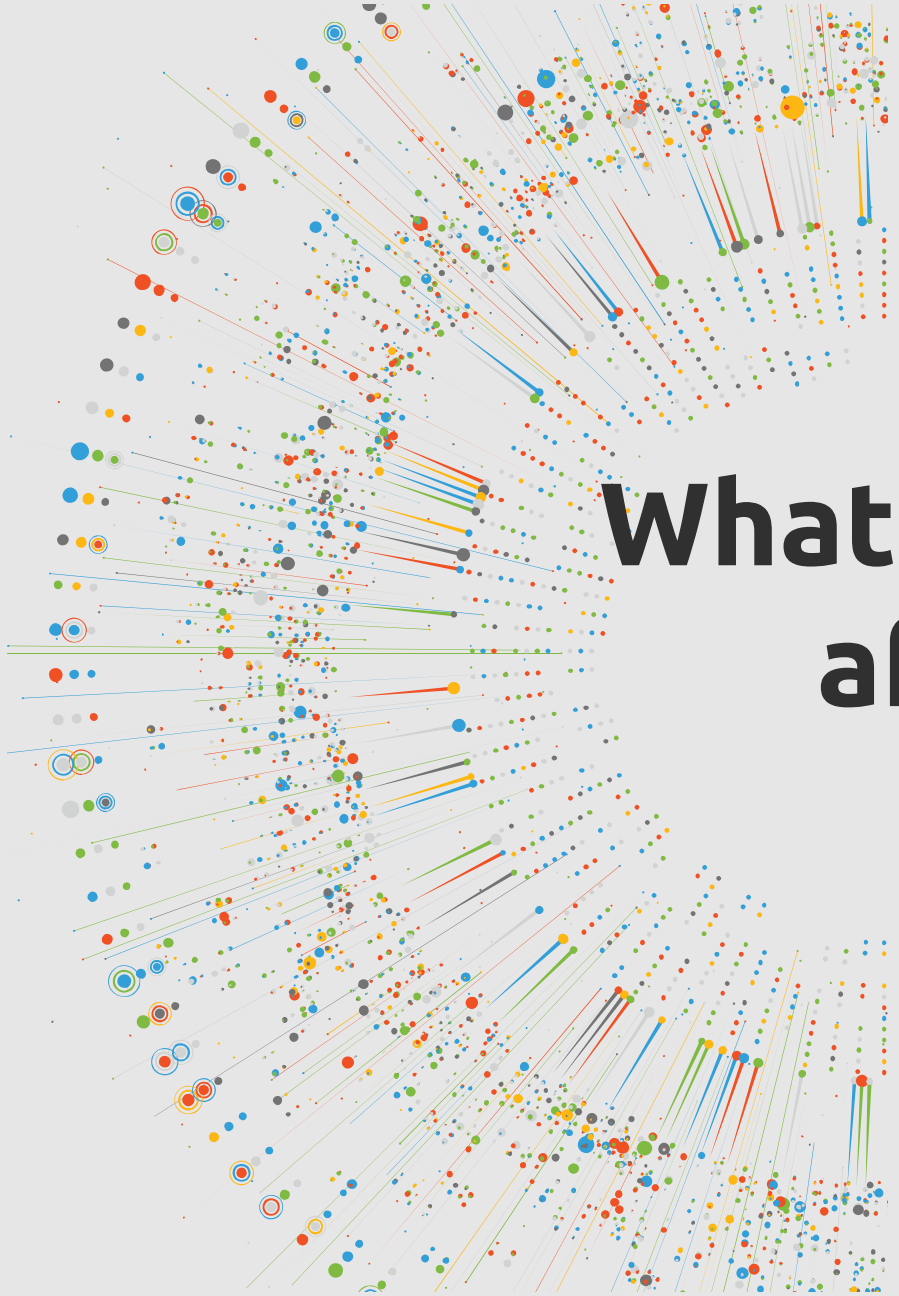


What is cgroups v2? and how it affects our application

(learning from the field)

Jun2-3 – 2023, KCD, Bangalore

Maheshkumar R

Cloud Solution Architect, Microsoft

@MahesKBlr

Resource allocations in a node

- Cluster consists of Nodes
  - Nodes contain resources such CPU's, Mem, Disk, GPUs etc..

