KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

cgroup v2 Is Coming Soon To a Cluster Near You -

David Porter (@bobbypage), Google & Mrunal Patel (@mrunalp), RedHat

cgroup v2 Is Coming Soon To a Cluster Near You

KubeCon | CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

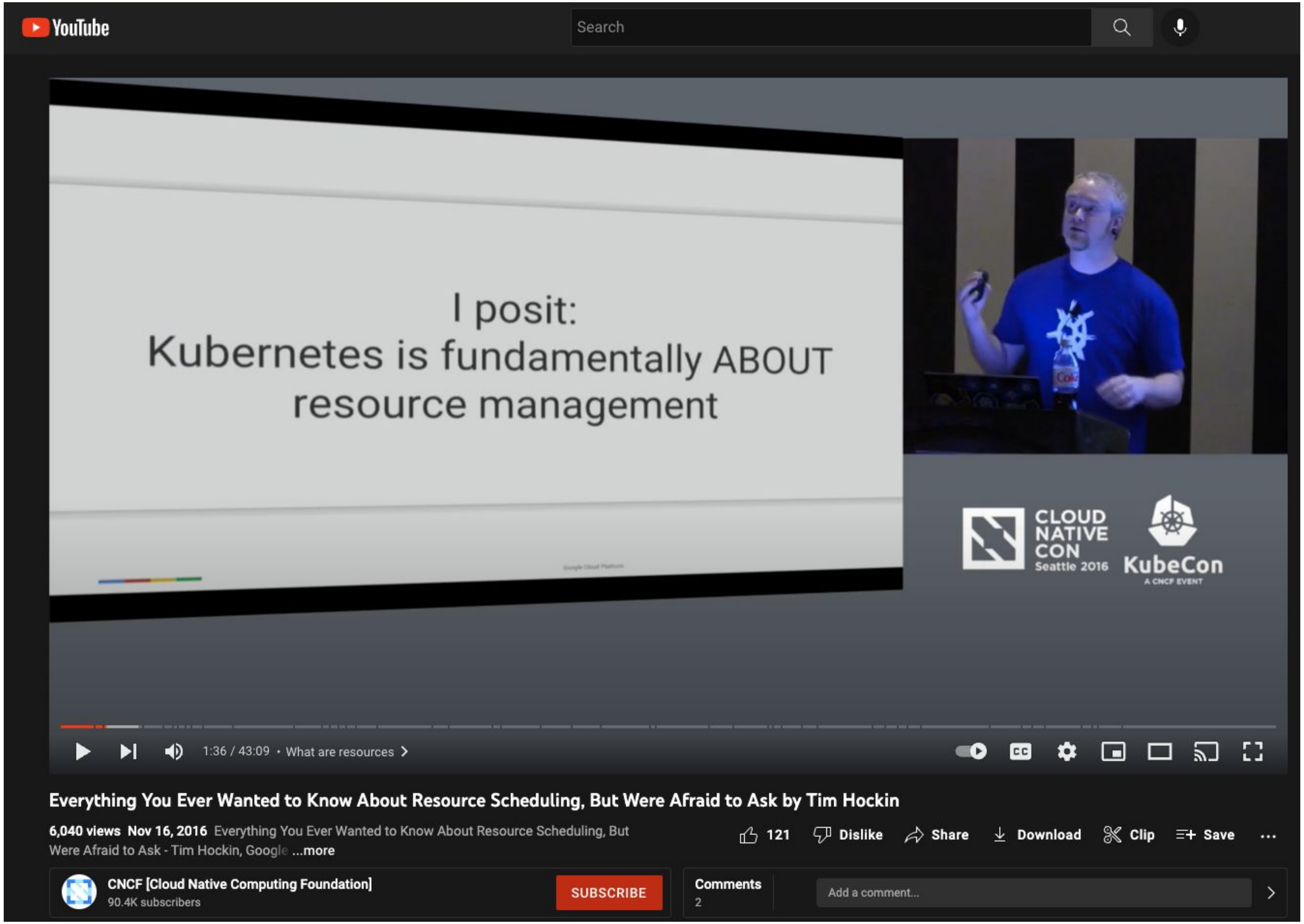
DETROIT 2022

October 24-28, 2021

David Porter
@bobbypage
Software Engineer
Google

Mrunal Patel
@mrunalp
Software Engineer
Red Hat

Resource Management



I posit:
Kubernetes is fundamentally ABOUT resource management

Everything You Ever Wanted to Know About Resource Scheduling, But Were Afraid to Ask by Tim Hockin

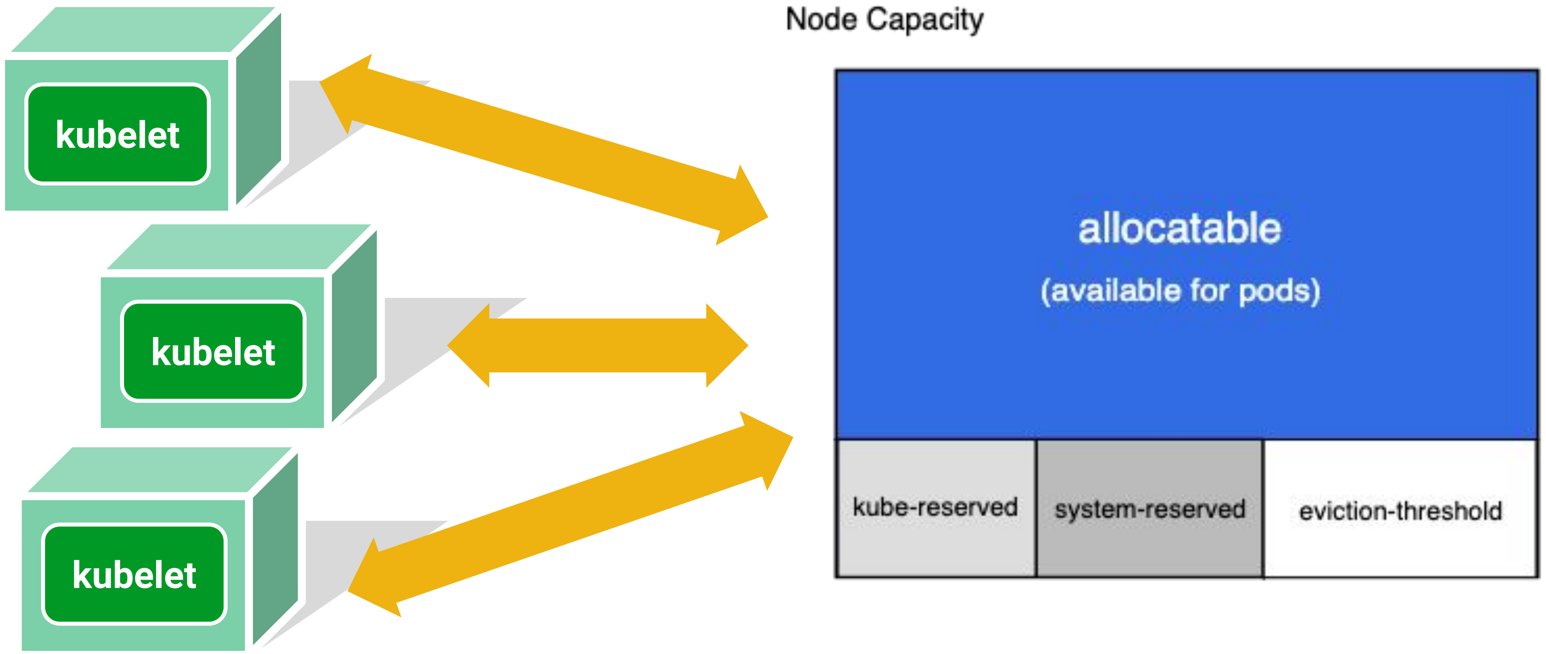
6,040 views Nov 16, 2016 Everything You Ever Wanted to Know About Resource Scheduling, But Were Afraid to Ask - Tim Hockin, Google ...more

