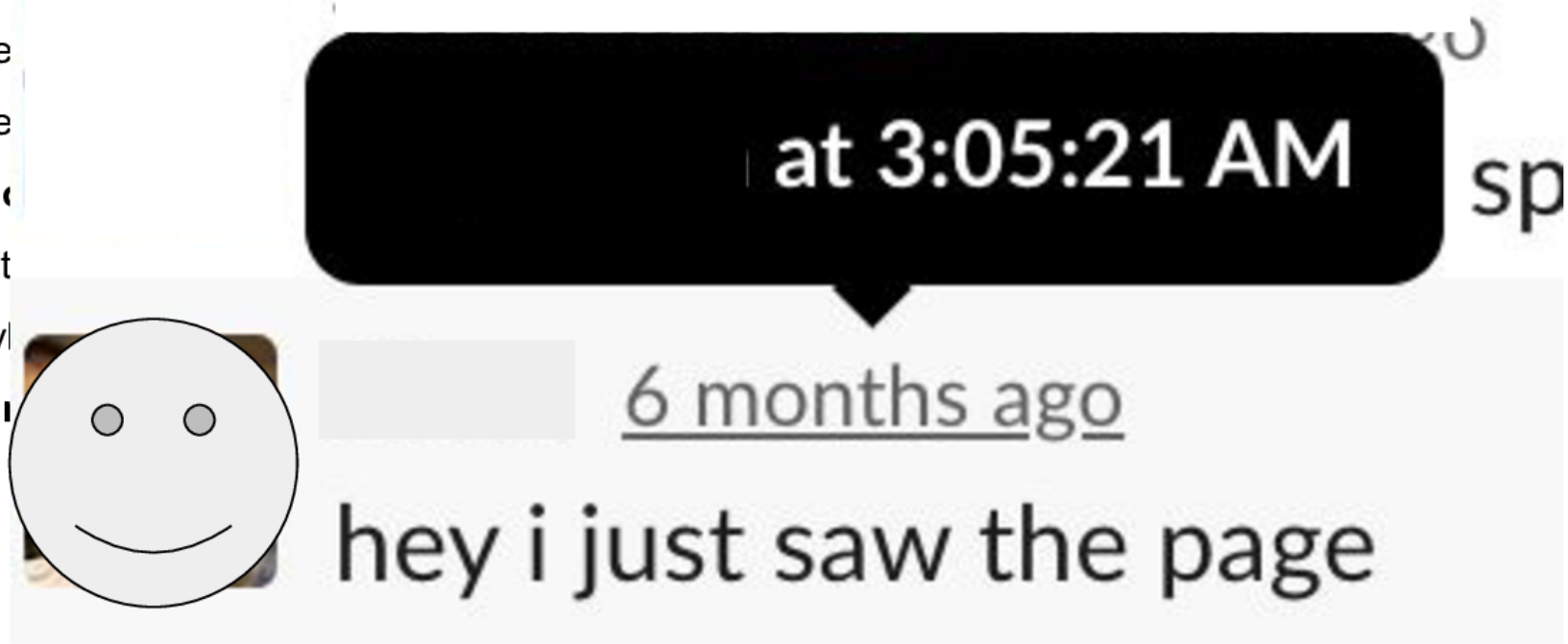
One hour later...

What have

- Dele
- Dele

  the
- We t
- May

  defi



@jiancheung & Joseph Kim

Remember the first page? 🤦🏾‍♀️

178 replies

at 2:10:54 AM

6 months ago

banana is experiencing high error rate. The service cannot be redeployed / scaled up due to K8s error: `Error creating: Internal error occurred: Internal error occurred: Unable to access invalid index: 5`

One hour later...

at 3:06:34 AM

6 months ago

I think it might be mutating admission controller blocking the new pod creation (edited)

6 months ago

the json patch looks like something it could create

jian.cheung 6 months ago

well, i guess that matches my google search for the error msg: https://github.com/kubernetes-sigs/controller-runtime/issues/281

(The issue here was an exact match for what we were seeing)

The Bug

```
# The input
{
 "spec": {
   "containers": [
     {
       "name": "sponge",
       "args": ["a", "b", "c", "d"]
     }
```

```
# The output
{
 "spec": {
   "containers": [
     {
       "name": "sponge",
       "args": ["a", "b"]
     }
```

The Bug

```
# The input
{
 "spec": {
   "containers": [
     {
       "name": "sponge",
       "args": ["a", "b", "c", "d"]
     }
```

```
# the generated output

[
  {
    "op": "remove",
    "path": "/spec/containers/0/args/2"
  },
  {
    "op": "remove",
    "path": "/spec/containers/0/args/3"
  }
]
```

```
# The output
{
 "spec": {
   "containers": [
     {
       "name": "sponge",
       "args": ["a", "b"]
     }
```

@jiancheung & Joseph Kim

The Bug

```
# The input
{
 "spec": {
   "containers": [
     {
       "name": "sponge",
       "args": ["a", "b", "c", "d"]
     }
```

```
# the generated output

[
  {
    "op": "remove",
    "path": "/spec/containers/0/args/2"
  },
  {
    "op": "remove",
    "path": "/spec/containers/0/args/3"
  }
]
```

```
# The output
{
 "spec": {
   "containers": [
     {
       "name": "sponge",
       "args": ["a", "b"]
     }
```

```
# the correct patch
[
  {
    "op": "remove",
    "path": "/spec/containers/0/args/3"
  },
  {
    "op": "remove",
    "path": "/spec/containers/0/args/2"
  }
]
```

@jiancheung & Joseph Kim

Mutating Time Bomb

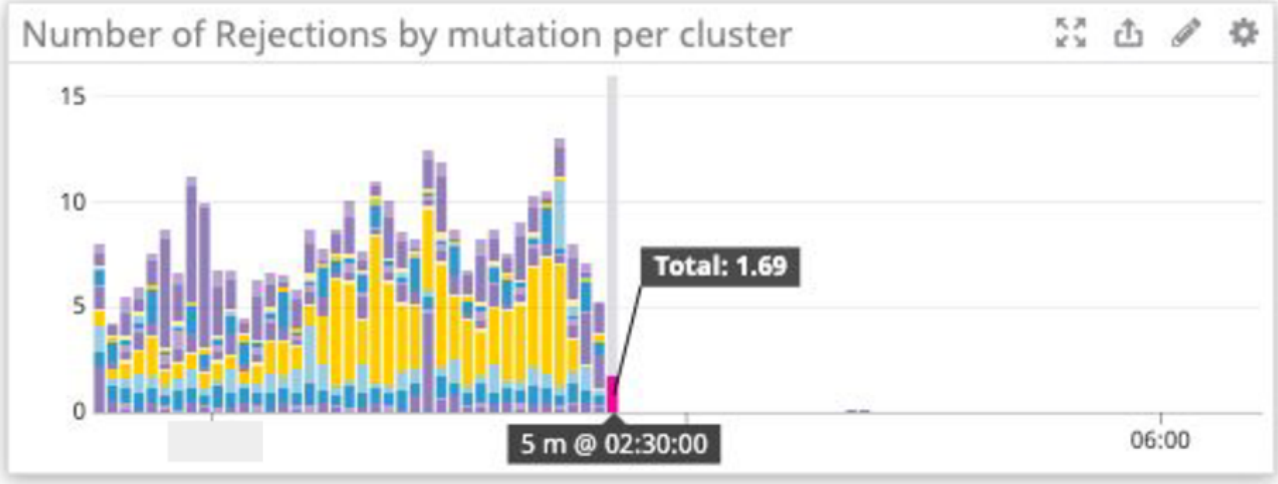
- We realized it was the Mutating Admission Controller
- Specifically it was an incompatible json patch change
- Confirmed it by deleting it in this specific cluster which immediately resulted in successful pods coming up
- Immediate fire is resolved. 🎉. So we're done right?

Mutating Time Bomb

- We realized it was the Mutating Admission Controller
- Specifically it was an incompatible json patch change
- Confirmed it by deleting it in this specific cluster which immediately resulted in successful pods coming up
- Immediate fire is resolved. 🎉. So we're done right? **No!**
-
- We rolled back the MAC change which happened **7 days before**
- We realized for ***some deployments*** over the last 7 days, new code wasn't actually being deployed.
- Services were being kept alive by old replicaSet
- And over time, pods of the old replicaSet slowly died off

Takeaway

- Be aware of the existence of Mutating Admission Controller
- Be willing to ask for help
- Google the error message
- Create alerts



@jiancheung & Joseph Kim

One to Autoscaling

https://github.com/kubernetes/kubernetes/issues/25238#issuecomment-406415297

Situation:

- You have a deployment with 3 replica
- Later, you set up HPA with a much larger number
- You update your deployment and apply it
- Boom! The number of replica resets down to 3

Suggested solution:

- Remove replicas from last deployment
- Remove replicas from current/future deployment

Lgtm, ship it!

Merged **fix autoscaling behavior** #1094

deployboard merged 11 commits into `master` from `evan-enable-correct-autosc…` on Oct 1, 2018

Fix

After running a bunch of manual `kubectl` commands I found that the reason the pods drops to 1 is because kubernetes tracks the `deployment.yml` file between deploys, and when enabling autoscaling for the first time, we're going from specifically set `replicas: n` field to replicas not specified. Kubernetes interprets this as changing the `replicas` to default (1) - thus we scale down to 1 replica on deploy.

To solve this, I used the `view-last-applied` and `set-last-applied` commands documented here: https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#apply
And changed the deployment process to:

1. call `view-last-applied` and edit the replicas out
2. call `set-last-applied` on the result so kubernetes no longer tracks the `replicas` field
3. apply the new `deployment.yml` that no longer scales pods to 1

@jiancheung & Joseph Kim

The Fix

If Deployment && HPA

1. Edit last deployment
2. Remove `replicas: X`
3. Apply the deployment

~~One~~ Zero to Autoscaling

@jiancheung & Joseph Kim

Edge case: ZERO

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#algorithm-details

Algorithm Details

From the most basic perspective, the Horizontal Pod Autoscaler controller operates on the ratio between desired metric value and current metric value:

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

[autoscaling] fix min==max=0 autoscaling #1946

Merged **deployboard** merged 3 commits into `master` from on May 24, 2019

Conversation 3 | Commits 3 | Checks 0 | Files changed 2

commented on May 23, 2019 • edited Member

Summary

[redacted] uncovered a bug where if there existed a deployment with replicas were 0, and autoscaling was turned on for the first time, the autoscaling would successfully deploy but the autoscaler wouldn't kick in.

Looked into this a bit and uncovered the following:

1. Previous fix of turning autoscaling on -> scale deployments down to 1 by viewing/editing last applied deployment caused us to keep whatever replicas we had from the previous deployment (in this case, 0)
2. The autoscaling algo: `desiredReplicas = ceil[currentReplicas * ( currentMetricValue / desiredMetricValue )]` will always calculate `desiredReplicas` to be 0 because the `currentMetricValue` of 0 pods will always be 0 as well.