- In a k8s node, CPU and memory are divided into,
  - Operating System
  - Kubelet, CNI, CRI, CSI (+ system daemons)
  - Pods
  - Eviction threshold

- Nodes advertise resource availability

  to the k8s scheduler
  - Node capacity
  - Node Allocatable = total capacity – Reserved

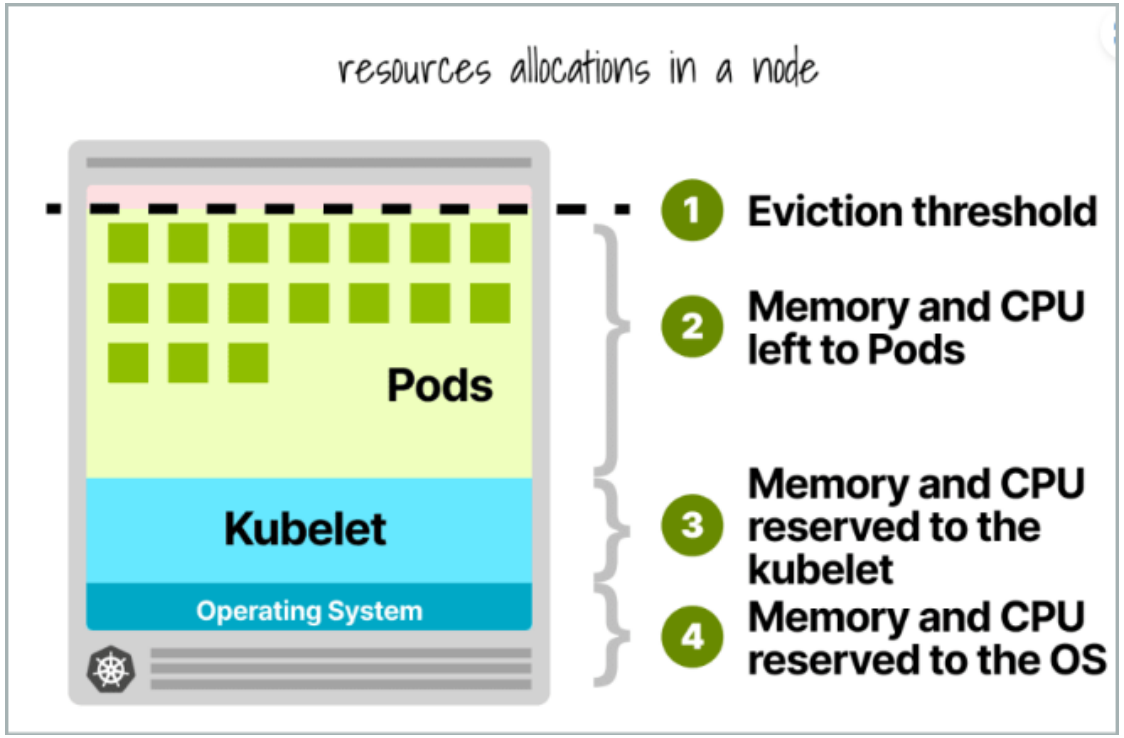


Image credit: Daniele Polencic

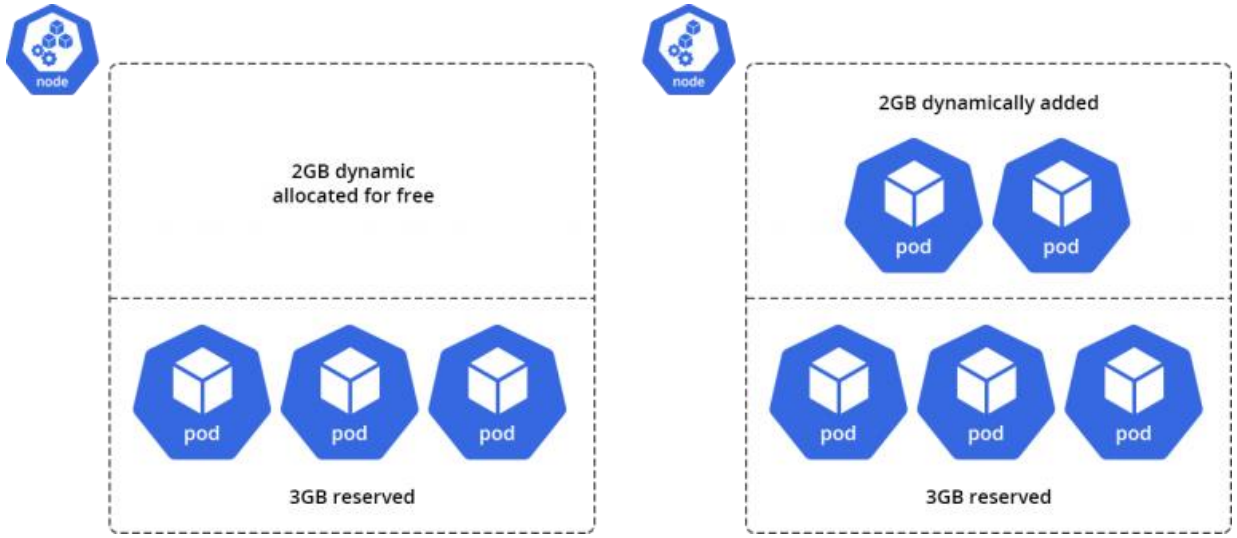
```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: app
    image: myapp
    resources:
      requests:
        cpu: "250m"
        memory: "64Mi"
      limits:
        cpu: "500m"
        memory: "128Mi
```