CNCF [Cloud Native Computing Foundation]
90.4K subscribers

Comments
2

The 10k view - Resources

- Clusters consistent of nodes
  - Nodes contain resources such as
    - CPUs, Memory, Disk, GPUs, etc…
- Nodes advertise resource availability to the kubernetes scheduler
  - Node Capacity
  - Node Allocatable = Capacity - Reserved



The 10k view - Pods use resources

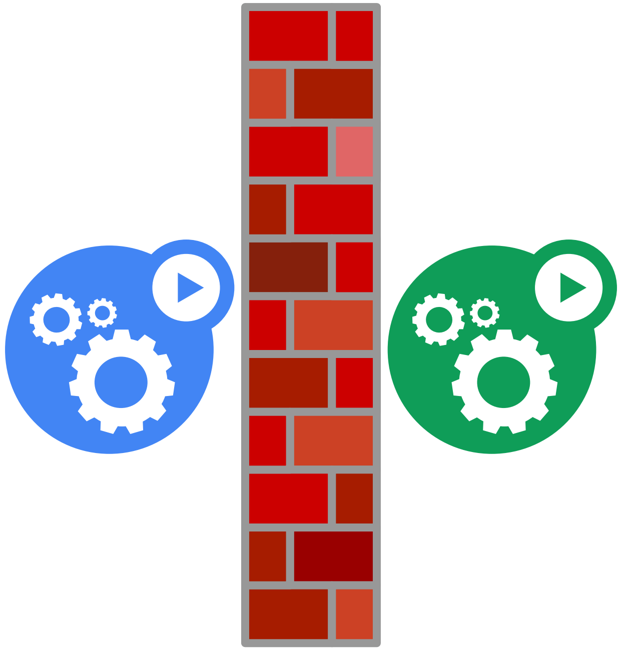
```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: app
    image: myapp
    resources:
      requests:
        cpu: "250m"
        memory: "64Mi"
      limits:
        cpu: "500m"
        memory: "128Mi
```

requests: → The minimum amount of resources this container needs

limits: → The maximum amount of resources this container can use

Resource Management Requirements

- Resource Isolation
  - Pods should not be able to hurt each other (or the system)
  - Pods should be able to receive consistent performance behavior based on their requests
  - Prevent:
    - Infinite Loops, Fork Bombs, Memory Leaks, Node lockups
- Sizing
  - Allocate proper resources for pods
- Utilization
  - Ensure resources are managed efficiently



Quality of Service - QoS

Defined in terms of Request and Limit

Guaranteed: highest protection

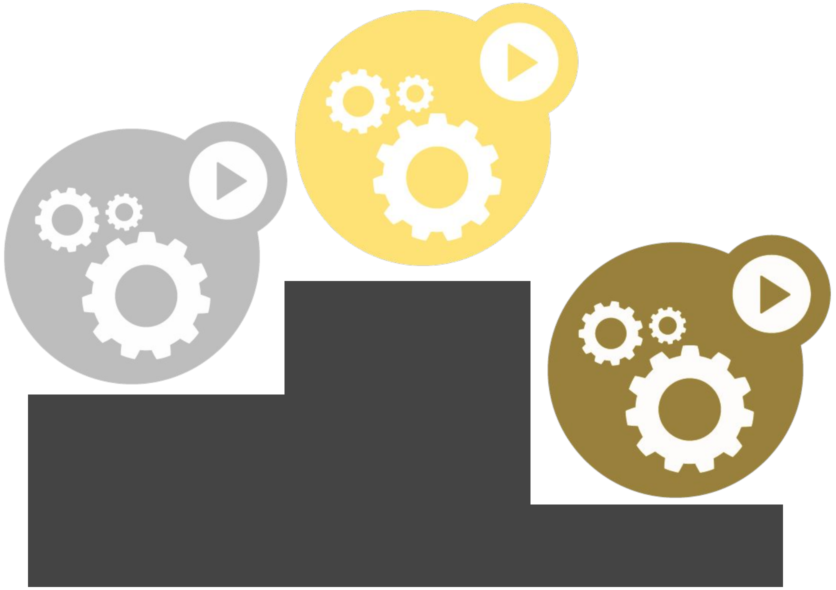
- request > 0 && limit == request [for all containers]
- Lowest OOM score

Burstable: medium protection

- request > 0 && limit > request [for at least one container]
- OOM score function of memory request