Fixed by checking if previous deployment replicas was 0, if so, don't edit previous deployment and let k8s do its thing: (aka not do the following)

@jiancheung & Joseph Kim

The Fix

If Deployment && HPA **&& replica != 0**

1. Edit last deployment
2. Remove `replicas: X`
3. Apply the deployment

~~One~~ Zero to Autoscaling

again

@jiancheung & Joseph Kim

10:07 PM

2019

for tomorrow: I deployed this evening to turn on autoscaling. I simply added:

```
autoscaling:
  minReplicas: 64
  maxReplicas: 100
  targetCPUUtilizationPercentage: 70
```

Upon deploy to prod, this happened (number of e replicas dropped to zero):

@jiancheung & Joseph Kim

10:07 PM

, 2019

for tomorrow: I deployed I this evening to turn on autoscaling. I simply added:

```
autoscaling:
  minReplicas: 64
  maxReplicas: 100
  targetCPUUtilizationPercentage: 70
```

Upon deploy to prod, this happened (number of replicas dropped to zero):

10:38 PM

TLDR, an apparently known issue when migrating to HPA for k8s https://github.com/kubernetes/kubernetes/issues/25238#issuecomment-406415297. Spinnaker does not have the standard workaround implemented yet. Will follow up w/ CD to post warnings until we submit this as a corrective action.

@jiancheung & Joseph Kim

The Fix

If Deployment && HPA && replica != 0

1. Edit last deployment
2. Remove `replicas: X`
3. Apply the deployment

// and remember to copy this logic to Spinnaker

~~One~~ Zero to Autoscaling

~~again~~

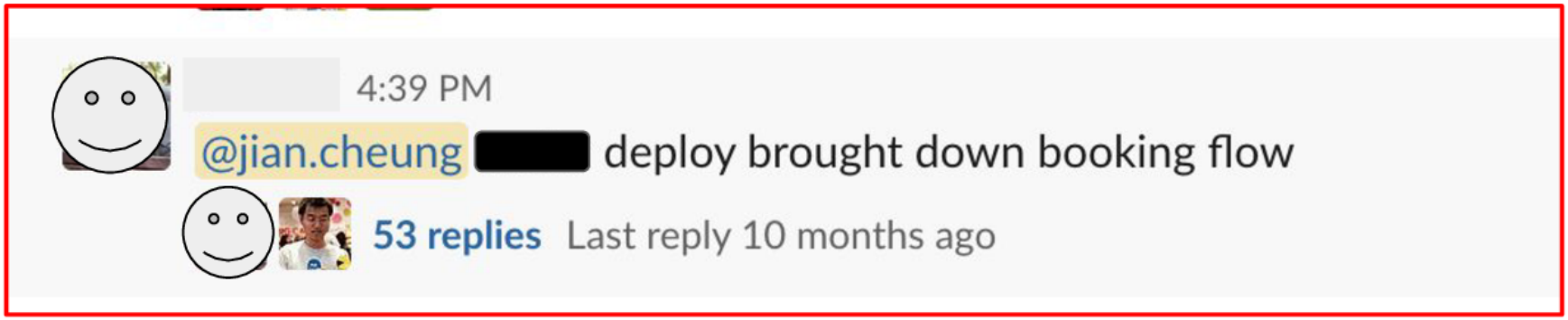
once more!

What do you think happens here?

1. A service is migrating to K8s and has old ec2 boxes running
2. Deploy this service to K8s with replicas = 0 (so now it is both k8s and non-k8s)
3. kubectl scale deployment --replicas=100 (nice! k8s is now taking traffic successfully)
4. Delete all the old ec2 boxes (yes, k8s is the best!)
5. Deploy again with HPA

What do you think happens here?

1. A service is migrating to K8s and has old ec2 boxes running
2. Deploy this service to K8s with replicas = 0 (so now it is both k8s and non-k8s)
3. **kubectl scale deployment --replicas=100** (nice! k8s is now taking traffic successfully)
4. Delete all the old ec2 boxes (yes, k8s is the best!)
5. Deploy again with HPA **(the most recent "proper" deploy was 0 replica)**



@jiancheung & Joseph Kim

The Fix

If Deployment && HPA && replica != 0

1. Edit last deployment
2. Remove `replicas: X`
3. Apply the deployment

// and remember to copy this logic to Spinnaker

// and add warning to `kubectl scale deployment` tool

// check what current replica count is

Takeaway

- Turning on Horizontal Pod Autoscaler for the first time is not trivial
- Test your edge cases (0, 1, etc)
- Remember the non-paved paths

Did we really delete all our master nodes?

at 10:43 AM

Hey compute-infra folks; we have drained master nodes on vanilla-a by mistake. And launching a new instance fails converging

@jiancheung & Joseph Kim

at 10:43 AM

Hey compute-infra folks; we have drained master nodes on vanilla-a by mistake. And launching a new instance fails converging

1 reply

8 months ago

apologies for the trouble. Not urgent since this is the vanilla cluster, but I'm sure you've getting the alerts too

@jiancheung & Joseph Kim

11:00 AM

the draining of the masters seems to have completely broken -vanilla-a

11:04 AM

we have two options, both of which would take some time (and have to be prioritized), 1. (easier) full factory reset of -vanilla-a, delete all data, bring it up as a brand new cluster 2. try to fix it in place, though it's unclear which dependencies are blocking it

12 replies Last reply 8 months ago

@jiancheung & Joseph Kim

8 months ago

it looks like I may have fixed it via the first operation in

There were a few things that made it seem like not the signature of that fix, but ultimately it's explainable by the same underlying cause

@jiancheung & Joseph Kim

Runbooks to the Rescue

1. Delete mutation controller as an immediate action via `kubectl delete ValidatingWebhookConfigration`
2. Then, terminate kube-dns pod to get it rescheduled

Takeaways

8 months ago

That is great news! Thank you so much . Any learnings for us w.r.t handling this in the future. (Apart from "dont drain master nodes"?)

8 months ago

don’t drain master nodes is the big one

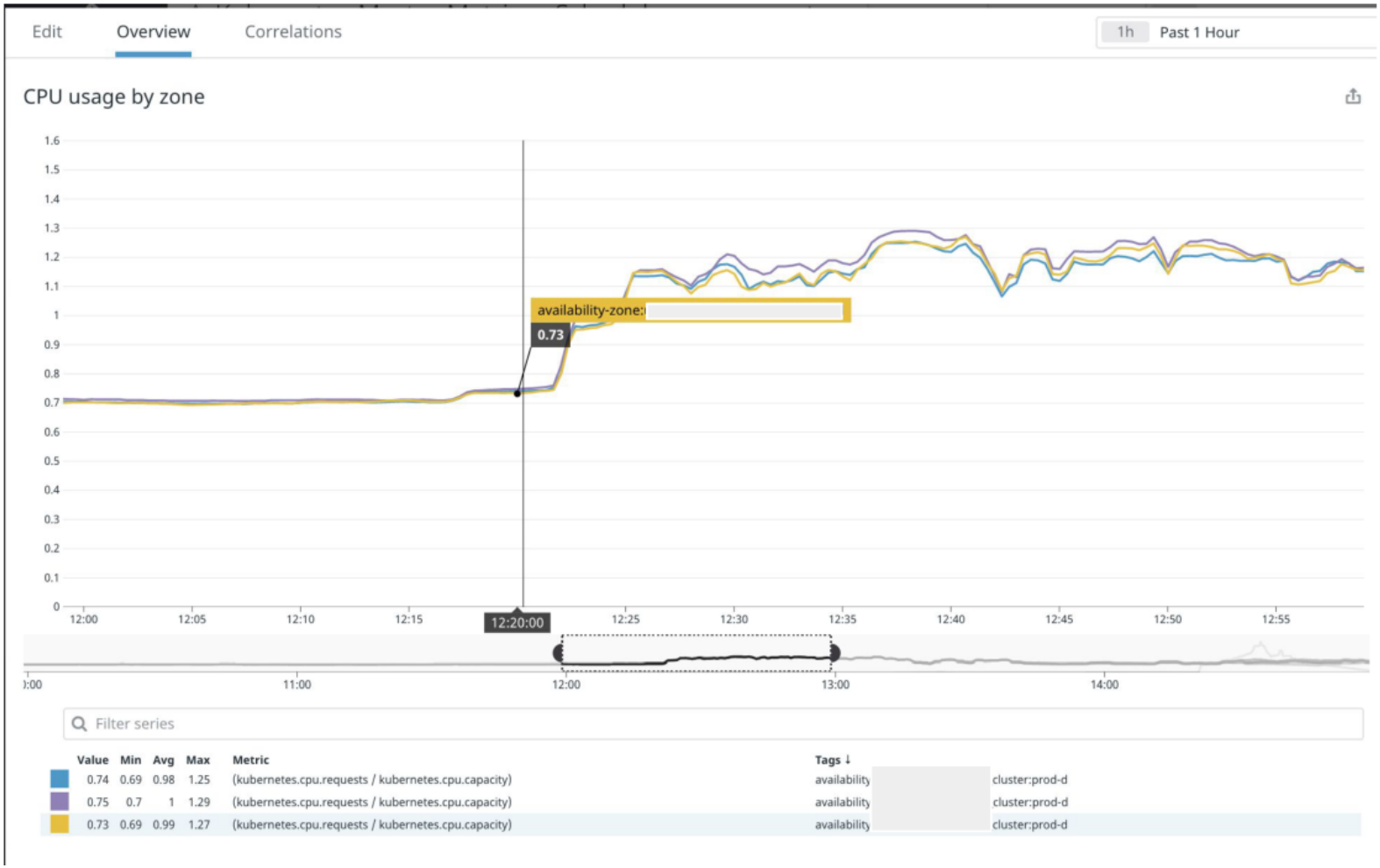
8 months ago

but if you do, check out the runbook

👍 1

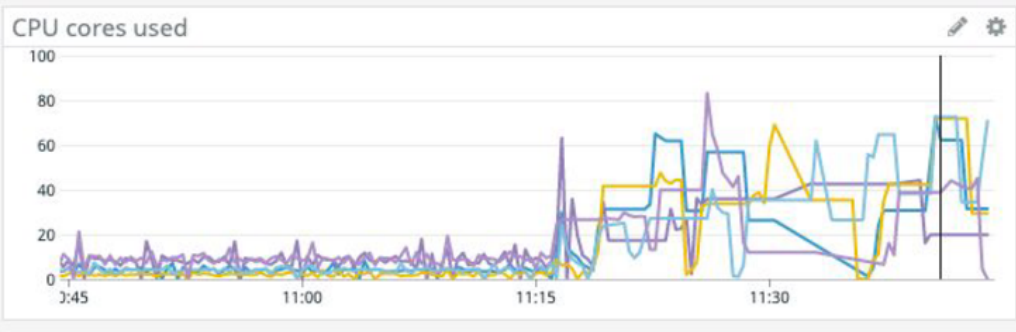
Masters Out of Memory

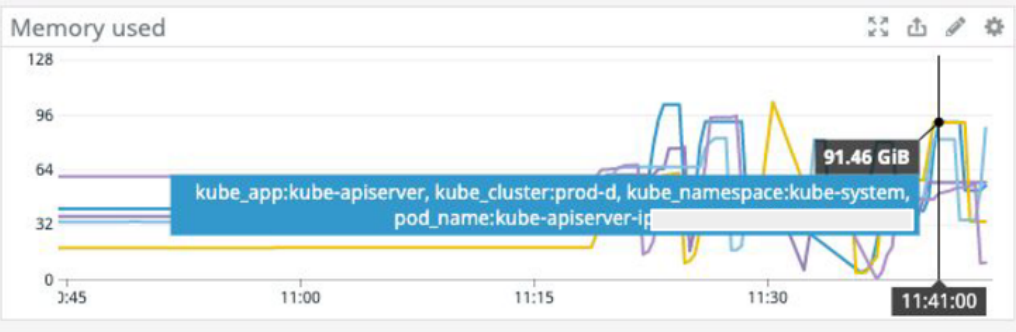
11:16, K8s master CPU goes up...



