KubeCon | CloudNativeCon

North America 2024

KubeCon | CloudNativeCon
North America 2024

Pod Power: Liberating Kubernetes Users from Container Resource Micromanagement

Dixita Narang @Google
Peter Hunt @Red Hat

Requests and Limits

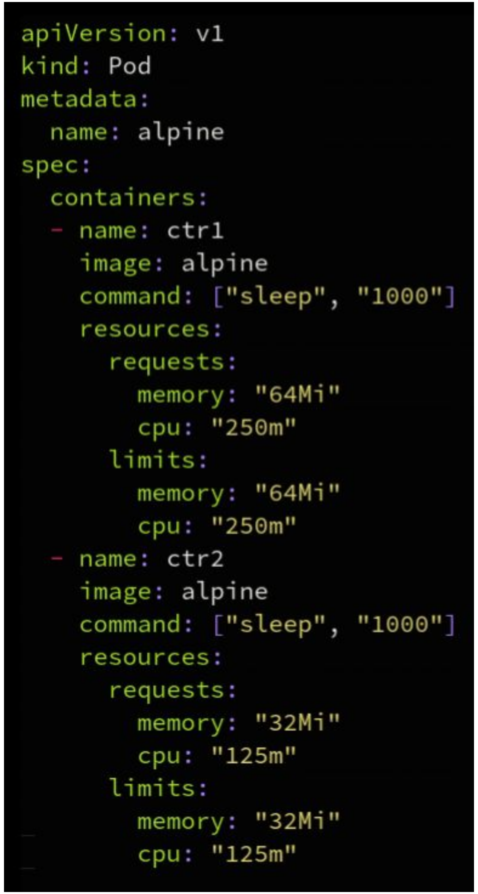
Requests:
“Make sure I have at least this much”

https://live.staticflickr.com/7770/27018180752_649caa1aec_b.jpg

Limits:
“I’m prepared for punishment if I use more than this much”

https://www.flickr.com/photos/47213689@N00/319188088

Requests and Limits: CPU



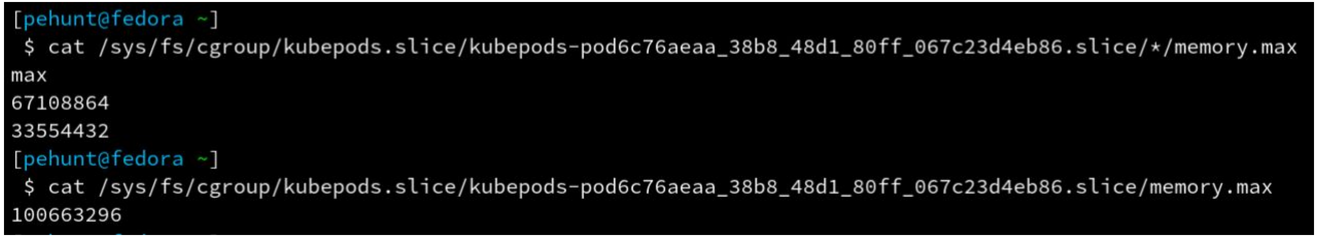
```
apiVersion: v1
kind: Pod
metadata:
  name: alpine
spec:
  containers:
  - name: ctr1
    image: alpine
    command: ["sleep", "1000"]
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "64Mi"
        cpu: "250m"
  - name: ctr2
    image: alpine
    command: ["sleep", "1000"]
    resources:
      requests:
        memory: "32Mi"
        cpu: "125m"
      limits:
        memory: "32Mi"
        cpu: "125m"
```

Memory Request

- Scheduler schedules with the Memory requests
- **CRI ignores requested Memory**