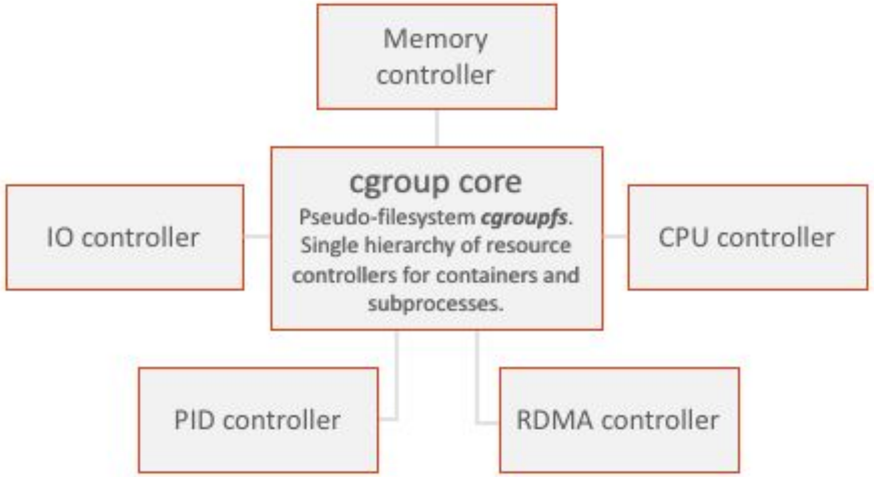
Best Effort: lowest protection

- request == 0 (unspecified)
- lowest CPU shares
- First to get OOM killed during low memory



cgroups

- Linux Kernel feature that provides ability to:
  - Group a set of process hierarchically
  - Set of **controllers** (cpu, memory, io, etc...) to manage resources in groups and provide monitoring
- Controlled via pseudo-filesystem cgroupfs
- Allow us to:
  - Limit usage of group of process (amount of CPU or memory or pids).
  - Measure resource usage for a group of processes



Memory controller

IO controller

cgroup core

Pseudo-filesystem *cgroupfs*. Single hierarchy of resource controllers for containers and subprocesses.

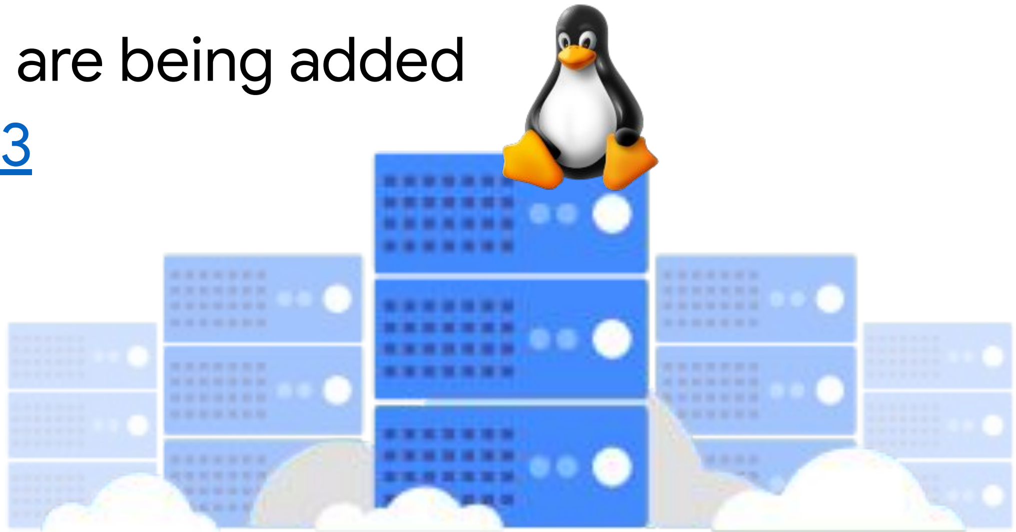
CPU controller

PID controller

RDMA controller

cgroups v1 & v2

- cgroup v1 - Introduced by Google in linux kernel in 2006, various controllers added one after the other.
- cgroup v2 - latest version of the Linux cgroup API
  - In development in the Linux Kernel since 2016:
  - Has matured across the container ecosystem
    - 2019 - Fedora moves to v2 by default
    - 2020 - Docker / runc cgroupv2 support
    - 2021 - Other distros enable cgroup v2 by default
- cgroups v1 is considered legacy - no new features are being added
  - Planned to be removed from systemd EoY 2023



cgroups v2 support

- Most new Linux distros today have adopted cgroup v2 by default:
  - Container Optimized OS (since M97)
  - Ubuntu (since 21.10)
  - Debian GNU/Linux (since Debian 11 Bullseye)
  - Fedora (since 31)
  - Arch Linux (since April 2021)
  - RHEL and RHEL-like distributions (since 9)
- Runtimes
  - containerd 1.4 and later
  - cri-o v1.20 and later
  - docker/moby > 20.10
  - runc > 1.0.0
  - crun > 0.7
- Kubernetes
  - alpha: v1.18
  - beta: v1.22
  - stable: v1.25

What’s new in cgroups v2?

- Single unified hierarchy design in API
- Enhanced resource allocation and isolation across multiple resources
  - Accounting improvements for:
    - non-immediate resource changes such as page cache writebacks
    - different types of memory allocations (user, network and kernel memory)
- Hard & Soft Memory Limits
- OOM killer is cgroup aware (`memory.oom.group`)
- PSI (Pressure Stall Information) metrics
  - Detect resource pressure for CPU, Memory, IO
- Improved rootless via delegation support

cgroups v1 vs v2 hierarchy

```
/sys/fs/cgroup/
├── cpu/
│   └── kubepods/
│       └── burstable/
│           └── pod1/
│               ├── container_main/
│               │   ├── cpu.shares
│               │   └── cpu.cfs_quota_us
│               └── sidecar/
│                   ├── cpu.shares
│                   └── cpu.cfs_quota_us
└── memory/
    └── kubepods/
        └── burstable/
            └── pod1/
                ├── container_main/
                │   └── memory.max_limit_in_bytes
                └── sidecar/
                    └── memory.max_limit_in_bytes
```

cgroup v1

```
/sys/fs/cgroup/
└── kubepods.slice/
    └── kubepods-burstable.slice/
        └── kubepods-burstable-pod1.slice/
            ├── cri-containerd-container_main.scope/
            │   ├── cpu.weight
            │   ├── cpu.max
            │   └── memory.max
            └── cri-containerd-container_sidecar.scope/
                ├── cpu.weight
                ├── cpu.max
                └── memory.max
```

cgroup v2

Mapping pod/container to cgroup

- Kubelet creates a cgroup for each pod
- Container runtime creates a cgroup for each container
- Depending on QoS class, pod cgroup may have cgroup resources enforced

