KubeCon | CloudNativeCon

North America 2023

Advancing Memory Management in Kubernetes: Next Steps with Memory QoS

Antti Kervinen (Intel) & Dixita Narang (Google)

Advancing Memory Management in Kubernetes: Next Steps with Memory QoS

KubeCon | CloudNativeCon

North America 2023

Antti Kervinen
Cloud Software Engineer at Intel
@askervin

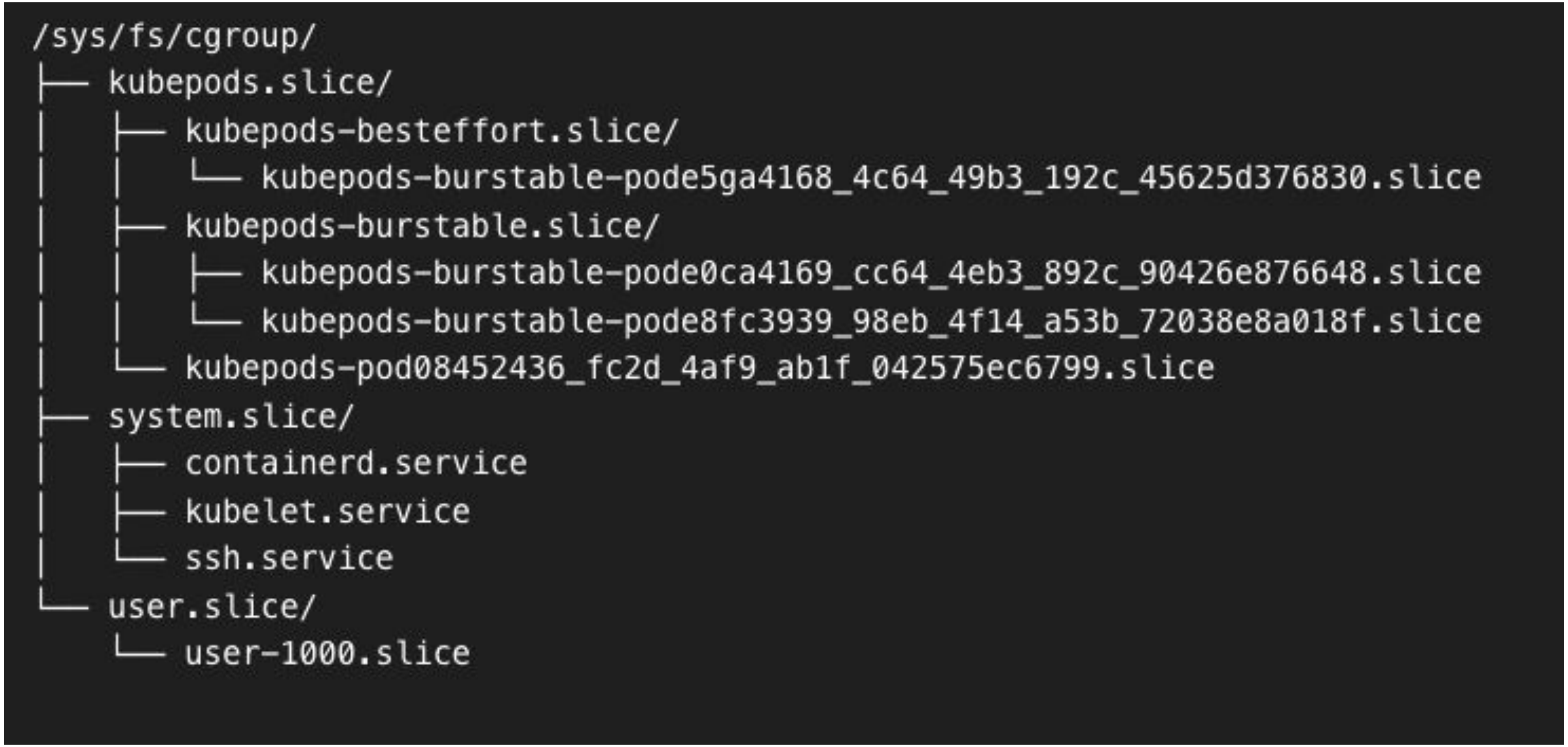
Dixita Narang
Software Engineer at Google
@ndixita

Memory QoS

✗ Memory QoS Beta efforts **Stalled** ✗

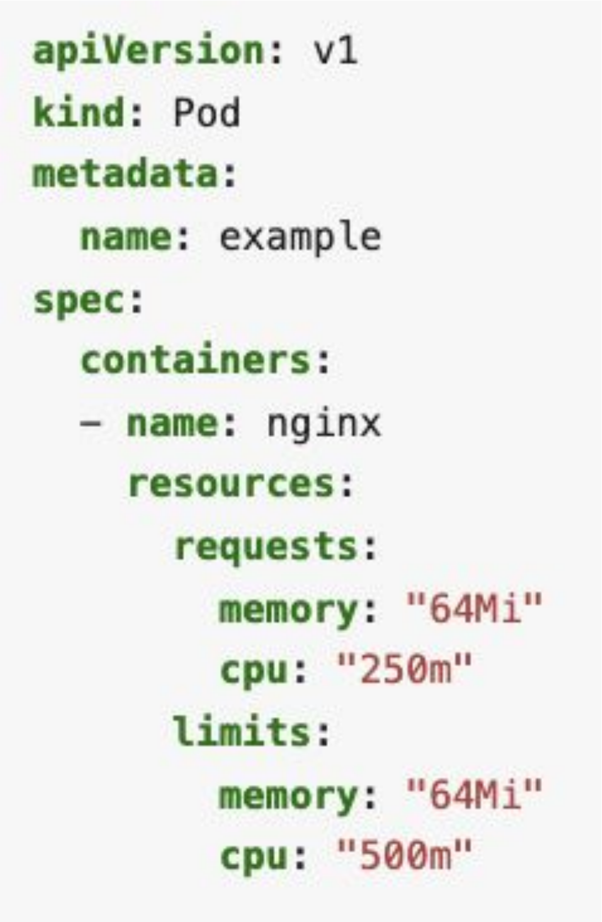
- KEP 2570
  - Initial Plan: Throttle memory using cgroup v2 “memory.high” controller.
  - Comprehension of “memory.high” didn’t align with the recommended guidelines.

