Kubernetes v1.33: Octarine

kubernetes

Nina Polshakova
v1.33 Release Lead

Dipesh Rawat
v1.33 Enhancements Lead

Ryota Sawada
v1.33 Comms Lead

Kubernetes v1.33 Release Team

Release Lead & Shadows
Nina Polshakova
Vyom Yadav
Matteo Bianchi
Rashan Smith
Sanchita Mishra
Daniel Chan

Emeritus Advisor
Angelos Kolaitis

Release Team Subproject Lead
Kat Cosgrove
Grace Nguyen

Branch Management
Jim Angel
Matteo Bianchi

Enhancements
Dipesh Rawat
Faeka Ansari
Eunji Jung
Arka Saha
Jenny Shu
Lauren Zung

Release Signal
Wendy Ha
Rajalakshmi Girish
Nitish Kumar
Sean McGinnis
Elieser Pereira
ChengHao Yang

Docs
Rayan Das
Shedrack Akintayo
Urvashi Choubey
Michelle Nguyen
Arvind Parekh
Mélony Qin
Sreeram Venkitesh

Comms
Ryota Sawada
Agustina Barbetta
Aakanksha Bhende
Udi Hofesh
Sneha Yadav

Kubernetes v1.33: Octarine

Kubernetes v1.33: Octarine

The theme for Kubernetes v1.33 is **Octarine: The Color of Magic**, inspired by Terry Pratchett's *Discworld* series.
This release highlights the open source magic that Kubernetes enables across the ecosystem.

"It's still magic even if you know how it's done." – Sir Terry Pratchett

Summary of Enhancements

64 Enhancements in v1.33

- 24 enhancements to **Alpha**
- 20 enhancements to **Beta**
- 18 enhancements to **Stable**
- 2 deprecated / withdrawn

For the complete list of enhancements:

http://bit.ly/k8s133-enhancements

Release Highlights: Stable

SIG-Network

Multiple Service CIDRs

This enhancement introduced a new implementation of allocation logic for Service IPs. Across the whole cluster, every Service of `type: ClusterIP` must have a unique IP address assigned to it. Trying to create a Service with a specific cluster IP that has already been allocated will return an error. The updated IP address allocator logic uses two newly stable API objects: `ServiceCIDR` and `IPAddress`. Now generally available, these APIs allow cluster administrators to dynamically increase the number of IP addresses available for `type: ClusterIP` Services (by creating new ServiceCIDR objects).

Status: Stable

- Alpha: v1.28
- Beta: v1.29
- Stable: v1.33

KEP

KEP-1880: Multiple Service CIDRs
(features.k8s.io/1880)

nftables backend for kube-proxy

The `nftables` backend for kube-proxy is now stable, adding a new implementation that significantly improves performance and scalability for Services implementation within Kubernetes clusters. For compatibility reasons, `iptables` remains the default on Linux nodes. Check the migration guide if you want to try it out.

Status: Stable

- Alpha: v1.29
- Beta: v1.31
- Stable: v1.33

KEP

KEP-3866: nftables kube-proxy backend
(features.k8s.io/3866)

Topology aware routing with `trafficDistribution: PreferClose`

This release graduates topology-aware routing and traffic distribution to GA, which would allow us to optimize service traffic in multi-zone clusters. The topology-aware hints in EndpointSlices would enable components like kube-proxy to prioritize routing traffic to endpoints within the same zone, thereby reducing latency and cross-zone data transfer costs. Building upon this, `trafficDistribution` field is added to the Service specification, with the `PreferClose` option directing traffic to the nearest available endpoints based on network topology. This configuration enhances performance and cost-efficiency by minimizing inter-zone communication.