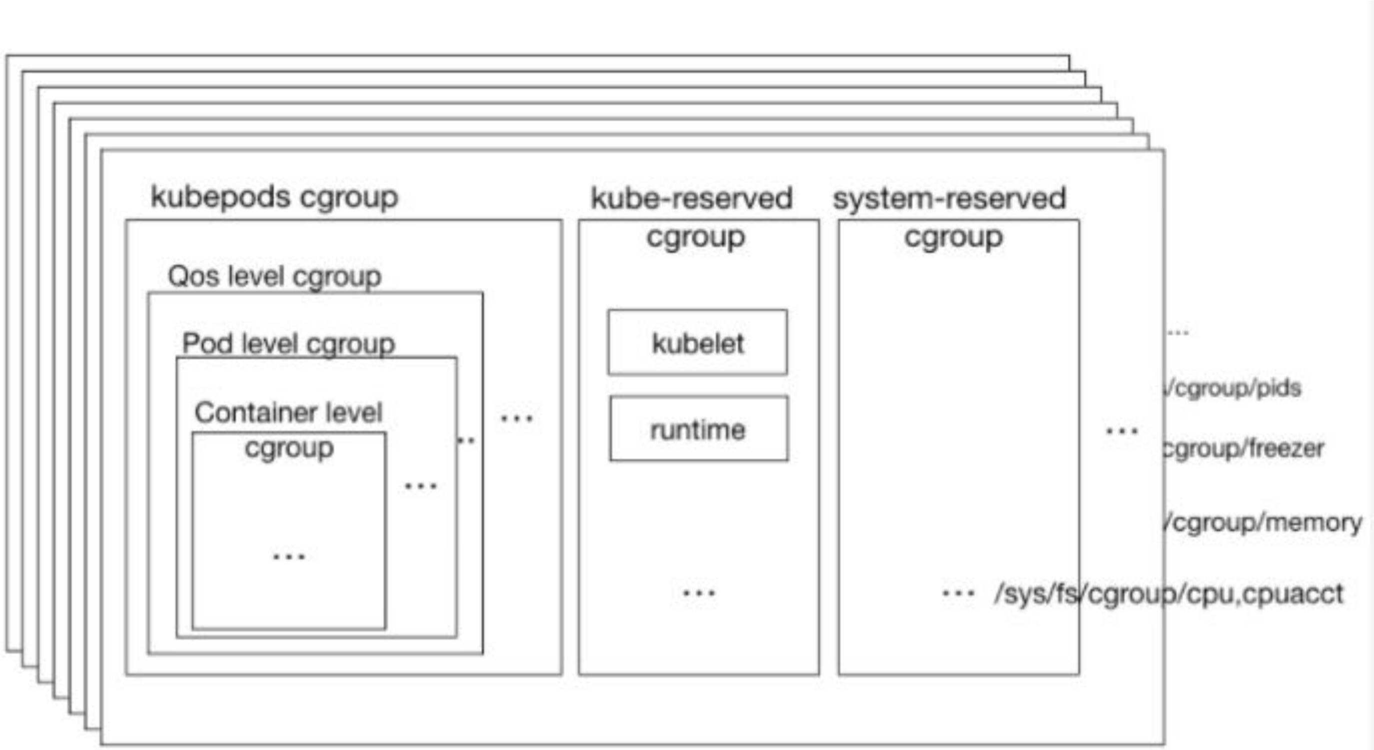
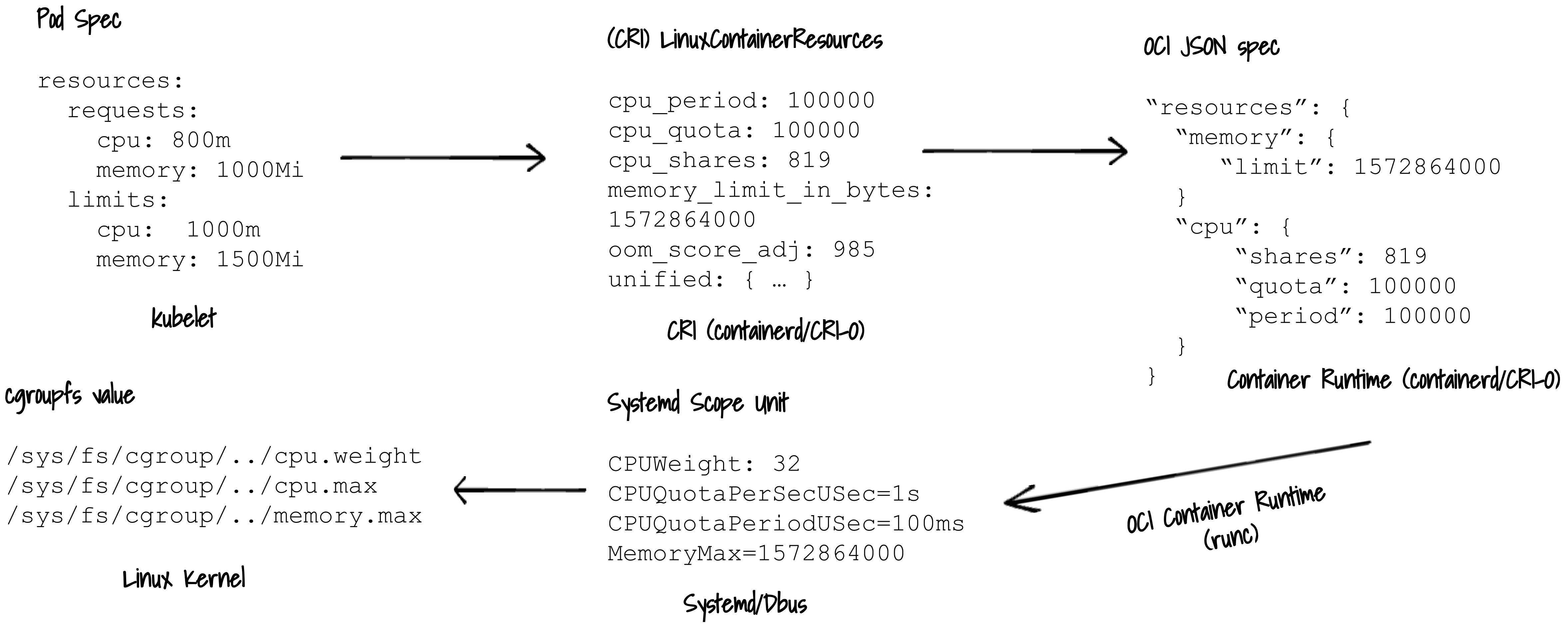


image: https://medium.com/geekculture/layer-by-layer-cgroup-in-kubernetes-c4e26bda676c

Journey of Pod Spec

Pod -> CRI (containerd/cri-o) -> OCI spec -> OCI runtime (runc) -> systemd (driver) -> cgroupfs kernel



Pod Spec

```
resources:
  requests:
    cpu: 800m
    memory: 1000Mi
  limits:
    cpu:  1000m
    memory: 1500Mi
```

kubelet

(CRI) LinuxContainerResources

```
cpu_period: 100000
cpu_quota: 100000
cpu_shares: 819
memory_limit_in_bytes:
1572864000
oom_score_adj: 985
unified: { … }
```

