SIG Node Intro and Deep Dive

Paco Xu, DaoCloud
Xiongxiong Yuan, JiHu(GitLab)

Intro us

Paco Xu

DaoCloud Shanghai.

Mainly worked on kubeadm & sig-node

Github: pacoxu

Twitter: xu_paco

❤️ → ⚽ & PUBG

Yuan xiongxiong

极狐 GITLAB Beijing.

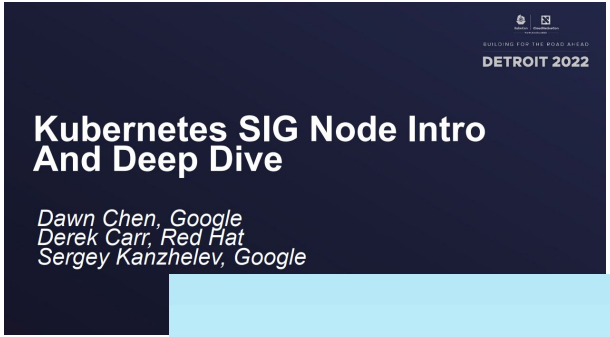
Mainly worked on helm & helmfile & dragonly & sig-node

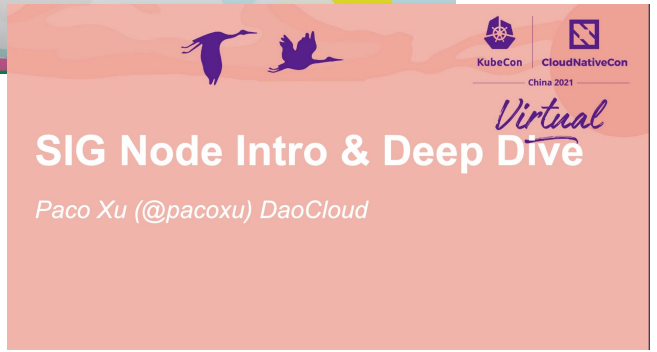
Github: yxxhero

❤️ → Arena Of Valor

Previous updates from SIG Node

- 2023 Q2 KubeCon EU(Virtual)
  - Recording
  - Sched: April 21st, 2023
- 2022 Q4 KubeCon Detroit
  - Recording
  - Sched: October 28th, 2022
  - Slides
- 2021 Kubecon China(Virtual)
  - Recording in **Chinese**
  - Sched: December 9th, 2021