Status: Stable

- Alpha: v1.21
- Beta: v1.23
- Stable: v1.33

(For KEP-2433)

KEPs

- KEP-4444: Traffic Distribution for Services (features.k8s.io/4444)
- KEP-2433: Topology Aware Routing (features.k8s.io/2433)

SIG-Apps

Backoff limits per index for indexed Jobs

This release graduates a feature that allows setting backoff limits on a per-index basis for Indexed Jobs. Traditionally, the `backoffLimit` parameter in Kubernetes Jobs specifies the number of retries before considering the entire Job as failed. This enhancement allows each index within an Indexed Job to have its own backoff limit, providing more granular control over retry behavior for individual tasks. This ensures that the failure of specific indices does not prematurely terminate the entire Job, allowing the other indices to continue processing independently.

Status: Stable

- Alpha: v1.28
- Beta: v1.29
- Stable: v1.33

KEP

KEP-3850: Backoff Limit Per Index For Indexed Jobs (features.k8s.io/3850)

Job success policy

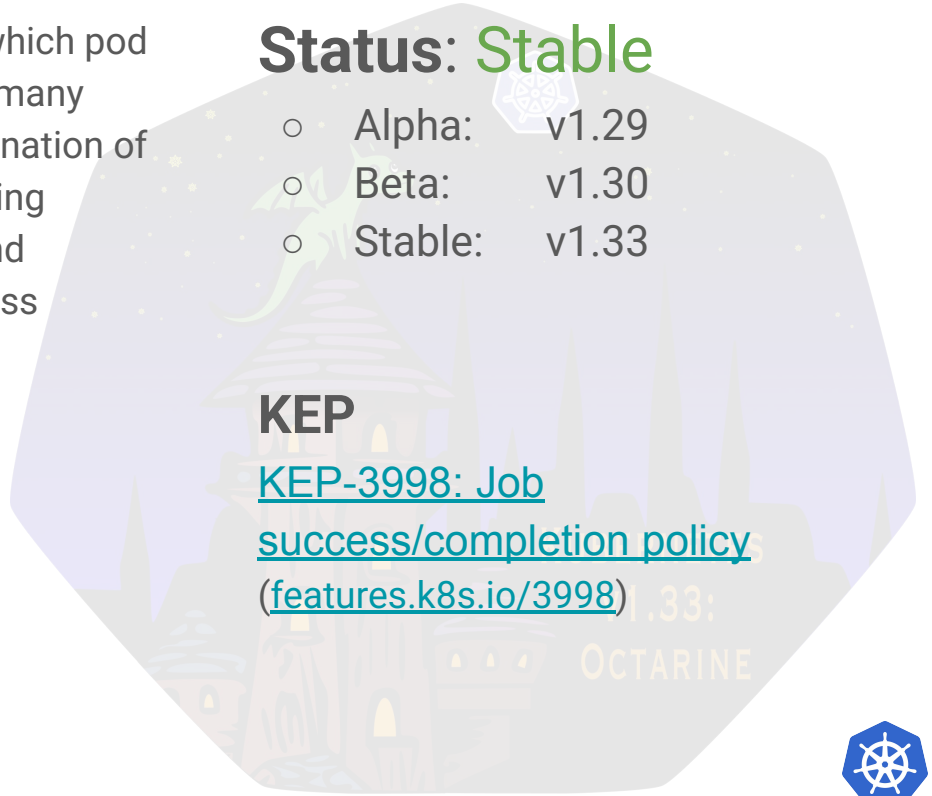
Using `.spec.successPolicy`, users can specify which pod indexes must succeed (`succeededIndexes`), how many pods must succeed (`succeededCount`), or a combination of both. This feature benefits various workloads, including simulations where partial completion is sufficient, and leader-worker patterns where only the leader's success determines the Job's overall outcome.

Status: Stable

- Alpha: v1.29
- Beta: v1.30
- Stable: v1.33

KEP

KEP-3998: Job success/completion policy (features.k8s.io/3998)



SIG-Auth

Bound ServiceAccount token security improvements

This enhancement introduced features such as including a unique token identifier (i.e. JWT ID Claim, also known as JTI) and node information within the tokens, enabling more precise validation and auditing. Additionally, it supports node-specific restrictions, ensuring that tokens are only usable on designated nodes, thereby reducing the risk of token misuse and potential security breaches. These improvements, now generally available, aim to enhance the overall security posture of service account tokens within Kubernetes clusters.