CRI (containerd/CRI-O)

OCI JSON spec

```
"resources": {
  "memory": {
     "limit": 1572864000
  }
  "cpu": {
      "shares": 819
      "quota": 100000
      "period": 100000
  }
}
```

Container Runtime (containerd/CRI-O)

OCI Container Runtime (runc)

Systemd Scope Unit

```
CPUWeight: 32
CPUQuotaPerSecUSec=1s
CPUQuotaPeriodUSec=100ms
MemoryMax=1572864000
```

Systemd/Dbus

cgroupfs value

```
/sys/fs/cgroup/../cpu.weight
/sys/fs/cgroup/../cpu.max
/sys/fs/cgroup/../memory.max
```

Linux Kernel

CPU Requests - CPU Shares

- CPU Request
  - Used for scheduling
  - Minimum Floor for CPU - you will *always* get CPU request
  - Converted to CPU shares (cpu.shares / cpu.weight)
  - Proportional to other containers

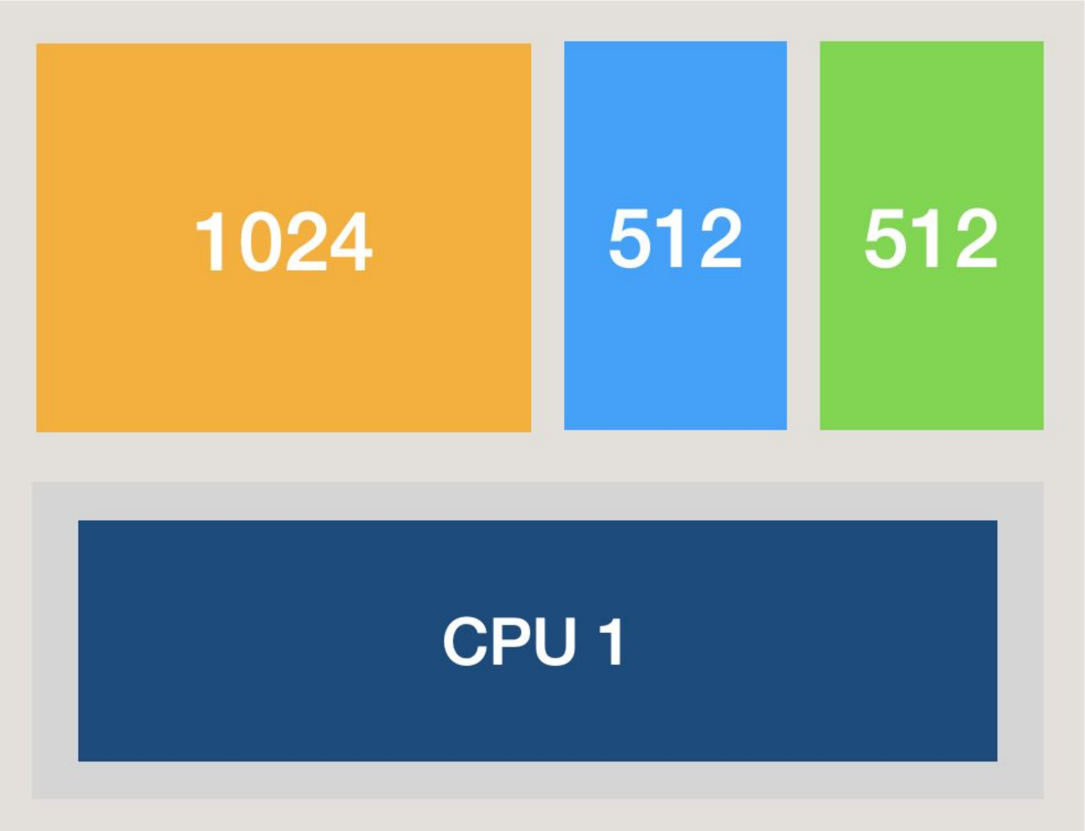
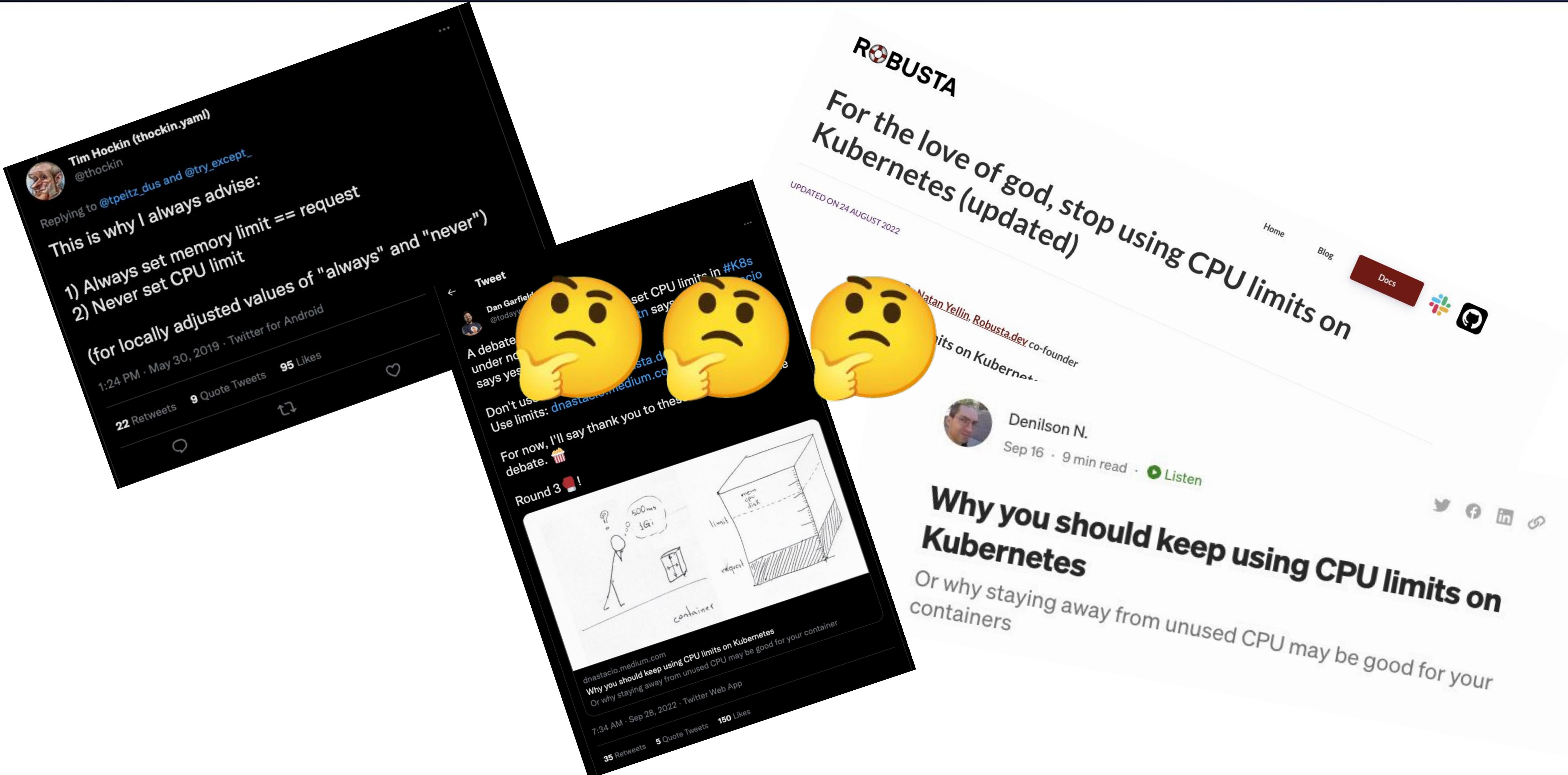