K8s node cgroups v2



```
/sys/fs/cgroup/
├── kubepods.slice/
│   ├── kubepods-besteffort.slice/
│   │   └── kubepods-burstable-pode5ga4168_4c64_49b3_192c_45625d376830.slice
│   ├── kubepods-burstable.slice/
│   │   ├── kubepods-burstable-pode0ca4169_cc64_4eb3_892c_90426e876648.slice
│   │   └── kubepods-burstable-pode8fc3939_98eb_4f14_a53b_72038e8a018f.slice
│   └── kubepods-pod08452436_fc2d_4af9_ab1f_042575ec6799.slice
├── system.slice/
│   ├── containerd.service
│   ├── kubelet.service
│   └── ssh.service
└── user.slice/
    └── user-1000.slice
```

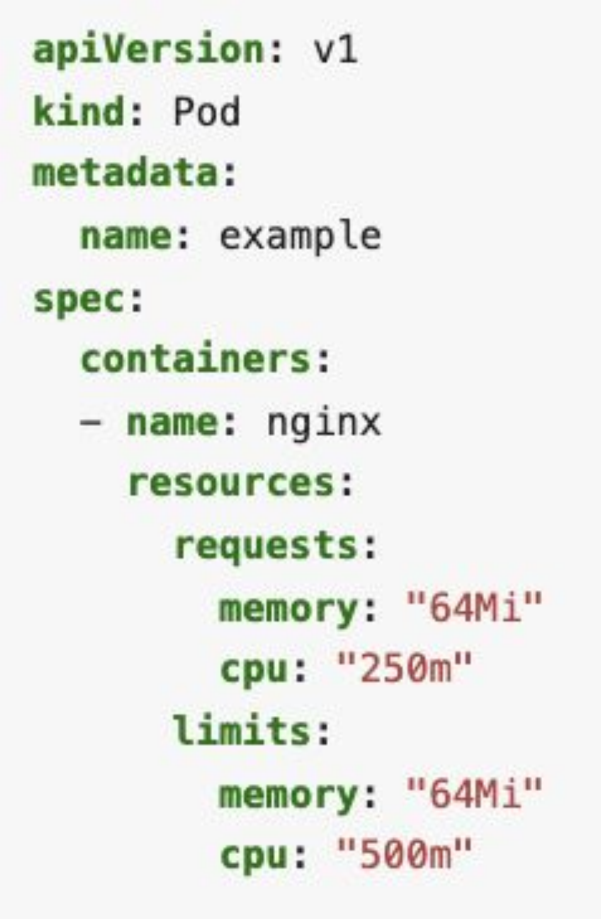
CPU Request & Limit



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
  containers:
  - name: nginx
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "64Mi"
        cpu: "500m"
```

- CPU Request
  - Kubernetes Scheduler uses the CPU requests from pod spec for scheduling.
  - Container runtime maps the requested CPU to “cpu.weight” cgroup parameter.
- CPU Limit
  - Kubernetes Scheduler ignores the CPU limits.
  - Container runtime maps the limits to “cpu.max”.

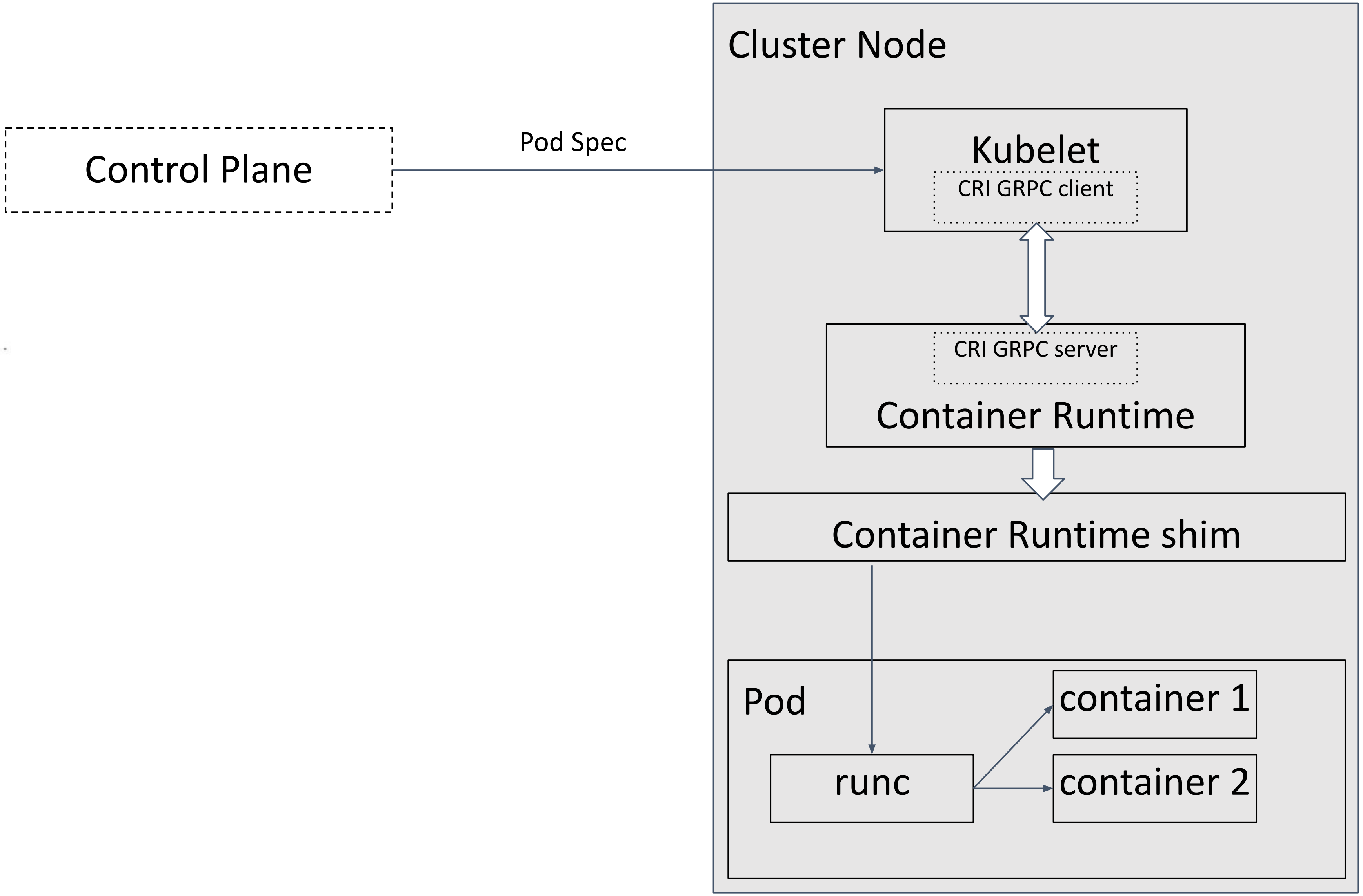
Memory Request & Limit



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
  containers:
  - name: nginx
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "64Mi"
        cpu: "500m"
```

- Memory Request
  - Kubernetes Scheduler uses the Memory requests from pod spec for scheduling.
  - Container Runtime ignores the requested memory value.
- Memory Limit
  - Kubernetes Scheduler ignores the Memory limits.
  - Container runtime maps the limits to “memory.max”.