Content

Agenda

01 SIG Node Intro

02 Feature updates in 1.28

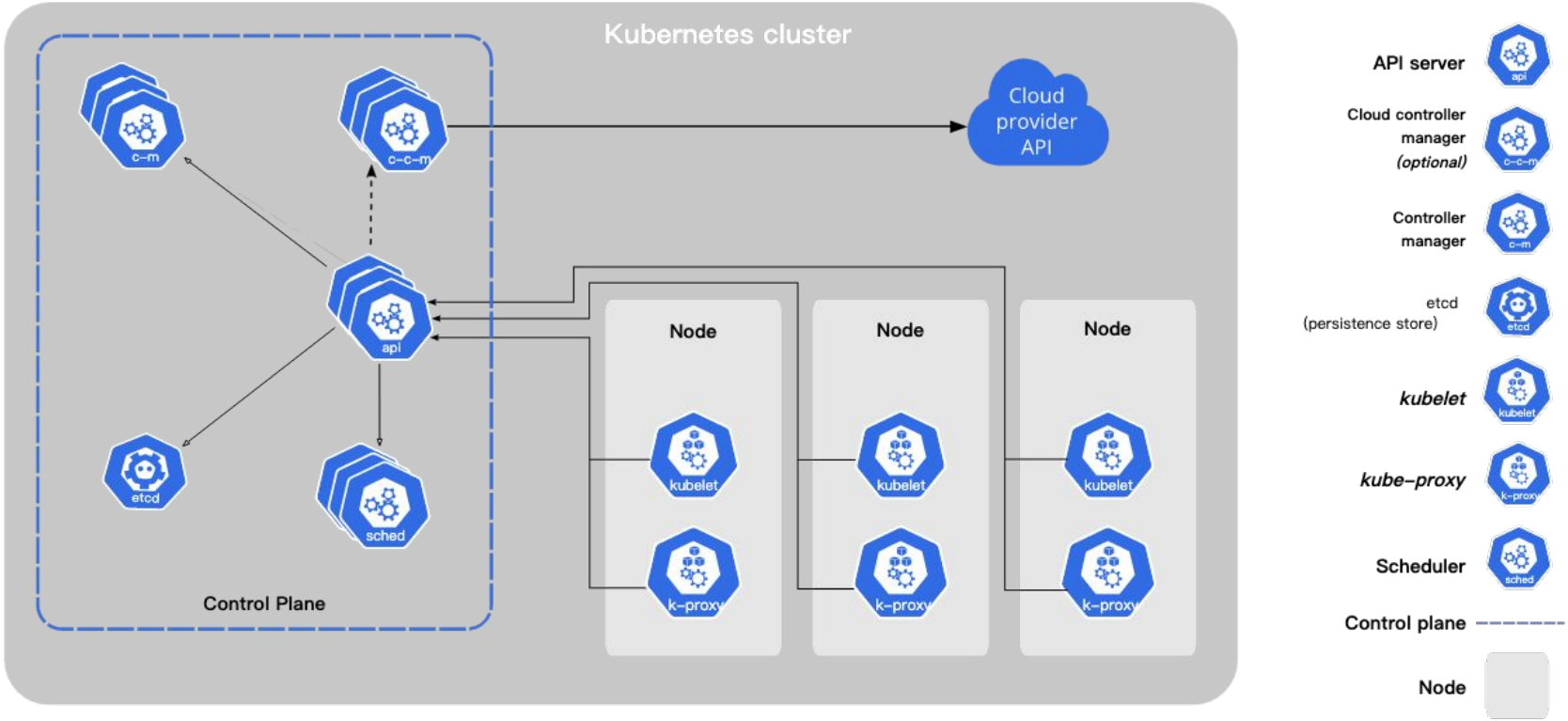
- *Swap*、*DRA*、*VPA*

03 Highlight: PLEG

04 Get involved

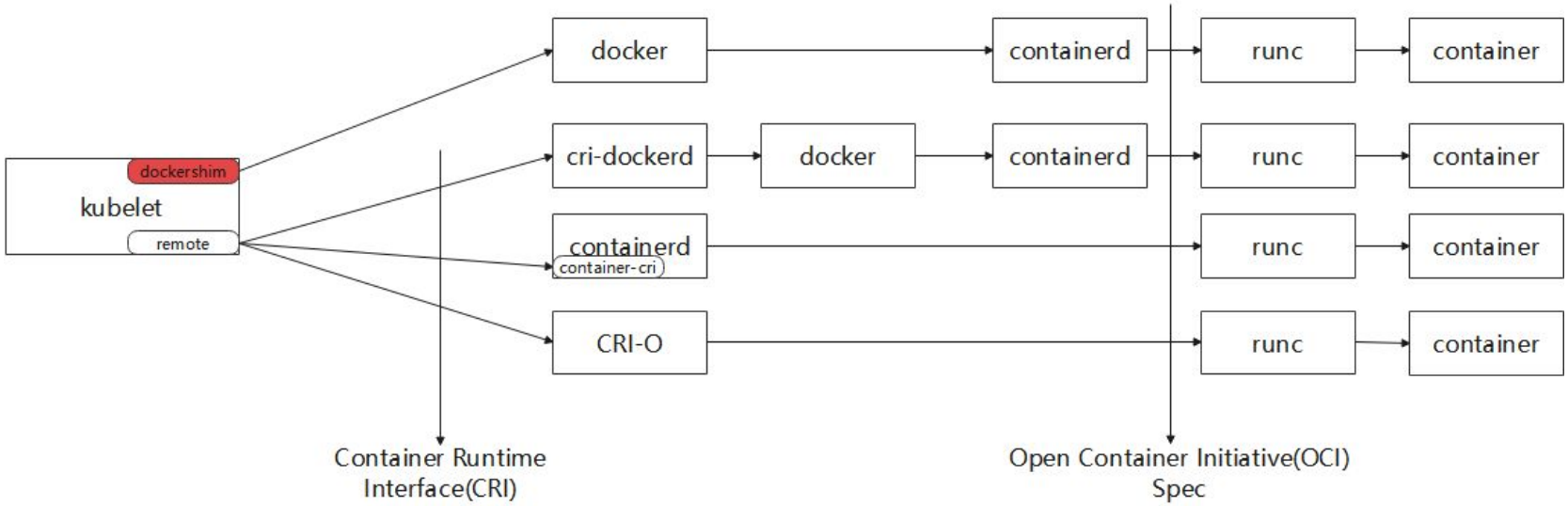
SIG Node Intro

Kubernetes Components



Components on a Node

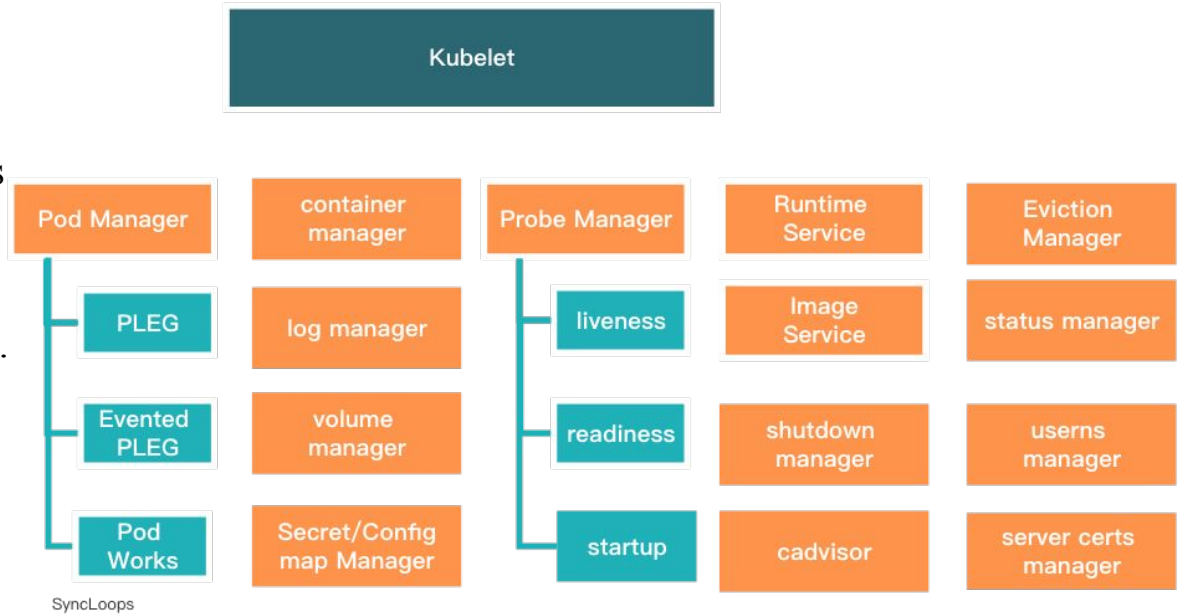
1. kubelet
2. container runtime



Kubelet Concept

An **agent** that runs on each node in the cluster. It makes sure that containers are **running in a Pod**.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are **running** and **healthy**. The kubelet doesn't manage containers which were not created by Kubernetes.



Node ⟷ Control Plane

Node to Control Plane

→ API Server is the only externally accessible point of the control plane.

It defaults to self-registration.

Control plane to node

→ API server to kubelet

For example：logs/exec

Self-registration of Nodes

For self-registration, the kubelet is started with the following options:

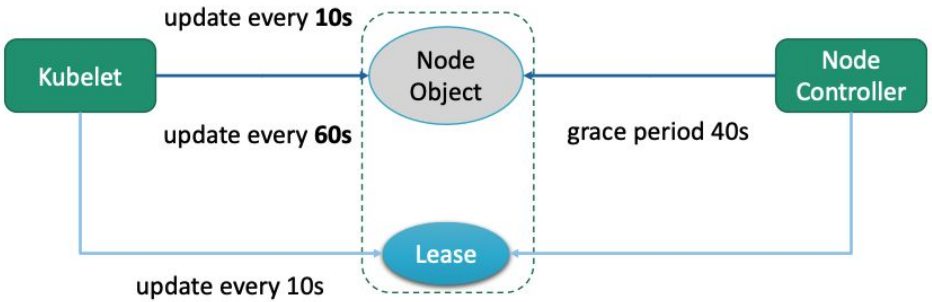
- `--kubeconfig` - Path to credentials to authenticate itself to the API server.
- `--cloud-provider` - How to talk to a cloud provider to read metadata about itself.
- `--register-node` - Automatically register with the API server.
- `--register-with-taints` - Register the node with the given list of taints (comma separated `<key>=<value>:<effect>`). No-op if `register-node` is false.
- `--node-ip` - IP address of the node.
- `--node-labels` - Labels to add when registering the node in the cluster (see label restrictions enforced by the NodeRestriction admission plugin).
- `--node-status-update-frequency` - Specifies how often kubelet posts its node status to the API server.