@jiancheung & Joseph Kim

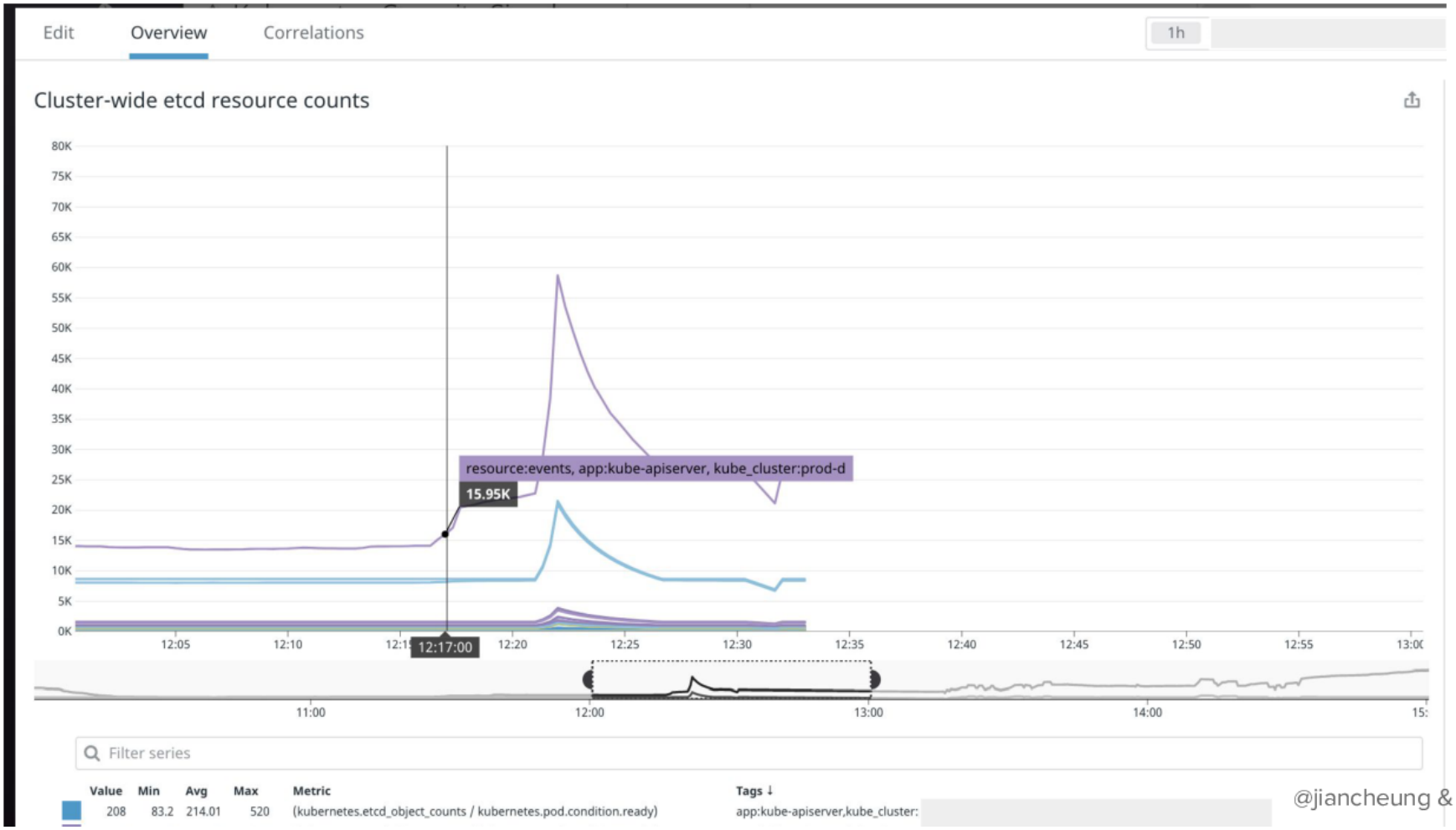
kube-apiserver goes wild





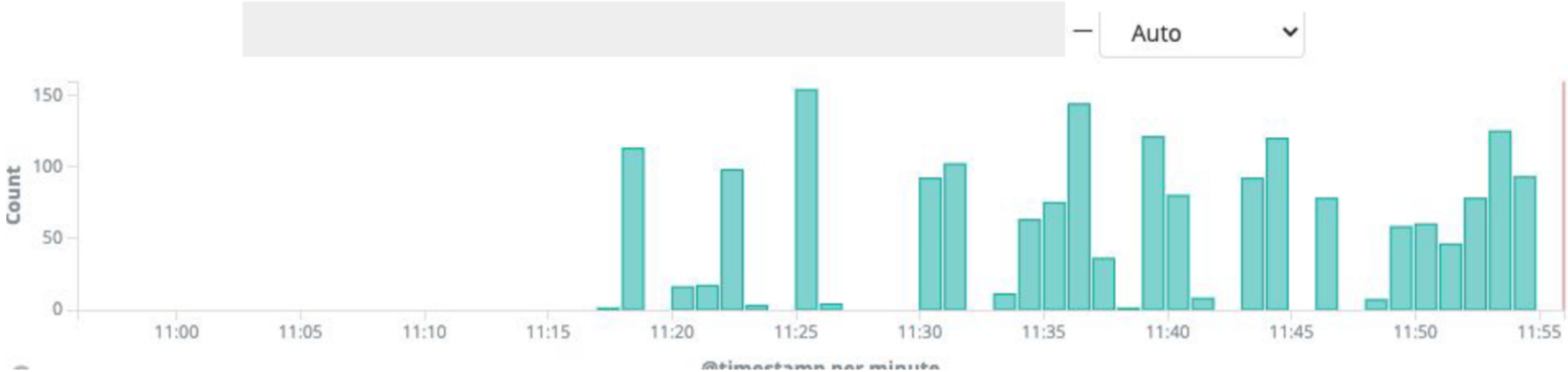
@jiancheung & Joseph Kim

etcd resource counts are up



@jiancheung & Joseph Kim

OOM Kills 😱


Auto
150
100
50
0
Count
11:00
11:05
11:10
11:15
11:20
11:25
11:30
11:35
11:40
11:45
11:50
11:55

Alerts start firing

APP 12:24 PM

No data: [K8s component] prod-d 'controller-manager' reported unhealthy

component 'controller-manager' is unhealthy, investigate why

```
kubectl --context=prod-d get componentstatus/controller-
manager -o yaml
kubectl --context=prod-d -n kube-system get pods -l
app=kube-controller-manager
```

Alerts start firing

APP 12:24 PM

No data: [K8s component] prod-d 'controller-manager' reported unhealthy

component 'controller-manager' is unhealthy, investigate why

```
kubectl --context=prod-d get componentstatus/controller-
ods -l
```

:25 **No data: Kubernetes component 'controller-manager' reported unhealthy on cluster prod-d**

The `controller-manager` component status has been reported unhealthy for at least 30 seconds by:

```
kubectl get componentstatus controller-manager
```

Alerts start firing

APP 12:24

No data: [K8s cor
unhealthy
component 'contr

```
kubectl --conte
```

:25 **No data: Kubernetes compor**
unhealthy on cluster prod-d

The `controller-manager` component status has been reported unhealthy for at least 30 seconds by:

```
kubectl get componentstatus controller-manager
```

Triggered: [K8s] prod-d master node condition reported Not Ready

Some master nodes have reported Not Ready.

```
kubectl --context=prod-d get nodes -l chef-role=kubernetes-master-prod-d --label-columns=kubernetes.io/hostname,chef-role
```

There should be at least 3 master nodes (balanced across AZs)

If this is limited to a single master, replace the host in Starforge using the
"Replace" button.

Runbook:
system/dc
ready

No access either

Sep 21st at 12:35 PM

ssh access into control plane nodes is inconsistent

3 replies

1 month ago

keep getting booted

1 month ago

intermittent access to k8s API

1 month ago

see some evidence of OOMs and high load

What is happening?

- api-server is OOMing and crashing
- Control-plane nodes are severely degraded (no ssh access)
- Is etcd having problems?
    - It has elevated objects being created but it’s actually healthy and fine. Phew!
- Existing workloads are fine :)

Response

- Spin up 3 bigger instances for the control plane
  - Just in time because the old instances died
- Memory gets eaten up super fast. Good thing these new instances are much **bigger**.

Postmortem

- This was actually on one of our largest clusters
- Incident lined up with deploy that had dramatically their maxSurge value
  - The deploy was crashlooping
  - The theory here is that this overwhelmed

Takeaways

- Restrict maxSurge
- Be ready to scale vertically

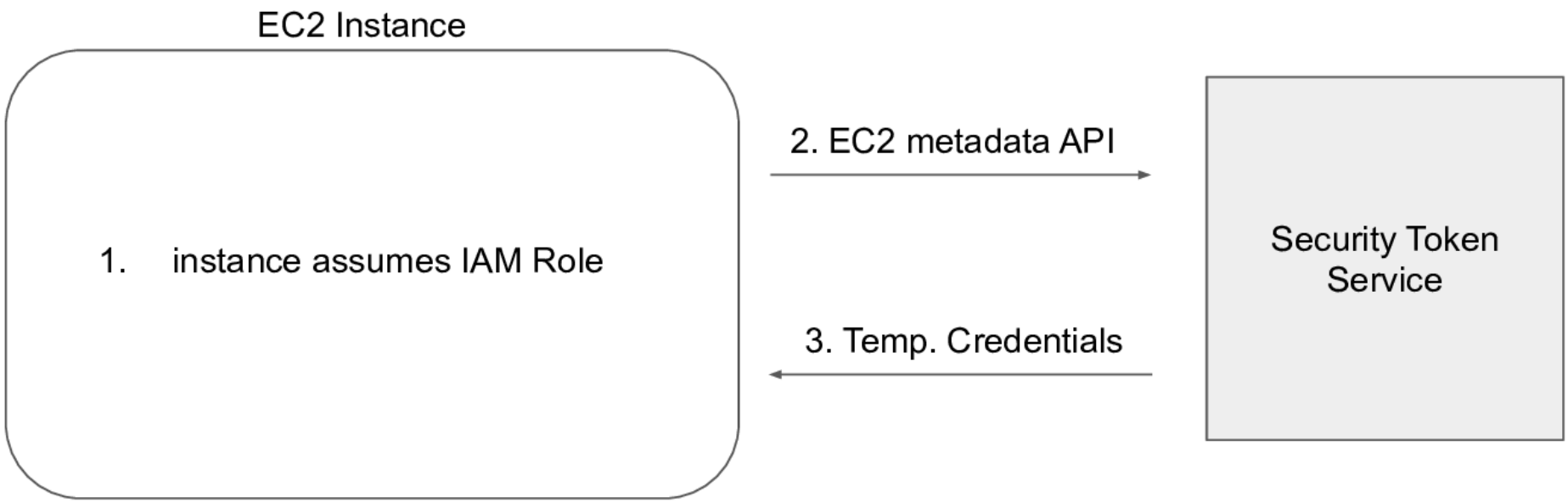
Ongoing Unknowns

- What is the exact breaking point in terms of pods, maxSurge, etc?
- We've had larger clusters (before we broke it up). What is different now?
- Why does memory usage jump suddenly?