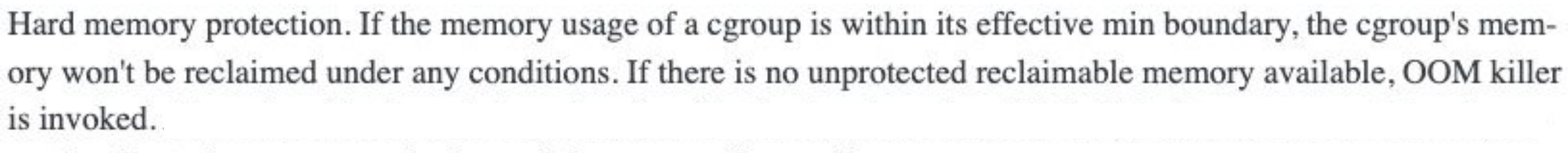
Pod Spec to Containers



cgroup v2 memory knobs in Memory QoS

- memory.max
  - memory.limit_in_bytes in cgroup v1

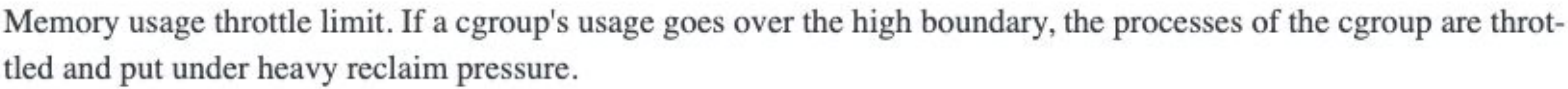
- memory.min
  - Memory QOS plan: map memory request to memory.min
  - From [kernel docs]()



Hard memory protection. If the memory usage of a cgroup is within its effective min boundary, the cgroup's memory won't be reclaimed under any conditions. If there is no unprotected reclaimable memory available, OOM killer is invoked.

cgroup v2 memory knobs in Memory QoS

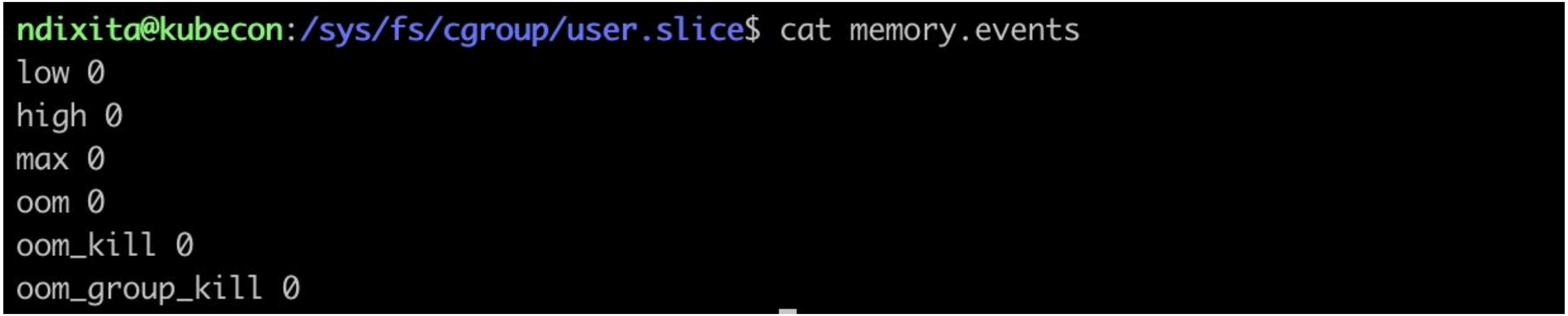
- memory.high
  - Memory QOS plan: set memory.high to throttle memory when usage nears limits.
  - From kernel docs



Memory usage throttle limit. If a cgroup's usage goes over the high boundary, the processes of the cgroup are throttled and put under heavy reclaim pressure.

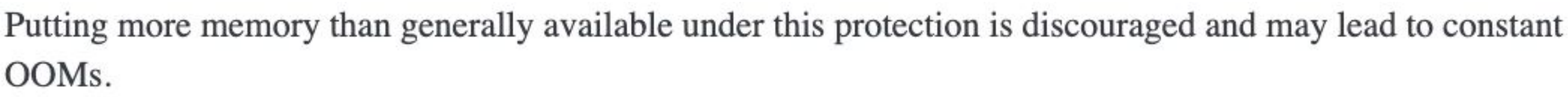
cgroup v2 memory knobs in Memory QoS

- memory.events



```
ndixita@kubecon:/sys/fs/cgroup/user.slice$ cat memory.events
low 0
high 0
max 0
oom 0
oom_kill 0
oom_group_kill 0
```

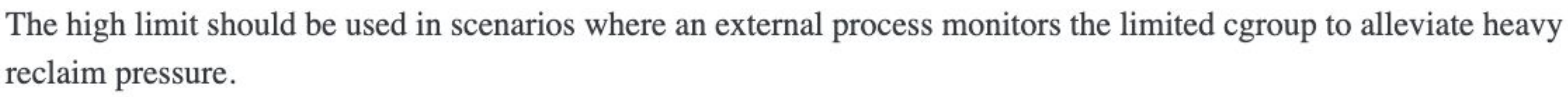
Possible side-effects of setting memory.min



Putting more memory than generally available under this protection is discouraged and may lead to constant OOMs.

- Setting memory.min can lead to more OOM kills, as memory becomes unreclaimable in each cgroup.
- When system is under memory pressure and reclaim cannot free memory, OOM killer is invoked.
- memory.min would work well for cases when a minimum amount of memory is required by processes in a cgroup to make