Heartbeats

For nodes there are two forms of heartbeats:

- updates to the `.status` of a Node
- Lease objects within the `kube-node-lease` namespace. Each Node has an associated Lease object.

● Add a new `Lease` build-in API

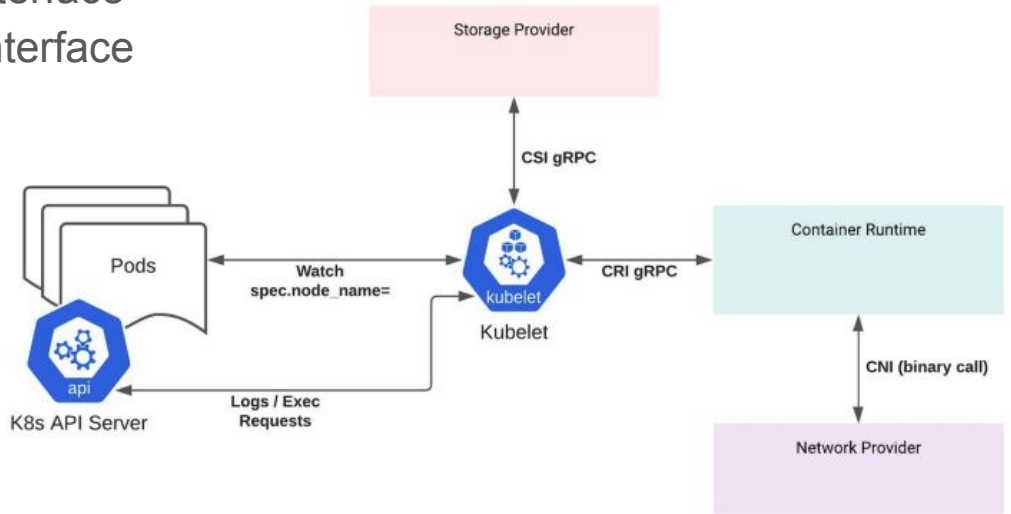


Lease objects ~100 bytes

Enabled @ 1.14

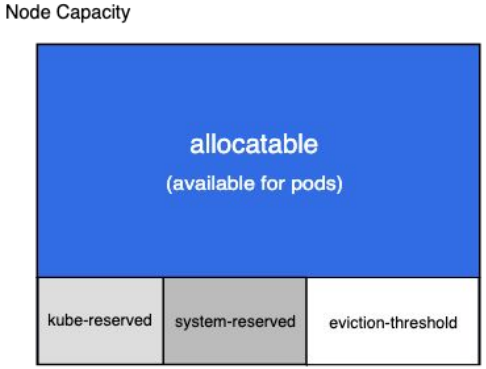
Interfaces：CSI CRI CNI

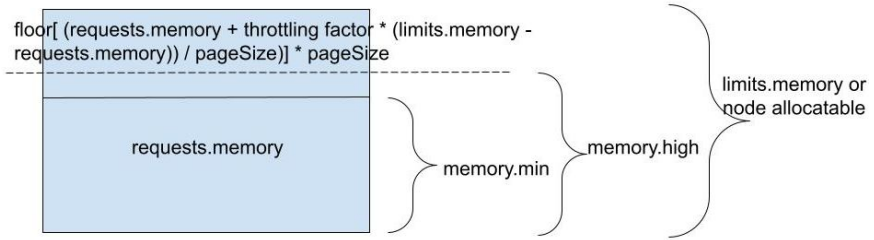
1. CRI: Container Runtime Interface
2. CSI: Container Storage Interface
3. CNI: Container Network Interface



Resource Management

- CPU
  - and NUMA nodes, GPU
- Memory
  - and HugePages, Swap
- Disk
- Ephemeral storage
- PIDs
- Devices





Eviction & Exit

What we should take care of?

- OOM-Killer
- kubelet eviction
- liveness probe
- process exit
- kubelet/container runtime restart
- node graceful shutdown

Feature updates in 1.28

SIG Node 1.28 retro

1.22 with 24 tracked and 13 merged
1.23 with 14 tracked and 8 merged
1.24 with 23 tracked and 6 merged
1.25 with 16 tracked and 10 merged
1.26 with 11 tracked and 8 merged
1.27 with 19 tracked and 13 merged
1.28 with 31 tracked and 17 merged

Things could have gone better:

- **Not good at merging code earlier**
- **Not too much review power**
- **Some big KEPs defer problems to beta**
- **split a big KEP into smaller ones?**
- **Progress not clear if there is no owner on a KEP**

Things that went well:

- **Memory QoS test: caught issues and prevent promotion**
- **Pushing the container requirements down to runtime**
- **Refactor on test infra went pretty well, moved to CNCF**

1.28 KEP Updates

Alpha:

- Sidecar containers
- Dynamic Resource Allocation
- Add CDI devices to device plugin API
- Discover cgroup driver from CRI
- Add support for a drop-in kubelet configuration directory
- Field status.hostIPs added for Pod
- Support User Namespaces
- Pod conditions around readiness to start containers after completion of pod sandbox creation

Beta

- Node memory swap support beta1(by default disabled)
- Improve multi-NUMA alignment in topology manager
- Retriable and non-retriable Pod failures for Jobs

GA

- graduate the kubelet podresources endpoint to GA
- Support Oldest Node With Newest Control Plane
- Extend podresources API to report allocatable resources
- Non-graceful node shutdown
- Add configurable grace period to probes
- Support 3rd party device monitoring plugins