Authentication in the World of Containerization

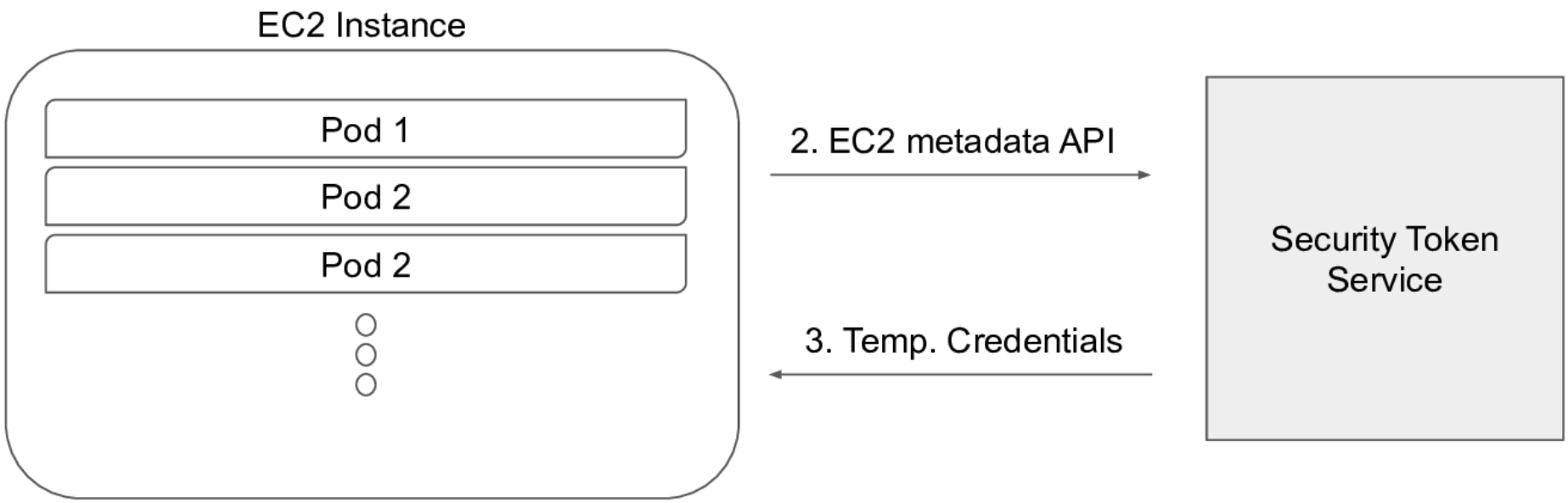
@jiancheung & Joseph Kim

Authentication via IAM Roles



Authentication via IAM Roles

with kubernetes??



@jiancheung & Joseph Kim

kube2iam!

Yay open source!

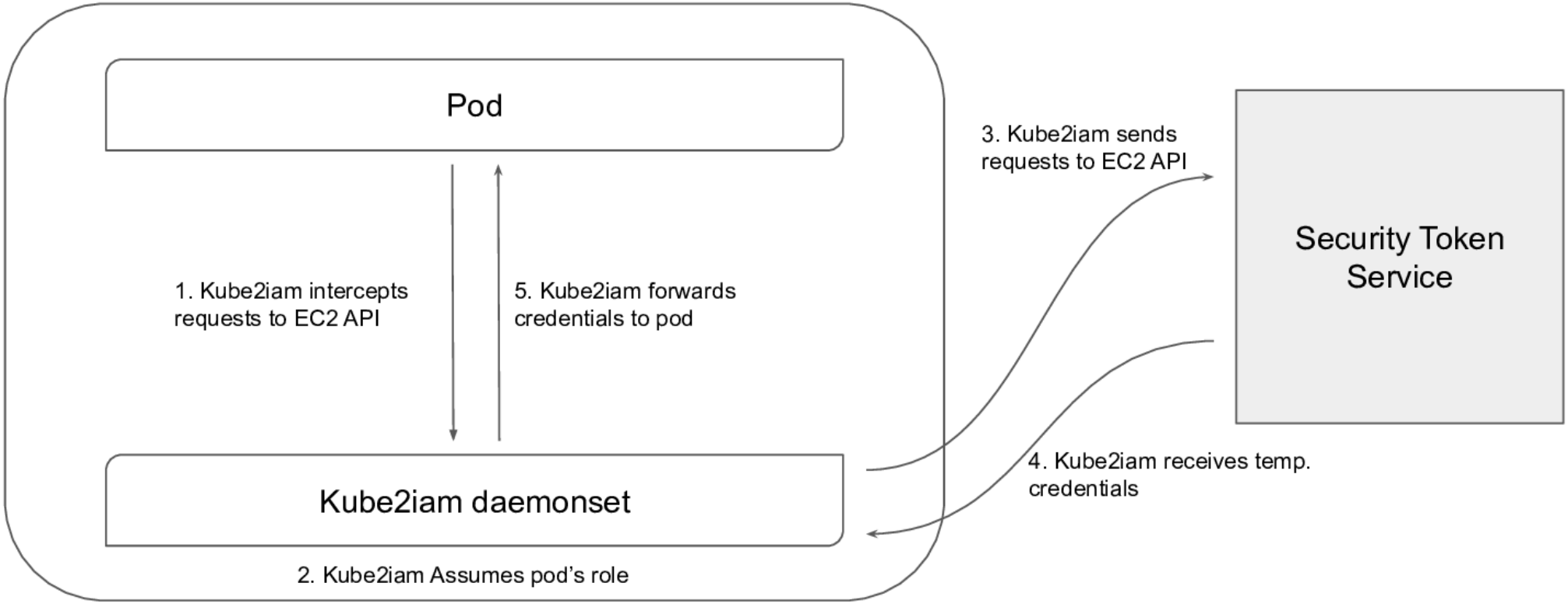
build passing | tag v0.10.11 | docker pulls 60M | go report A+ | license not identifiable by github

kube2iam

Provide IAM credentials to containers running inside a kubernetes cluster based on annotations.

kube2iam!

Yay open source!



@jiancheung & Joseph Kim

Race Conditions

!

Expected course of events:

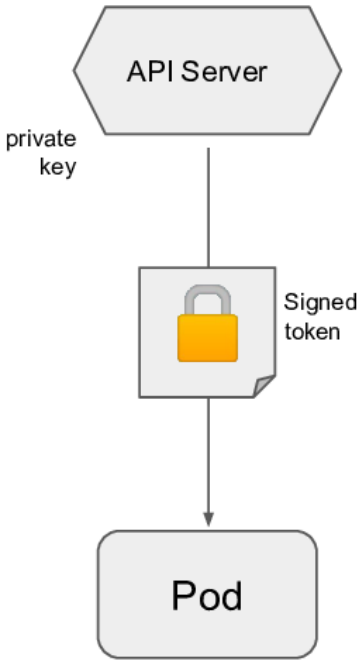
1. Node starts up
2. kube2iam pod starts up
3. Iptables rule forward ec2 api ip address → kube2iam pod
4. kube2iam starts watching for new pods
5. <app> pod starts up
6. kube2iam notices new pod, caches <app> pod IP address ↔ AWS IAM Role mapping
7. Containers in <app> pod makes request to ec2 api

More Init Containers!

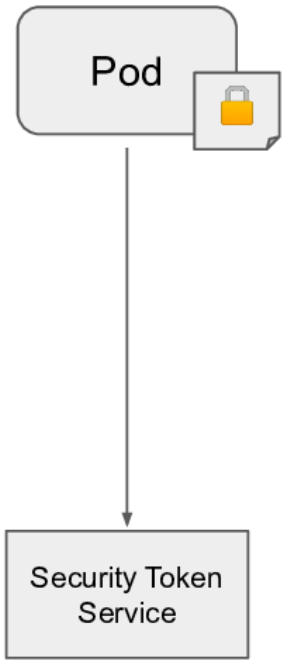
```
name: kube2iam-wait
Command:
  check() {
    # pings given endpoint
    ...
  }
  check “ec2.api.ip/healthz”
  check “ec2.api.ip/latest/meta-data/iam/security-credentials”
```

Service Account IAM Auth

1. Kube API Server uses its private key to create and sign a token that is injected into the pod

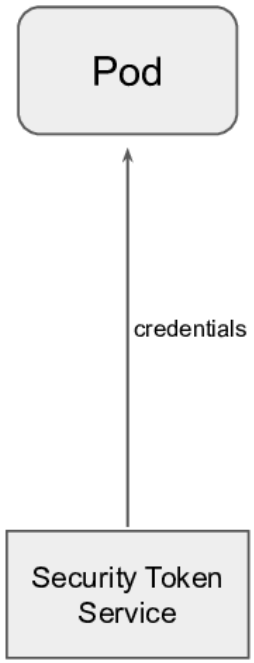


2. Pod calls sts:AssumeRoleWithWebIdentity to get credentials for the pod's IAM Role



3. STS verifies the token signature is valid with public keys

4. STS returns temporary credentials to the pod



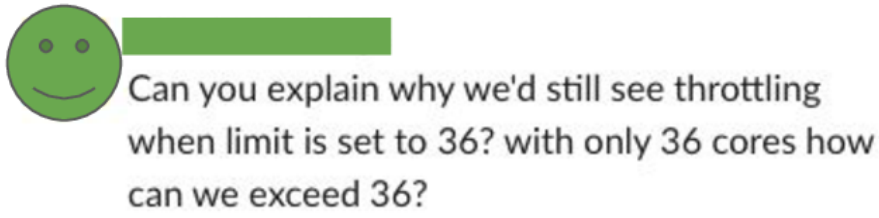
@jiancheung & Joseph Kim

Takeaways

- Init containers are versatile despite their simplicity, and are good solutions even if they're temporary!

Cpu limit == number of cpu cores on a node still has throttling??

36 cores on machine, 36 limits on cpu



@jiancheung & Joseph Kim

K8s limits pod’s cpu usage using linux cgroups & CFS

Turns k8s CPU limits → linux cpu quotas

K8s limits pod’s cpu usage using linux cgroups & CFS

Turns k8s CPU limits → linux cpu quotas

Example:

- 36 core machine
- 100ms default scheduling period

K8s limits pod's cpu usage using linux cgroups & CFS

Turns k8s CPU limits → linux cpu quotas

Example:

- 36 core machine
- 100ms default scheduling period

K8s manifest

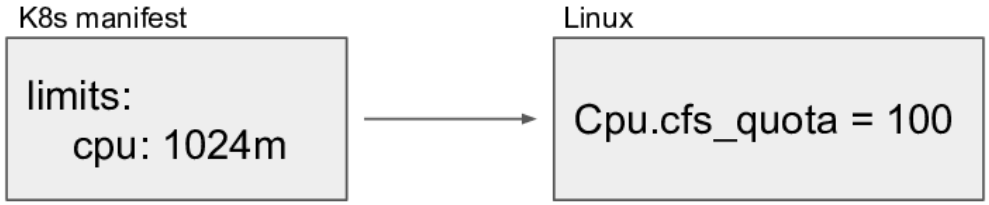
```
limits:
    cpu: 1024m
```

K8s limits pod's cpu usage using linux cgroups & CFS

Turns k8s CPU limits → linux cpu quotas

Example:

- 36 core machine
- 100ms default scheduling period

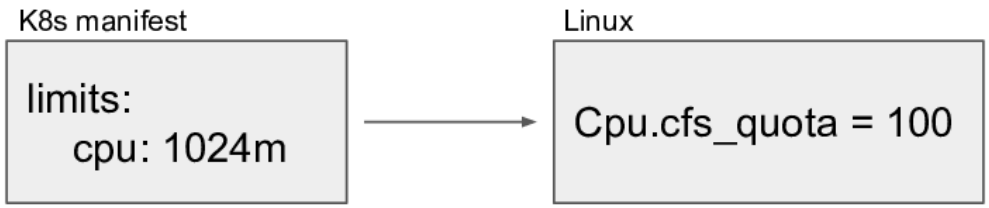


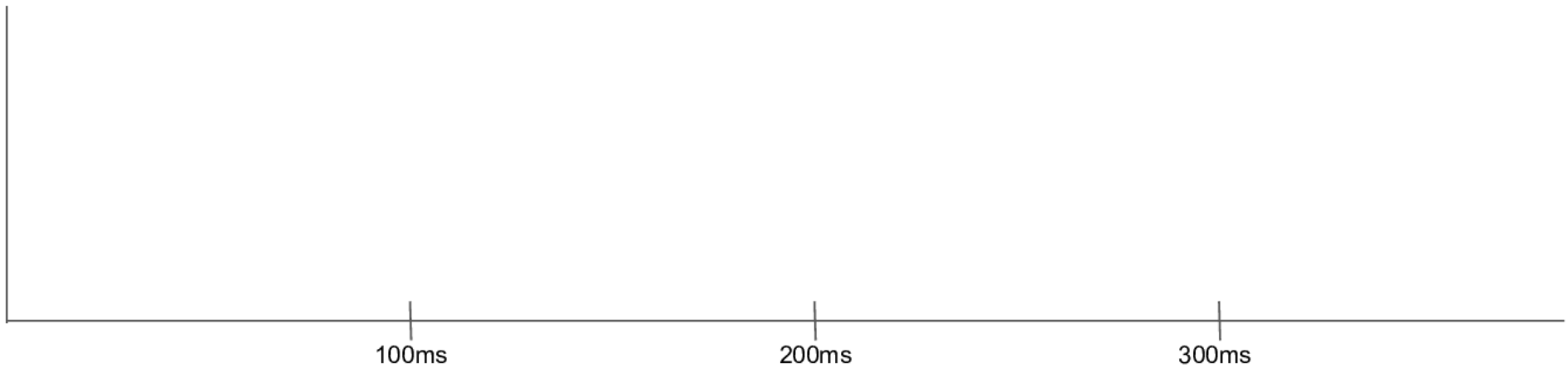
K8s limits pod's cpu usage using linux cgroups & CFS

Turns k8s CPU limits → linux cpu quotas

Example:

- 36 core machine
- 100ms default scheduling period



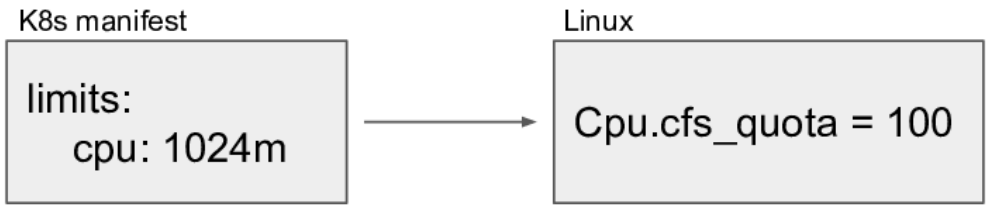


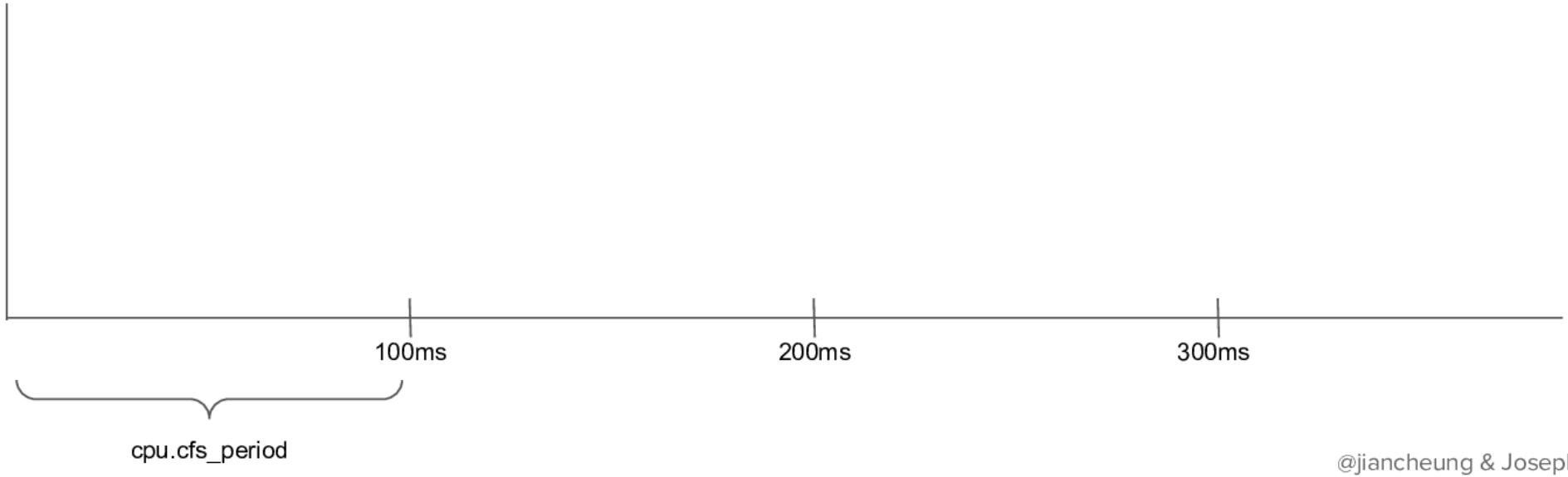
K8s limits pod's cpu usage using linux cgroups & CFS

Turns k8s CPU limits → linux cpu quotas

Example:

- 36 core machine
- 100ms default scheduling period





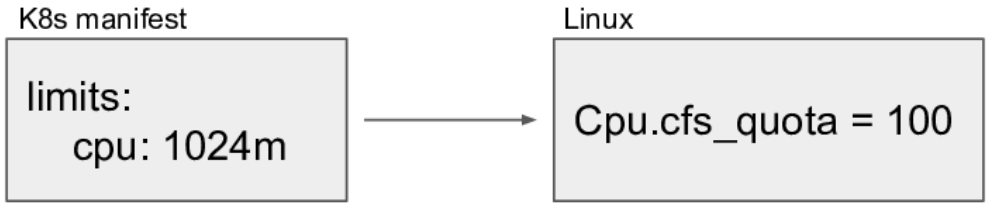
@jiancheung & Joseph Kim

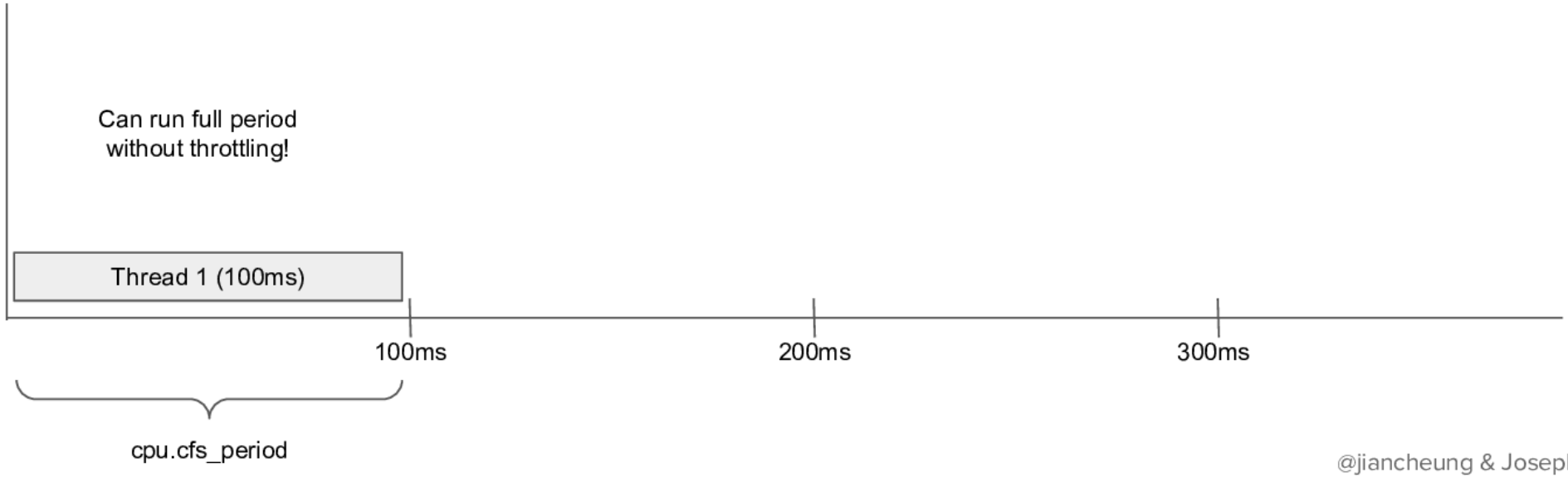
K8s limits pod's cpu usage using linux cgroups & CFS

Turns k8s CPU limits → linux cpu quotas

Example:

- 36 core machine
- 100ms default scheduling period





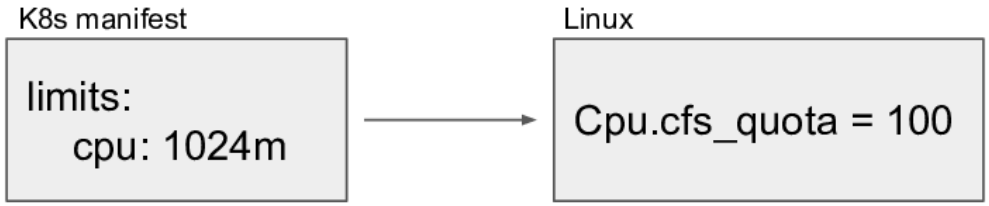
@jiancheung & Joseph Kim

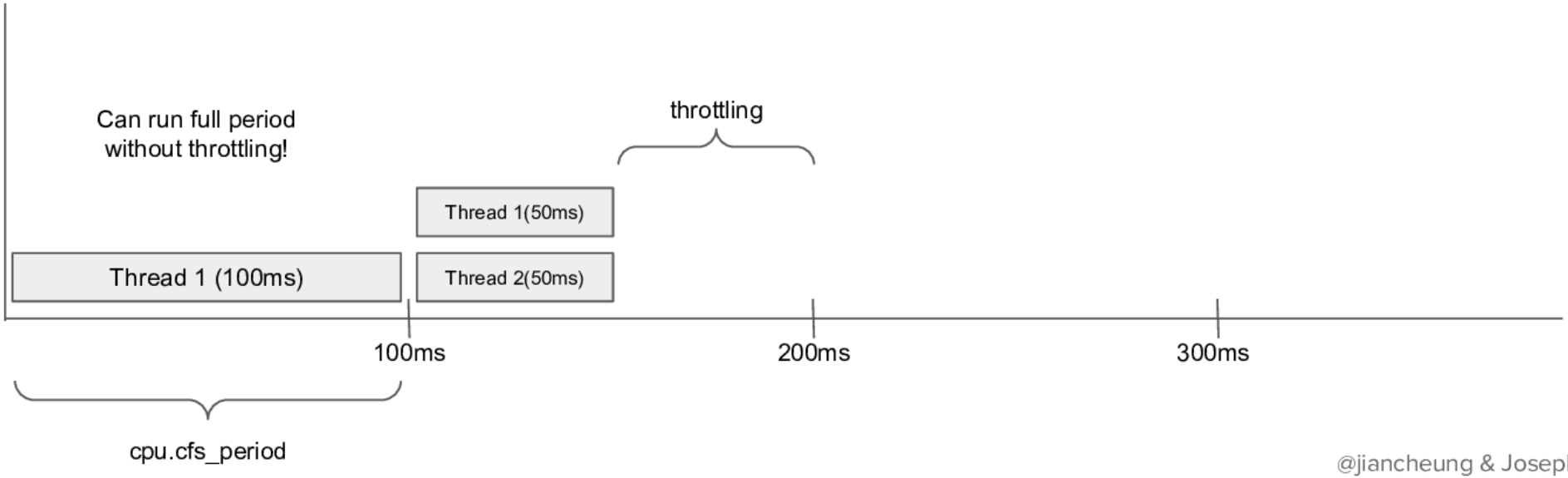
K8s limits pod's cpu usage using linux cgroups & CFS