The minimum amount of resources this container needs

The maximum amount of resources this container can use

Resource Management Requirements

- Resource Isolation
  - Pods should stay within the limits – not hurt each other (or the system)
  - Pods should be able to receive consistent performance behavior based on their request
  - Prevent : Infinite Loops, Mem leak, node lockups
- Sizing
  - Allocate proper resources for pods
- Utilization
- Ensure resources area managed efficiently



How do we do this?

- **Cgroups – Linux kernel feature** – short for ‘control groups’
  - Way to group a set of process hierarchically
  - Set of **controllers to put limits on** cpu, memory, io, etc to manage resources in group and provide monitoring
- Controlled via pseudo-filesystem cgroupfs

Allow us to :

- Writes these files - to set limit usage of group of process (% of cpu/memory/pids)
- Measure resource usage for a group processes

https://kubernetes.io/docs/concepts/architecture/cgroups/

More on cgroup

Without cgroup, there are no containers on k8s

Constrain(limit) resources allocated to processes

kubelet and CRI interface with cgroups to enforce resource management for pods and containers

cgroup v1 & cgroup v2. cgroup v2 is the new gen and defacto

How does K8s make use of cgroups?

- Container resource enforcement
- Container resource metrics

https://kubernetes.io/docs/concepts/architecture/cgroups/

History of cgroups v1,v2

- cgroup v1 was 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 Lx 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

- cgroupv1 is considered legacy – no new features are being added

- 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)

