zalando

Optimizing Kubernetes Resource Requests/Limits

JAX DEVOPS

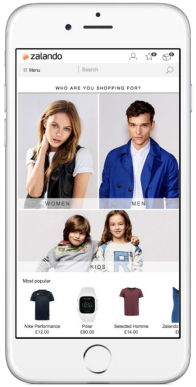
LONDON

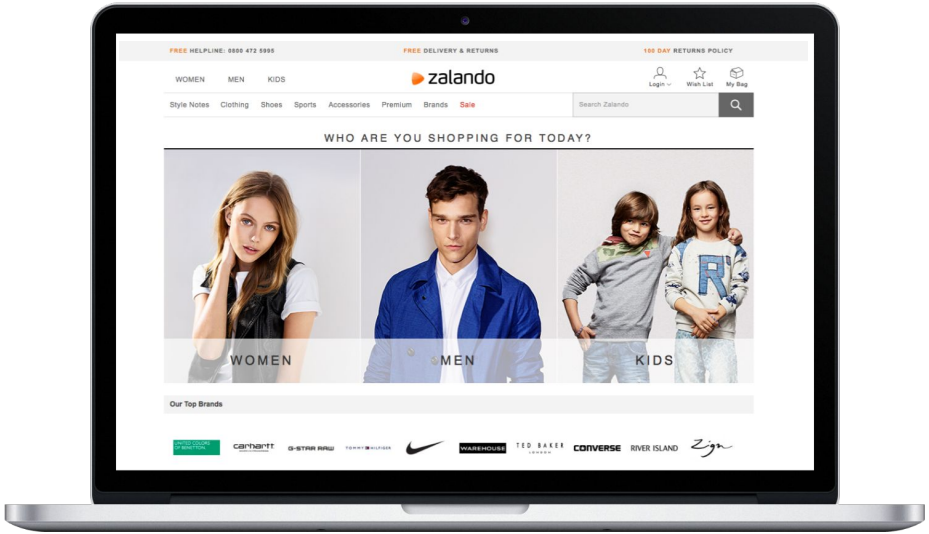
2019-05-15

HENNING JACOBS

@try_except_

EUROPE’S LEADING ONLINE FASHION PLATFORM





zalando

ZALANDO AT A GLANCE

~ 5.4 billion EUR
revenue 2018

> 250 million
visits per month

> 15.000
employees in Europe

> 79%
of visits via mobile devices

> 26 million
active customers

> 300.000
product choices

~ 2.000
brands

17
countries

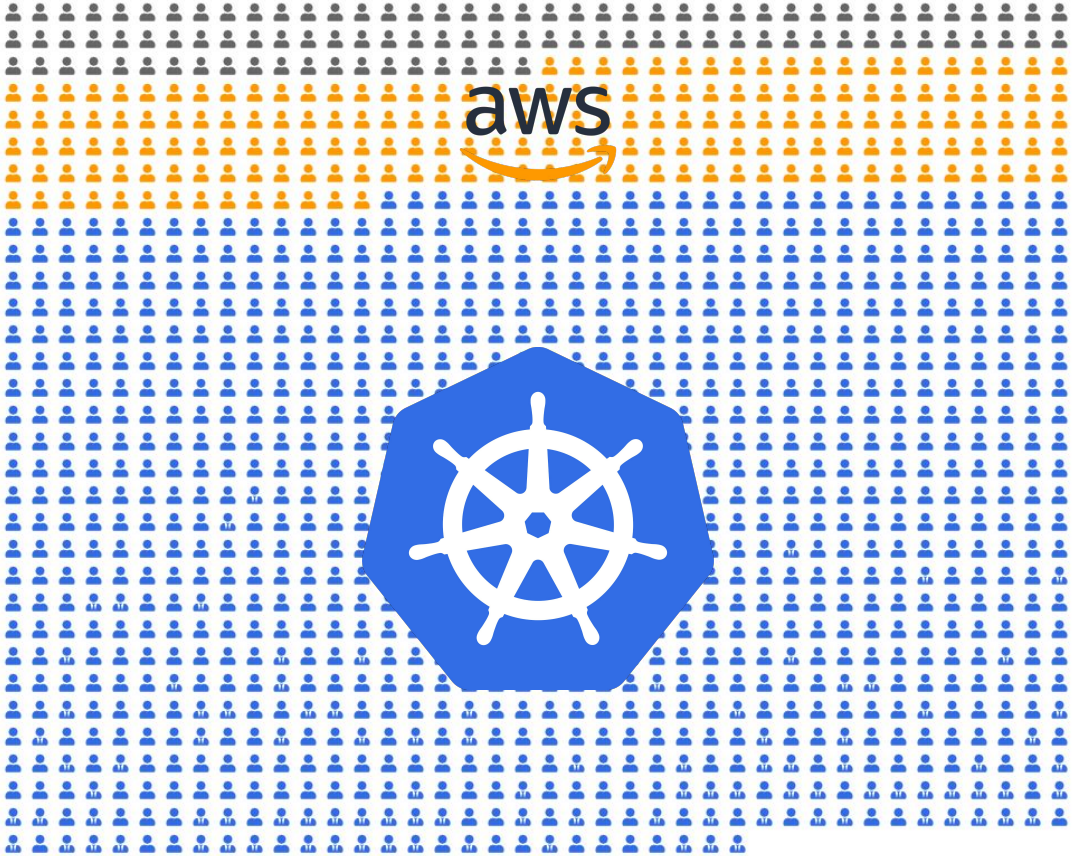
zalando

SCALE

380 Accounts

118 Clusters

DEVELOPERS USING KUBERNETES



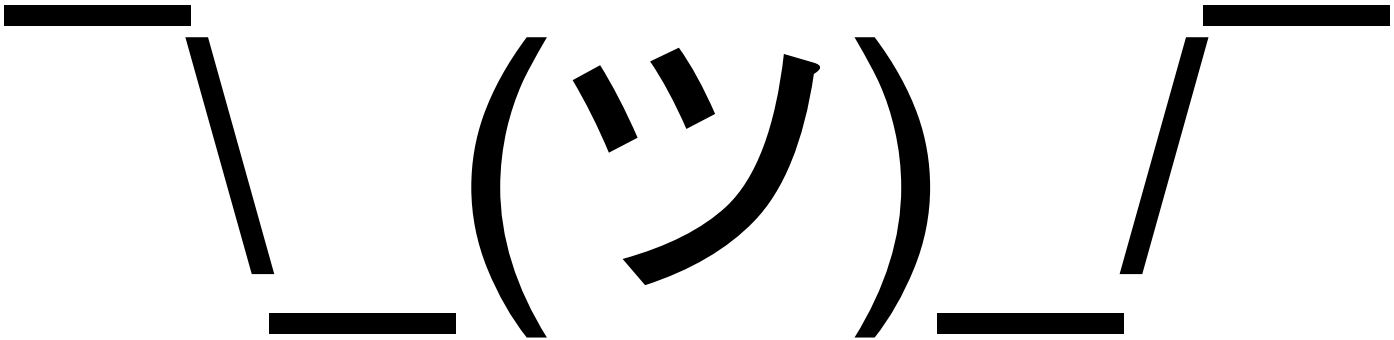
zalando

Lyft has to pay Amazon's cloud at least $8 million a month until the end of 2021

- In its IPO filings, Lyft said it's on the hook to pay Amazon Web Services (AWS) at least $300 million by the end of 2021 for cloud-computing services.

Is this a lot? Is this cost efficient?