Possible side-effects of setting memory.high



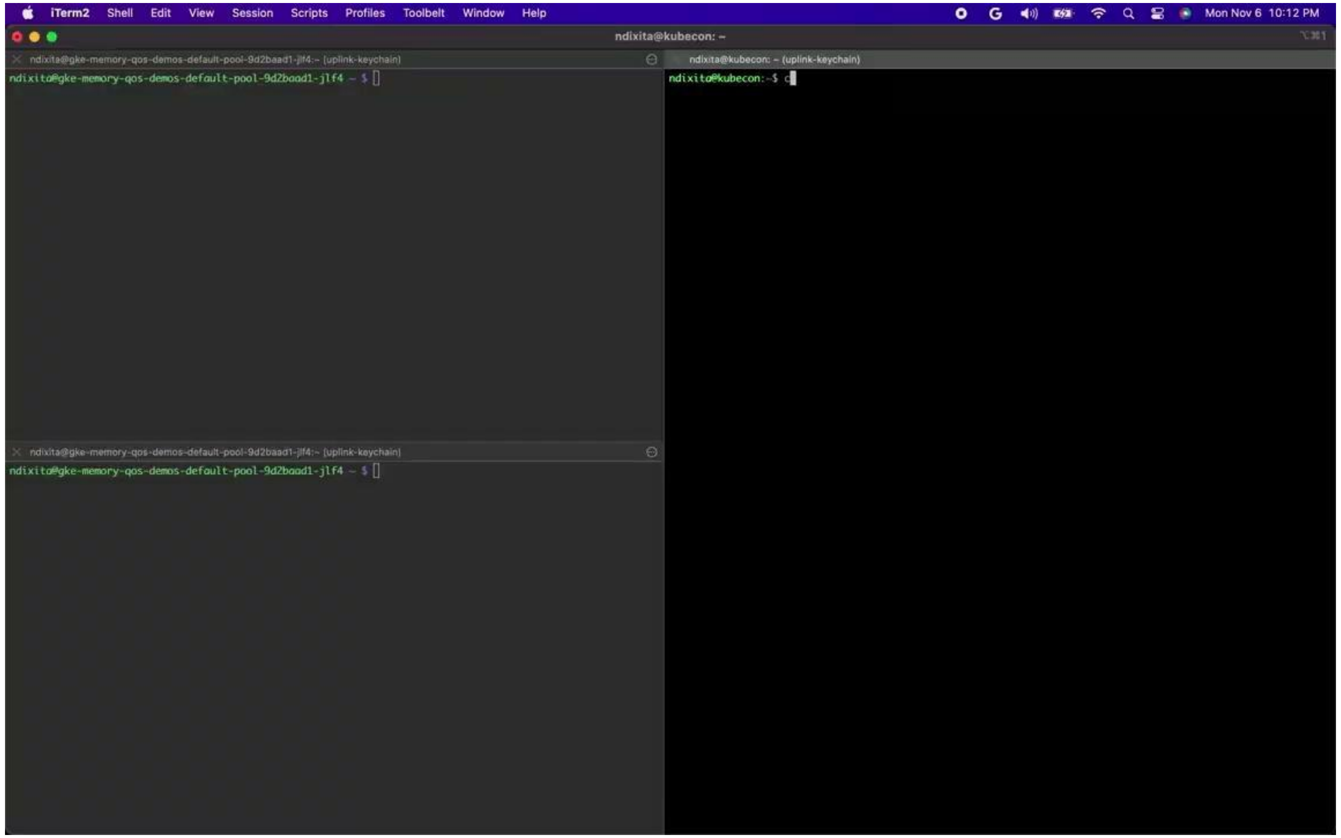
The high limit should be used in scenarios where an external process monitors the limited cgroup to alleviate heavy reclaim pressure.

- Livelock scenario when the process consumes memory at the faster pace than what memory reclaim can recover on reachin throttling limit i.e. memory.high.
  - the process is stuck indefinitely

- memory.high recommended to be uses in feedback loop ✓
  - external process to act when memory is throttled at memory.high level.

Demo

- When memory.high is set, and there’s no external process to alleviate heavy reclaim pressure
  - demonstrate throttling
  - demonstrate livelock scenario


iTerm2
Shell
Edit
View
Session
Scripts
Profiles
Toolbelt
Window
Help
Mon Nov 6 10:12 PM
ndixita@kubecon: ~
⌥⌘1
ndixita@gke-memory-qos-demos-default-pool-9d2baad1-jlf4:~ (uplink-keychain)
ndixita@kubecon: ~ (uplink-keychain)
ndixita@gke-memory-qos-demos-default-pool-9d2baad1-jlf4:~ (uplink-keychain)

```
ndixita@gke-memory-qos-demos-default-pool-9d2baad1-jlf4 ~ $
```

```
ndixita@kubecon:~$ c
```

```
ndixita@gke-memory-qos-demos-default-pool-9d2baad1-jlf4 ~ $
```

https://drive.google.com/file/d/1oWTfh4t1Np8OsGBZlVhwEW26bYa9ejHj/view?usp=drive_link

Takeaways

- Better to be OOM killed than being throttled forever.
- Involve subject-matter experts in KEP approval process.
- memory.high can be used with an external process for other use cases.

Use cases for memory.high

- Throttle with Liveness probe as an external process

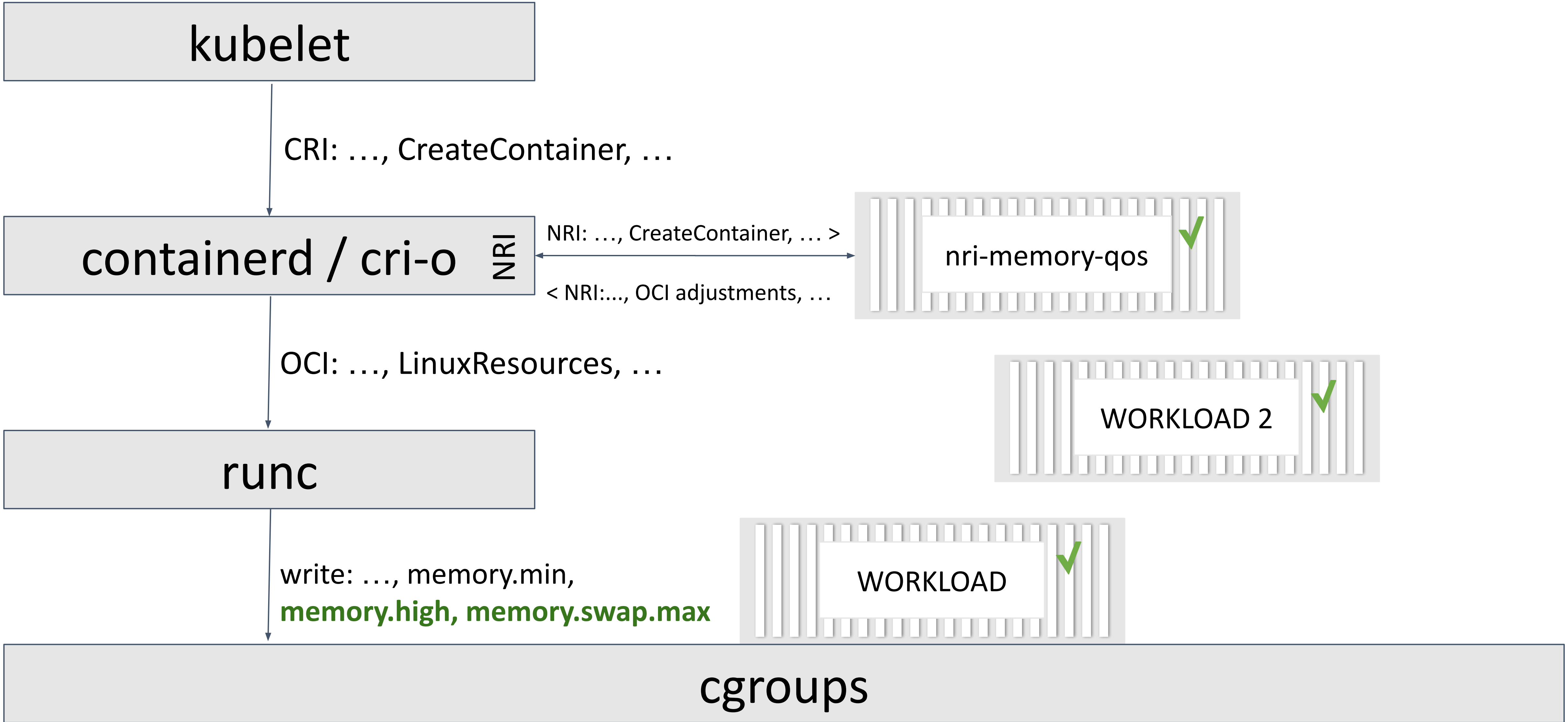
Use cases for memory.high

- memory.high as a signal to vertically scale the pods
  - External process can increase the memory limits when memory is throttled.
  - Reset memory.high to a new value based upon new memory limits.
  - Can use in place pod vertical scaling to scale the pods.