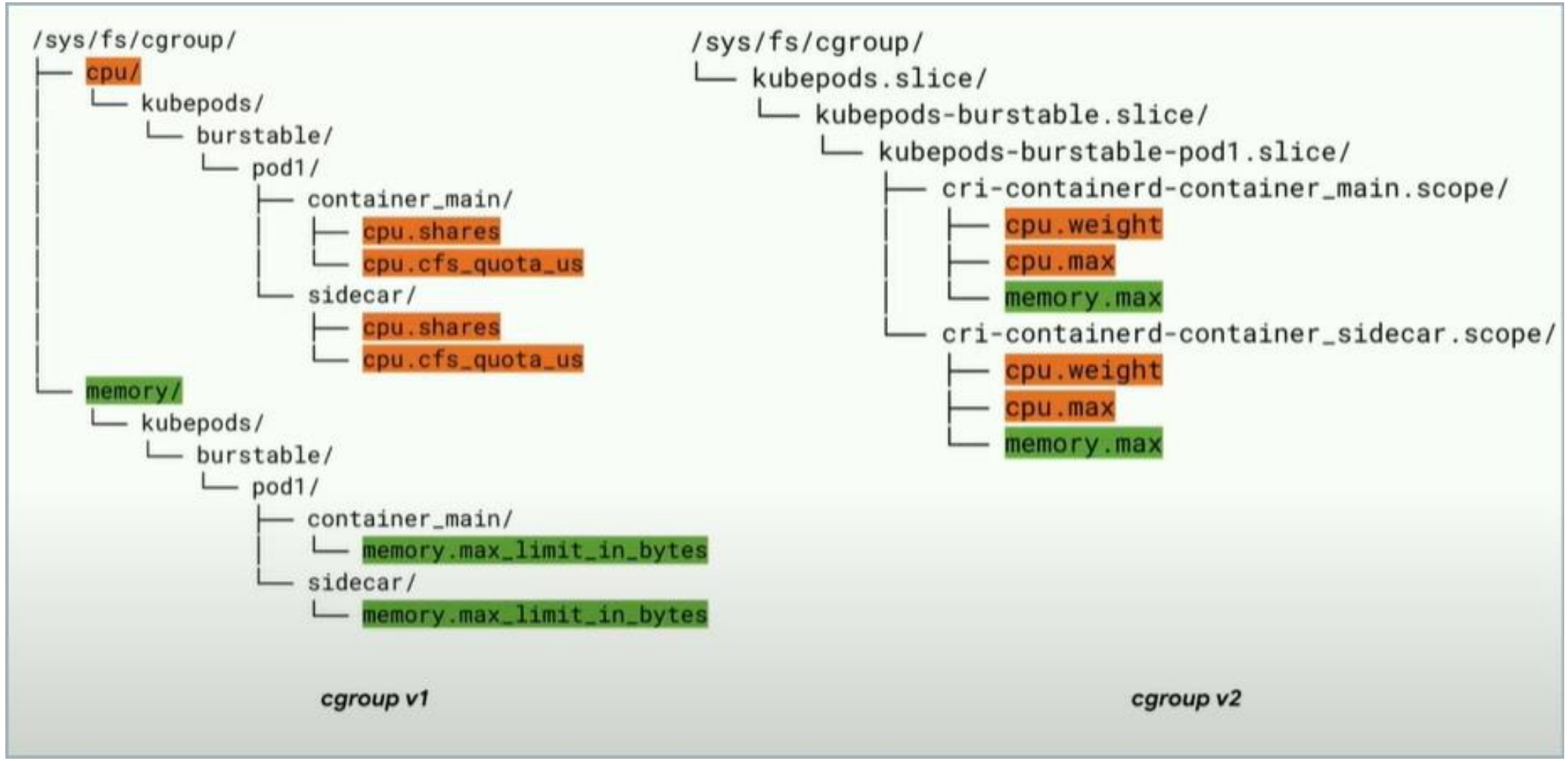
- Runtimes
  - containerd 1.4 and later
  - cri-o v1.20 and later
  - docker/moby > 20.10
  - runc > 1.0.0
  - crun > 0.7

- 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
- Hard and Soft memory limits
- OOM killer is cgroup aware
- PSI (Pressure stall information) metrics
  - Detect resource pressure of CPU, Memory and IO
- Improved rootles via delegation support

cgroups v1 vs v2 hierarchy



Accessible via virtual filesystem (like everything in Linux)