Status: Stable

- Alpha: v1.30
- Beta: v1.31
- Stable: v1.33

KEP

KEP-4193: Bound service account token improvements (features.k8s.io/4193)

SIG-CLI

Subresource support in kubectl

The `--subresource` argument is now generally available for kubectl subcommands such as `get`, `patch`, `edit`, `apply` and `replace`, allowing users to fetch and update subresources for all resources that support them. To learn more about the subresources supported, visit the kubectl reference.

Status: Stable

- Alpha: v1.24
- Beta: v1.27
- Stable: v1.33

KEP

KEP-2590: Add subresource support to kubectl (features.k8s.io/2590)

SIG-Node

Sidecar containers

The sidecar pattern involves deploying separate auxiliary container(s) to handle extra capabilities in areas such as networking, logging, and metrics gathering. Sidecar containers graduate to stable in v1.33.

Kubernetes implements sidecars as a special class of init containers with `restartPolicy: Always`, ensuring that sidecars start before application containers, remain running throughout the pod's lifecycle, and terminate automatically after the main containers exit.

Status: Stable

- Alpha: v1.28
- Beta: v1.29
- Stable: v1.33

KEP

KEP-753: Sidecar Containers
(features.k8s.io/753)

Options to reject non SMT-aligned workload

This feature added policy options to the CPU Manager, enabling it to reject workloads that do not align with Simultaneous Multithreading (SMT) configurations. This enhancement, now generally available, ensures that when a pod requests exclusive use of CPU cores, the CPU Manager can enforce allocation of entire core pairs (comprising primary and sibling threads) on SMT-enabled systems, thereby preventing scenarios where workloads share CPU resources in unintended ways.

Status: Stable

- Alpha: v1.22
- Beta: v1.23
- Stable: v1.33

KEP

KEP-2625: node: cpumanager: add options to reject non SMT-aligned workload (features.k8s.io/2625)

SIG-Scheduling

Defining Pod affinity or anti-affinity using `matchLabelKeys` and `mismatchLabelKeys`

The `matchLabelKeys` and `mismatchLabelKeys` fields are available in Pod affinity terms, enabling users to finely control the scope where Pods are expected to co-exist (Affinity) or not (AntiAffinity). These newly stable options complement the existing `labelSelector` mechanism. The affinity fields facilitate enhanced scheduling for versatile rolling updates, as well as isolation of services managed by tools or controllers based on global configurations.

Status: Stable

- Alpha: v1.29
- Beta: v1.30
- Stable: v1.33

KEP

KEP-3633: Introduce MatchLabelKeys to Pod Affinity and Pod Anti Affinity (features.k8s.io/3633)

Considering taints and tolerations when calculating Pod topology spread skew

This enhanced `PodTopologySpread` by introducing two fields: `nodeAffinityPolicy` and `nodeTaintsPolicy`. These fields allow users to specify whether node affinity rules and node taints should be considered when calculating pod distribution across nodes. By default, `nodeAffinityPolicy` is set to `Honor`, meaning only nodes matching the pod's node affinity or selector are included in the distribution calculation. The `nodeTaintsPolicy` defaults to `Ignore`, indicating that node taints are not considered unless specified. This enhancement provides finer control over pod placement, ensuring that pods are scheduled on nodes that meet both affinity and taint toleration requirements, thereby preventing scenarios where pods remain pending due to unsatisfied constraints.

Status: Stable

- Alpha: v1.25
- Beta: v1.26
- Stable: v1.33

KEP

KEP-3094: Take taints/tolerations into consideration when calculating PodTopologySpread skew
(features.k8s.io/3094)

SIG-Storage

Volume populators

After being released as beta in v1.24, *volume populators* have graduated to GA in v1.33. This newly stable feature provides a way to allow users to pre-populate volumes with data from various sources, and not just from PersistentVolumeClaim (PVC) clones or volume snapshots. The mechanism relies on the `dataSourceRef` field within a PersistentVolumeClaim. This field offers more flexibility than the existing `dataSource` field, and allows for custom resources to be used as data sources.