Turns k8s CPU limits → linux cpu quotas

Example:

- 36 core machine
- 100ms default scheduling period





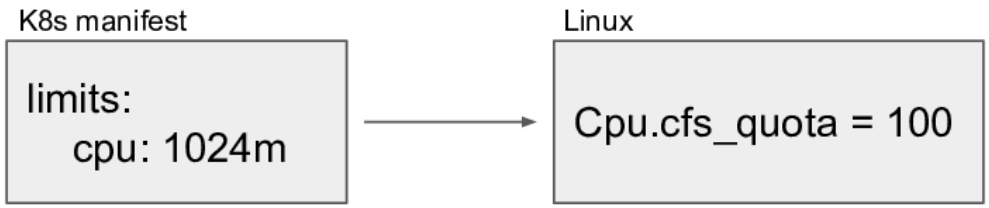
@jiancheung & Joseph Kim

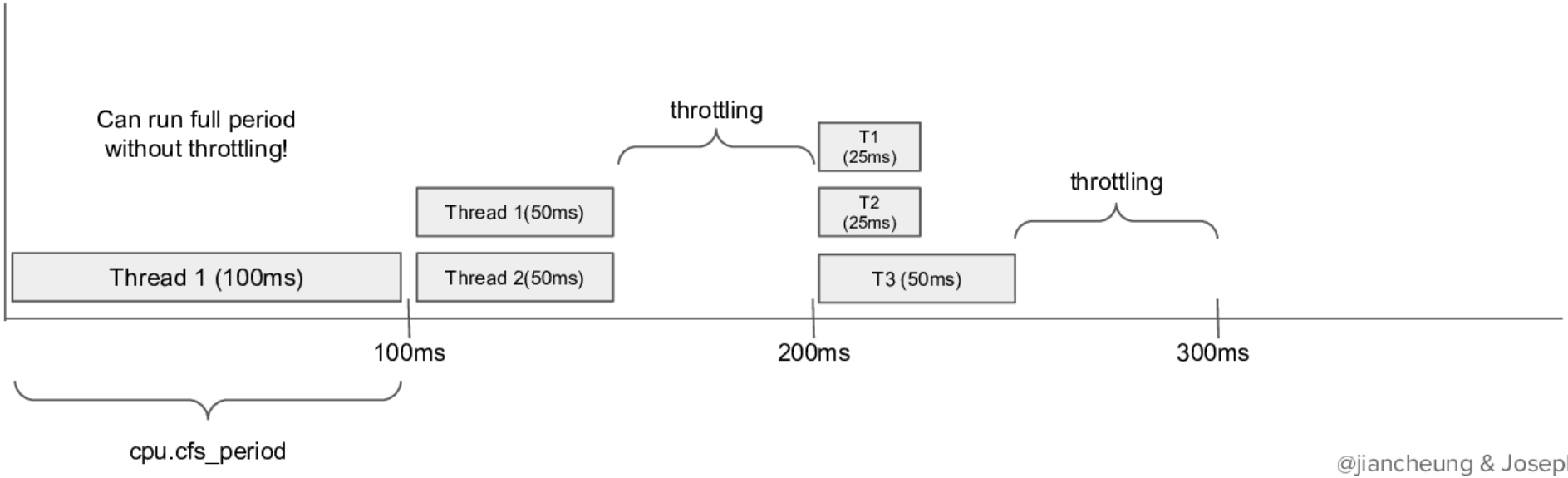
K8s limits pod's cpu usage using linux cgroups & CFS

Turns k8s CPU limits → linux cpu quotas

Example:

- 36 core machine
- 100ms default scheduling period

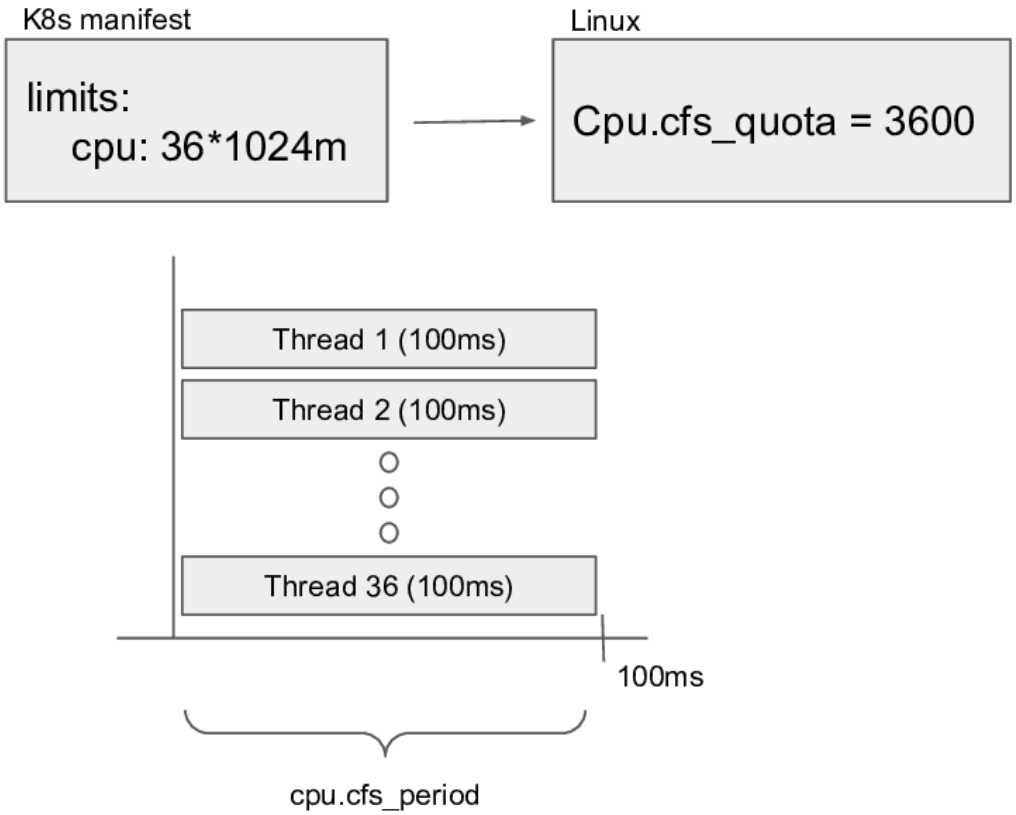




@jiancheung & Joseph Kim

So with 36 cores, 36 limits...

Should never get throttled!



How is it affecting performance?

No limits

```
General statistics:
    total time:                          9.7048s
    total number of events:              10000
    total time taken by event execution: 387.2722s
    response time:
         min:                                  29.95ms
         avg:                                  38.73ms
         max:                                 161.46ms
         approx.  95 percentile:               66.28ms
Threads fairness:
    events (avg/stddev):           250.0000/28.86
    execution time (avg/stddev):   9.6818/0.01
```

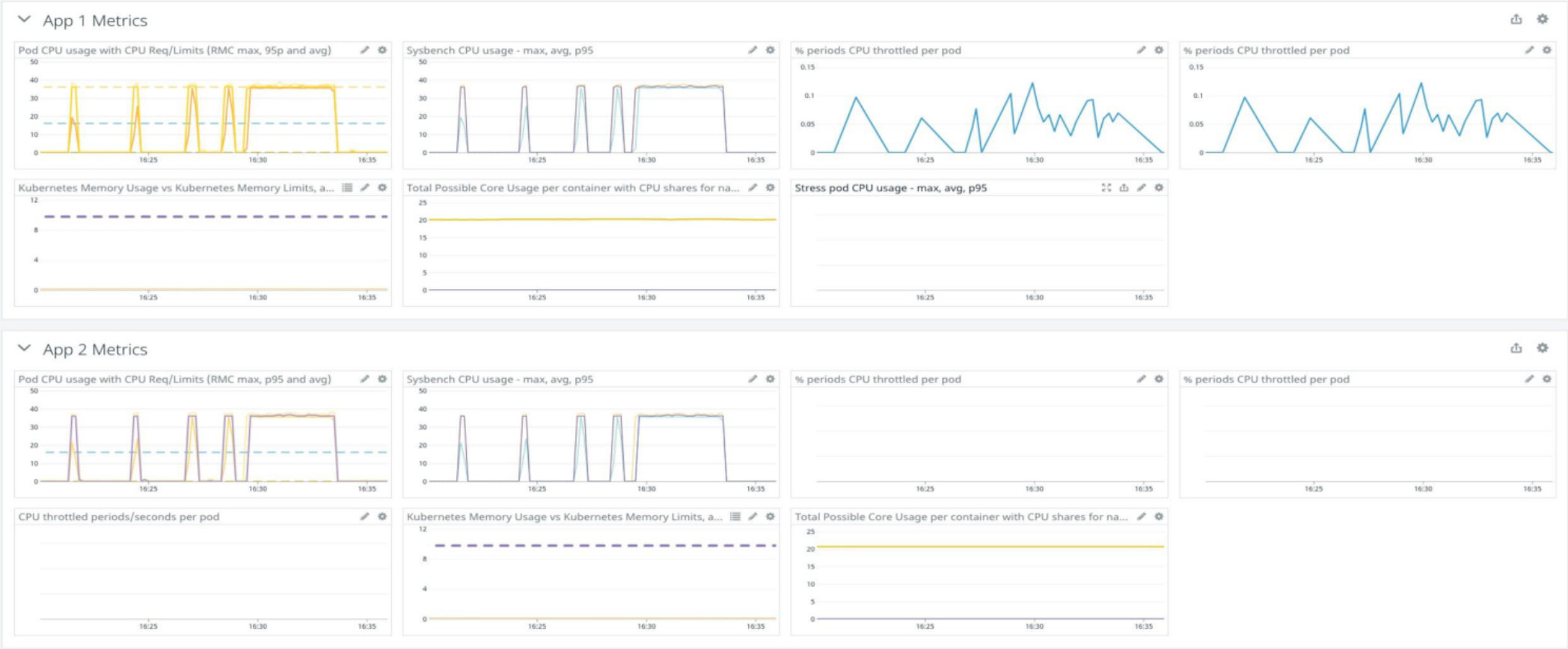
Limits = 36 cpu

```
General statistics:
    total time:                          9.6844s
    total number of events:              10000
    total time taken by event execution: 386.5394s
    response time:
         min:                                  29.59ms
         avg:                                  38.65ms
         max:                                 185.93ms
         approx.  95 percentile:               66.32ms
Threads fairness:
    events (avg/stddev):           250.0000/29.58
    execution time (avg/stddev):   9.6635/0.01
```

@jiancheung & Joseph Kim

How is it affecting performance?

Not very different, even though there was throttling!