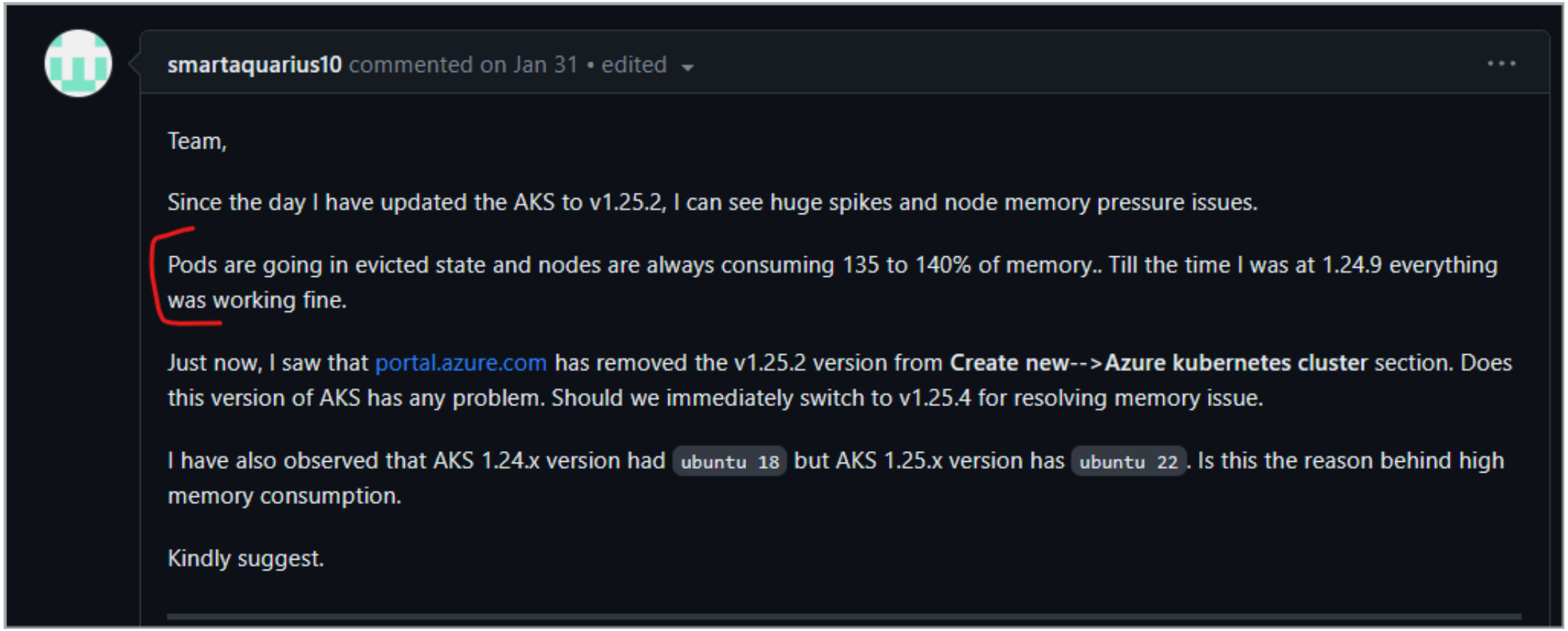
Test your apps on cgroup v2

- Most apps do not have cgroup dependencies, but some apps may

- Third party monitoring and security agents
  - Contact vendor and ensure agents support cgroup v2

  - Java apps uses JDK to read cgroup settings for auto-tuning, upgrade to JDK 11.0.16 and later or JDK 15+ fully support cgroup v2

Field learning



smartaquarius10 commented on Jan 31 • edited

Team,

Since the day I have updated the AKS to v1.25.2, I can see huge spikes and node memory pressure issues.

Pods are going in evicted state and nodes are always consuming 135 to 140% of memory.. Till the time I was at 1.24.9 everything was working fine.

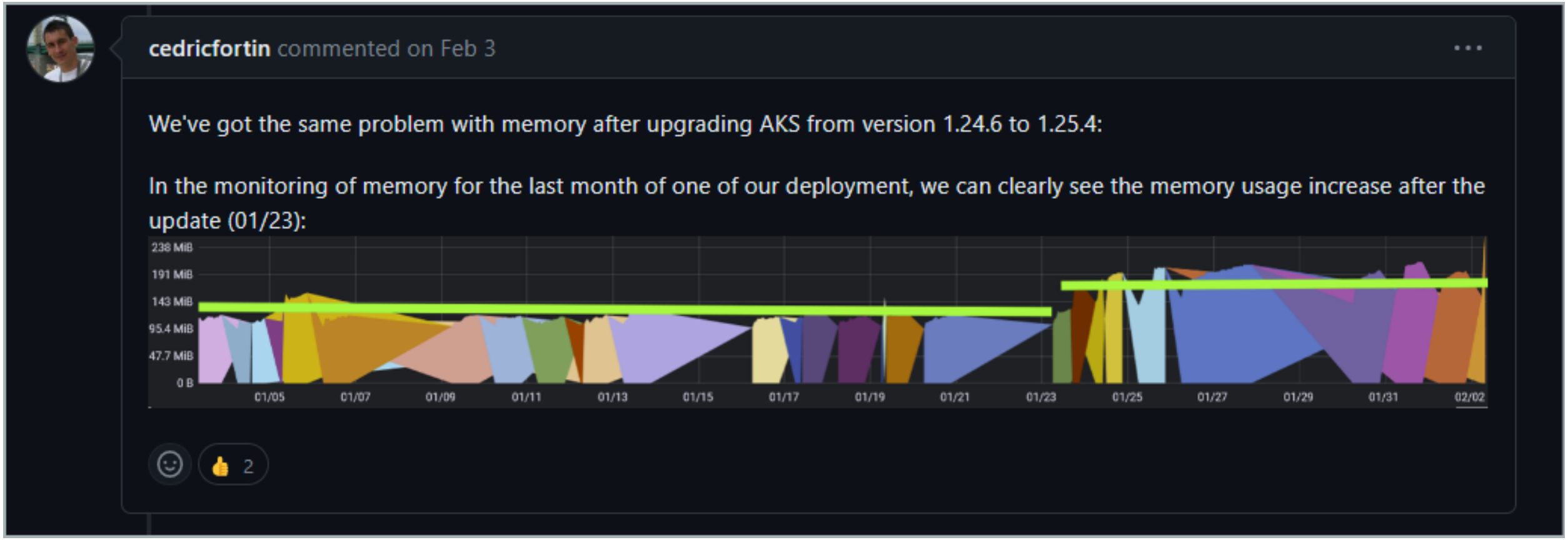
Just now, I saw that portal.azure.com has removed the v1.25.2 version from **Create new-->Azure kubernetes cluster** section. Does this version of AKS has any problem. Should we immediately switch to v1.25.4 for resolving memory issue.