1.28 Postponed KEPs

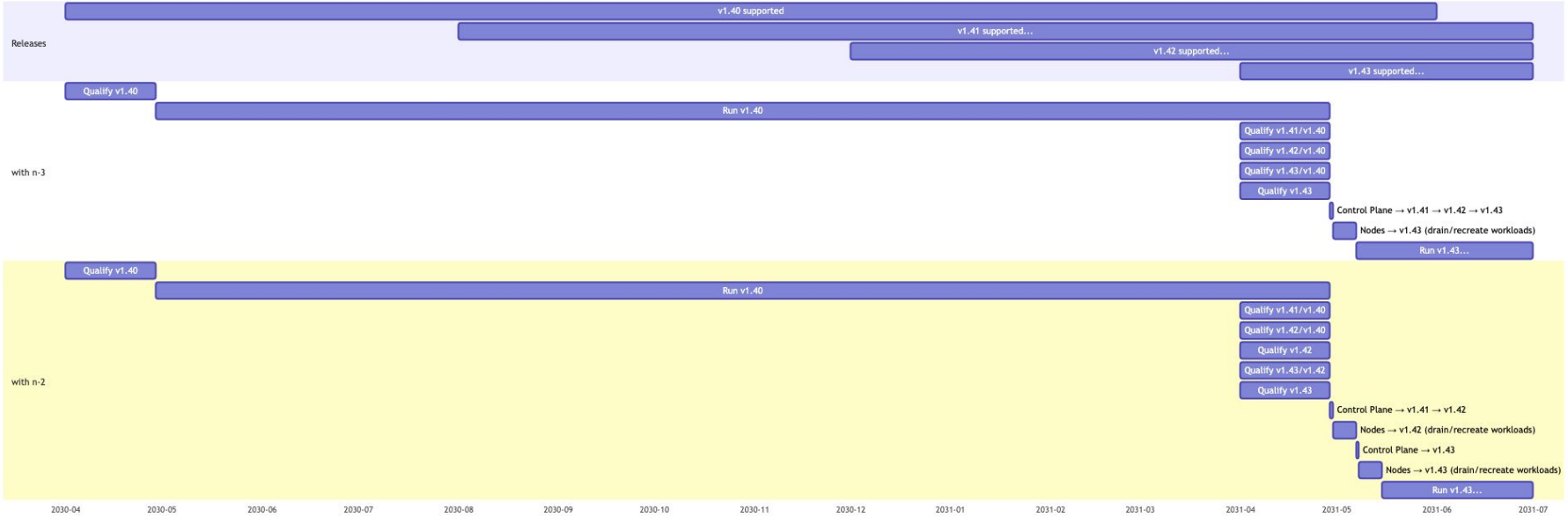
- kubelet Resource Metrics Endpoint Stay beta(GA in v1.29 **merged**)
- Cloud Dual-Stack --node-ip Handling Stay alpha (beta in v1.29 **merged**)
- Support memory qos with cgroups v2 Stay alpha(blocking issue found)
- Limit on parallel image pull Stay alpha: needs more e2e tests
- In-Place Update of Pod Resources Stay alpha (Pending on Windows node support)
- cAdvisor-less, CRI-full Container and Pod Stats Stay alpha
- Fine-grained SupplementalGroups control Stay alpha
- Split stdout and stderr log stream PR WIP
- Ensure secret pulled images PR in Review
- Introducing Sleep Action for PreStop Hook PR in Review
- Decouple TaintManager from NodeLifecycleController PR in Review
- Add a new field maxRestartTimes to podSpec when running into RestartPolicyOnFailure KEP in Review
- Sub-second / More granular probes KEP in Review
- QoS-class resources KEP in Review

Node Skew: Annual Node Upgrade

The `kube-apiserver` instances the `kubelet` communicates with are at **1.28**

Optionally upgrade `kubelet` instances to **1.28** (or they can be left at **1.27**, **1.26**, or **1.25**)

[#KEP-3935-oldest-node-newest-control-plane]



Swap only supports cgroup v2 since v1.28

Backgrounds：

- Blog: Clarification of Common Misconceptions About Swap ✨✨

Implement History

- Alpha 1.22 with Feature gate `*NodeSwap*`
- Beta1 1.28 but by default disabled
  - Only support cgroup v2（v1.28+）
  - Only Burstable Pods can use Swap

KEP 2400

- Swap Support in Kubernetes

User Namespace Support

- v1.25: introduces a new field in *pod.Spec* called **hostUsers**, which allows users to enable or disable user namespaces for container groups.
- v1.27: Support is limited to **stateless** Pods only.
- v1.28: Support is extended to Pods with **volumes**, and the feature gate name is changed from UserNamespacesStatelessPodsSupport to UserNamespacesSupport.
- v1.29 (WIP): There are plans to integrate User Namespace with **Pod Security**. If the user namespace feature is enabled, certain security restrictions can be relaxed.

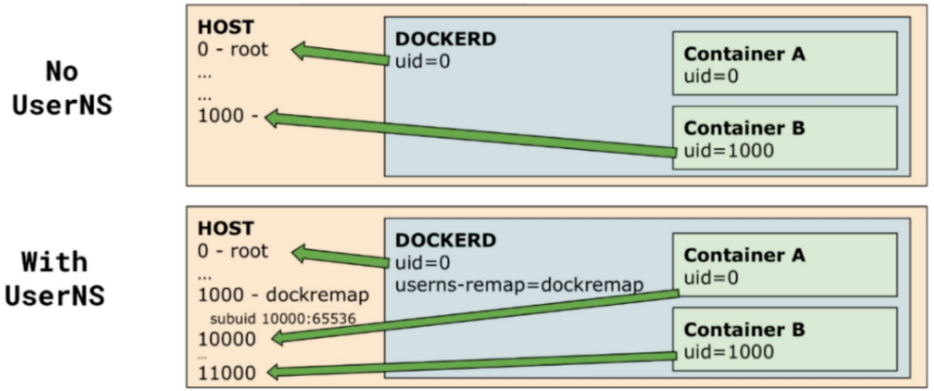
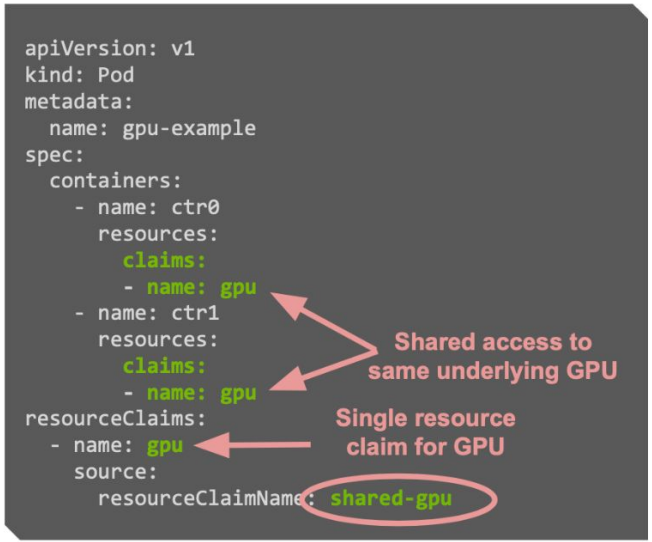