Status: Stable

- Alpha: v1.18
- Beta: v1.23
- Stable: v1.33

KEP

KEP-1495: Generic data populators (features.k8s.io/1495)

Always honor PersistentVolume reclaim policy

This enhancement addressed an issue where the Persistent Volume (PV) reclaim policy is not consistently honored, leading to potential storage resource leaks. Specifically, if a PV is deleted before its associated Persistent Volume Claim (PVC), the "Delete" reclaim policy may not be executed, leaving the underlying storage assets intact. To mitigate this, Kubernetes now sets finalizers on relevant PVs, ensuring that the reclaim policy is enforced regardless of the deletion sequence. This enhancement prevents unintended retention of storage resources and maintains consistency in PV lifecycle management.

Status: Stable

- Alpha: v1.23
- Beta: v1.31
- Stable: v1.33

KEP

KEP-2644: Always Honor PersistentVolume Reclaim Policy (features.k8s.io/2644)

Deprecations & Removals

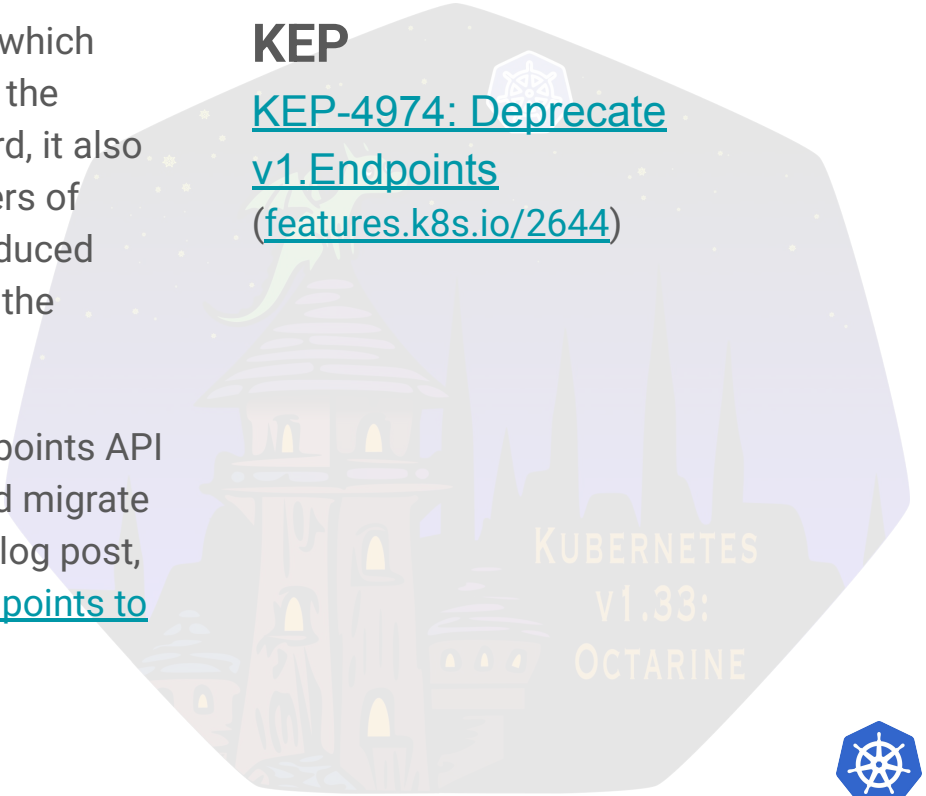
Deprecation of the stable Endpoints API

The EndpointSlices API has been stable since v1.21, which effectively replaced the original Endpoints API. While the original Endpoints API was simple and straightforward, it also posed some challenges when scaling to large numbers of network endpoints. The EndpointSlices API has introduced new features such as dual-stack networking, making the original Endpoints API ready for deprecation.