I have also observed that AKS 1.24.x version had `ubuntu 18` but AKS 1.25.x version has `ubuntu 22`. Is this the reason behind high memory consumption.

Kindly suggest.

https://github.com/Azure/AKS/issues/3443

Field learning



https://github.com/Azure/AKS/issues/3443

Field learning

- **cgroups v2 is default from v1.25 onwards (k8s)**
- Java apps running < JDK ver11 moving to K8s 1.25 showing memory issue - **increased resource consumption and at times OOM issue** ☹
- We also noticed someone upgrading to v1.25 with older SDK doubling the cluster size – scaled to meet the app resource demand otherwise OOM issue ☹
- We scaled to from 17 nodes (old setup) -> 35+ nodes (in v1.25) ☺

Why v1.25 & Java

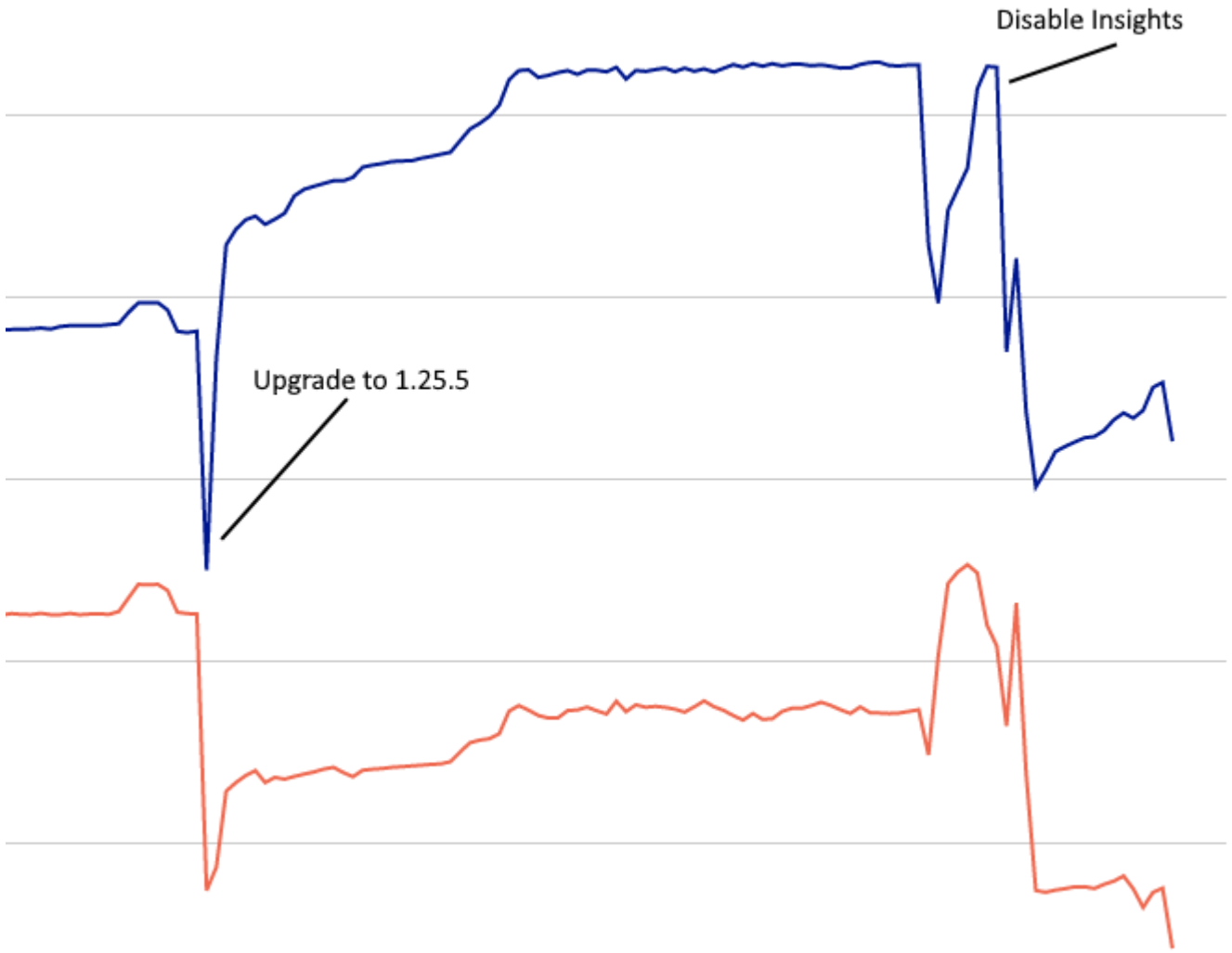
- Kubernetes cgroups v2 reached GA on the version 1.25.x and with this change K8s(AKS) changed the OS of the nodes from Ubuntu18.04 to Ubuntu22.04 that already uses cgroups v2 by default.