zalando

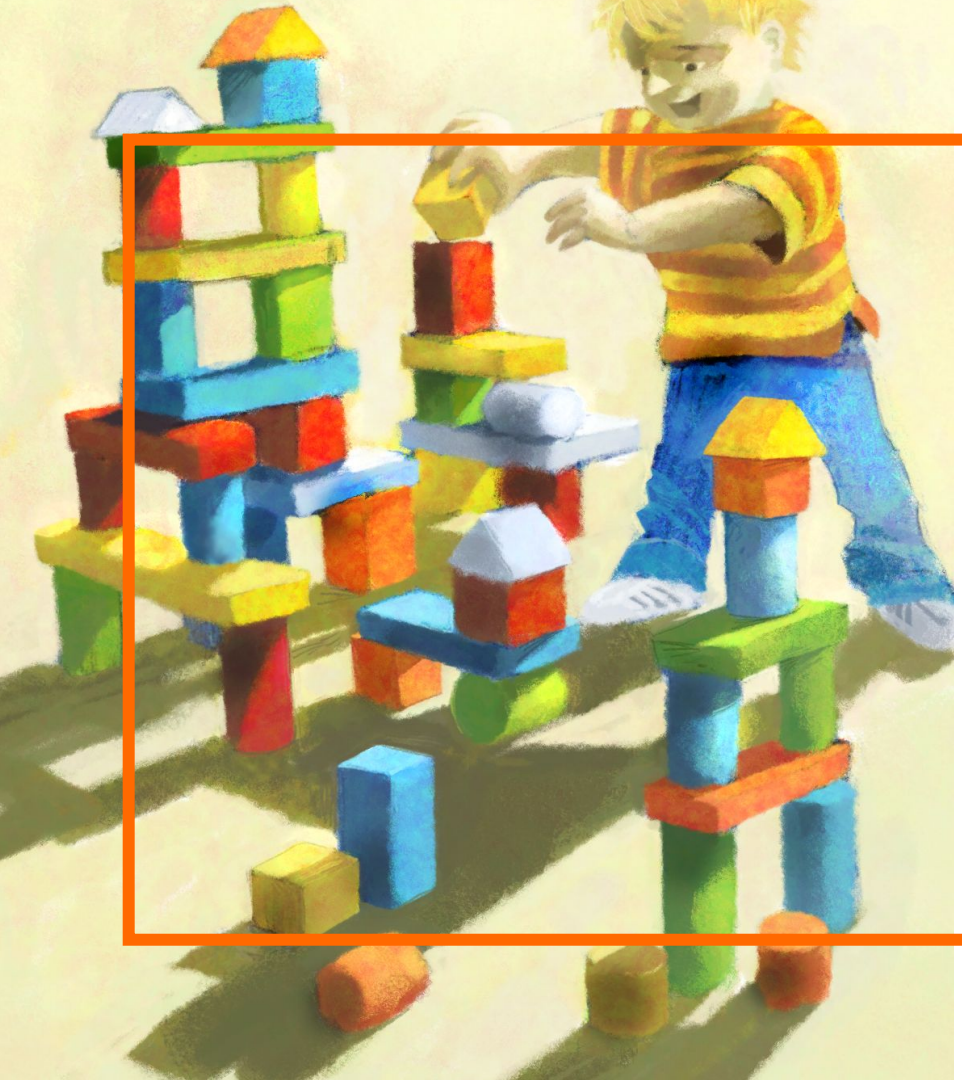


*Do you know **your** per unit costs?*

THE MAGIC DIAL

Speed
Stability
Overprovision
Higher Cost

Efficiency
Risk
Overcommit
Lower Cost



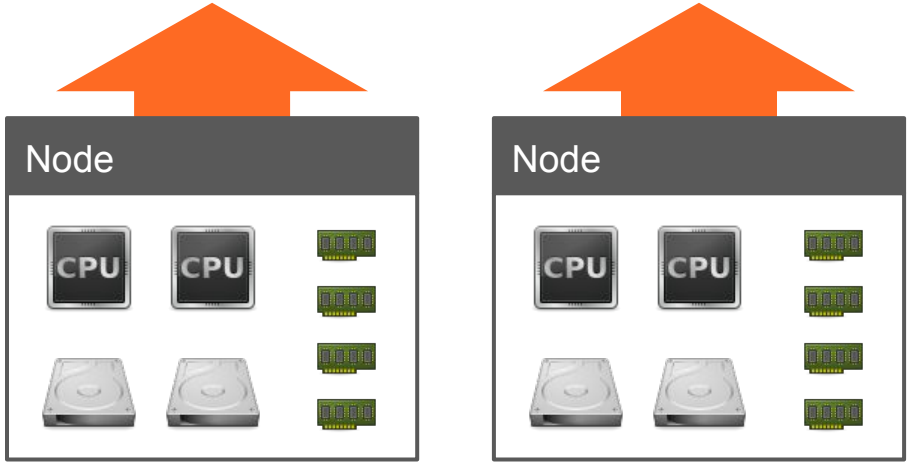
THE BASICS

KUBERNETES: IT'S ALL ABOUT RESOURCES

Pods demand capacity

Scheduler

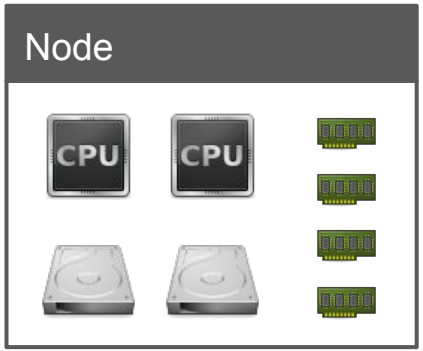
Nodes offer capacity

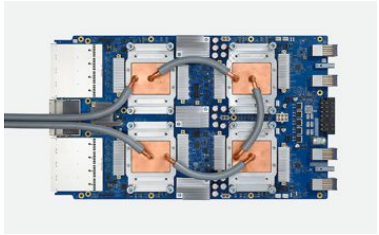


zalando

COMPUTE RESOURCE TYPES

- CPU
- Memory
- *Local ephemeral storage (1.12+)*
- *Extended Resources*
  - *GPU*
  - *TPU?*





KUBERNETES RESOURCES

CPU

- Base: 1 AWS vCPU (or GCP Core or ..)
- Example: **`100m`** (0.1 vCPU, "100 Millicores")

Memory

- Base: 1 Byte
- Example: **`500Mi`** (500 MiB memory)

REQUESTS / LIMITS

Requests

- Affect Scheduling Decision
- Priority (CPU, OOM adjust)

```
resources:
  requests:
    cpu: 100m
    memory: 300Mi
  limits:
    cpu: 1
    memory: 300Mi
```

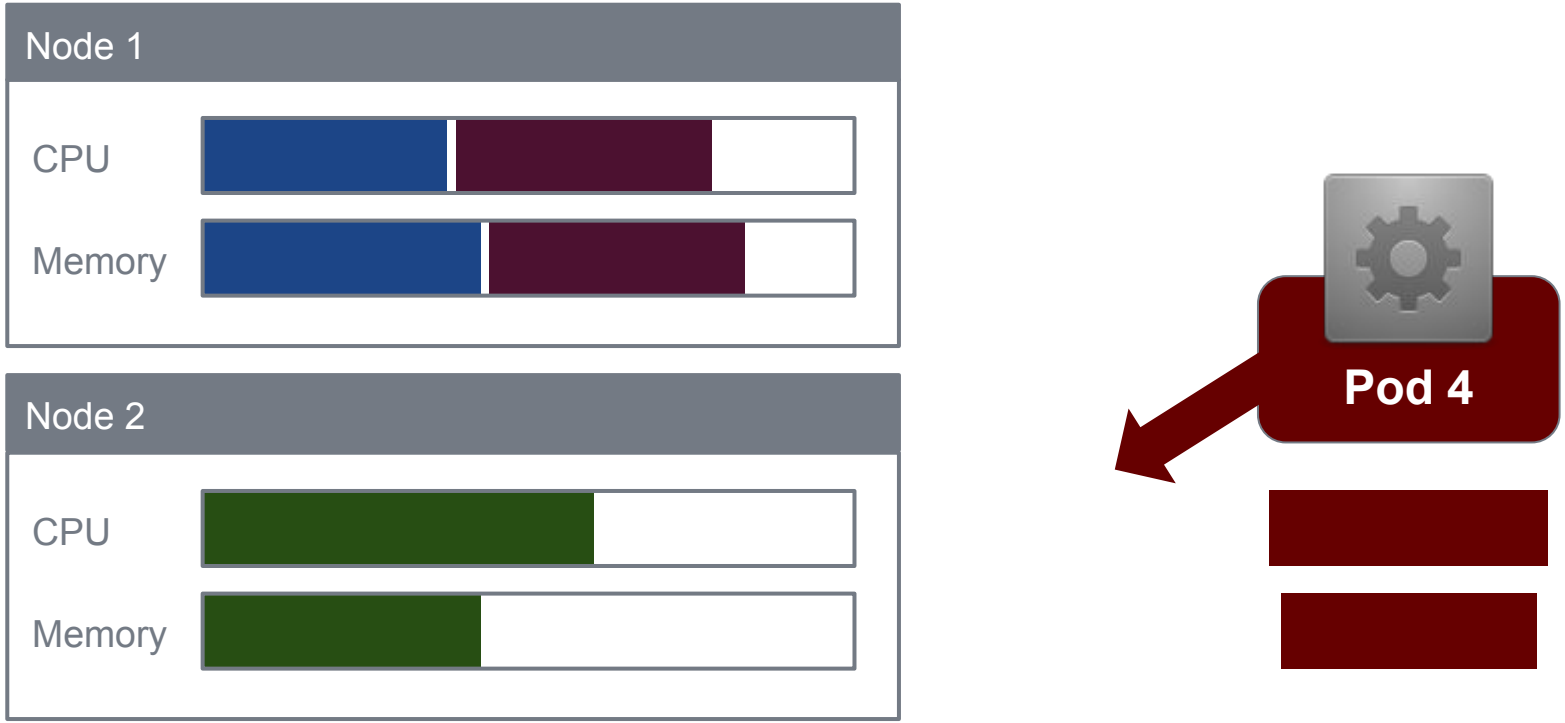
Limits

- Limit maximum container usage

REQUESTS: POD SCHEDULING

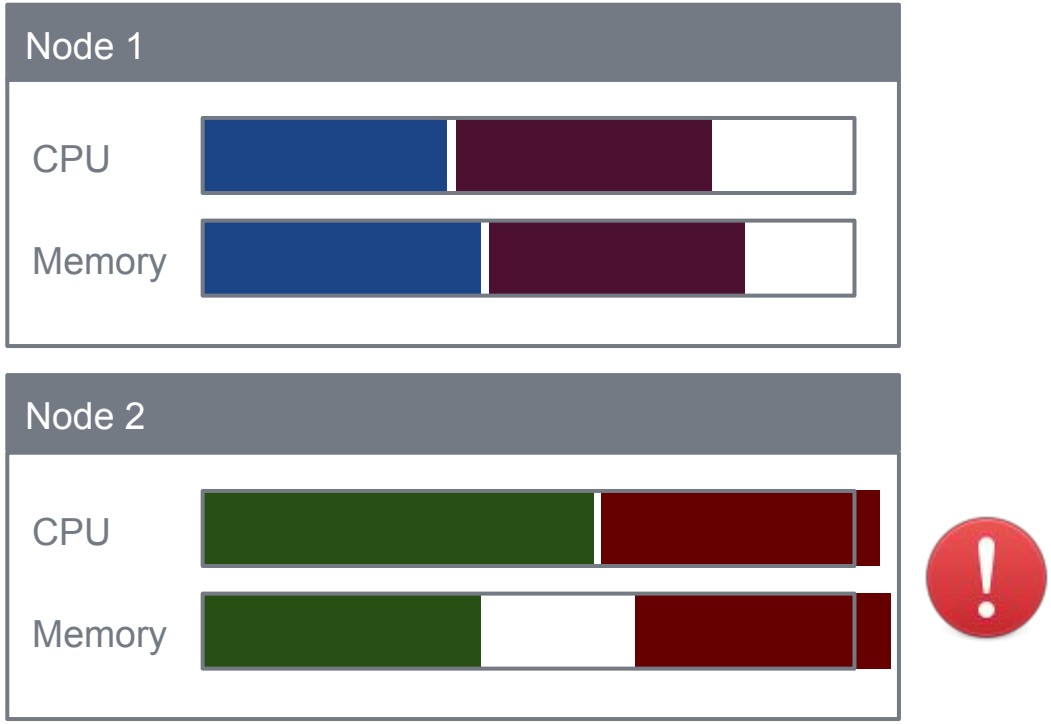
zalando

POD SCHEDULING



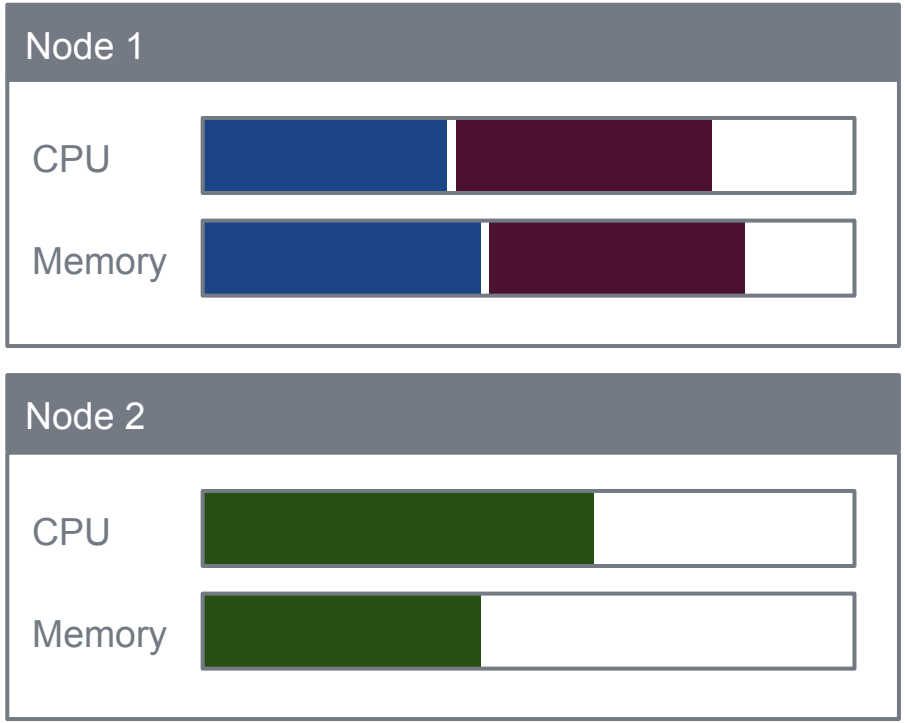
zalando

POD SCHEDULING: TRY TO FIT



zalando

POD SCHEDULING: NO CAPACITY



"PENDING"

zalando

REQUESTS: CPU SHARES

```
kubectl run --requests=cpu=10m/5m ..sha512()..

cat /sys/fs/cgroup/cpu/kubepods/burstable/pod5d5..0d/cpu.shares
10      // relative share of CPU time

cat /sys/fs/cgroup/cpu/kubepods/burstable/pod6e0..0d/cpu.shares
5       // relative share of CPU time

cat /sys/fs/cgroup/cpuacct/kubepods/burstable/pod5d5..0d/cpuacct.usage
    /sys/fs/cgroup/cpuacct/kubepods/burstable/pod6e0..0d/cpuacct.usage
13432815283 // total CPU time in nanoseconds
7528759332  // total CPU time in nanoseconds
```