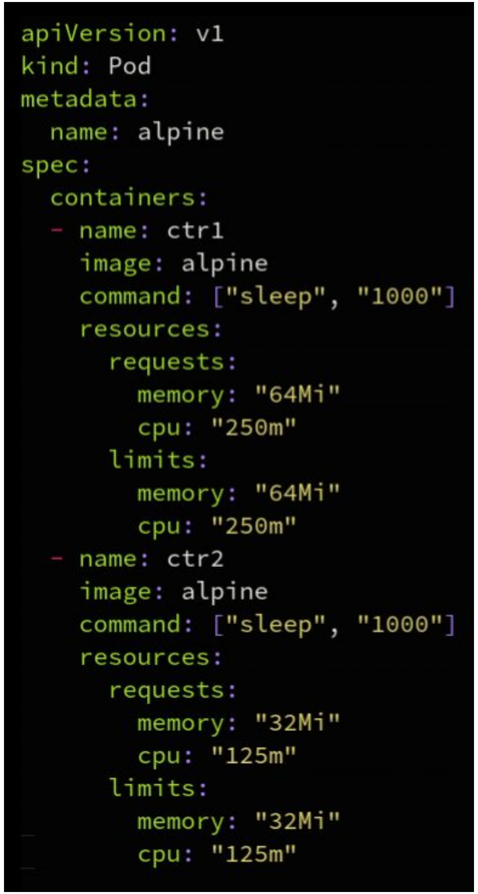
Memory Limit

- Scheduler ignores the Memory limits
- Container runtime maps the limits to “memory.max”.



```
[pehunt@fedora ~]
 $ cat /sys/fs/cgroup/kubepods.slice/kubepods-pod6c76aeaa_38b8_48d1_80ff_067c23d4eb86.slice/*/memory.max
max
67108864
33554432
[pehunt@fedora ~]
 $ cat /sys/fs/cgroup/kubepods.slice/kubepods-pod6c76aeaa_38b8_48d1_80ff_067c23d4eb86.slice/memory.max
100663296
```

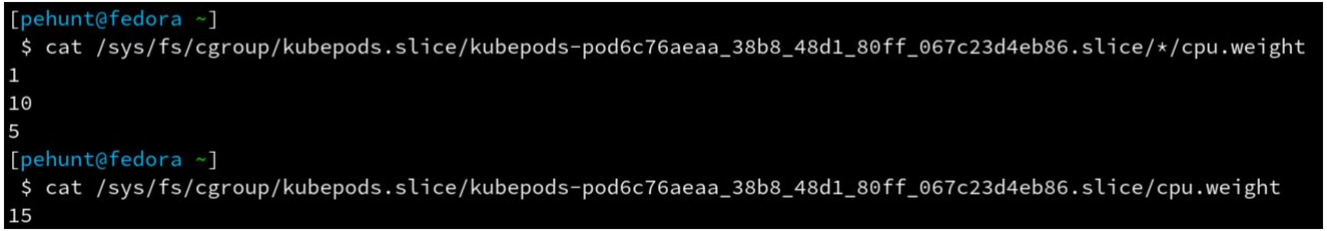
Requests and Limits: CPU



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: alpine
spec:
  containers:
  - name: ctr1
    image: alpine
    command: ["sleep", "1000"]
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "64Mi"
        cpu: "250m"
  - name: ctr2
    image: alpine
    command: ["sleep", "1000"]
    resources:
      requests:
        memory: "32Mi"
        cpu: "125m"
      limits:
        memory: "32Mi"
        cpu: "125m"
```

CPU Request

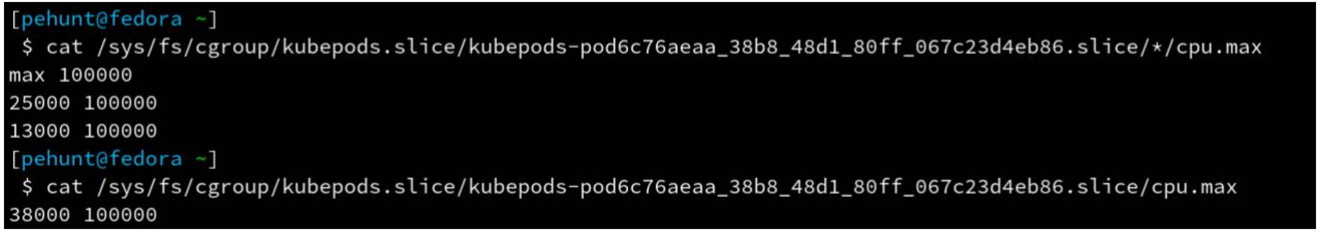
- Scheduler schedules with the CPU requests
- **CRI maps the requested CPU to “cpu.weight”**