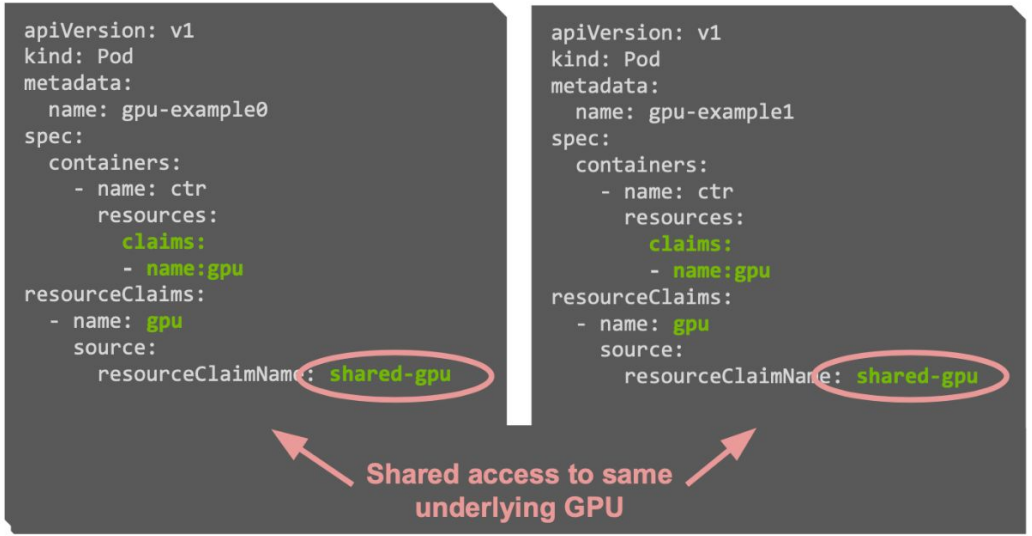
Image resource: https://medium.com/@flavienb/installing-and-securing-docker-rootless-for-production-use-8e358d1c0956

KEP-3063: Dynamic resource allocation

The new API is more flexible than Kubernetes' **Device Plugins** functionality because it allows Pods to request special types of resources, which can be provided at the node-level, cluster-level, or in other ways as configured by the user.

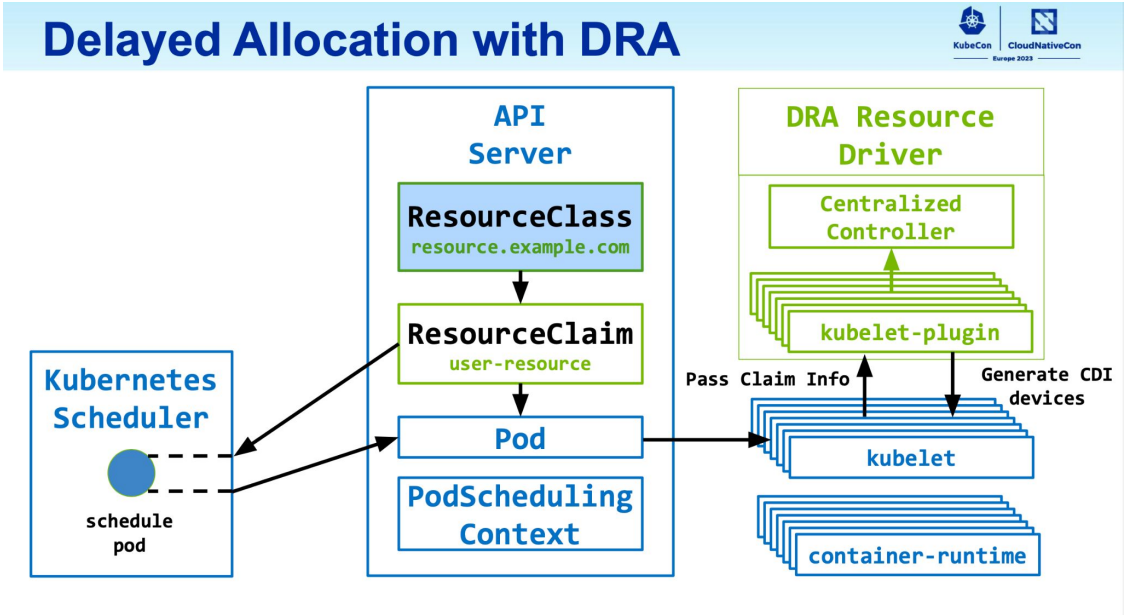
- Introduced in v1.26 as v1alpha1; updated to v1alpha2 in v1.27; and planned to be beta in v1.29.
- In v1.28, significant optimizations and changes were made in scheduling, including improvements for scenarios with unallocated or unreserved resources





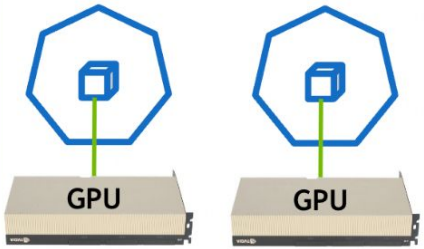
Details about DRA

From: Kubecon EU v1.26/v1.27 update



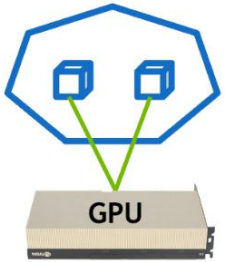
Build your own DRA driver

gpu-test1.yaml



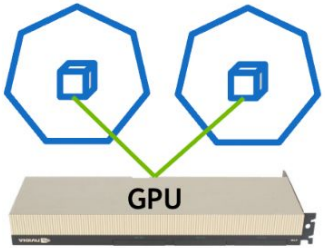
- 2 pods
- 1 container each
- 1 GPU per container