LIMITS: COMPRESSIBLE RESOURCES

Can be taken away quickly,

"only" cause slowness

CPU Throttling

200m CPU limit

⇒ container can use **0.2s of CPU time per second**

CPU THROTTLING

```
docker run --cpus CPUS -it python
 python -m timeit -s 'import hashlib' -n 10000 -v
 'hashlib.sha512().update(b"foo")'
```

CPUS=1.0	3.8 - 4ms
CPUS=0.5	3.8 - 52ms
CPUS=0.2	6.8 - 88ms
CPUS=0.1	5.7 - 190ms

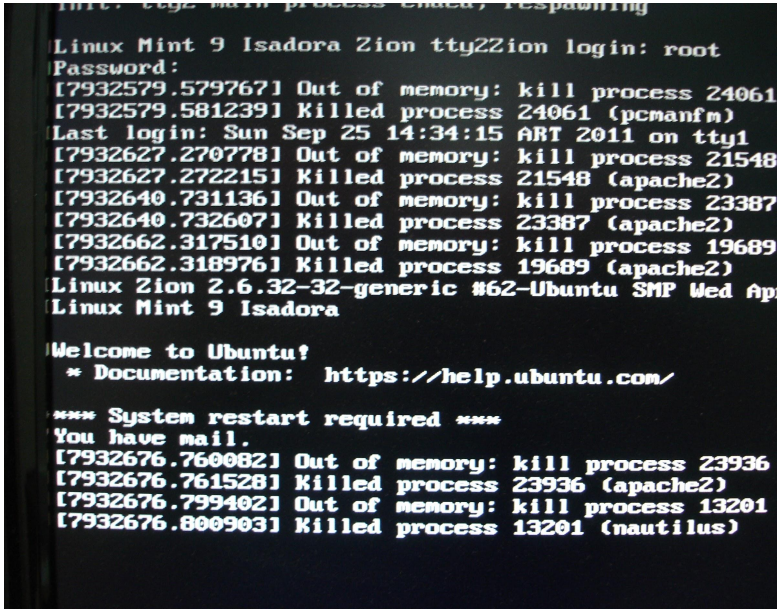
more CPU throttling,
slower hash computation

LIMITS: NON-COMPRESSIBLE RESOURCES

Hold state,

are slower to take away.

⇒ **Killing** (OOMKill)



zalando

MEMORY LIMITS: OUT OF MEMORY

```
kubectl get pod
NAME                          READY    STATUS             RESTARTS    AGE
kube-ops-view-7bc-tcwkt       0/1      CrashLoopBackOff   3           2m


kubectl describe pod kube-ops-view-7bc-tcwkt
 ...
 Last State:    Terminated
   Reason:      OOMKilled
   Exit Code:   137
```

QUALITY OF SERVICE (QOS)

Guaranteed: all containers have limits == requests
Burstable: some containers have limits > requests
BestEffort: no requests/limits set

```
kubectl describe pod ...

  Limits:
    memory:     100Mi
  Requests:
    cpu:        100m
    memory:     100Mi

QoS Class:      Burstable
```

OVERCOMMIT

Limits > Requests ⇒ Burstable QoS ⇒ Overcommit

For CPU: fine, running into completely fair scheduling

For memory: fine, as long as demand < node capacity

Might run into unpredictable OOM situations when demand reaches node's memory capacity (Kernel OOM Killer)

https://code.fb.com/production-engineering/oomd/

LIMITS: CGROUPS

```
docker run --cpus 1 -m 200m --rm -it busybox

cat /sys/fs/cgroup/cpu/docker/8ab25..1c/cpu.{shares,cfs_*}
1024    // cpu.shares (default value)
100000  // cpu.cfs_period_us (100ms period length)
100000  // cpu.cfs_quota_us (total CPU time in µs consumable per period)

cat /sys/fs/cgroup/memory/docker/8ab25..1c/memory.limit_in_bytes
209715200
```

LIMITS: PROBLEMS

1. CPU CFS Quota: **Latency**
2. Memory: accounting, **OOM behavior**

PROBLEMS: LATENCY

szuecs commented on Mar 15 • edited by hjacobs ▾ Member

Reduce latency of ingress 5-10 times, because of 100ms time slices set by Kubernetes for having the limits.

limits were set to `200m` before, which sets:

```
cfs_quota_us /* quota = 20ms */
cpu.cfs_period_us /* period = 100ms */
```

This means that we get 20ms in every 100ms slice and if a request is processed not within one time period of 100ms it will use >100ms for processing the request.

https://github.com/zalando-incubator/kubernetes-on-aws/pull/923

PROBLEMS: HARDCODED PERIOD

```
        // 100000 is equivalent to 100ms
        QuotaPeriod     = 100000
        MinQuotaPeriod = 1000
)

// MilliCPUToQuota converts milliCPU to CFS quota and period values.
func MilliCPUToQuota(milliCPU int64) (quota int64, period uint64) {
        // CFS quota is measured in two values:
        //  - cfs_period_us=100ms (the amount of time to measure usage across)
        //  - cfs_quota=20ms (the amount of cpu time allowed to be used across a period)
        // so in the above example, you are limited to 20% of a single CPU
        // for multi-cpu environments, you just scale equivalent amounts

        if milliCPU == 0 {
                return
        }

        // we set the period to 100ms by default
        period = QuotaPeriod
```

PROBLEMS: HARDCODED PERIOD

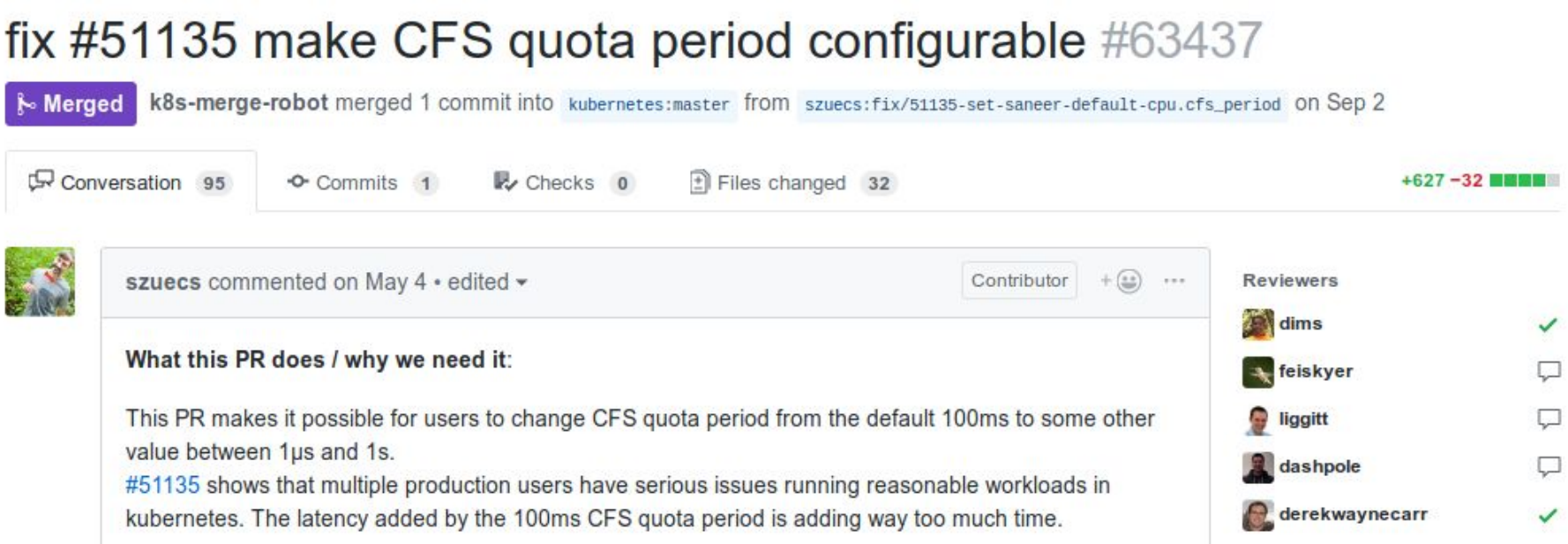
obeattie commented on Apr 27

For what it's worth, we (Monzo) have found 5ms to be a "good" value for *most* of our latency-sensitive workloads and run with this. There are some exceptions though.