- JRE didn't had support for cgroups v2 container awareness – bug JRE 11.0.14. This means that the container were not able to respect the imposed memory quotas defined on the deployment descriptor.

- In cgroup v2, the location of these files has changed and Java applications prior to JDK 15 will exhibit significant memory consumption which may make your environments unstable.

- Oracle and OpenJDK addressed this issue by supporting it natively on JRE 17 and backporting this fix to JRE 15 and JRE 11.0.16++

V1.25 Mem chart



From:
Sent: Tuesday, February 28, 2023 4:31:26 PM
To: Maheshkumar Rajarathinavel <[redacted]@microsoft.com>
Cc: [redacted]soft.com>; [redacted]rosoft.com>
Subject: Re: [EXTERNAL]

Hi Mahesh,

I think it is better to analyze it together over a call.

If the recommendation is to move to jdk 15+ do we have any other option or a way to control the memory consumption?
As you know it is not so easy to change 150+ services to jdk 15 in a week's time frame.
While we are at it, we need an intermediate solution to alleviate our expenditure.
Previously we managed with a 24 node cluster, but now, we have to scale it beyond 30+ nodes to handle the same traffic.

Thanks,

V1.25 Mem chart

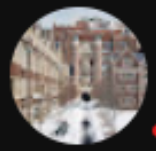
With JRE 11.0.13 (after 10 minutes running)

CPU(cores)	MEMORY(bytes)
3m	1312Mi
2m	1332Mi
9m	1964Mi
3m	1299Mi
22m	1278Mi
4m	1167Mi
17m	1487Mi

and with JRE 11.0.18 (after 10 minutes running)

CPU(cores)	MEMORY(bytes)
3m	1312Mi
2m	1332Mi
3m	1965Mi
3m	1299Mi
14m	507Mi
5m	1194Mi
16m	1487Mi

Only the workload with the arrow was upgraded



aquarius 2 months ago

Is it possible to disable Cgroupsv2 and re-enable the v1... For enabling this, cloud provider has started using ubuntu 22 which is causing high memory consumption issues. All of a sudden low memory node sku, especially consisting of 4 GB RAM, have stopped working. What can we do in such cases because downgrading the kubernetes is not possible when we are using...

Reply

RE: CGroups v1 in AKS 1.25.

Maheshkumar Rajarathinavel

To

Cc

General

Reply

In both the cases, upgrading java version confirms the issue.

I deployed JDK 11 application on V1.25.5 and V1.24.9 and deployed other JDK 17 application on V1.25.5 and V1.24.9. Here is the difference in memory utilization of pods .