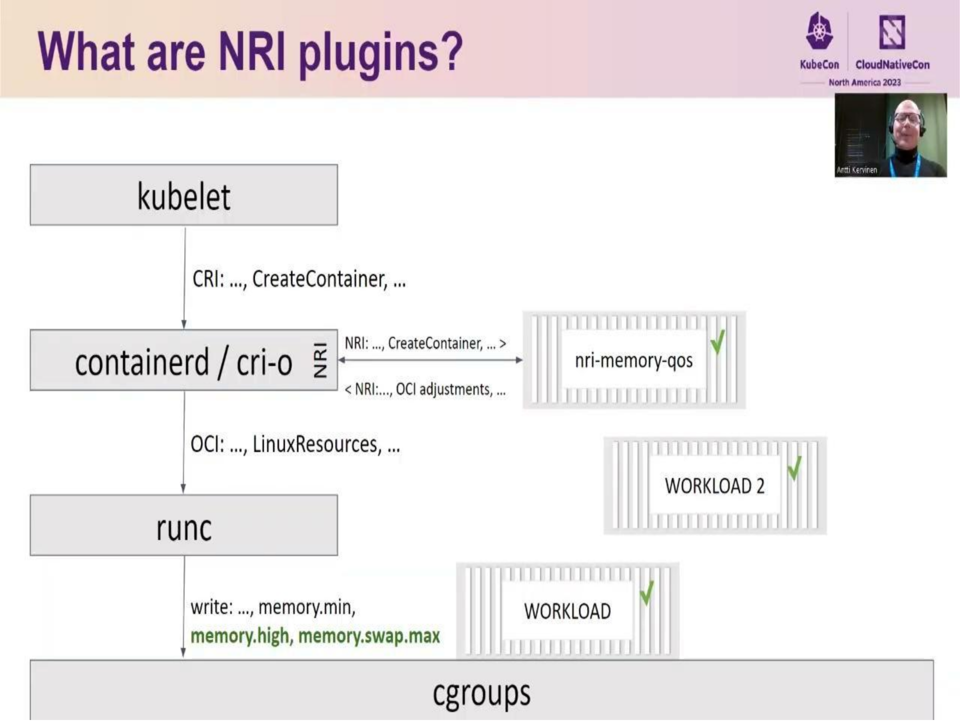
Use cases for memory.high

- memory.high to swap out container memory.
    - External process to swap out container memory when usage nears the limits.
    - Memory QoS NRI plugin can be used.

What are NRI plugins?


kubelet
CRI: …, CreateContainer, …
containerd / cri-o
NRI
NRI: …, CreateContainer, … >
< NRI:..., OCI adjustments, …
nri-memory-qos
✓
OCI: …, LinuxResources, …
WORKLOAD 2
✓
runc
write: …, memory.min,
memory.high, memory.swap.max
WORKLOAD
✓
cgroups

Demo
Swap memory using memory QoS NRI plugin



What are NRI plugins?

kubelet

CRI: ..., CreateContainer, ...

containerd / cri-o

NRI

NRI: ..., CreateContainer, ... >

< NRI:..., OCI adjustments, ...

nri-memory-qos ✓

OCI: ..., LinuxResources, ...

WORKLOAD 2 ✓

runc

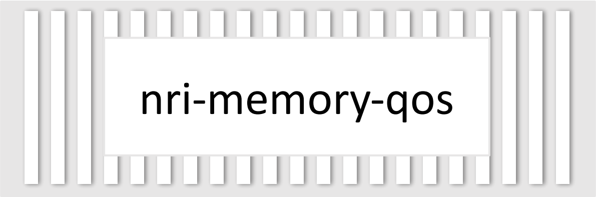
write: ..., memory.min,
memory.high, memory.swap.max

WORKLOAD ✓

cgroups

https://drive.google.com/file/d/1JVQgyJfZp6uNgdhpvUv3Zyh8fB9KD9tg/view?usp=drive_link

Installing nri-memory-qos



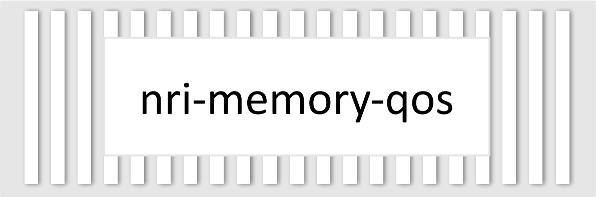
Project: https://github.com/containers/nri-plugins

Documentation: https://containers.github.io/nri-plugins/

Install:

```
helm repo add nri-plugins https://containers.github.io/nri-plugins
helm install nri-memory-qos nri-plugins/nri-memory-qos --namespace kube-system
```

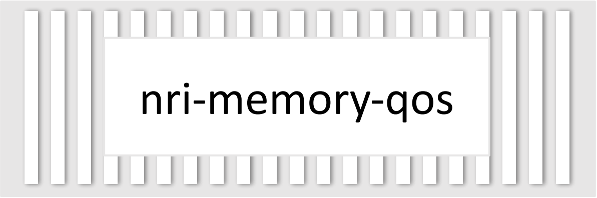
Using nri-memory-qos



ConfigMap

```
# Class-based access
classes:
- name: silver
  swaplimitratio: 0.2
- name: bronze
  swaplimitratio: 0.5

# Allow direct access
unifiedannotations:
- memory.swap.max
- memory.high
```