👍 2

https://github.com/kubernetes/kubernetes/issues/51135

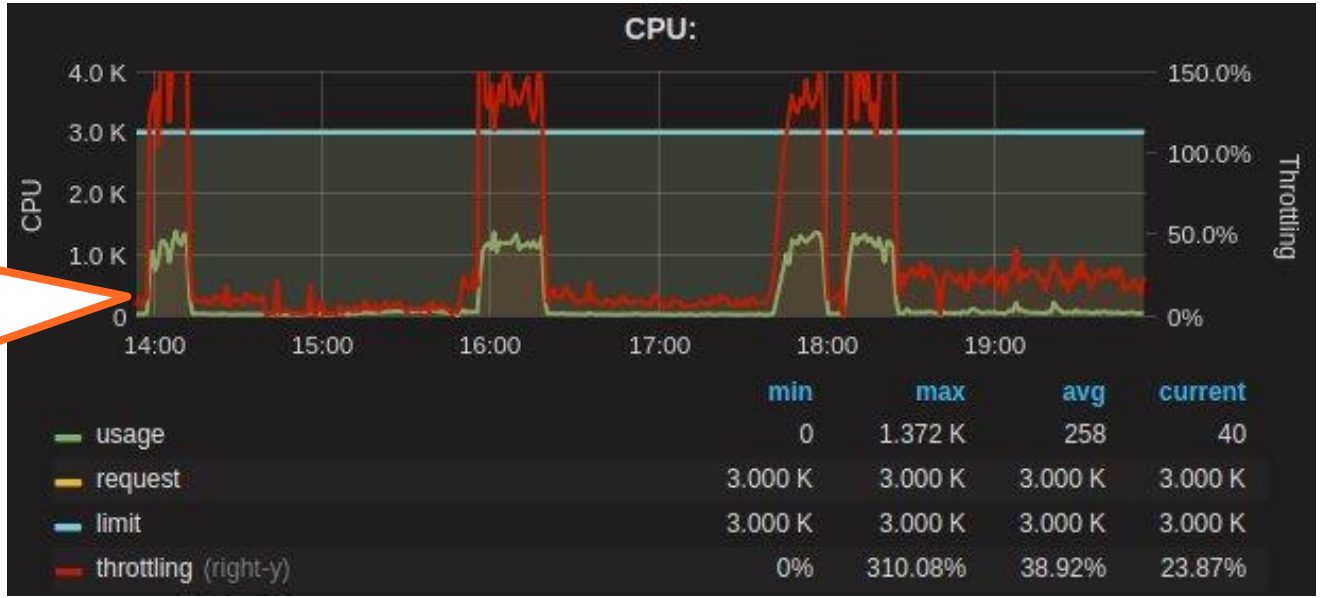
NOW IN KUBERNETES 1.12



https://github.com/kubernetes/kubernetes/pull/63437

OVERLY AGGRESSIVE CFS

Usage < Limit, but heavy throttling



zalando

OVERLY AGGRESSIVE CFS: EXPERIMENT #1

CPU Period: 100ms

CPU Quota: None

Burn 5ms and sleep 100ms

⇒ Quota disabled

⇒ **No Throttling expected!**

https://gist.github.com/bobrik/2030ff040fad360327a5fab7a09c4ff1

EXPERIMENT #1: NO QUOTA, NO THROTTLING

```
2018/11/03 13:04:02 [0] burn took 5ms, real time so far: 5ms, cpu time so far: 6ms
2018/11/03 13:04:03 [1] burn took 5ms, real time so far: 510ms, cpu time so far: 11ms
2018/11/03 13:04:03 [2] burn took 5ms, real time so far: 1015ms, cpu time so far: 17ms
2018/11/03 13:04:04 [3] burn took 5ms, real time so far: 1520ms, cpu time so far: 23ms
2018/11/03 13:04:04 [4] burn took 5ms, real time so far: 2025ms, cpu time so far: 29ms
2018/11/03 13:04:05 [5] burn took 5ms, real time so far: 2530ms, cpu time so far: 35ms
2018/11/03 13:04:05 [6] burn took 5ms, real time so far: 3036ms, cpu time so far: 40ms
2018/11/03 13:04:06 [7] burn took 5ms, real time so far: 3541ms, cpu time so far: 46ms
2018/11/03 13:04:06 [8] burn took 5ms, real time so far: 4046ms, cpu time so far: 52ms
2018/11/03 13:04:07 [9] burn took 5ms, real time so far: 4551ms, cpu time so far: 58ms
```

OVERLY AGGRESSIVE CFS: EXPERIMENT #2

CPU Period: 100ms

CPU Quota: 20ms

Burn 5ms and sleep 500ms

⇒ No 100ms intervals where possibly 20ms is burned

⇒ **No Throttling expected!**

EXPERIMENT #2: OVERLY AGGRESSIVE CFS

```
2018/11/03 13:05:05 [0] burn took 5ms, real time so far: 5ms, cpu time so far: 5ms
2018/11/03 13:05:06 [1] burn took 99ms, real time so far: 690ms, cpu time so far: 9ms
2018/11/03 13:05:06 [2] burn took 99ms, real time so far: 1290ms, cpu time so far: 14ms
2018/11/03 13:05:07 [3] burn took 99ms, real time so far: 1890ms, cpu time so far: 18ms
2018/11/03 13:05:07 [4] burn took 5ms, real time so far: 2395ms, cpu time so far: 24ms
2018/11/03 13:05:08 [5] burn took 94ms, real time so far: 2990ms, cpu time so far: 27ms
2018/11/03 13:05:09 [6] burn took 99ms, real time so far: 3590ms, cpu time so far: 32ms
2018/11/03 13:05:09 [7] burn took 5ms, real time so far: 4095ms, cpu time so far: 37ms
2018/11/03 13:05:10 [8] burn took 5ms, real time so far: 4600ms, cpu time so far: 43ms
2018/11/03 13:05:10 [9] burn took 5ms, real time so far: 5105ms, cpu time so far: 49ms
```

OVERLY AGGRESSIVE CFS: EXPERIMENT #3

CPU Period: 10ms

CPU Quota: 2ms

Burn 5ms and sleep 100ms

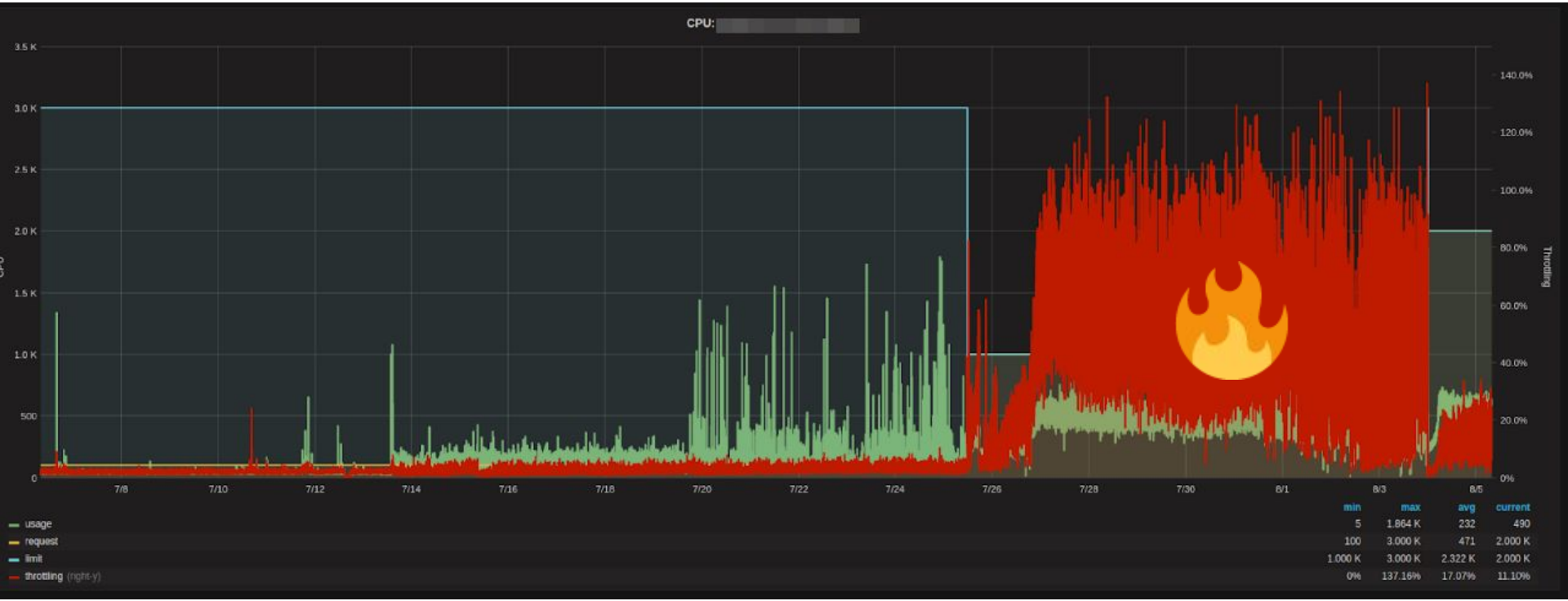
⇒ Same 20% CPU (200m) limit, but smaller period

⇒ **Throttling expected!**

SMALLER CPU PERIOD ⇒ BETTER LATENCY

```
2018/11/03 16:31:07 [0] burn took 18ms, real time so far: 18ms, cpu time so far: 6ms
2018/11/03 16:31:07 [1] burn took 9ms, real time so far: 128ms, cpu time so far: 8ms
2018/11/03 16:31:07 [2] burn took 9ms, real time so far: 238ms, cpu time so far: 13ms
2018/11/03 16:31:07 [3] burn took 5ms, real time so far: 343ms, cpu time so far: 18ms
2018/11/03 16:31:07 [4] burn took 30ms, real time so far: 488ms, cpu time so far: 24ms
2018/11/03 16:31:07 [5] burn took 19ms, real time so far: 608ms, cpu time so far: 29ms
2018/11/03 16:31:07 [6] burn took 9ms, real time so far: 718ms, cpu time so far: 34ms
2018/11/03 16:31:08 [7] burn took 5ms, real time so far: 824ms, cpu time so far: 40ms
2018/11/03 16:31:08 [8] burn took 5ms, real time so far: 943ms, cpu time so far: 45ms
2018/11/03 16:31:08 [9] burn took 9ms, real time so far: 1068ms, cpu time so far: 48ms
```

INCIDENT INVOLVING CPU THROTTLING



https://k8s.af

zalando

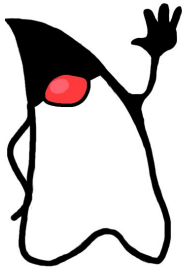
LIMITS: VISIBILITY

```
docker run --cpus 1 -m 200m --rm -it busybox top
```

```
Mem: 7369128K used, 726072K free, 128164K shrd, 303924K buff, 1208132K cached
CPU0: 14.8% usr  8.4% sys  0.2% nic 67.6% idle  8.2% io  0.0% irq  0.6% sirq
CPU1:  8.8% usr 10.3% sys  0.0% nic 75.9% idle  4.4% io  0.0% irq  0.4% sirq
CPU2:  7.3% usr  8.7% sys  0.0% nic 63.2% idle 20.1% io  0.0% irq  0.6% sirq
CPU3:  9.3% usr  9.9% sys  0.0% nic 65.7% idle 14.5% io  0.0% irq  0.4% sirq
```

LIMITS: VISIBILITY

- Container-aware memory configuration
  - JVM MaxHeap
- Container-aware processor configuration
  - Thread pools
  - GOMAXPROCS
  - node.js cluster module



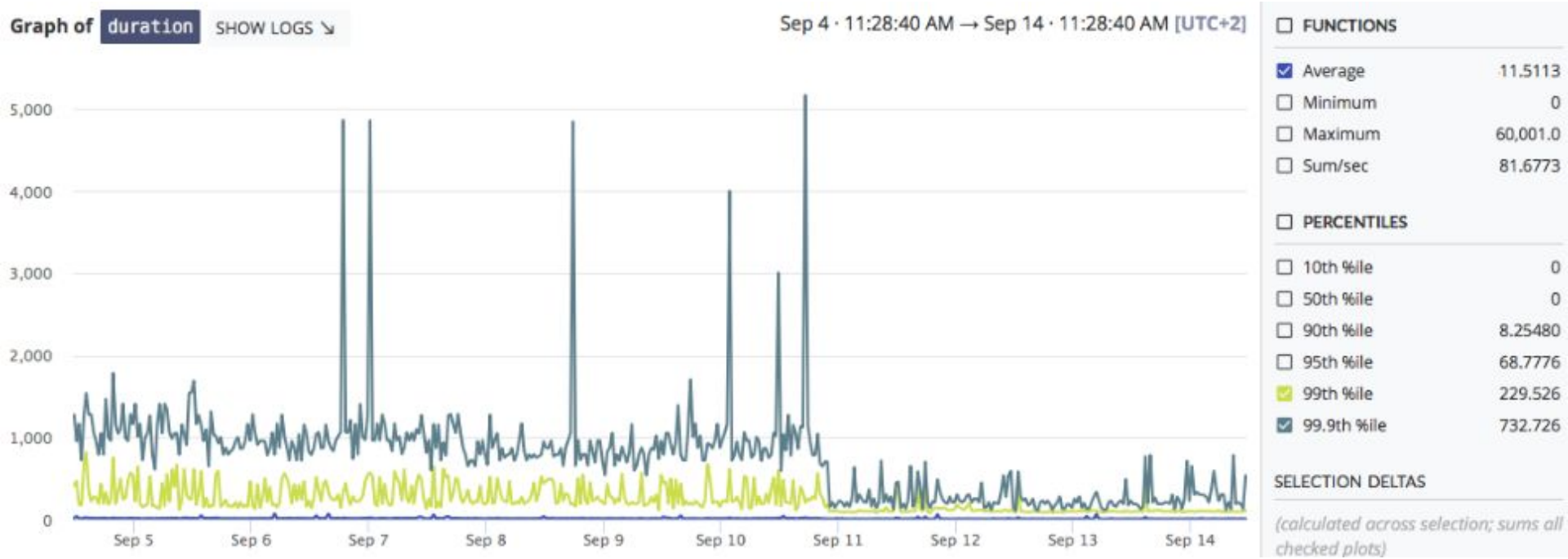

Henning Jacobs
@try_except_
Plot twist: Google engineers recommend setting Kubelet flag --cpu-cfs-quota=false to disable CPU quota in #Kubernetes #KubeCon 😮
Tweet übersetzen
Avoid setting CPU limits for Guaranteed pods · Is...
The effect of CPU throttling is non obvious to users and throws them off when they try out kubernetes. It also complicates CPU capacity planning for pods. Pods t...
github.com
16:50 - 3. Mai 2018
5 Retweets 8 „Gefällt mir"-Angaben

zalando

ZALANDO: DECISION

1. **Forbid Memory Overcommit**
   - Implement mutating admission webhook
   - Set requests = limits
2. **Disable CPU CFS Quota in all clusters**
   - `--cpu-cfs-quota=false`

INGRESS LATENCY IMPROVEMENT



zalando

CLUSTER AUTOSCALER

Simulates the Kubernetes scheduler internally to find out..