This deprecation affects only those who use the Endpoints API directly from workloads or scripts; these users should migrate to use EndpointSlices instead. There is a dedicated blog post, Kubernetes v1.33: Continuing the transition from Endpoints to EndpointSlices, with more details on the deprecation implications and migration plans.

KEP

KEP-4974: Deprecate v1.Endpoints (features.k8s.io/2644)



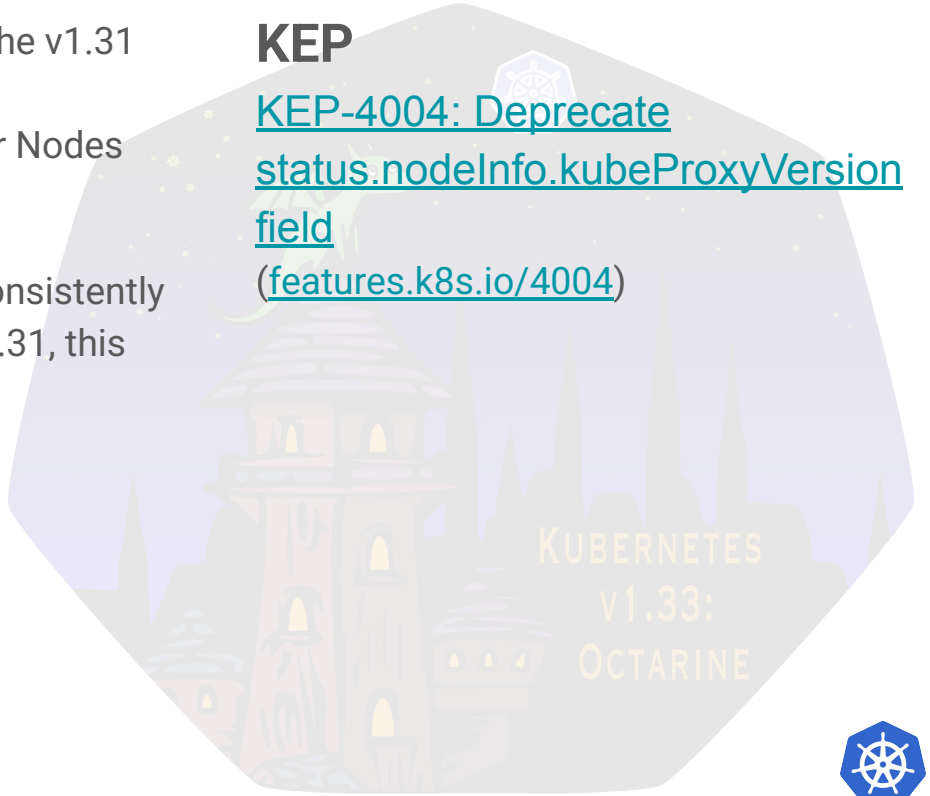
Removal of kube-proxy version information in node status

Following its deprecation in v1.31, as highlighted in the v1.31 release announcement, the `.status.nodeInfo.kubeProxyVersion` field for Nodes was removed in v1.33.

This field was set by kubelet, but its value was not consistently accurate. As it has been disabled by default since v1.31, this field has been removed entirely in v1.33.