Using nri-memory-qos



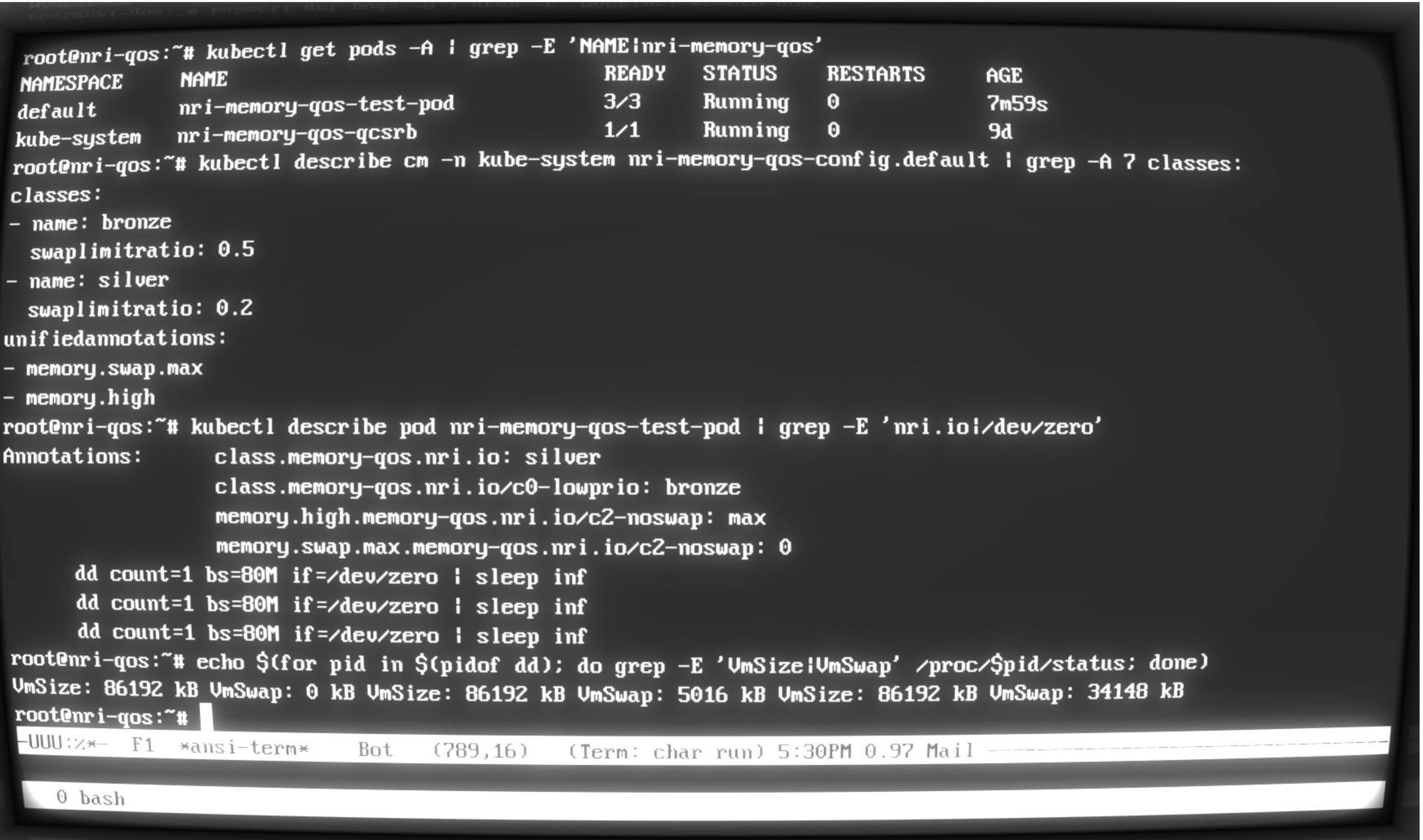
nri-memory-qos

```
ConfigMap

# Class-based access
classes:
- name: silver
  swaplimitratio: 0.2
- name: bronze
  swaplimitratio: 0.5

# Allow direct access
unifiedannotations:
- memory.swap.max
- memory.high
```

WORKLOAD

```
annotations:
  # Memory QoS class for all containers
  class.memory-qos.nri.io: silver

  # Memory QoS class for container B
  class.memory-qos.nri.io/B: bronze

  # Never swap memory of container A
  memory.swap.max.memory-qos.nri.io/A: "0"
  memory.high.memory-qos.nri.io/A: max
```

nri-memory-qos in action



```
root@nri-qos:~# kubectl get pods -A | grep -E 'NAME|nri-memory-qos'
NAMESPACE     NAME                            READY   STATUS    RESTARTS     AGE
default       nri-memory-qos-test-pod         3/3     Running   0            7m59s
kube-system   nri-memory-qos-qcsrb            1/1     Running   0            9d
root@nri-qos:~# kubectl describe cm -n kube-system nri-memory-qos-config.default | grep -A 7 classes:
classes:
- name: bronze
  swaplimitratio: 0.5
- name: silver
  swaplimitratio: 0.2
unifiedannotations:
- memory.swap.max
- memory.high
root@nri-qos:~# kubectl describe pod nri-memory-qos-test-pod | grep -E 'nri.io|/dev/zero'
Annotations:      class.memory-qos.nri.io: silver
                  class.memory-qos.nri.io/c0-lowprio: bronze
                  memory.high.memory-qos.nri.io/c2-noswap: max
                  memory.swap.max.memory-qos.nri.io/c2-noswap: 0
      dd count=1 bs=80M if=/dev/zero | sleep inf
      dd count=1 bs=80M if=/dev/zero | sleep inf
      dd count=1 bs=80M if=/dev/zero | sleep inf
root@nri-qos:~# echo $(for pid in $(pidof dd); do grep -E 'VmSize|VmSwap' /proc/$pid/status; done)
VmSize: 86192 kB VmSwap: 0 kB VmSize: 86192 kB VmSwap: 5016 kB VmSize: 86192 kB VmSwap: 34148 kB
root@nri-qos:~#
-UUU:%*-  F1  *ansi-term*    Bot   (789,16)   (Term: char run) 5:30PM 0.97 Mail ------------------------
  0 bash
```