- ..if any of the pods wouldn't fit on existing nodes
  ⇒ upscale is needed
- ..if it's possible to fit some of the pods on existing nodes
  ⇒ downscale is needed

⇒ **Cluster size is determined by resource requests**
(+ constraints)

github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler

AUTOSCALING BUFFER

- Cluster Autoscaler only triggers on Pending Pods
- Node provisioning is slow

⇒ Reserve extra capacity via low priority Pods

"Autoscaling Buffer Pods"

AUTOSCALING BUFFER

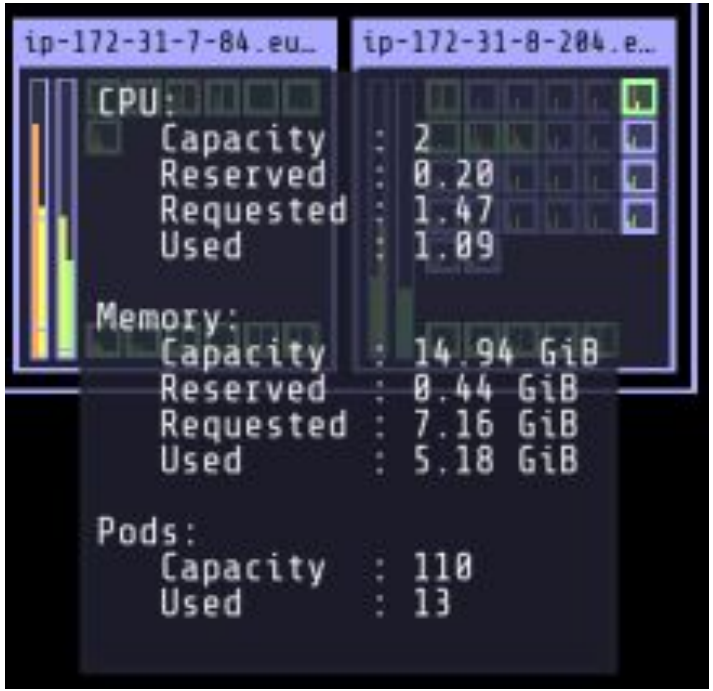
```
kubectl describe pod autoscaling-buffer-..zjq5 -n kube-system
...
Namespace:                  kube-system
Priority:                   -1000000
PriorityClassName:          autoscaling-buffer
Containers:
  pause:
    Image:                  teapot/pause-amd64:3.1
    Requests:
      cpu:                  1600m
      memory:               6871947673
```

Evict if higher priority (default) Pod needs capacity

ALLOCATABLE

Reserve resources for
system components,
Kubelet, and container runtime:

```
--system-reserved=\
   cpu=100m,memory=164Mi
--kube-reserved=\
   cpu=100m,memory=282Mi
```

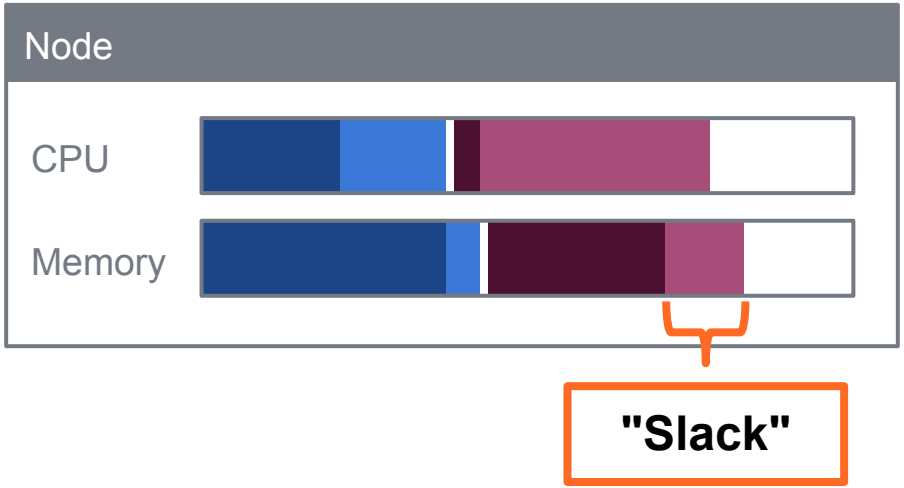


zalando

SLACK

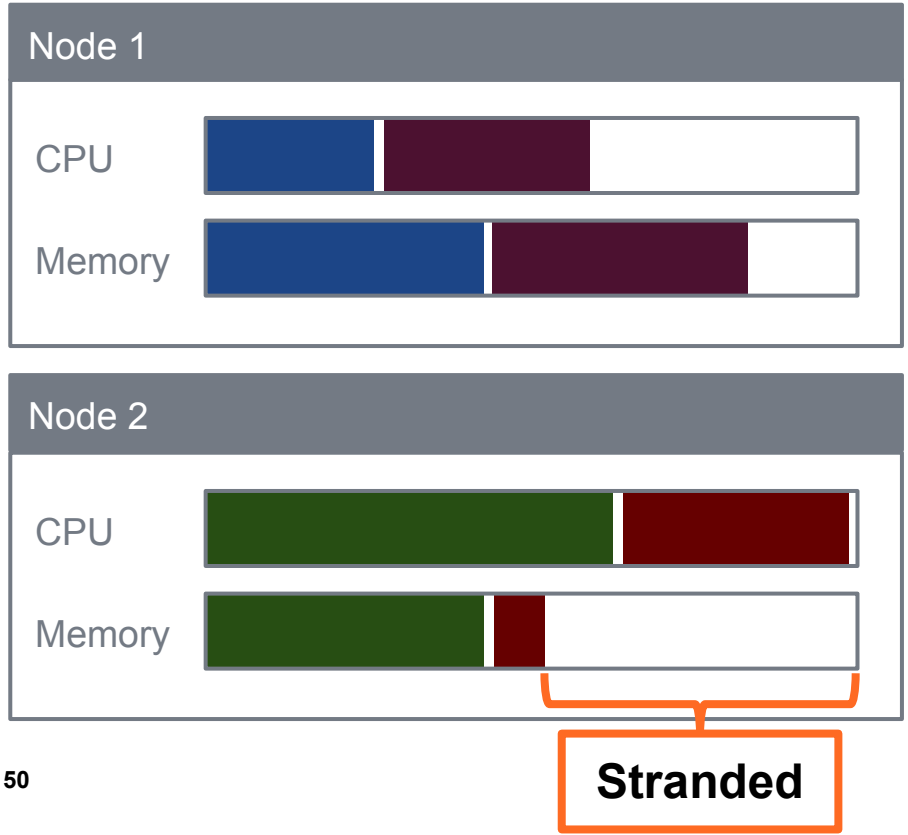
CPU/memory requests "block" resources on nodes.

Difference between actual usage and requests → **Slack**



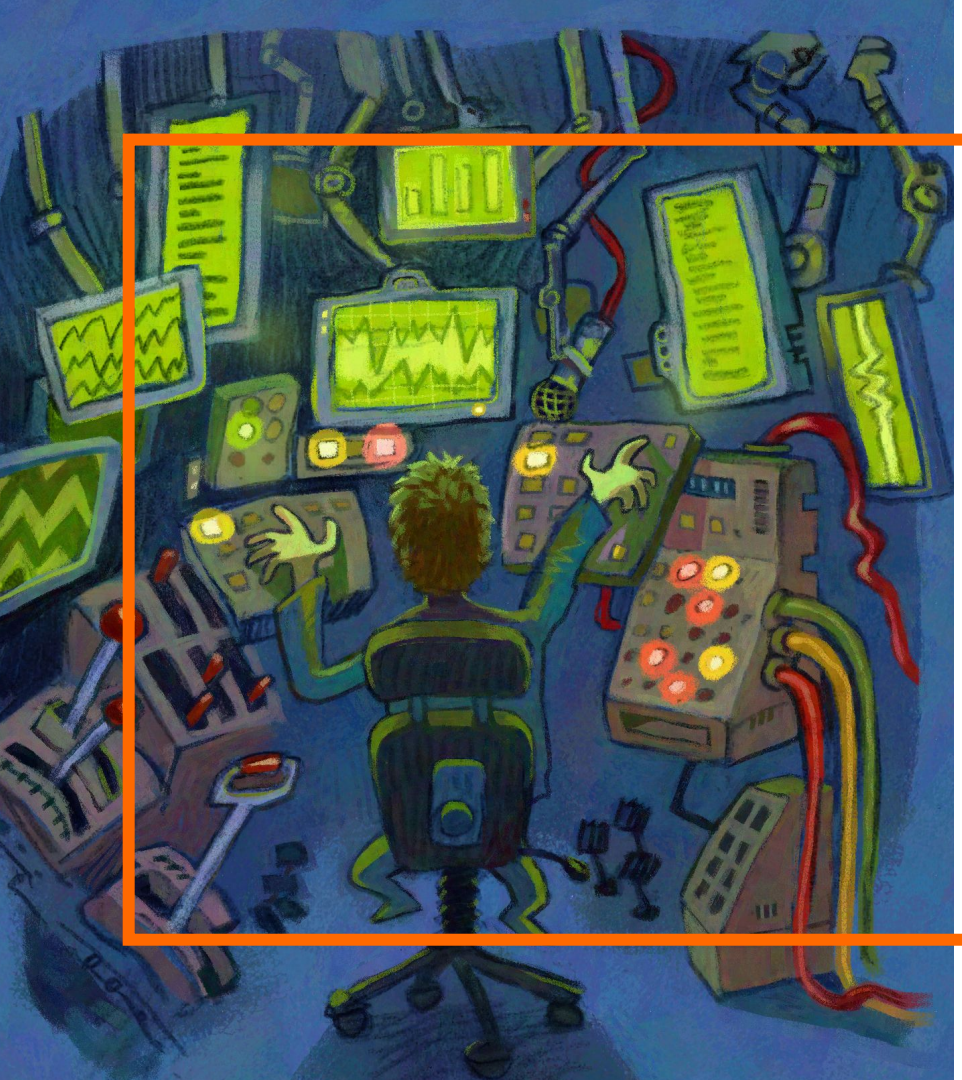
zalando

STRANDED RESOURCES



Some available capacity can become unusable / stranded.