KEP

KEP-4004: Deprecate status.nodeInfo.kubeProxyVersion field
(features.k8s.io/4004)



Removal of in-tree gitRepo volume driver

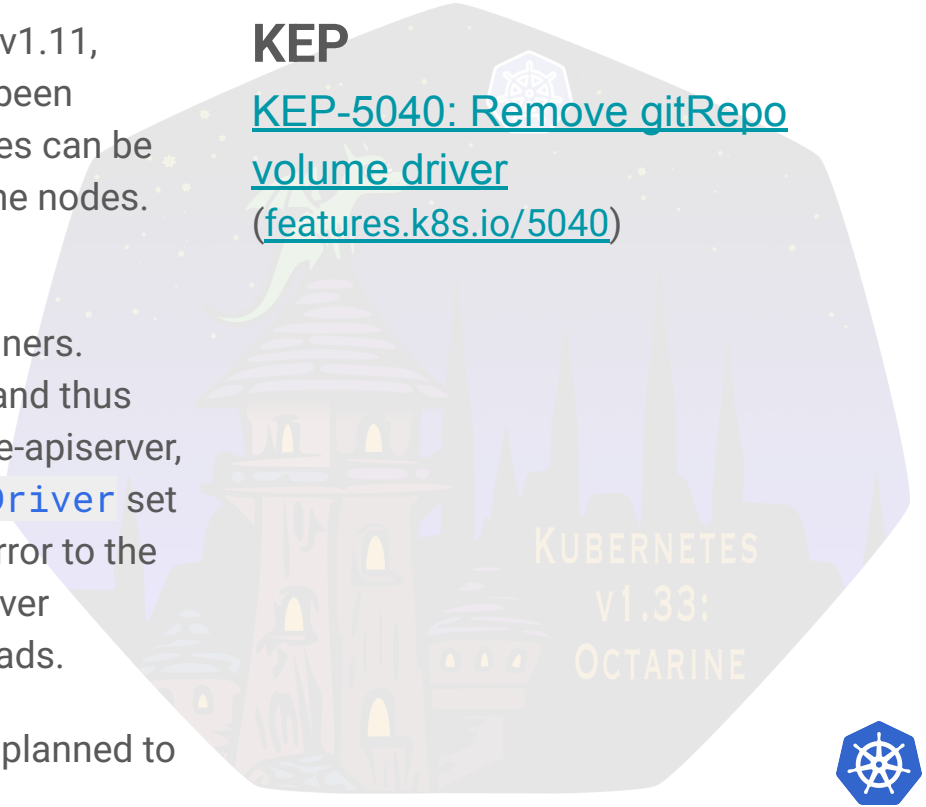
The gitRepo volume type has been deprecated since v1.11, nearly 7 years ago. Since its deprecation, there have been security concerns, including how gitRepo volume types can be exploited to gain remote code execution as root on the nodes. In v1.33, the in-tree driver code is removed.

There are alternatives such as git-sync and initContainers. `gitVolumes` in the Kubernetes API is not removed, and thus pods with `gitRepo` volumes will be admitted by kube-apiserver, but kubelets with the feature-gate `GitRepoVolumeDriver` set to false will not run them and return an appropriate error to the user. This allows users to opt-in to re-enabling the driver for 3 versions to give them enough time to fix workloads.

The feature gate in kubelet and in-tree plugin code is planned to be removed in the v1.39 release.

KEP

KEP-5040: Remove gitRepo volume driver (features.k8s.io/5040)