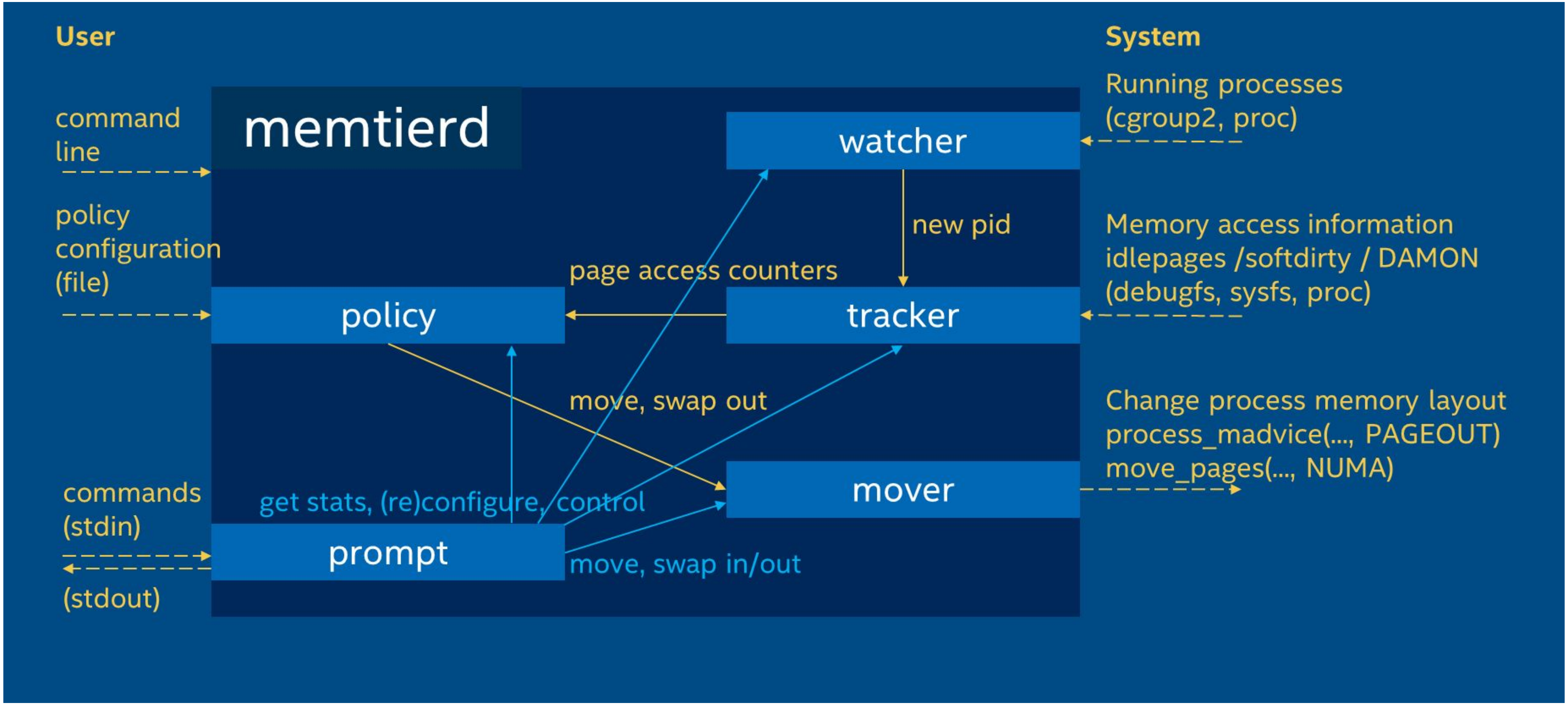
nri-memory-qos in action

```
## Do It Yourself: compressed in-RAM swap
## Run commands in a single-node cluster

# 1. Enable swap if not already enabled
modprobe zram
echo 4G > /sys/block/zram0/disksize
mkswap /dev/zram0
swapon /dev/zram0
column -t < /proc/swaps

# 2. Enable NRI in your container runtime and install nri-memory-qos
helm repo add nri-plugins https://containers.github.io/nri-plugins
helm install nri-memory-qos nri-plugins/nri-memory-qos --namespace kube-system --set
nri.patchRuntimeConfig=true

# 3. Create a test pod, 3 containers, all of them running dd that only allocated memory
kubectl apply -f
https://raw.githubusercontent.com/containers/nri-plugins/main/test/e2e/files/nri-memory-qos-test-pod.yaml

# 4. Show test pod’s annotations and container’s memory requests and limits
kubectl describe pod nri-memory-qos-test-pod | grep -E 'nri.io|c[0-2].*:|Request|Limit|memory:'

# 5. Show how differently dd’s are swapped
for pid in $(pidof dd); do
    echo dd pid: $pid
    grep -E 'VmSize|VmSwap' /proc/$pid/status
done
```

Controlling memory beyond cgroups

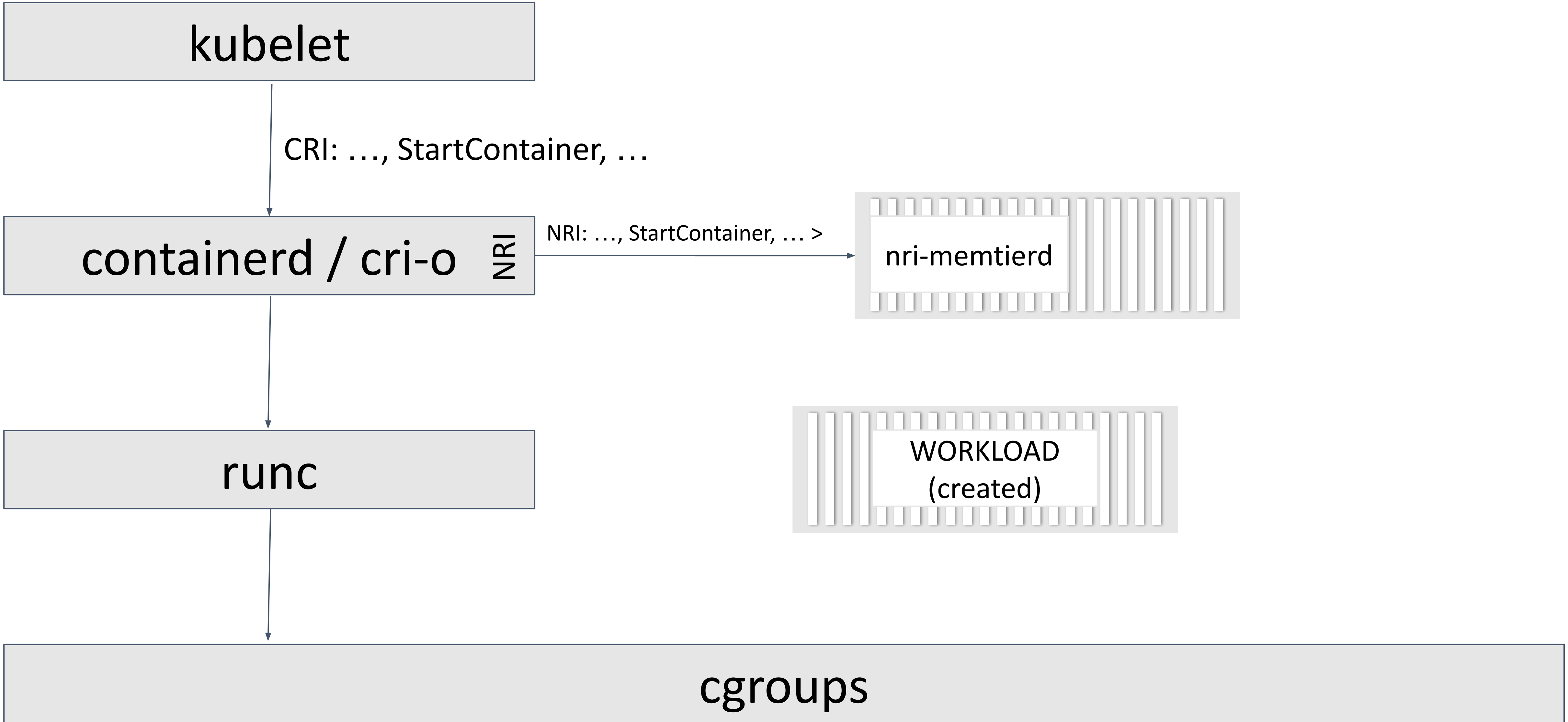
KubeCon | CloudNativeCon
North America 2023