```
[pehunt@fedora ~]
 $ cat /sys/fs/cgroup/kubepods.slice/kubepods-pod6c76aeaa_38b8_48d1_80ff_067c23d4eb86.slice/*/cpu.weight
1
10
5
[pehunt@fedora ~]
 $ cat /sys/fs/cgroup/kubepods.slice/kubepods-pod6c76aeaa_38b8_48d1_80ff_067c23d4eb86.slice/cpu.weight
15
```

CPU Limit

- Scheduler ignores the CPU limits
- Container runtime maps the limits to “cpu.max”.



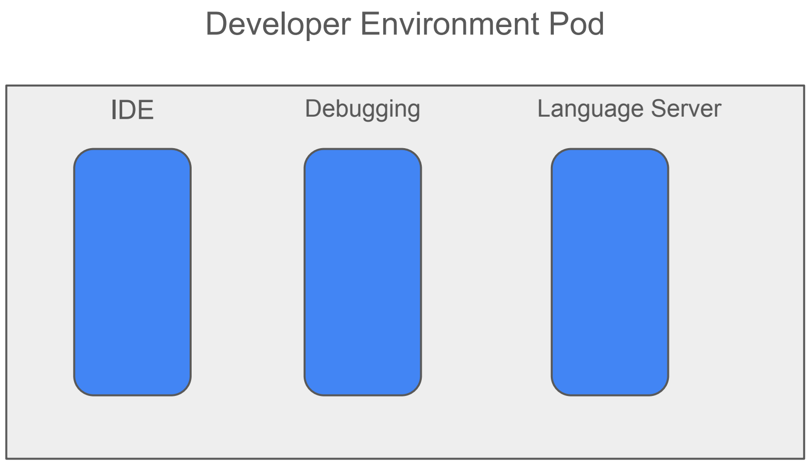
```
[pehunt@fedora ~]
 $ cat /sys/fs/cgroup/kubepods.slice/kubepods-pod6c76aeaa_38b8_48d1_80ff_067c23d4eb86.slice/*/cpu.max
max 100000
25000 100000
13000 100000
[pehunt@fedora ~]
 $ cat /sys/fs/cgroup/kubepods.slice/kubepods-pod6c76aeaa_38b8_48d1_80ff_067c23d4eb86.slice/cpu.max
38000 100000
```

Problem: Accurately Estimating Resources

https://www.flickr.com/photos/54942754@N02/24800603239

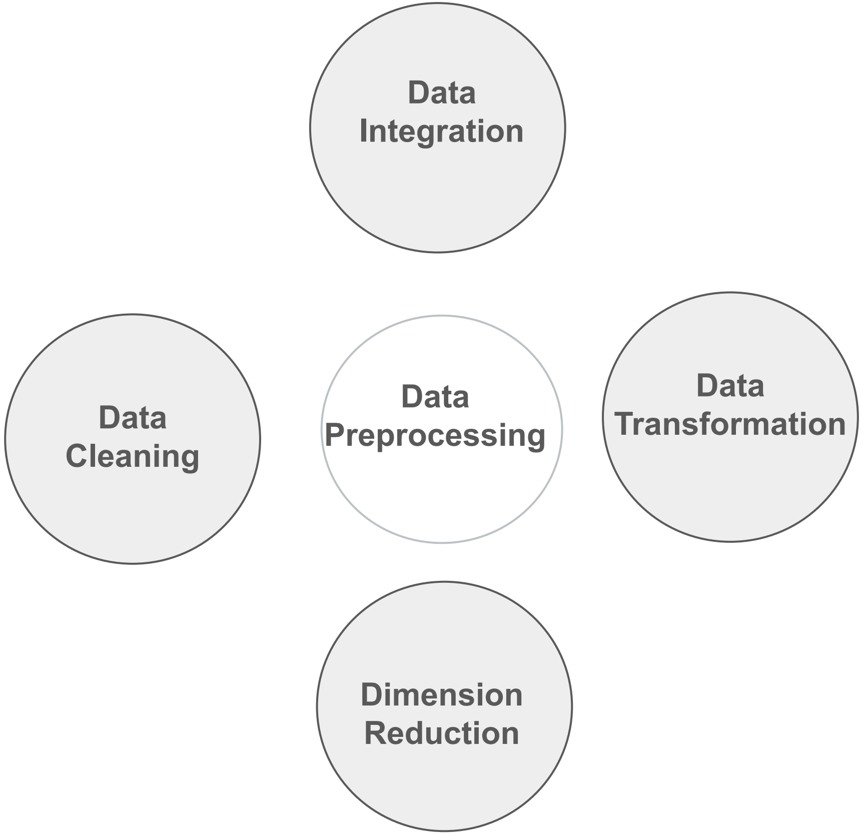
Problem: Multi-Container Pods

- Requires meticulous configurations of individual containers.
- Complicated Resource Allocation for Unpredictable Workloads.
- Difficult to manage resources at pod-level for multi-container pods.

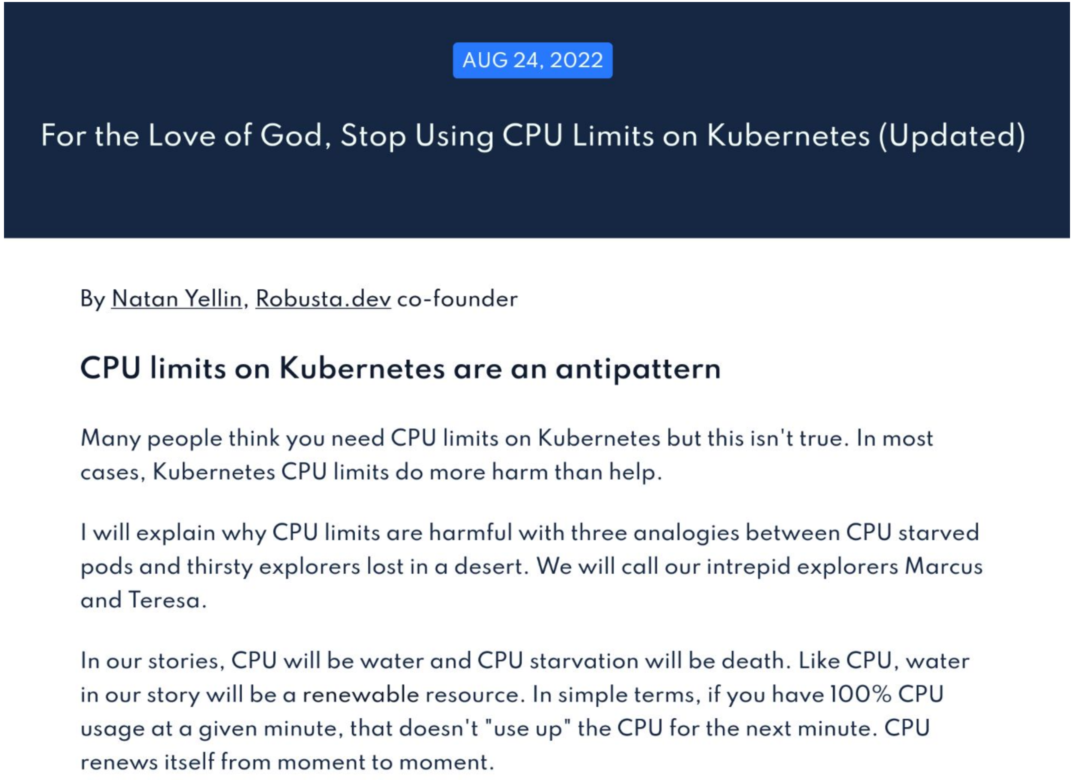


Problem: High-Burst Applications

- No access to idle resources.
- Increased Cost