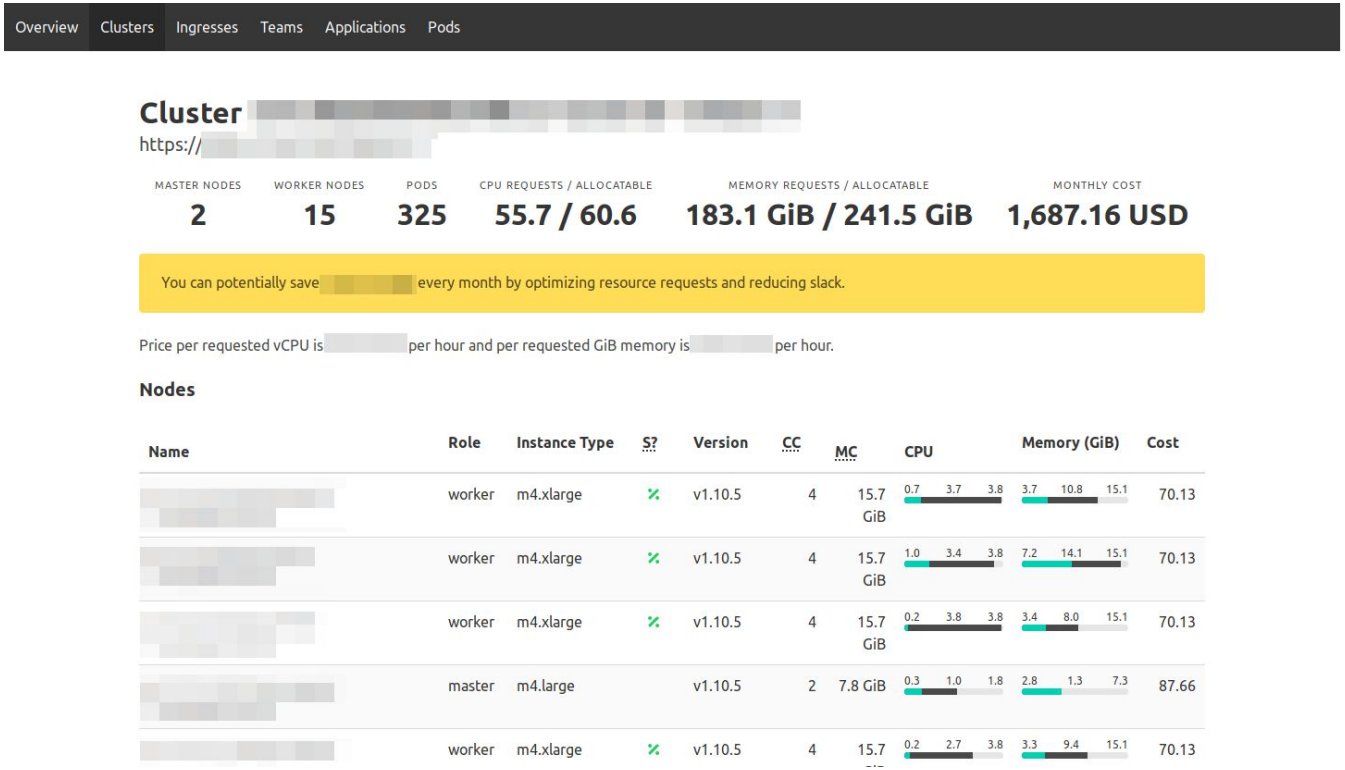
⇒ Reschedule, bin packing



MONITORING
COST EFFICIENCY

zalando

KUBERNETES RESOURCE REPORT



github.com/hjacobs/kube-resource-report

zalando

RESOURCE REPORT: TEAMS

ID	C	A	P	CR	MR	CPU	Memory (MiB)	Cost	Slack Cost
[redacted]	3	14	114	457.9	1.7 TiB	69.01 / 457.9	1,074,024 / 1,780,420	27,558.50	13,327.47
[redacted]	1	9	251	428.95	426.2 GiB	137.99 / 428.95	164,613 / 436,384	19,406.82	13,111.24

Sorting teams by Slack Costs

github.com/hjacobs/kube-resource-report

RESOURCE REPORT: APPLICATIONS

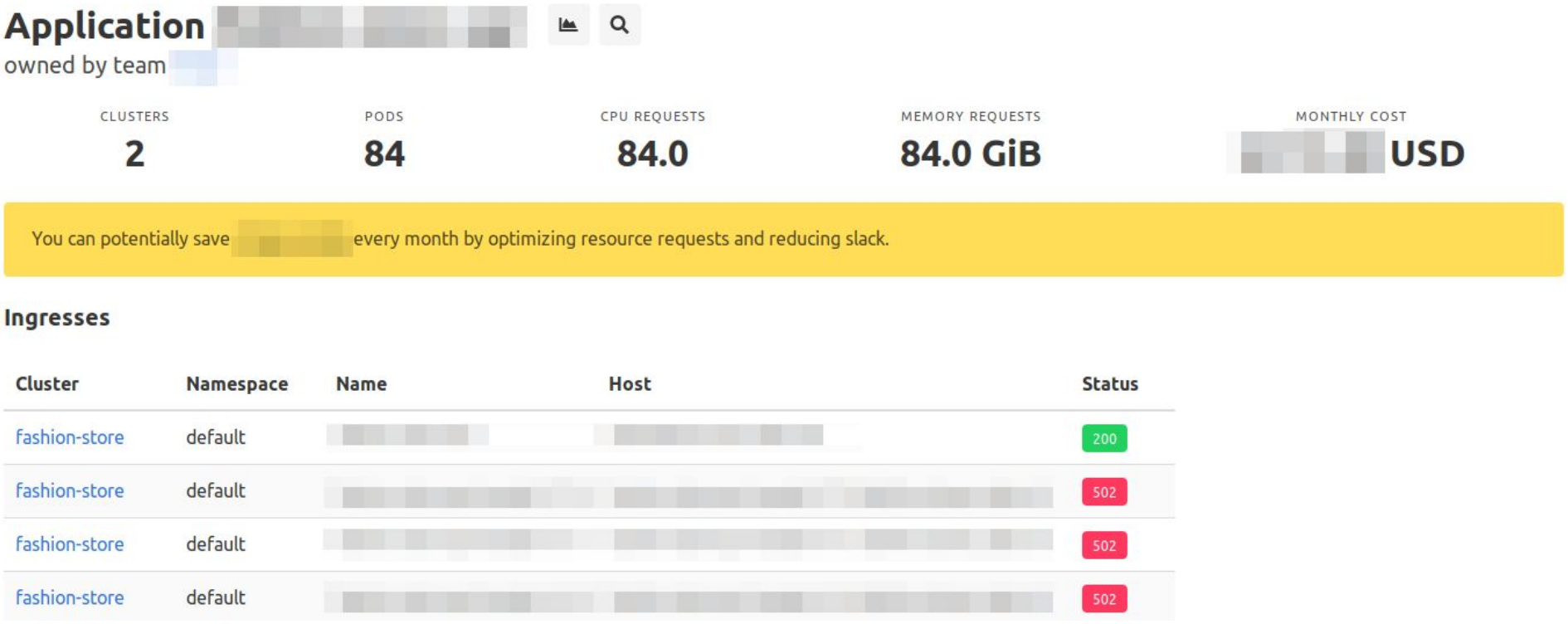
You can potentially save **2,474.14 USD** every month by optimizing resource requests and reducing slack.

Applications

ID	A?	C	P	CR	MR	CPU	Memory (MiB)	Cost	Slack Cost
felicity	✓	2	16	40.0	109.4 GiB	26.97 / 40.0	75,333 / 112,000	2,126.64	598.24
live-kafka-	✓	1	12	24.0	175.8 GiB	5.93 / 24.0	125,274 / 180,000	1,858.43	565.02
live-elasticsearch-	✓	1	6	9.0	82.0 GiB	0.01 / 9.0	30,107 / 84,000	970.18	659.34
live-	✓	1	16	14.0	87.5 GiB	7.47 / 14.0	85,167 / 89,600	925.08	45.77

"Slack"

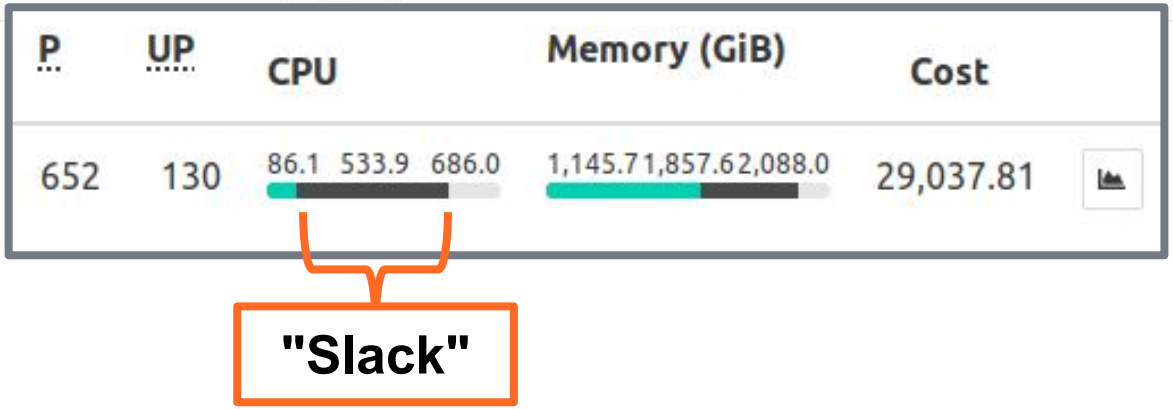
RESOURCE REPORT: APPLICATIONS



zalando

RESOURCE REPORT: CLUSTERS

Cluster	MN	WN	Inst. Types	S?	Version	I	P	UP	CPU	Memory (GiB)	Cost
[redacted]	2	58	c5d.2xlarge, m5.large, m5.4xlarge, c5.2xlarge, m5d.2xlarge, c5.4xlarge		v1.12.5-custom.master-1		26	652	130	86.1 533.9 686.0	1,145.7 1,857.6 2,088.0	29,037.81
[redacted]	1	18	m5d.xlarge, r4.xlarge	%	v1.12.5-custom.master-1		13	223	36	7.7 32.2 70.2	195.9 227.1 374.5	3,986.41