The memtierd daemon: track, swap & move memory

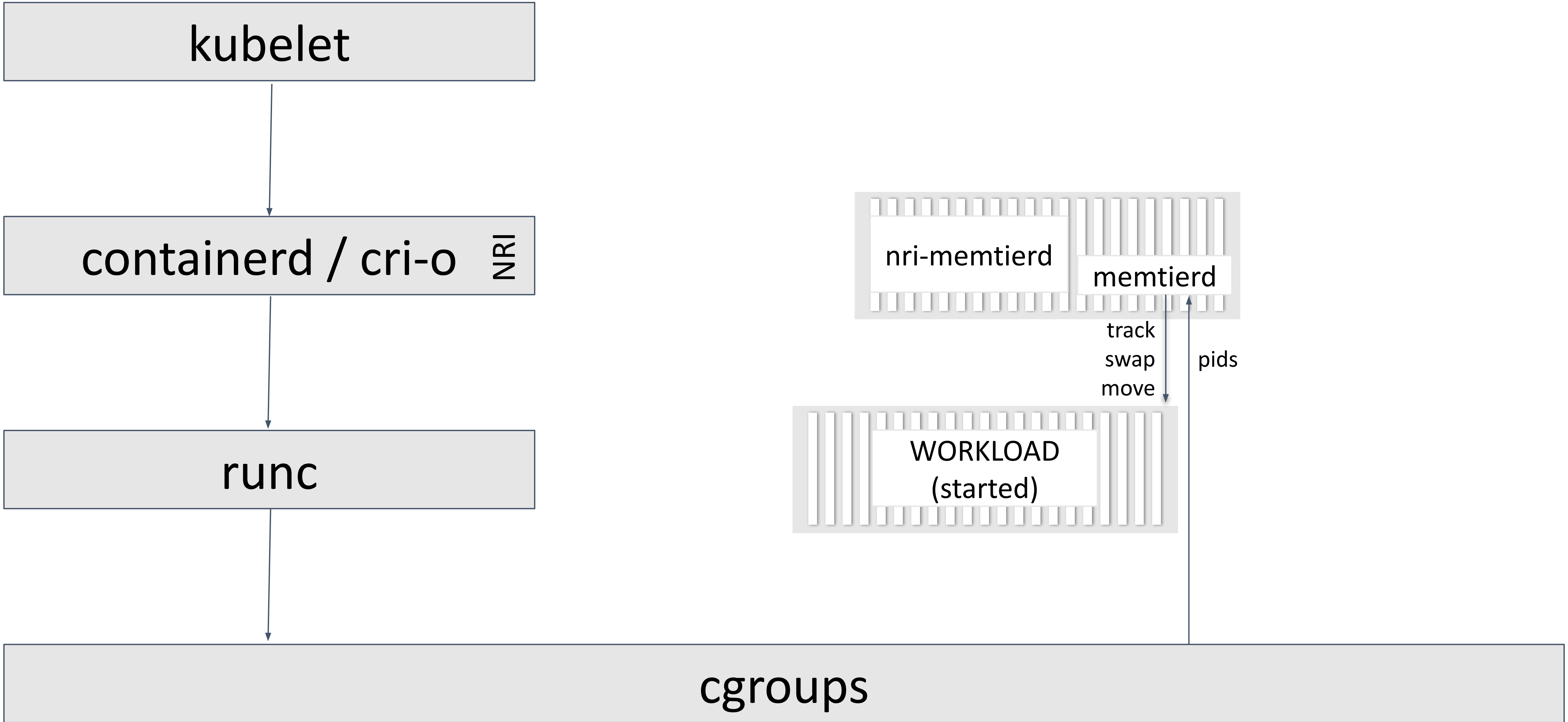

User
command line
policy configuration (file)
commands (stdin)
(stdout)
memtierd
watcher
new pid
page access counters
policy
tracker
move, swap out
get stats, (re)configure, control
mover
prompt
move, swap in/out
System
Running processes (cgroup2, proc)
Memory access information
idlepages /softdirty / DAMON
(debugfs, sysfs, proc)
Change process memory layout
process_madvice(..., PAGEOUT)
move_pages(..., NUMA)

NRI brings memtierd to Kubernetes

KubeCon | CloudNativeCon
North America 2023



NRI brings memtierd to Kubernetes



Installing nri-memtierd

Project: https://github.com/containers/nri-plugins

Documentation: https://containers.github.io/nri-plugins/

Install:

```
helm repo add nri-plugins https://containers.github.io/nri-plugins
helm install nri-memtierd nri-plugins/nri-memtierd --namespace kube-system
```

Using nri-memtierd

nri-memtierd

```
ConfigMap

# Class-based access
classes:
- name: swap-idle-data
  allowswap: true
  memtierdconfig: |
    policy:
    …
```

WORKLOAD

```
annotations:
  # Swap idle memory of containers in this pod
  # even if there is no memory pressure.
  class.memtierd.nri.io: "swap-idle-data"
  # Except for container A. Do not manage it.
  class.memtierd.nri.io/A: ""
```

How to improve? How to participate?

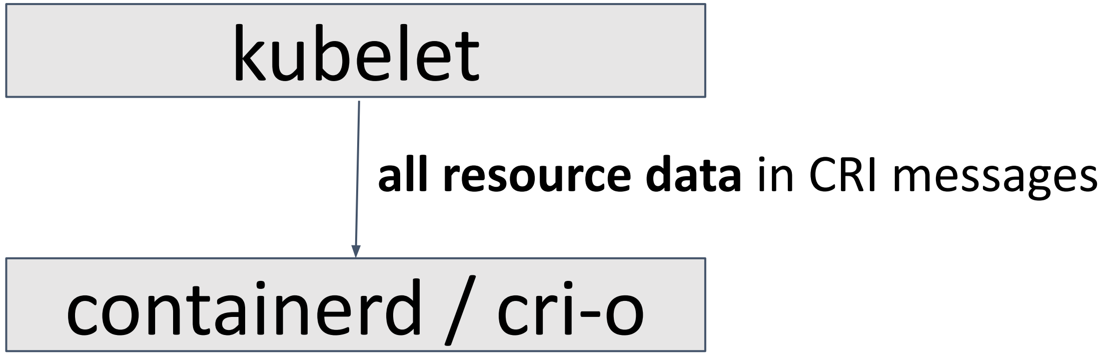
- QoS in annotations? Validating? Scheduling? QoS quotas?
  QoS Class Resources (KEP 3008) makes QoS a first class citizen in Kubernetes.

```
annotations:
  class.memory-qos.nri.io: "bronze"
```

⇒

```
containers:
- name: …
  resources:
    qosResources:
    - name: memory-qos
      class: bronze
```

- Not all pod/container resource information is available for NRI plugins.
  Pass down resources to CRI (KEP 4113)


kubelet

all resource data in CRI messages

containerd / cri-o

How to improve? How to participate?

- Help redesign KEP 2570

Thank You

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