- Reduced Efficiency



Problem: Idling Workloads



AUG 24, 2022

For the Love of God, Stop Using CPU Limits on Kubernetes (Updated)

By Natan Yellin, Robusta.dev co-founder

CPU limits on Kubernetes are an antipattern

Many people think you need CPU limits on Kubernetes but this isn't true. In most cases, Kubernetes CPU limits do more harm than help.

I will explain why CPU limits are harmful with three analogies between CPU starved pods and thirsty explorers lost in a desert. We will call our intrepid explorers Marcus and Teresa.

In our stories, CPU will be water and CPU starvation will be death. Like CPU, water in our story will be a renewable resource. In simple terms, if you have 100% CPU usage at a given minute, that doesn't "use up" the CPU for the next minute. CPU renews itself from moment to moment.

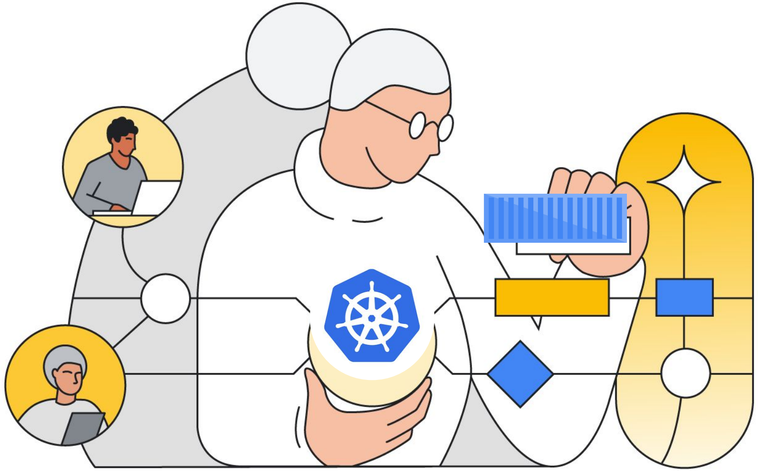
https://home.robusta.dev/blog/stop-using-cpu-limits

Pod Level Resource Specifications

Pod Level Resource Specifications

- Requests and Limits for the entire pod.
- Containers dynamically share unused resources.

A Holistic Approach to Resource Management



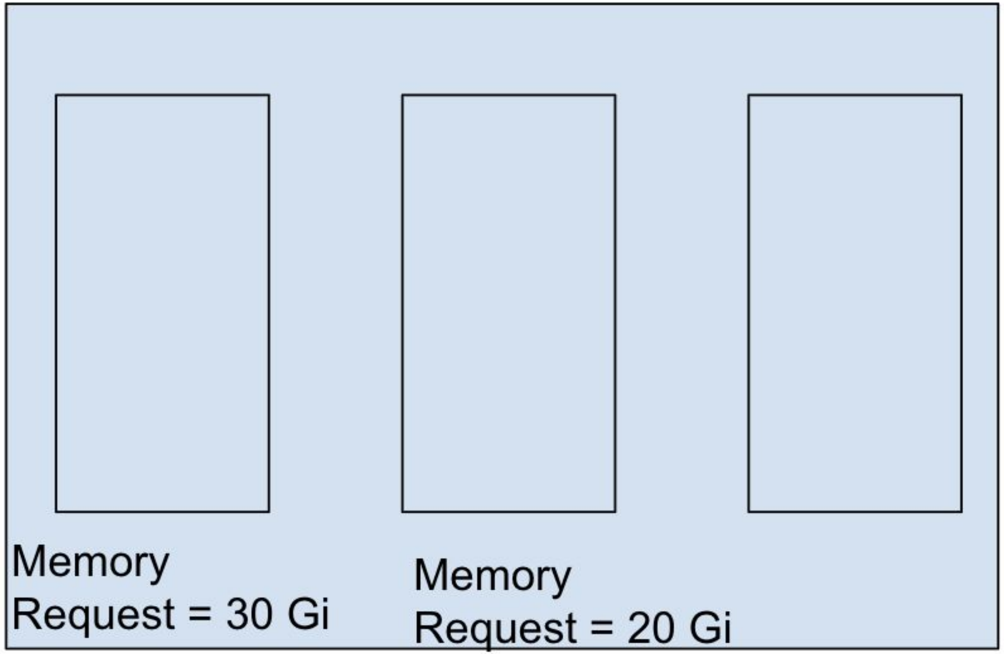
```
apiVersion: v1