github.com/hjacobs/kube-resource-report

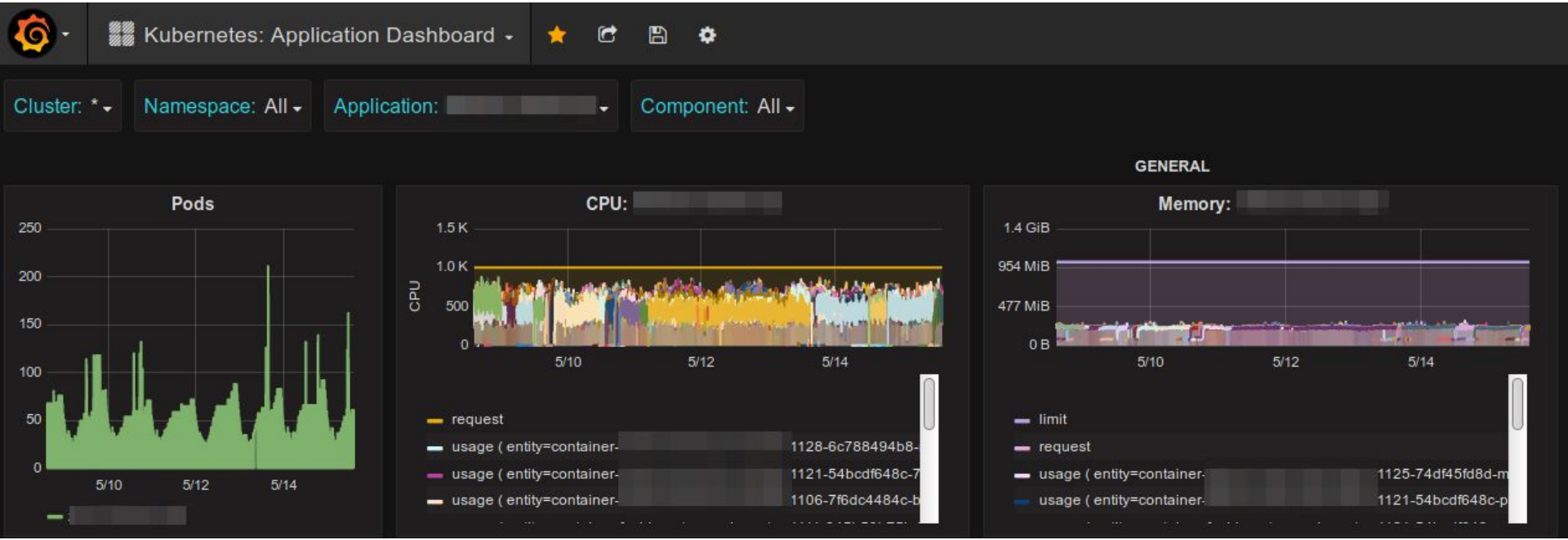
zalando

RESOURCE REPORT METRICS

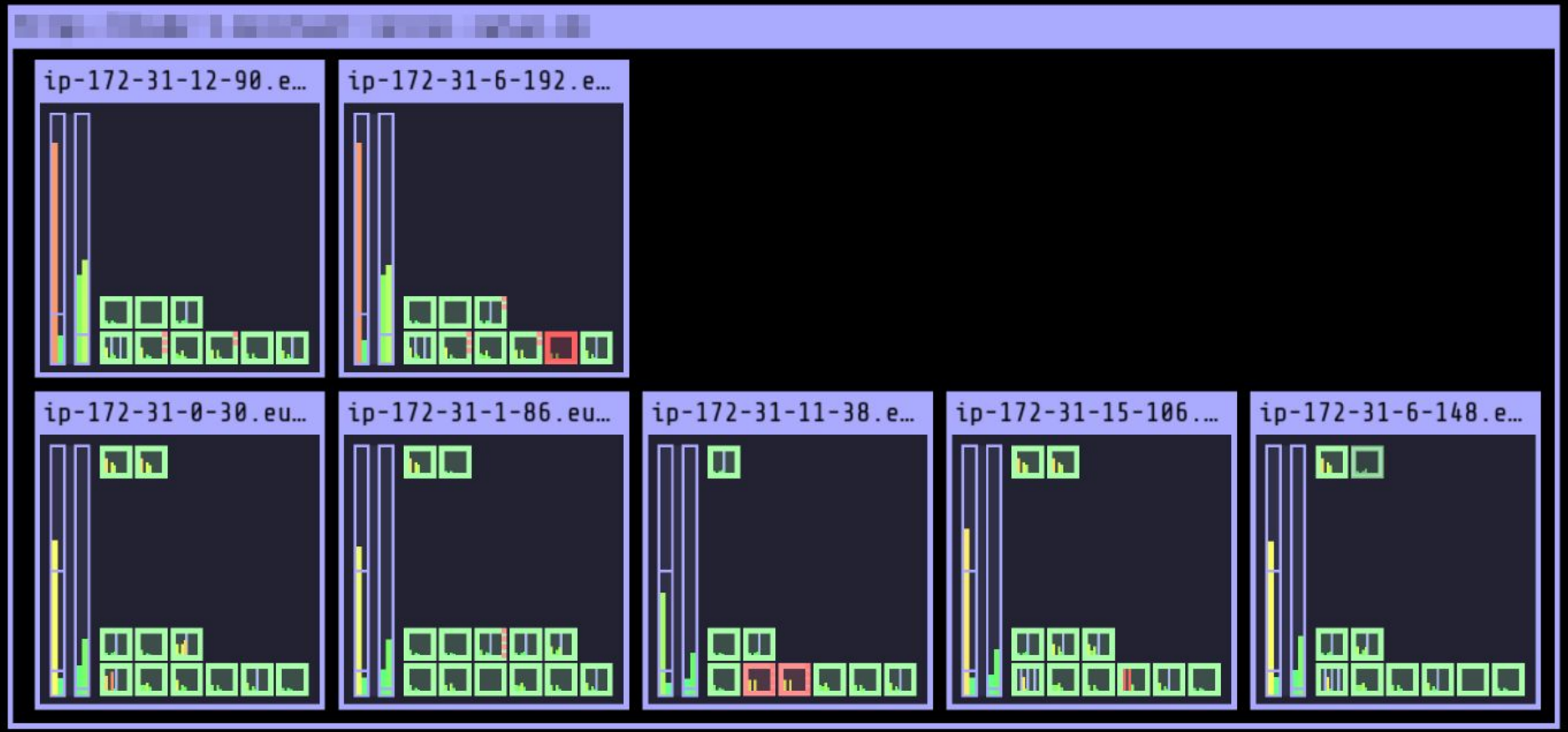
Slack				
Metric	Min	Max	Avg ▾	Current
.slack_cost)	$3.30K	$9.42K	$6.57K	$5.91K
.slack_cost)	$4.97K	$7.11K	$6.28K	$5.75K
slack_cost)	$813.44	$1.85K	$1.23K	$1.40K
.slack_cost)	$77.38	$468.11	$218.58	$326.99
.slack_cost)	$62.80	$160.74	$94.54	$79.25
slack_cost)	$1.84	$91.43	$58.43	$59.43
.slack_cost)	$3.10	$45.72	$12.56	$26.67
.slack_cost)	$4.37	$9.02	$7.28	$6.77
.slack_cost)	$-6.24	$11.43	$2.95	$1.23
.slack_cost)	$1.08	$2.19	$1.46	$1.34
.slack_cost)	$-14.74	$2.82	$-8.41	$-13.44

github.com/hjacobs/kube-resource-report

KUBERNETES APPLICATION DASHBOARD



zalando



https://github.com/hjacobs/kube-ops-view

https://github.com/hjacobs/kube-ops-view

OPTIMIZING
COST EFFICIENCY

zalando

VERTICAL POD AUTOSCALER (VPA)

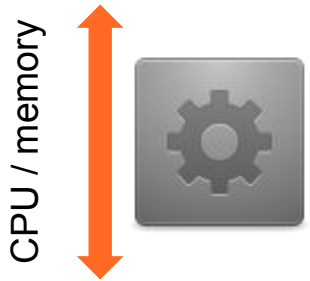
"Some 2/3 of the (Google) Borg users use autopilot."

- *Tim Hockin*

VPA: Set resource requests automatically based on usage.

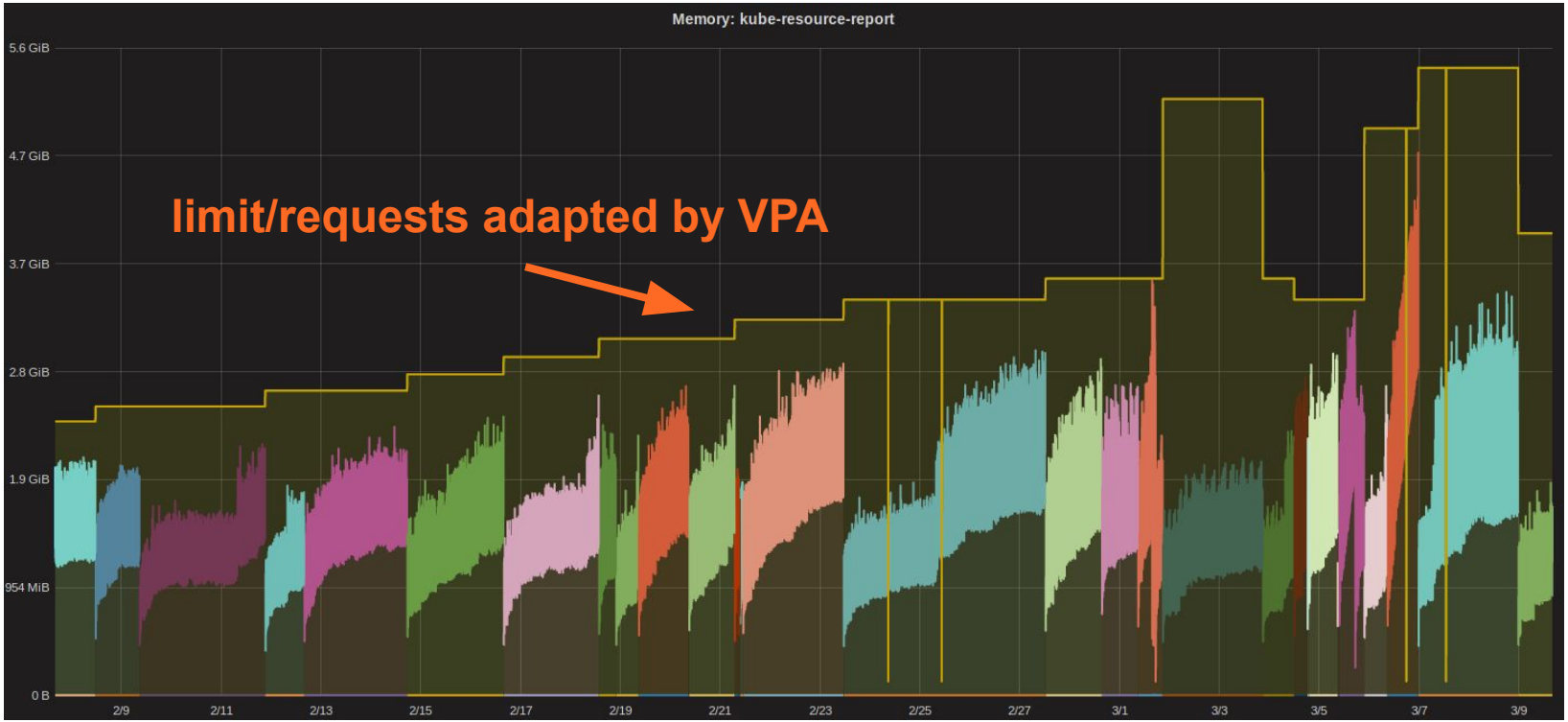
VPA FOR PROMETHEUS

```
apiVersion: autoscaling.k8s.io/v1beta2
kind: VerticalPodAutoscaler
metadata: ...
spec:
  targetRef:
     apiVersion: apps/v1
     kind: StatefulSet
     name: prometheus
  updatePolicy: { updateMode: Auto }
  resourcePolicy:
     containerPolicies: { containerName: prometheus }
     minAllowed:
        memory: 512Mi
     maxAllowed:
        memory: 10Gi
```



zalando

VERTICAL POD AUTOSCALER



zalando

VERTICAL POD AUTOSCALER

zalando

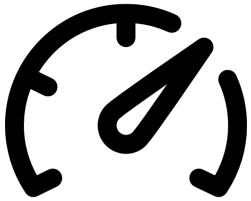
HORIZONTAL POD AUTOSCALER

```yaml
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: myapp
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: myapp
  minReplicas: 3
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 100
```

target: ~100% of CPU requests

HORIZONTAL POD AUTOSCALING (CUSTOM METRICS)