image: https://www.batey.info/cgroup-cpu-shares-for-kubernetes.html

CPU Limits - Cfs Bandwidth Control

- CPU Limit
  - Ignored in scheduling
  - Ceiling for CPU - you will *be* throttled going above limit
    - *even if there are spare CPU cycles available*
  - Uses cpu quota / cpu period (v1) and cpu.max (v2)
- Quota = Limit from Pod spec
- CPU period = 100ms
- “You can use quota amount of CPU in each wall clock period”

The CPU Limits Debate



Tim Hockin (thockin.yaml)
@thockin

Replying to @tpeitz_dus and @try_except_

This is why I always advise:

1) Always set memory limit == request
2) Never set CPU limit

(for locally adjusted values of "always" and "never")

1:24 PM · May 30, 2019 · Twitter for Android

22 Retweets 9 Quote Tweets 95 Likes

Tweet

Dan Garfield
@today

A debate … set CPU limits in #K8s
under no … cio
says yes … tn says … sta.d … e

Don't us… medium.co
Use limits: dnastacio.medium.co

For now, I'll say thank you to thes… debate. 🍿

Round 3 🥊!

dnastacio.medium.com
Why you should keep using CPU limits on Kubernetes
Or why staying away from unused CPU may be good for your container

7:34 AM · Sep 28, 2022 · Twitter Web App

35 Retweets 5 Quote Tweets 150 Likes

ROBUSTA

Home Blog Docs

For the love of god, stop using CPU limits on Kubernetes (updated)

UPDATED ON 24 AUGUST 2022

Natan Yellin, Robusta.dev co-founder

…its on Kubernet…

Denilson N.
Sep 16 · 9 min read · Listen

Why you should keep using CPU limits on Kubernetes

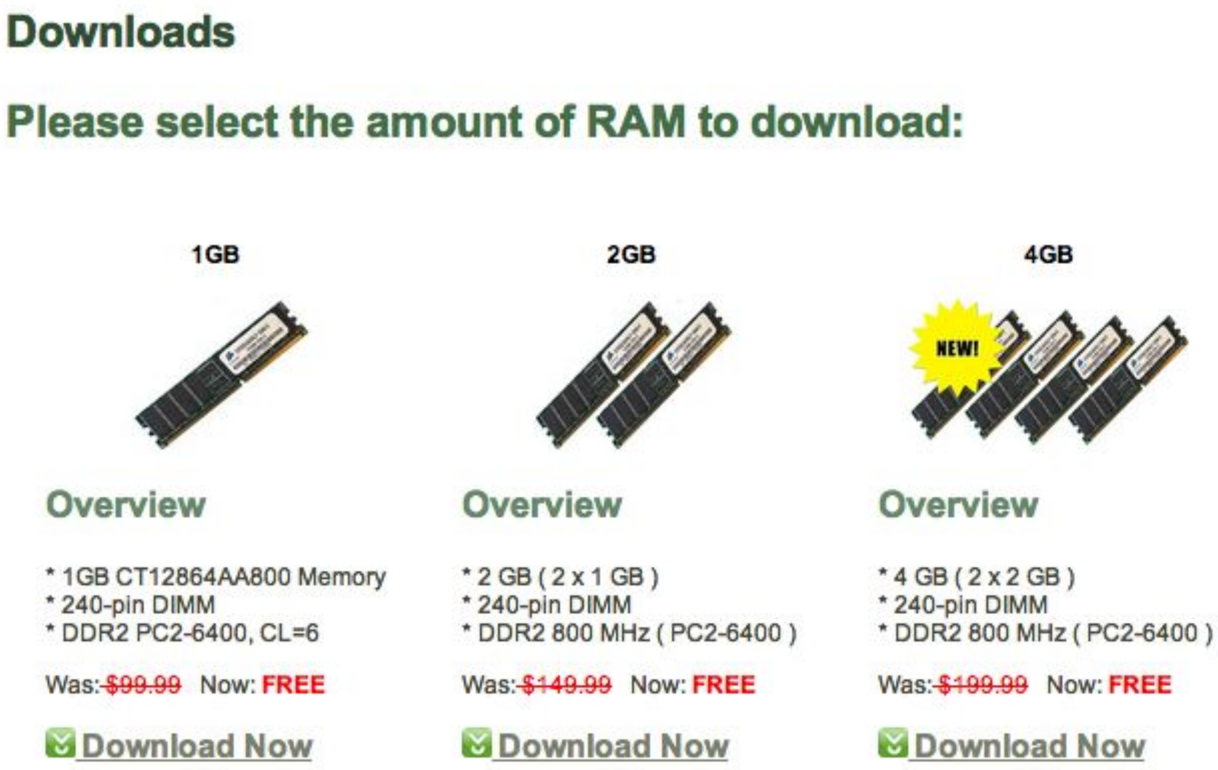
Or why staying away from unused CPU may be good for your containers

CPU Debate

- Always set a CPU request - used for scheduling
  - otherwhile it will be best effort (lowest priority QoS)
  - You will always get CPU request (enforced by CPU shares)
- Cons of CPU Limits
  - 👎 Can’t use spare CPU cycles
  - 👎 “Throwing away unused CPU cycles”
  - 👎 You may introduce “artificial” throttling to your application
- Pros of CPU Limits
  - 👍 Avoid reliance on unpredictable spare CPU cycles due to low CPU requests
  - 👍 More predictable - Guaranteed QoS
  - 👍 Multi-tenant environment (e.g. chargeback)