kind: Pod
metadata:
  name: developer-environment
spec:
  resources:
    requests:
      memory: 64Mi
      cpu:    100m
    limits:
      memory: 128Mi
      cpu:    500m
  containers:
  - name: ide
    image: busybox
  initContainers:
  - name: debugging
    image: busybox
    restartPolicy: Always
  - name: language-server
    image: busybox
    restartPolicy: Always
status:
  ...
  qosClass: Burstable
```

Unlocking The Benefits

- Simplified Resource Management
  - Users specify one set of requests and limits, making the configuration simpler.
- Greater Flexibility
  - Kubernetes allocates resources among containers dynamically, based on need.
- Better Resource Utilization
  - Reduces underutilized resources, leading to cost savings and better node utilization.

Unlocking The Benefits

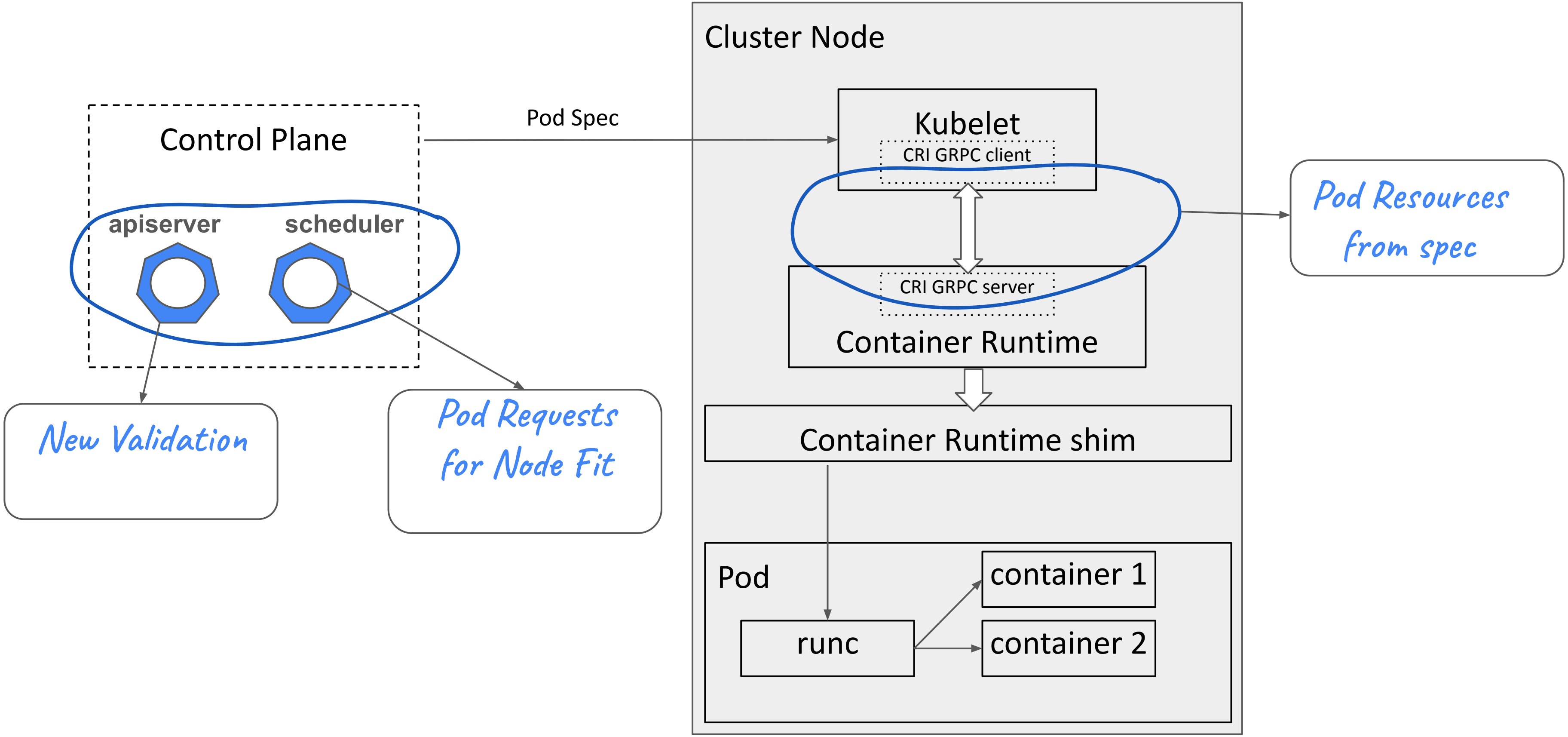
Pod Level Memory Limit = 100 Gi



Memory Request = 30 Gi

Memory Request = 20 Gi

Deep Dive: What has changed?



New Validation

Pod Requests for Node Fit

Pod Resources from spec

Deep Dive: What has changed?

- QoS Determination
  - Priotizes pod-level resources spec.
- OOM Killer
  - OOM Score Adjustment formula accounts for pod-level requests.