Queue Length



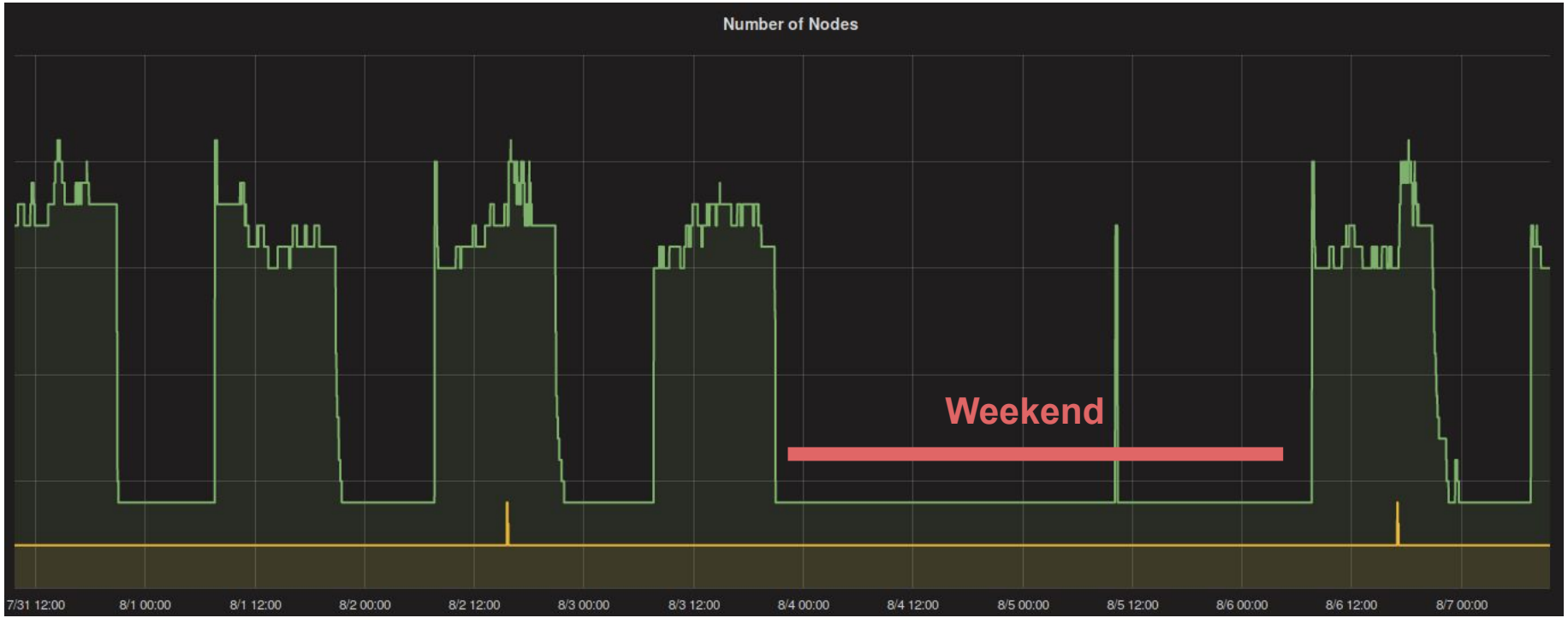
Ingress Req/s

Prometheus Query

ZMON Check

github.com/zalando-incubator/kube-metrics-adapter

DOWNSCALING DURING OFF-HOURS



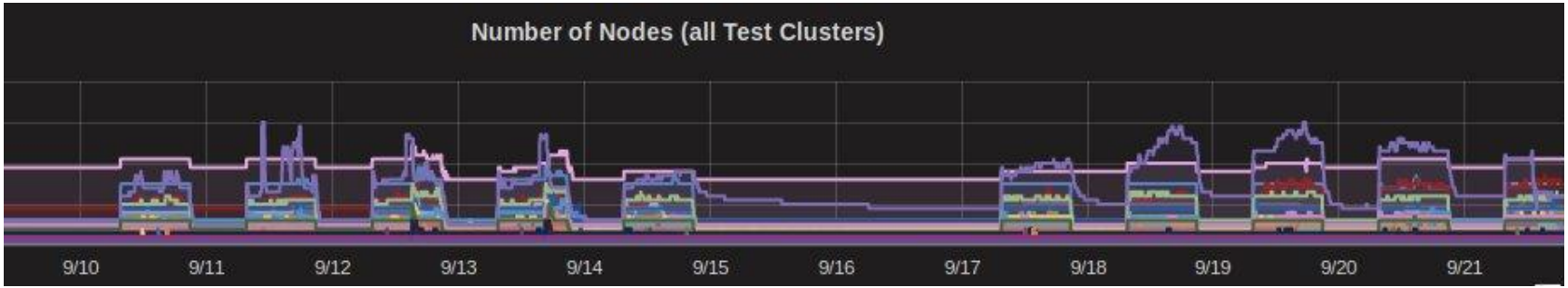
github.com/hjacobs/kube-downscaler

zalando

DOWNSCALING DURING OFF-HOURS

```
DEFAULT_UPTIME="Mon-Fri 07:30-20:30 CET"
```

```
annotations:
    downscaler/exclude: "true"
```



github.com/hjacobs/kube-downscaler

ACCUMULATED WASTE

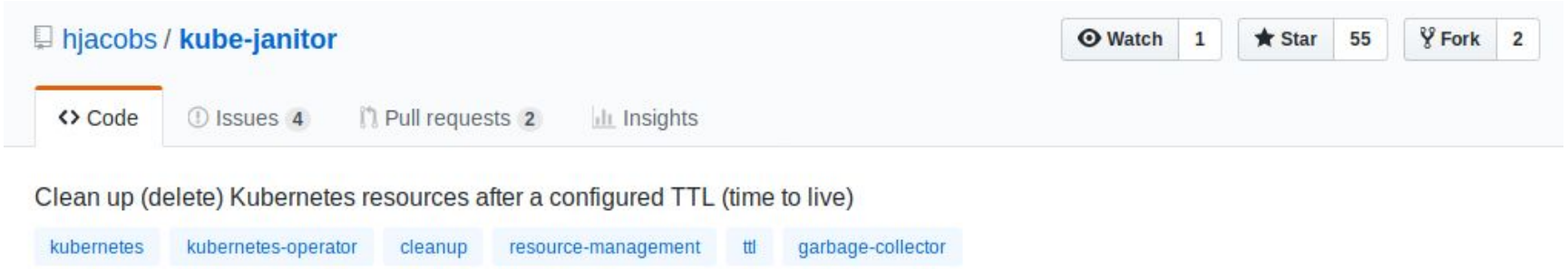
- Prototypes
- Personal test environments
- Trial runs
- Decommissioned services
- Learning/training deployments

Sounds familiar?

HOUSEKEEPING

- Delete prototypes after X days
- Clean up temporary deployments
- Remove resources without owner

KUBERNETES JANITOR



- **TTL** and **expiry date** annotations, e.g.
  - set time-to-live for your test deployment
- **Custom rules**, e.g.
  - delete everything without "app" label after 7 days

github.com/hjacobs/kube-janitor

JANITOR TTL ANNOTATION

```
# let's try out nginx, but only for 1 hour
kubectl run nginx --image=nginx
kubectl annotate deploy nginx janitor/ttl=1h
```


github.com/hjacobs/kube-janitor

CUSTOM JANITOR RULES

```
# require "app" label for new pods starting April 2019
- id: require-app-label-april-2019
  resources:
  - deployments
  - statefulsets
  jmespath: "!(spec.template.metadata.labels.app) &&
             metadata.creationTimestamp > '2019-04-01'"
  ttl: 7d
```

github.com/hjacobs/kube-janitor

EC2 SPOT NODES

Role	Instance Type	S?	Version	CC	MC	CPU	Memory (GiB)	Cost
worker	m4.2xlarge		v1.12.5-custom.master-1	8	31.4 GiB	1.6 6.9 7.8	8.8 14.3 30.9	350.64
worker	m4.4xlarge	%	v1.12.5-custom.master-1	16	62.9 GiB	4.1 9.0 15.8	51.2 62.3 62.4	193.73

72% savings

SPOT ASG / LAUNCH TEMPLATE

```
10    AutoScalingGroup:
11      CreationPolicy:
12        ResourceSignal:
13          Count: '0'
14          Timeout: PT15M
15      Properties:
16        HealthCheckGracePeriod: 300
17        HealthCheckType: EC2
18  {{ if gt (len .NodePool.InstanceTypes) 1 }}
19        MixedInstancesPolicy:
20          InstancesDistribution:
21            OnDemandPercentageAboveBaseCapacity: {{if eq .NodePool.Discour
22            SpotInstancePools: {{ len .NodePool.InstanceTypes }}
23          LaunchTemplate:
24            LaunchTemplateSpecification:
25              LaunchTemplateId: !Ref LaunchTemplate
26              Version: !GetAtt LaunchTemplate.LatestVersionNumber
27            Overrides:
28  {{ range $type := .NodePool.InstanceTypes }}
29              - InstanceType: "{{ $type }}"
```

Not upstream in cluster-autoscaler (yet)

zalando

CLUSTER OVERHEAD: CONTROL PLANE

- GKE cluster: free
- EKS cluster: $146/month
- Zalando prod cluster: **$635/month**
  (etcd nodes + master nodes + ELB)

Potential: fewer etcd nodes, no HA, shared control plane.

WHAT WORKED FOR US

- Disable CPU CFS Quota in all clusters
- Prevent memory overcommit
- Kubernetes Resource Report
- Downscaling during off-hours
- EC2 Spot

STABILITY ↔ EFFICIENCY

Slack

Autoscaling Buffer

Disable Overcommit

Cluster Overhead

Resource Report

HPA

VPA

Downscaler

Janitor

EC2 Spot

OPEN SOURCE

Kubernetes on AWS
github.com/zalando-incubator/kubernetes-on-aws

AWS ALB Ingress controller
github.com/zalando-incubator/kube-ingress-aws-controller

External DNS
github.com/kubernetes-incubator/external-dns

Postgres Operator
github.com/zalando/postgres-operator

Kubernetes Resource Report
github.com/hjacobs/kube-resource-report

Kubernetes Downscaler
github.com/hjacobs/kube-downscaler

Kubernetes Janitor
github.com/hjacobs/kube-janitor

OTHER TALKS/POSTS

- Everything You Ever Wanted to Know About Resource Scheduling
- Inside Kubernetes Resource Management (QoS) - KubeCon 2018
- Setting Resource Requests and Limits in Kubernetes (Best Practices)
- Effectively Managing Kubernetes Resources with Cost Monitoring

zalando

QUESTIONS?

HENNING JACOBS

HEAD OF

DEVELOPER PRODUCTIVITY

henning@zalando.de

@try_except_

Illustrations by @01k