Memory

- Memory Request is only used for scheduling (v1)
- Memory Limit = `memory.max` (v2) / `memory.max_limit_in_bytes` (v1)
    - Going over the limit results in OOM
- Recommendation - Set `Memory Requests = Limit`



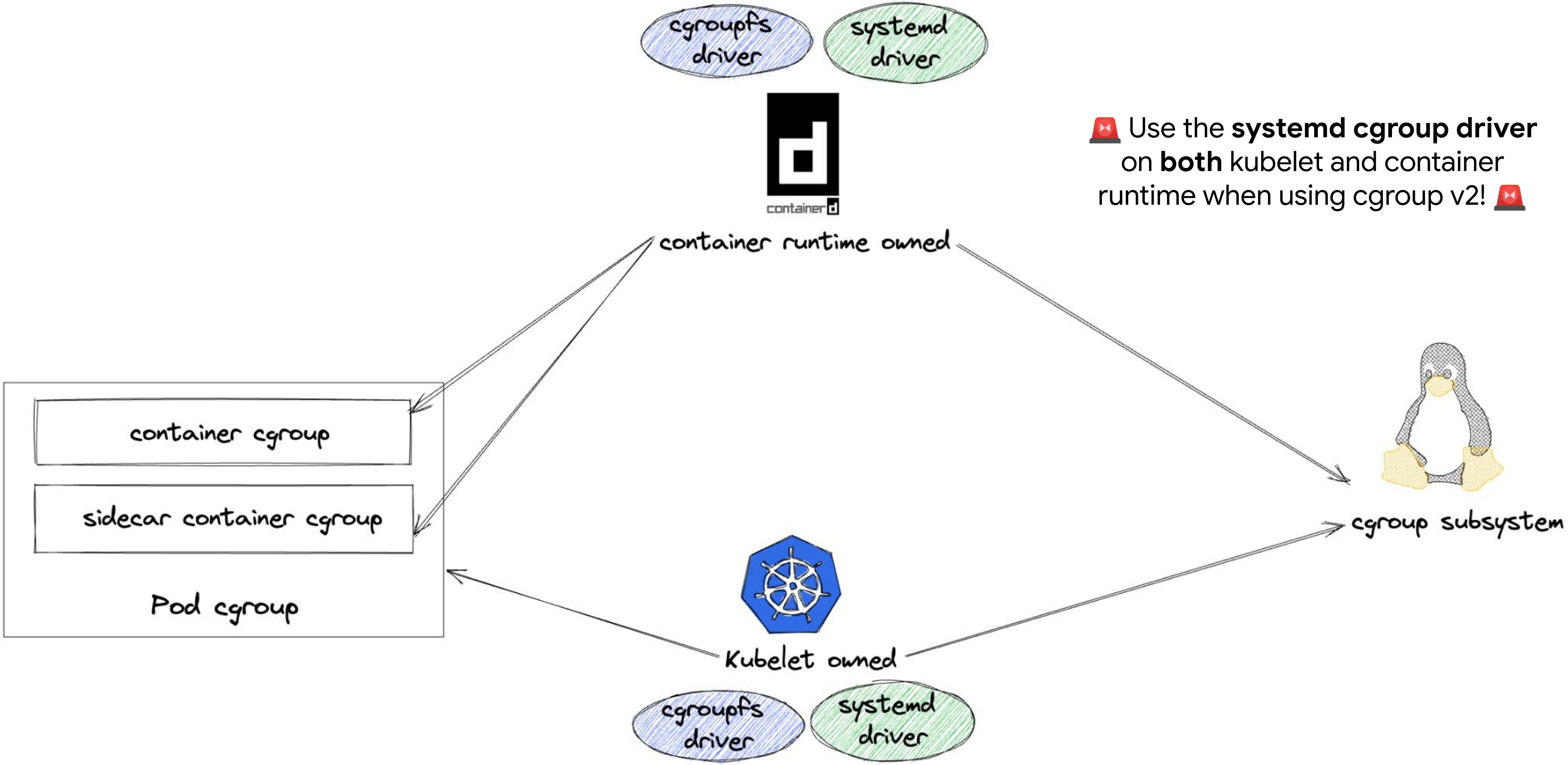
Downloads

Please select the amount of RAM to download:

1GB	2GB	4GB
Overview	Overview	Overview
* 1GB CT12864AA800 Memory * 240-pin DIMM * DDR2 PC2-6400, CL=6	* 2 GB (2 x 1 GB) * 240-pin DIMM * DDR2 800 MHz (PC2-6400)	* 4 GB (2 x 2 GB) * 240-pin DIMM * DDR2 800 MHz (PC2-6400)
Was: ~~$99.99~~ Now: FREE	Was: ~~$149.99~~ Now: FREE	Was: ~~$199.99~~ Now: FREE
Download Now	Download Now	Download Now

image: https://knowyourmeme.com/photos/367171

Kubelet and Container Runtime Cgroup Drivers



🚨 Use the **systemd cgroup driver** on **both** kubelet and container runtime when using cgroup v2! 🚨

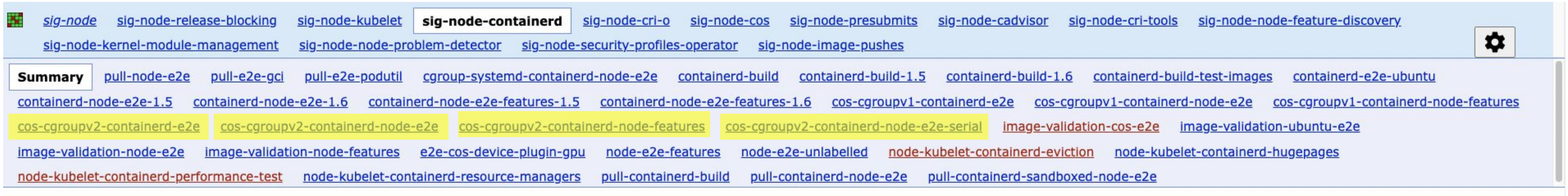
Monitoring

- cgroups provide monitoring of container / pod resource usage
- cAdvisor and container runtime monitors cgroups and reports the information to kubelet
    - kubelet depends on cAdvisor as a library
    - cAdvisor updated to support cgroup v2 in v0.43
- KEP 2371 - CRI Pod & Container Stats in progress to move all metrics to runtime and remove cAdvisor dependency

image: http://github.com/google/cadvisor

cgroups v2 testing

- Large testing effort in SIG-Node to get cgroupv2 jobs CI jobs
  - Conformance, Serial Node E2E, Cluster E2E, Node E2E Features
- Gathering feedback
- Support in the broader community



sig-node sig-node-release-blocking sig-node-kubelet **sig-node-containerd** sig-node-cri-o sig-node-cos sig-node-presubmits sig-node-cadvisor sig-node-cri-tools sig-node-node-feature-discovery
sig-node-kernel-module-management sig-node-node-problem-detector sig-node-security-profiles-operator sig-node-image-pushes