gpu-test2.yaml



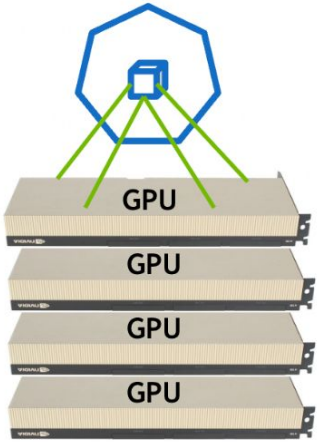
- 1 pod
- 2 containers
- 1 shared GPU

gpu-test3.yaml



- 2 pods
- 1 container each
- 1 shared GPU

gpu-test4.yaml



- 1 pod
- 1 container
- 1 claim
- 4 GPUs per claim

https://github.com/kubernetes-sigs/dra-example-driver

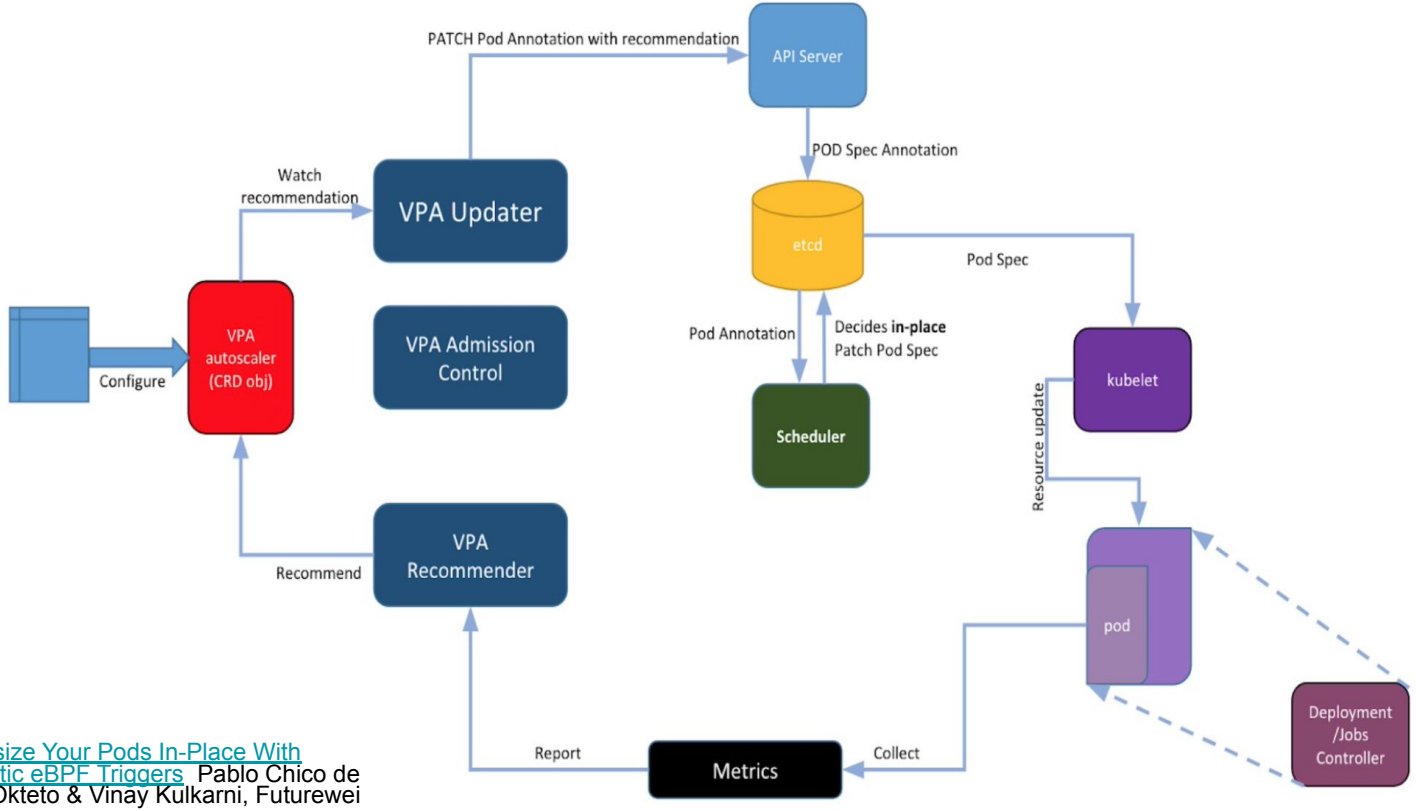
In-Place Resource Resize

It is typically used with VPA (Vertical Pod Autoscaling).

- In v1.25, CRI supports for adjusting Pod resource reservations and limits.
- In v1.27, an Alpha implementation was introduced.
- Originally planned for v1.28, Windows support has been postponed to v1.29.
- Additionally, ByteDance works on VPA combined with core-binding in v1.29.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: qos-demo-5
spec:
  restartPolicy: Never
  containers:
  - name: qos-demo-ctr-5
    image: docker.m.daocloud.io/library/nginx:1.14.1
    resizePolicy:
    - resourceName: cpu
      restartPolicy: NotRequired
    - resourceName: memory
      restartPolicy: RestartContainer
    resources:
      limits:
        memory: "200Mi"
        cpu: "700m"
      requests:
        memory: "200Mi"
        cpu: "700m"
```

Vertical Pod Autoscaler



Demo: [Resize Your Pods In-Place With Deterministic eBPF Triggers](#) Pablo Chico de Guzman, Okteto & Vinay Kulkarni, Futurewei Technologies

VPA vs HPA

Advantages of VPA Compared to HPA:

- No restarts/in-place updates.
- Faster scaling response:
  - New Pods spawned by HPA require cold starts.
  - VPA directly modifies resource limits, achieving scaling goals more directly.
- Better at addressing resource wastage issues.