JDK 11 :

V1.25.5

```
C:\Users\pallavidesai\java-docker-build-tutorial>kubectl top pods
NAME                                      CPU(cores)   MEMORY(bytes)
flightbookingsystemsample-7cff9b84c9-7vwdt   2m           418Mi

C:\Users\pallavidesai\java-docker-build-tutorial>kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-nodepool1-24667080-vmss000000   Ready    agent   10h   v1.25.5
aks-nodepool1-24667080-vmss000001   Ready    agent   10h   v1.25.5
aks-nodepool1-24667080-vmss000002   Ready    agent   10h   v1.25.5

C:\Users\pallavidesai\java-docker-build-tutorial>
```

418Mi

V1.24.9

```
C:\Users\pallavidesai\java-docker-build-tutorial>kubectl top pods
NAME                                      CPU(cores)   MEMORY(bytes)
flightbookingsystemsample-86d7c475f9-f2jth   2m           354Mi

C:\Users\pallavidesai\java-docker-build-tutorial>kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-nodepool1-16134279-vmss000000   Ready    agent   20h   v1.24.9
aks-nodepool1-16134279-vmss000001   Ready    agent   20h   v1.24.9
aks-nodepool1-16134279-vmss000002   Ready    agent   20h   v1.24.9

C:\Users\pallavidesai\java-docker-build-tutorial>
```

354Mi

JDK 17

```
pallavisample-757b5b6fbd-2hbp7            2m           44Mi

C:\Users\pallavidesai\java-docker-build-tutorial>kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-nodepool1-24667080-vmss000000   Ready    agent   10h   v1.25.5
aks-nodepool1-24667080-vmss000001   Ready    agent   10h   v1.25.5
aks-nodepool1-24667080-vmss000002   Ready    agent   10h   v1.25.5

C:\Users\pallavidesai\java-docker-build-tutorial>
```

44Mi

```
pallavisample-5d9bfd676-k2m5f             2m           42Mi

C:\Users\pallavidesai\java-docker-build-tutorial>kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-nodepool1-16134279-vmss000000   Ready    agent   21h   v1.24.9
aks-nodepool1-16134279-vmss000001   Ready    agent   21h   v1.24.9
aks-nodepool1-16134279-vmss000002   Ready    agent   21h   v1.24.9
```

42Mi

Here is the reference: https://blog.kintone.io/entry/2022/03/08/170206#Java

Suggestion for CX is to move to JDK 15+.

Recommendations

1) As cgroup v2 is GA in 1.25, and is also the default on Ubuntu 22.04, we should migrate our applications to JDK 15+
   - Update our JDK version which has the fix i.e. 11.0.16/17 or above of course - JDK https://bugs.openjdk.org/browse/JDK-8230305
   - V2 awareness fix: If cgroups v2 unified hierarchy is available only, use the cgroups v2 backend. Otherwise fall back to existing cgroups v1 container support.

2) If we can't update our JDK in-time (given 1.24 EOL), then try revert which is a temp solution;
   - this will allow us to upgrade to 1.25 but keep cgroupsv1 as the effective cgroups mode

Revert cgroup (temp solution)

- An alternative temporary solution is to revert the cgroup version on your nodes using this Daemonset
- The Daemonset by default will apply to all nodes in your cluster and will reboot them to apply the cgroup change.
- Please set a nodeSelector to control how this gets applied.

- https://github.com/Maheshk-MSFT/CGroupsV2_in_AKS_1.25_JDKBelow11_OOM

Thank you

@MahesKBlr

Slide deck will be posted in my GH https://github.com/Maheshk-MSFT

References

- https://community.cncf.io/events/details/cncf-kcd-bengaluru-presents-kubernetes-community-days-bengaluru-2023-in-person/
- https://medium.com/geekculture/layer-by-layer-cgroup-in-kubernetes-c4e26bda676c
- https://dev.to/danielepolencic/reserved-cpu-and-memory-in-kubernetes-nodes-2f31
- https://blog.kintone.io/entry/2022/03/08/170206

Talks and resources to check out

- See more talks this week!

- *Cgroupv2 Is Coming Soon To a Cluster Near You*
  - David Porter, Google & Mrunal Patel, RedHat
  - https://sched.co/182JZ on Friday @ 2:00pm

- *Kubernetes SIG Node Intro And Deep Dive*
  - Sergey Kanzhelev & Dawn Chen, Google; Derek Carr, Red Hat
  - https://sched.co/182Pi on Friday @ 4:00pm

- Release Blog
  - *Kubernetes 1.25: cgroup v2 graduates to GA*
  - https://kubernetes.io/blog/2022/08/31/cgroupv2-ga-1-25/