Deep Dive: What has changed?

$$oomScoreAdjust = 1000 - \left[1000 \times \frac{memoryRequest}{memoryCapacity}\right]$$

$$\Big\downarrow$$

$$oomScoreAdjust = 1000 - \left[1000 \times \frac{containerMemoryRequest + remainingPodMemRequestPerContainer}{memoryCapacity}\right]$$

$$remainingPodMemRequestPerContainer = \frac{\left[PodRequest - \sum(containerMemoryRequests)\right]}{no.ofcontainers}$$

Common Questions & Misconceptions

- Why are we doing this?
- Can I use pod-level and container-level specs together?
- Will this affect my monitoring tools?

- Upcoming changes in Beta
  - Topology, CPU, memory manager
  - Eviction manager
  - In-place Pod resize

The Future of Pod Power

Current State

- Alpha in 1.32

Support for features in 1.33

- Huge pages
- Topology Manager
- Memory Manager
- CPU Manager

Upcoming KEP support

- Other KEPs: In place pod resize
- Dynamic containers

Community Feedback and Collaboration

- Join the Pod Power Revolution!
- Planning Beta in 1.33

Call to Action: Try it out and provide feedback, share use cases, contribute.

- KEP-2837
- Reach out to SIG Node

Beta support: Topology Manager, Autoscaler, InPlace Pod Resize, etc.

THANK YOU