Summary pull-node-e2e pull-e2e-gci pull-e2e-podutil cgroup-systemd-containerd-node-e2e containerd-build containerd-build-1.5 containerd-build-1.6 containerd-build-test-images containerd-e2e-ubuntu
containerd-node-e2e-1.5 containerd-node-e2e-1.6 containerd-node-e2e-features-1.5 containerd-node-e2e-features-1.6 cos-cgroupv1-containerd-e2e cos-cgroupv1-containerd-node-e2e cos-cgroupv1-containerd-node-features
cos-cgroupv2-containerd-e2e cos-cgroupv2-containerd-node-e2e cos-cgroupv2-containerd-node-features cos-cgroupv2-containerd-node-e2e-serial image-validation-cos-e2e image-validation-ubuntu-e2e
image-validation-node-e2e image-validation-node-features e2e-cos-device-plugin-gpu node-e2e-features node-e2e-unlabelled node-kubelet-containerd-eviction node-kubelet-containerd-hugepages
node-kubelet-containerd-performance-test node-kubelet-containerd-resource-managers pull-containerd-build pull-containerd-node-e2e pull-containerd-sandboxed-node-e2e

Migrating to cgroup v2 with k8s

- Use one of the many Linux distros that enables cgroup v2 out of the box
  - Kernel 5.8+
- Use an up-to-date of CRI runtime (containerd/cri-o)
- Use the systemd cgroup driver on **both** kubelet and container runtime
  - ⚠️ SIG-Node does not support nor test the cgroupfs driver. ⚠️
  - 🚨 Do not use cgroupfs driver on cgroup v2 🚨
- For hosted kubernetes offerings, understand how your vendor is adopting cgroup v2

Test your apps on cgroup v2

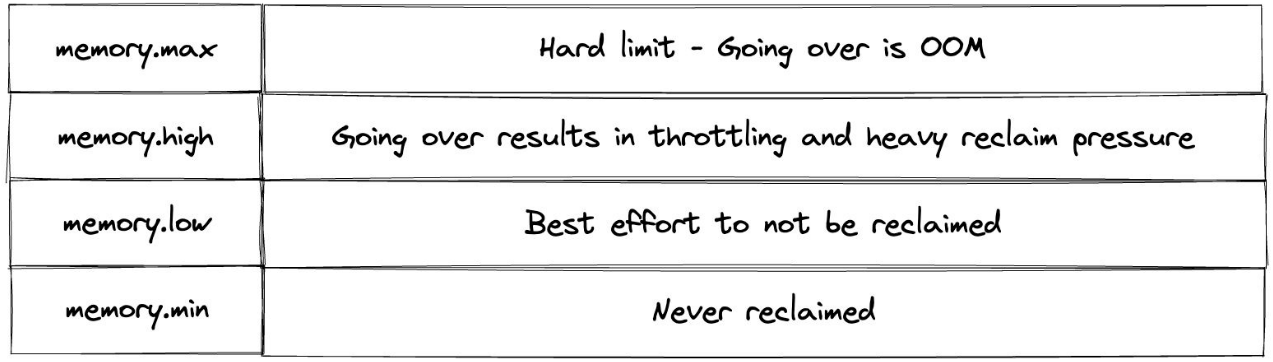
- Most applications do not have cgroup dependencies, but some applications may
- Third party monitoring and security agents
    - Contact vendor and ensure agents support cgroup v2
    - cAdvisor standalone (upgrade to v0.43.0+)
    - github.com/uber-go/automaxprocs v1.5.0+
    - Java apps uses JDK to read cgroup settings for auto-tuning, upgrade to JDK 11.0.16 and later or JDK 15+ which fully support cgroup v2

Future work in this space

We have many opportunities to improve resource management and build upon cgroup v2!

Memory QoS

- *KEP-2570* - Quality-of-Service for Memory Resources (alpha in 1.22)
    - cgroupv1 - only provided max memory limit, (i.e. pod memory requests ignored; only used for scheduling)
    - cgroupv2 provides `memory.{min,low,high,max}`
    - Idea
        - Map container `spec.containers[].resources.requests[memory]` to `memory.low`
        - Map `spec.containers[].resources.limits[memory] * throttling factor` to `memory.high`
    - Results - Less frequent OOMs, ensure memory is throttled as memory approaches the limit



memory.max	Hard limit - Going over is OOM
memory.high	Going over results in throttling and heavy reclaim pressure
memory.low	Best effort to not be reclaimed
memory.min	Never reclaimed

Future work

- PSI Pressure Metrics
  - Integrate Kubelet with PSI pressure metrics
  - Improve out of pressure eviction to evict pods if pressure
  - Will help to preserve node stability
- Disk throttling
  - cgroupv2 has a new IO controller
  - Can be used to limit I/O of pods (currently not possible)
- OOMd
  - systemd offers new userspace OOM killer based on PSI metrics
  - Investigate integrating with kubernetes to provide more predictable OOM killing behavior taking into account Pod QoS, Usage, etc

image: https://www.usenix.org/conference/lisa21/presentation/down

Demo

Thank you

- SIG Node Community
- Container runtime community
  - Shout out to Giuseppe Scrivano (@gscrivano)
  - Containerd, CRI-O, Moby/Docker maintainers
- Systemd Maintainers
- Linux Kernel Maintainers

More resources

- cgroupv2 GA k8s blog - https://kubernetes.io/blog/2022/08/31/cgroupv2-ga-1-25/
- cgroupv2 k8s docs - https://kubernetes.io/docs/concepts/architecture/cgroups/
- cgroupv2 kernel docs - https://docs.kernel.org/admin-guide/cgroup-v2.html
- Kubecon talks
  - 2020 - Kubernetes on cgroup v2
  - https://kccnceu20.sched.com/event/ZeoS/kubernetes-on-cgroup-v2-giuseppe-scrivano-red-hat
  - 2022 - cgroups v2 before you jump in https://kccncna2022.sched.com/event/182J2/cgroups-v2-before-you-jump-in-tony-gosselin-mike-tougeron-adobe-systems

Please scan the QR Code above to
leave feedback on this session