```
spec:
  containers:
  - args:
    - -c
    - while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    command:
    - /bin/sh
    image: registry.k8s.io/ubuntu-slim:0.1
    imagePullPolicy: IfNotPresent
    name: hamster
    resizePolicy:
    - resourceName: cpu
      restartPolicy: NotRequired
    - resourceName: memory
      restartPolicy: NotRequired
    resources:
      requests:
        cpu: 587m
        memory: 262144k
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-79z8m
      readOnly: true
```

不重启

```
[root@daocloud vertical-pod-autoscaler]# kubectl describe vpa
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  <none>
API Version:  autoscaling.k8s.io/v1
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2023-07-28T05:20:25Z
  Generation:          10
  Resource Version:    194649
  UID:                 bebc2de8-d30c-41e9-88d6-558c85038a37
Spec:
  Resource Policy:
    Container Policies:
      Container Name:  *
      Controlled Resources:
        cpu
        memory
      Max Allowed:
        Cpu:     1
        Memory:  500Mi
      Min Allowed:
        Cpu:     100m
        Memory:  50Mi
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
  Update Policy:
    Update Mode:  Auto
Status:
  Conditions:
    Last Transition Time:  2023-07-28T05:17:34Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     423m
        Memory:  262144k
      Target:
        Cpu:     627m
        Memory:  262144k
      Uncapped Target:
        Cpu:     627m
        Memory:  262144k
      Upper Bound:
        Cpu:     1
        Memory:  500Mi
```

PLEG

- the most critical piece

PLEG

1. What is PLEG?
2. Generic PLEG
3. Evented PLEG
4. Comparision
5. Troubleshooting

Why we need PLEG?

- Kubelet Overview:
  - Kubelet is a per-node daemon in Kubernetes.
  - It manages Pods on nodes, ensuring their status matches their Pod Spec.
- Challenges with Kubelet:
  - Kubelet needs to react to changes in Pod Specs and container states.
  - It observes changes in Pod Specs.
  - It periodically polls the container runtime (e.g., every 10 seconds) for the latest status of all containers.
  - Increased pod/container count leads to **performance** overhead.
  - Parallel processing intensifies this issue, with each pod having a separate goroutine for querying the CR.
  - Periodic, concurrent, and high request loads result in elevated CPU usage (even without Spec/Status changes), performance issues, and reliability problems due to an overwhelmed container runtime.
  - Ultimately, it limits Kubelet's scalability.
- Introducing PLEG (Pod Lifecycle Event Generator):
  - PLEG addresses these challenges.
  - It monitors and reacts to changes in Pod lifecycles efficiently.
  - PLEG significantly reduces the need for periodic polling.
  - It enhances Kubelet's scalability and overall performance.

How?

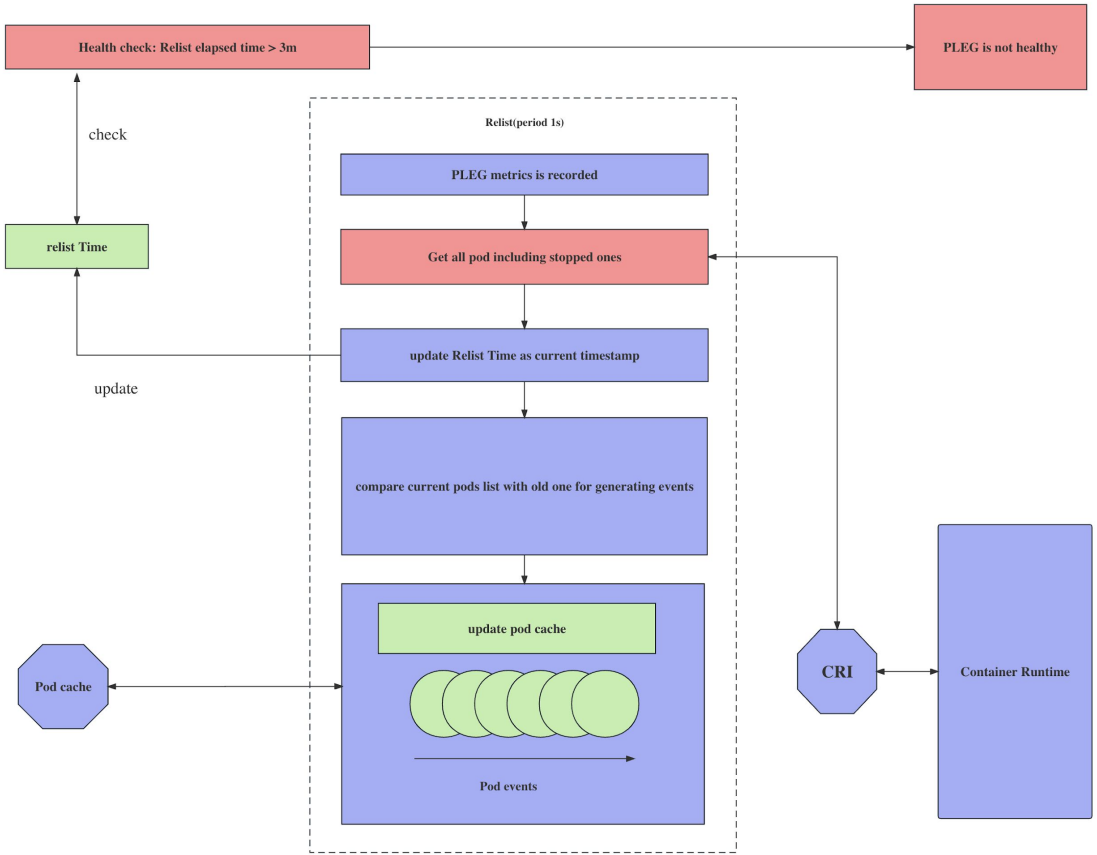
PLEG, short for Pod Lifecycle Event Generator, is responsible for generating events related to the lifecycle of pods.

- Core Logic:
  - querying information from the container runtime that belongs to containers/sandboxes
  - compares this information with its own maintained pods
- Concurrency：Executes with a single thread, avoiding simultaneous access to the container runtime by all pod worker threads.
- Efficiency: Only relevant Pod Workers are awakened for synchronization.

PLEG improves performance through the following two methods:

1. Reducing unnecessary work during inactive periods, when there are no spec/status changes.
2. Lowering concurrent requests to the container runtime.

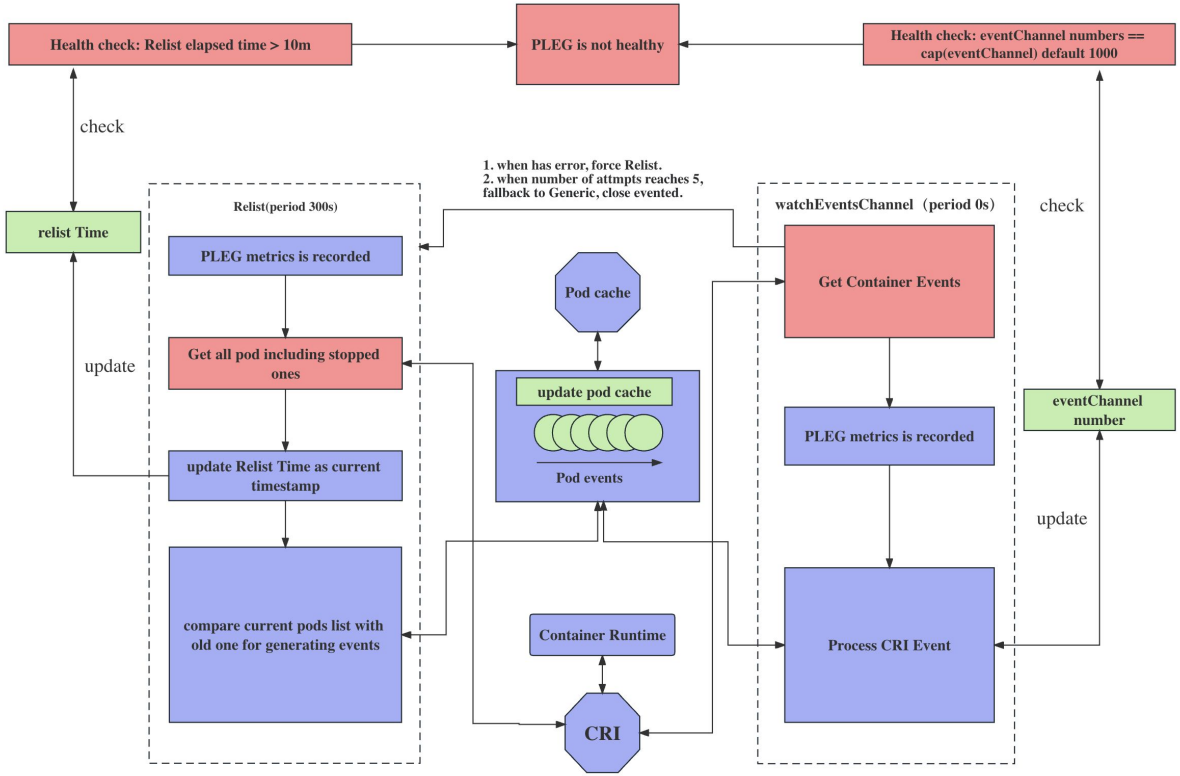
Generic PLEG