@jiancheung & Joseph Kim

Takeaways

- Some throttling is inherent and is okay
- Be careful of your metrics -- docker built in cpu usage information may not be granular enough
- Percentage of throttled periods is a better metric than absolute value of periods

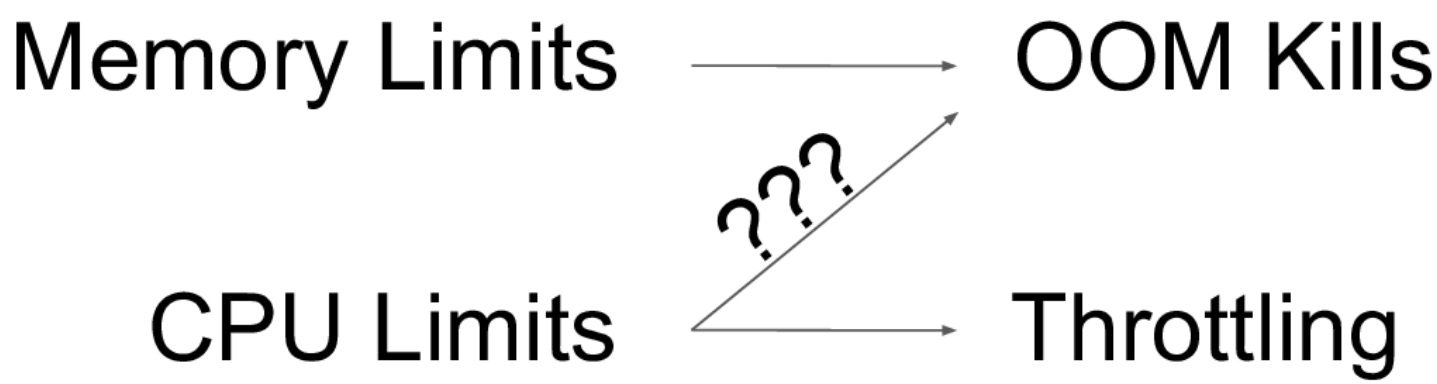
CPU Limits cause ...
OOM Kills??

Intuitively...

Memory Limits → OOM Kills

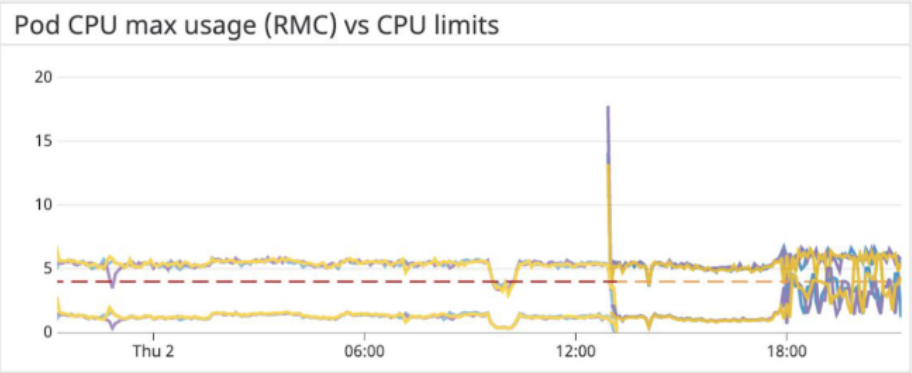
CPU Limits → Throttling

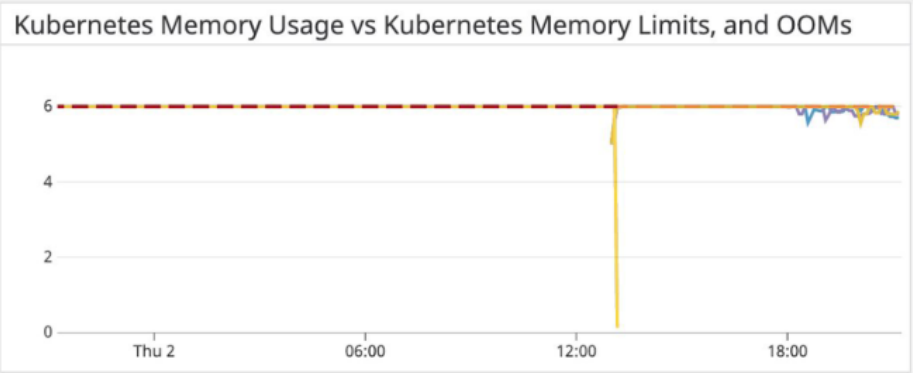
~~**Intuitively...**~~



@jiancheung & Joseph Kim

Correlation between throttling and OOMs





@jiancheung & Joseph Kim

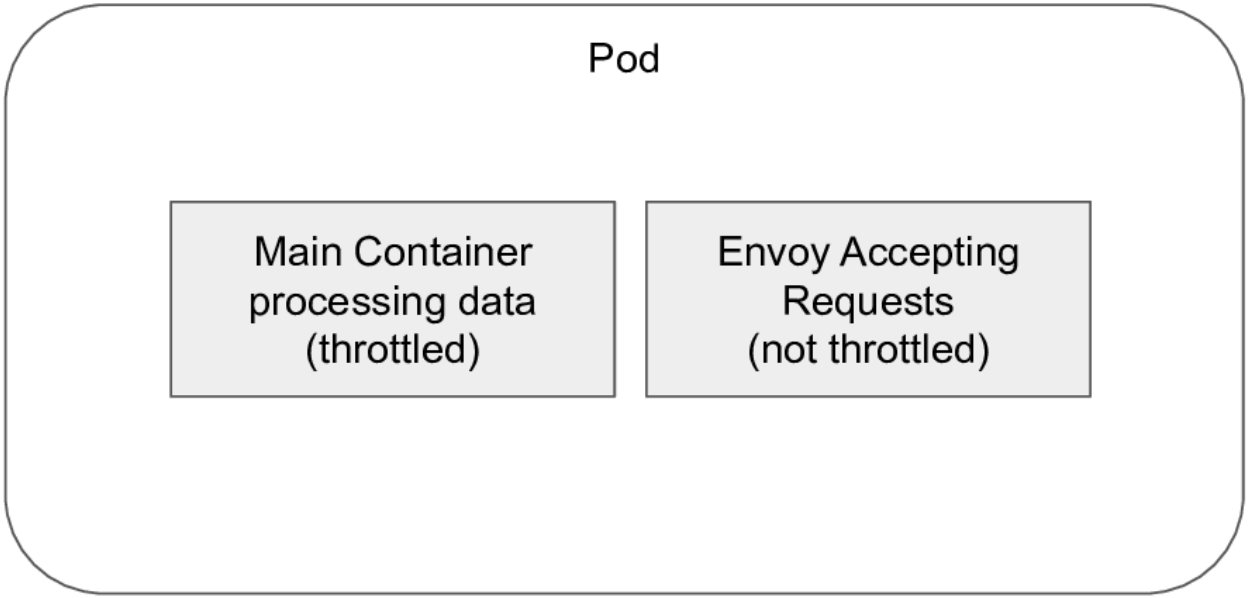
Oops #1

JVM Garbage Collector is intensive & spawns a lot of threads

- Two flags
  - ParallelGCThreads
  - ConcGCThreads

Oops #2 & #3

Only certain containers get throttled & backpressure isn't working!

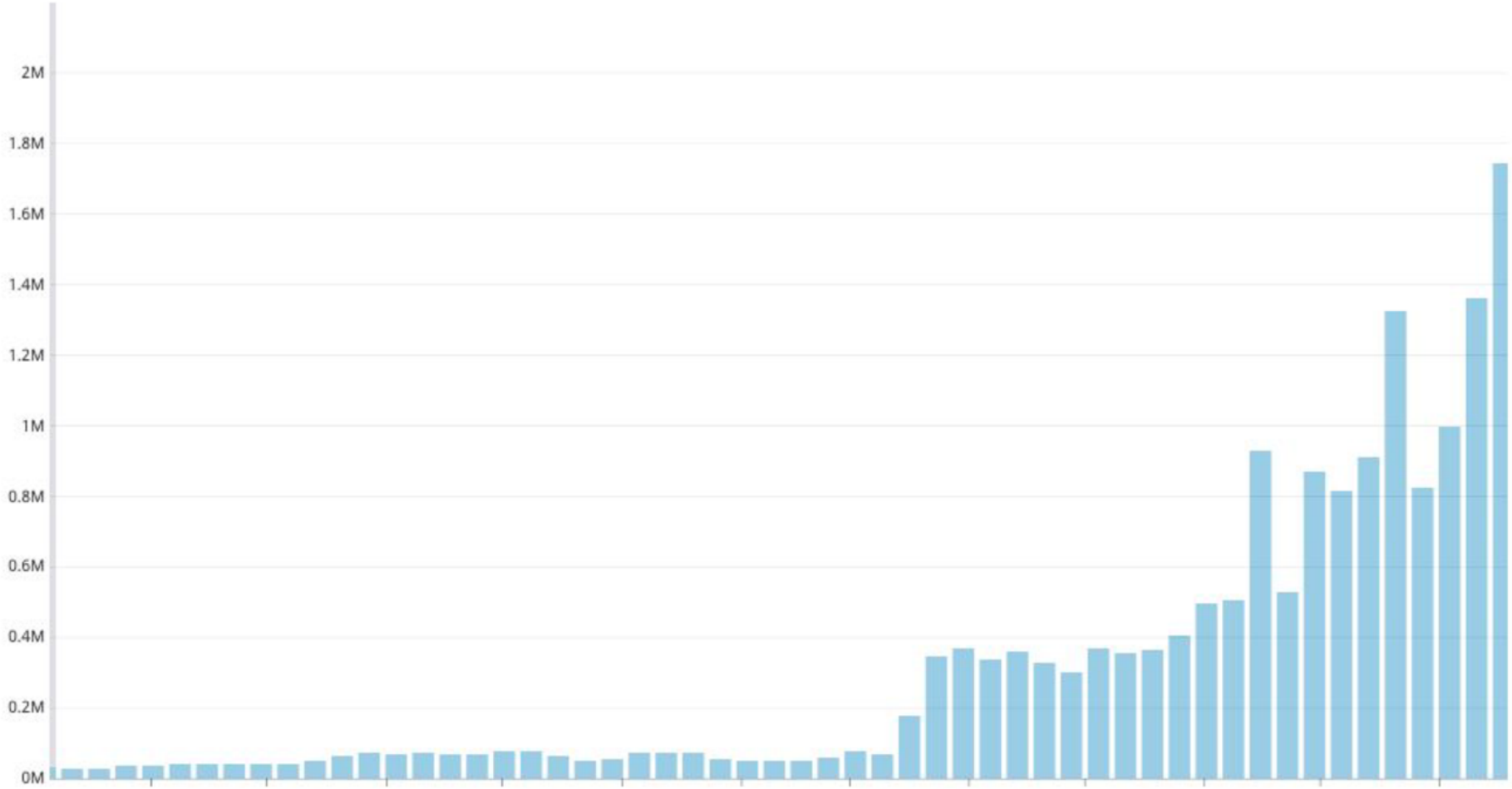


The Fix & Takeaways

- Be wary of what other processes may be running on the pod and adjust limits accordingly
- Tune JVM GC Threads if necessary
- Make sure to have proper backpressure to avoid overloading your services

Rogue ECR Cleaner

We make a lot of docker images



@jiancheung & Joseph Kim

ECR Cleaner

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

```
find_all_images_in_use()

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

ECR Cleaner started deleting images in production

ECR Cleaner

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

```
find_all_images_in_use()

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

Oops #1

Improper error handling

```
def find_all_clusters():
    ...

clusters = find_all_clusters()
active_images = []
for cluster in clusters:
    active_images.extend(find_all_images())
    ...
return active_images
```

Oops #1

Improper error handling

```
def find_all_clusters():
    ...

clusters = find_all_clusters()
active_images = []
for cluster in clusters:
    active_images.extend(find_all_images())
    ...
return active_images
```

```
def find_all_clusters():
    ...
    if err: return []

clusters = find_all_clusters()
active_images = []
for cluster in clusters:
    active_images.extend(find_all_images())
    ...
return active_images
```