Generic PLEG is the initial implementation of PLEG, with its key logic residing in the 'relist' function.

- **'relist' function** runs at a 1-second interval, querying the list of pods/containers running in the container runtime. It compares this information with the internal pods/containers cache and generates events accordingly.
- kubelet performs health checks using the registered **'healthy' function**, and if it exceeds 3 minutes, it prompts us to address the commonly encountered 'not healthy' issue.

Evented PLEG



Generic will still start, with the difference inteval and threshold.

*genericPlegRelistPeriod = time.Second * 1*
*genericPlegRelistThreshold = time.Minute * 3*

*eventedPlegRelistPeriod = time.Second * 300*
*eventedPlegRelistThreshold = time.Minute * 10*

http://kep.k8s.io/3386 alpha: v1.25 beta: v1.27

If enabled, Generic relisting will be forced in the following two exceptions:

1. When obtaining container events from CRI fails.
2. After 5 retries, it will fall back entirely to Generic

Comparision

Generic

- Simple
- Guarantees consistency
- Performance is acceptable
- Works with any runtime

Evented

- Streaming and eventing
- Faster detection
- Improved resource utilization
- Modern runtimes are needed

Why is Evented PLEG better?

- Evented + Relist: Reduce unnecessary work during inactivity(no event)
- Modern container runtime: higher performance

PLEG Troubleshooting

"PLEG is not healthy" can happen due to various causes.

1. Container runtime latency or timeout (performance degradation, deadlock, bugs) during remote requests.
2.Too many running pods for host resources or too many running pods on high-spec hosts to complete the relist within 3 minutes. Events and latency are proportional to the pod numbers regardless of host resources.
3.Bugs in Kubernetes.

Solutions:

1. Kill containers which are hanging or restart the container runtime, and if necessary, reboot the node.
2. Set `maxPods` to be more lower number and restart the kubelet.
3. Foucs on kubuernetes on Github.

Get Involved

Contribution Priorities

- Stability first!
  - Tests > bug fixes/open issues > features
  - Test infrastructure monitoring and health
- Improve the user and developer experience
  - Documentation
  - Enhance logging and metrics
  - Keeping on top of PRs/issues

How to contribute

- Attend our SIG meetings! 🤝
  - SIG meetings cover features, KEPs, and more
  - CI/Triage meetings are a smaller, hands-on group
    - Good setting for learning how to triage, improve CI
- Participate in reviews, issues, and docs! 🙋💁‍♂️
  - Triage Guide
  - Main Node Board / CI and Test Enhancements Board
- Adopt new features and give feedback!🎁
  - Testing in real environments is critical for graduating features

Where to find us?

- SIG Meetings:
  - Regular meeting, weekly on Tuesdays at 1am CST
  - CI/Triage meeting, weekly on Wednesdays at 1am CST
- Slack channel: #sig-node
- Mailing list: kubernetes-sig-node
- Chair
  - Sergey Kanzhelev (@SergeyKanzhelev), Google
  - Mrunal Patel (@mrunalp), Red Hat
- Technical Leads
  - Dawn Chen (@dchen1107), Google
  - Derek Carr (@derekwaynecarr), Red Hat
  - Mrunal Patel (@mrunalp), Red Hat

What's More

Kubelet & Scheduler

1. kubelet update node capacity and node allocatable status
2. scheduler set pod nodeName
3. kubelet start the pod that has the same nodeName and re-calculate the allocatable status.

- Node Label
- Node Taint
- Resource Management
  - CPU
  - Memory
  - Ephemeral Storage

cgroups v2 vs cgroup v1

- API 中单个统一的层次结构设计
- 为容器提供更安全的子树委派能力
- 压力阻塞信息[4]等新功能
- 增强的资源分配管理和跨多个资源的隔离
  - 统一核算不同类型的内存分配(网络和内核内存等)
  - 考虑非即时资源更改，例如页面缓存回写