Oops #1

Improper error handling

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

```
find_all_images_in_use()

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

Oops #2

Lack of alerts

- ECR Cleaner failing
- ImagePullBackOff in production

The Fix

Very tedious

[redacted]

plan is to:

1. ✅ shut down `ecr-cleaner`
2. 🟥 look for crashlooping containers (since we don't track ImagePullBackoff / Pods may remain healthy until it actually gets restarted)
3. ✅ parse out deleted images from the log
4. ✅ querying all `prod*` clusters for list of images in use
5. ✅ write script to cross-check container images ECR
6. ✅ ^ run onetouch build CI job again for those (edited)

[redacted]

actually easier to find the buildkite link here using this URL:

[redacted]

...thank you kind teammate

@jiancheung & Joseph Kim

Takeaways

- The more critical the service, the more crucial thorough testing & review is
- Have proper error handling & alerts for infrastructural issues
- Try to make sure your fixes aren’t causing problems elsewhere

Be careful what you break

Firedrills 🔥

[redacted] 11:18 AM

Firedrill 🔥 🏗️ CI is broken and we cannot do a OneTouch build for ECR cleaner. We need to ship a new version to production that displays a new log line on startup without relying on CI @evan who is oncall Disclaimer: CI is not really broken, do not panic, this is a drill, this is only a drill! 🚨

🚨 2 🔥 1

15 replies Last reply 4 months ago

@jiancheung & Joseph Kim

Firedrills 🔥

We did a lot of em

[redacted] 10:14 AM

Uh Oh @joseph.kim it looks like [redacted] is crashlooping. 🔥 🚧 🔥 🚧 🚨 could you PTAL when you get a chance?

👀 1 🚨 1

[redacted] 11:18 AM

Firedrill 🔥 🚧 CI is broken and we cannot do a OneTouch build for ECR cleaner. We need to ship a new version to production that displays a new log line on startup without relying on CI @evan who is oncall Disclaimer: CI is not really broken, do not panic, this is a drill, this is only a drill! 🚨

🚨 2 🔥 1

15 replies

[redacted] 12:08 PM

🔥 🚧 *We recently discovered an issue with* [redacted] *and need to make sure it is patched today for all the services* [redacted] *cluster. For this urgent chance we need to bypass the freeze duration of the mutator and cycle the* [redacted] *cluster to make sure all the pods pick up the change by the end of the day.* More info here

joseph.kim 10:59 AM

🔥 🚧 🔥 🚧 🚨 🚧 Fire drill! [redacted] Looks like [redacted] in a crash-loop! Please take a look when you have time 🙂

🍩 1

74 replies

@jiancheung & Joseph Kim

Firedrills 🔥

Block the creation of replicaset (do not merge // firedrill) #3040

Closed [redacted] wants to merge 1 commit into `master` from `lc--block-deployment-firedrill`

Conversation 3 | Commits 1 | Checks 0 | Files changed 2

[redacted] Member

Summary

This is a test change that we will deploy to a test cluster to test debugging skills of our oncall.

@jiancheung & Joseph Kim

Firedrills 🔥

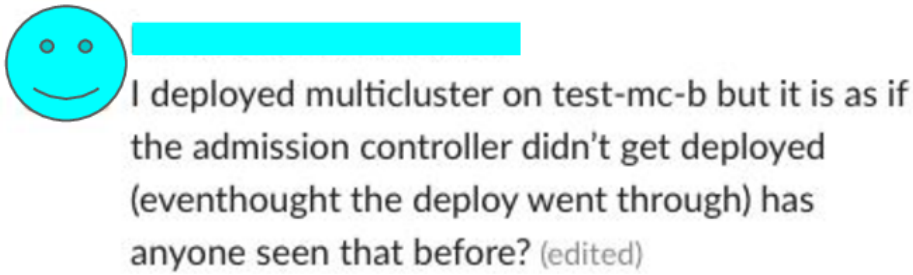
Deny all replicasets!

```go
func init() {
	RegisterDenyValidator(DenyValidator{
		Name:        "deny-new-replicasets-creation",
		Description: "deny all new replicasets creation (meant for a firedrill)",
		Func: func(vr ValidationRequest, params map[string]interface{}) error {
			if vr.Resource != "replicasets" {
				return nil
			}
			if op := vr.Operation; op == v1beta1.Create {
				return fmt.Errorf("admission controller denied creation of new replicasets!")
			}
			return nil
		},
	})
}
```

@jiancheung & Joseph Kim

A few days later...

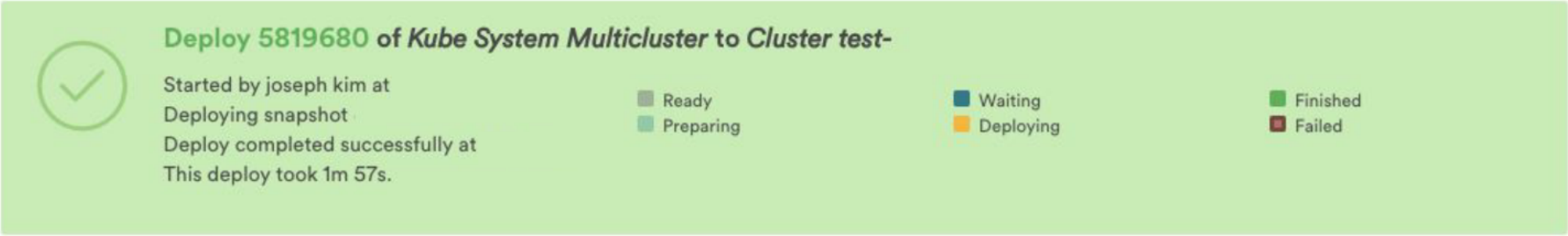
This is no longer a drill



@jiancheung & Joseph Kim

Oops #1

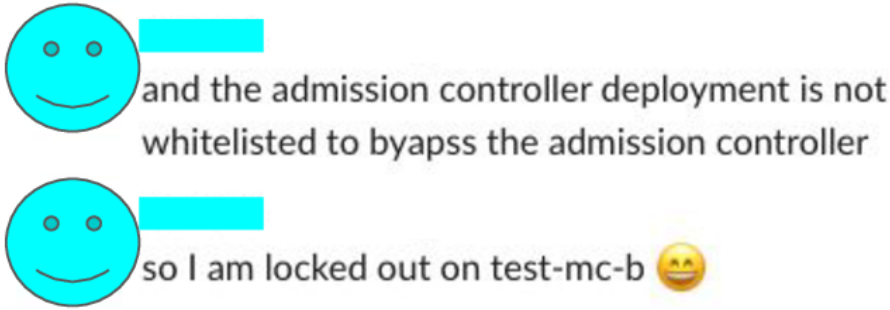
Bug in our deploy process reporting success for failed deploys



@jiancheung & Joseph Kim

Oops #2

Admission-Controller is not whitelisted by Admission-Controller!



The fix

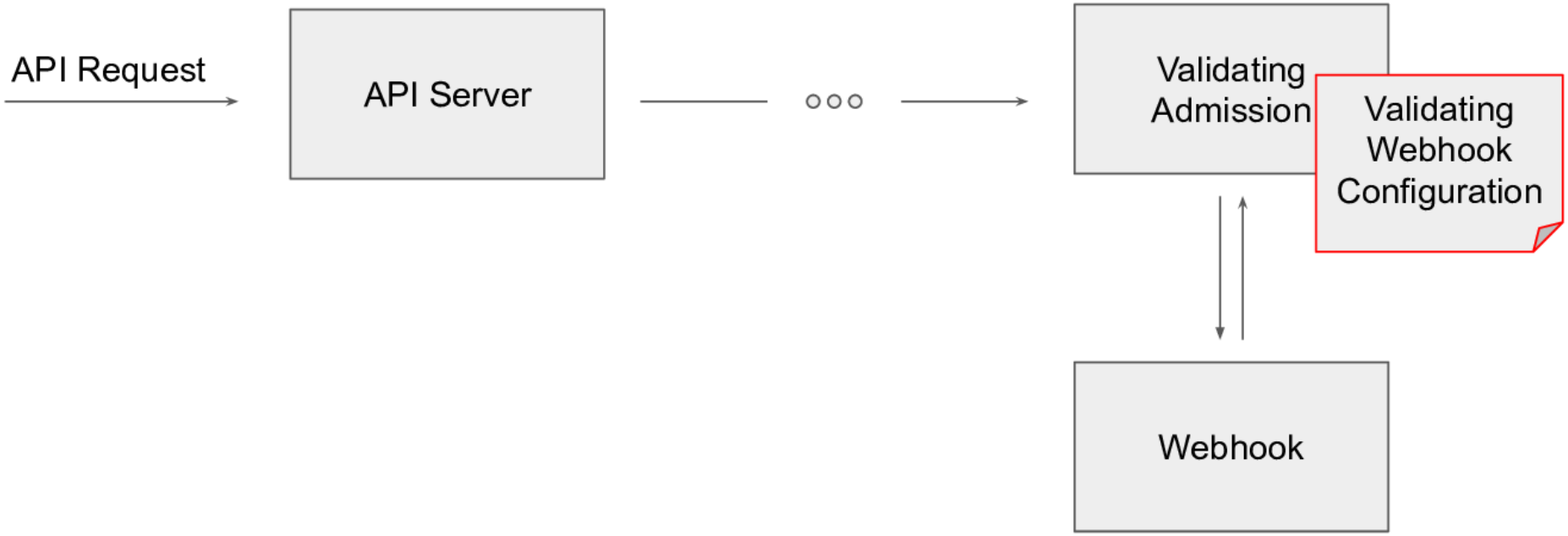
Just temporarily delete the admission-controller webhook configuration

Straight from our runbook!

```
> kubectl --context=prod delete \
validatingwebhookconfiguration/admission-controller-production
```

Oops #3

Admission-Controller webhook configuration is applied before the webhook deployment



@jiancheung & Joseph Kim

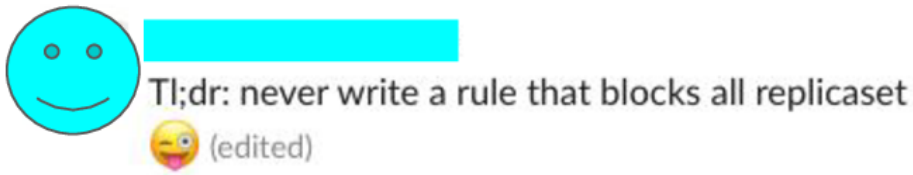
The fix #2

☢☢☢

I'd probably nuke the whole thing... delete the mutaitngadmissioncontroller webhook, and then delete the admission-controller deployment

Takeaways

- Double check that your deploys went through
- Make sure the thing that controls your deploys can fix itself with a deploy
- Be aware of k8s apply ordering
- Even a simple drill can reveal a lot of insights / bugs



Recap

Top 10 Takeaways

- Do test new features in test clusters
- Be aware of the existence of Mutating Admission Controller
- Remember the non-paved paths
- Don’t drain master nodes
- Be ready to scale vertically
- Init containers are versatile despite their simplicity, and are good solutions even if they’re temporary!
- Some throttling is inherent and is okay
- Be wary of what other processes may be running on the pod and adjust limits accordingly
- Have proper error handling & alerts for infrastructural issues
- Be aware of k8s apply ordering

Thanks!

- learn more @ medium.com/airbnb-engineering
- jobs @ airbnb.com/careers
- Contact us at @jiancheung🐦‍⬛ or joseph.kim@airbnb.com!