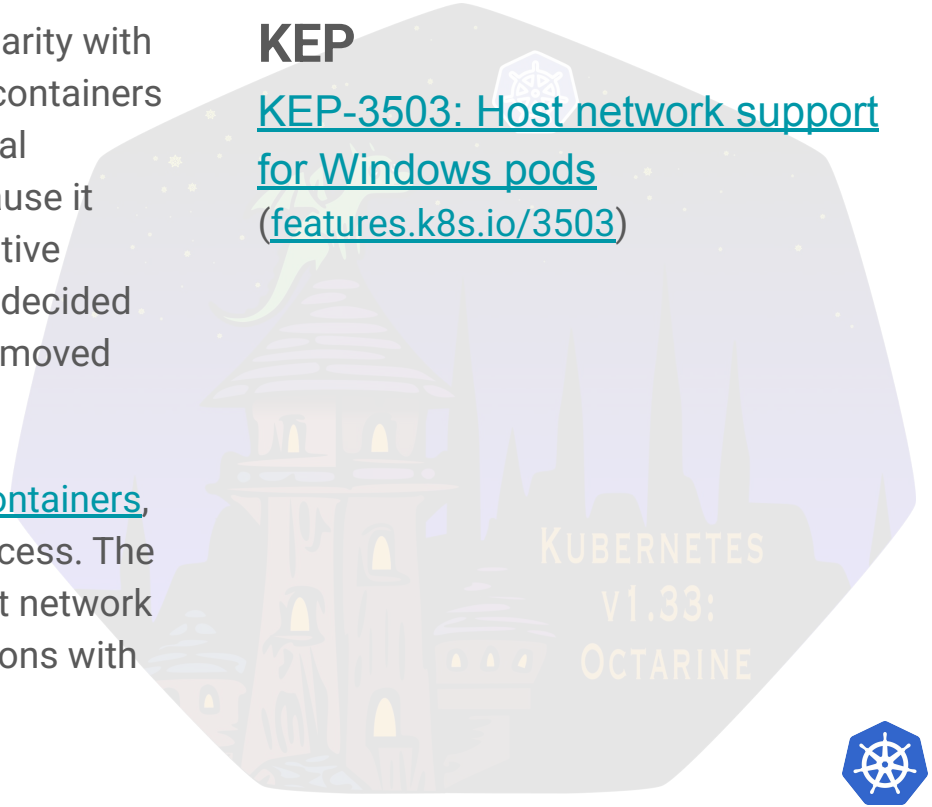
Removal of host network support for Windows pods

Windows Pod networking aimed to achieve feature parity with Linux and provide better cluster density by allowing containers to use the Node's networking namespace. The original implementation landed as alpha with v1.26, but because it faced unexpected containerd behaviours and alternative solutions were available, the Kubernetes project has decided to withdraw the associated KEP. Support was fully removed in v1.33.

Please note that this does not affect HostProcess containers, which provides host network as well as host level access. The KEP withdrawn in v1.33 was about providing the host network only, which was never stable due to technical limitations with Windows networking logic.

KEP

KEP-3503: Host network support for Windows pods (features.k8s.io/3503)



Release Highlights: Other Notable Changes

Dynamic Resource Allocation Galore!

Kubernetes v1.33 continues to develop Dynamic Resource Allocation (DRA) with features designed for today's complex infrastructures. DRA is an API for requesting and sharing resources between pods and containers inside a pod. Typically those resources are devices such as GPUs, FPGAs, and network adapters.

- Similar to Node taints, by enabling the `DRADeviceTaints` feature gate, devices support taints and tolerations.
- By enabling the feature gate `DRAPrioritizedList`, DeviceRequests get a new field named `firstAvailable`.
- With feature gate `DRAAdminAccess` enabled, only users authorized to create ResourceClaim or ResourceClaimTemplate objects in namespaces labeled with `resource.k8s.io/admin-access: "true"` can use the `adminAccess` field.

KEPs

- KEP-5055: DRA: device taints and tolerations (features.k8s.io/5055)
- KEP-4816: DRA: Prioritized Alternatives in Device Requests (features.k8s.io/4816)
- KEP-5018: DRA: AdminAccess for ResourceClaims and ResourceClaimTemplates (features.k8s.io/5018)
- KEP-4815: DRA: Partitionable Devices (features.k8s.io/4815)

In-place resource resize for vertical scaling of Pods

Before this enhancement, container resources defined in a Pod's spec were immutable, and updating any of these details within a Pod template would trigger Pod replacement.

We can now have such in-place Pod updates without Pod replacement. It was released as alpha in v1.27, and has graduated to beta in v1.33. This opens up various possibilities for vertical scale-up of stateful processes without any downtime, seamless scale-down when the traffic is low, and even allocating larger resources during startup, which can then be reduced once the initial setup is complete.

Status: Beta

- Alpha: v1.23
- Beta: v1.31

KEP

KEP-1287: In-Place Update of Pod Resources (features.k8s.io/1287)

Release Team Updates

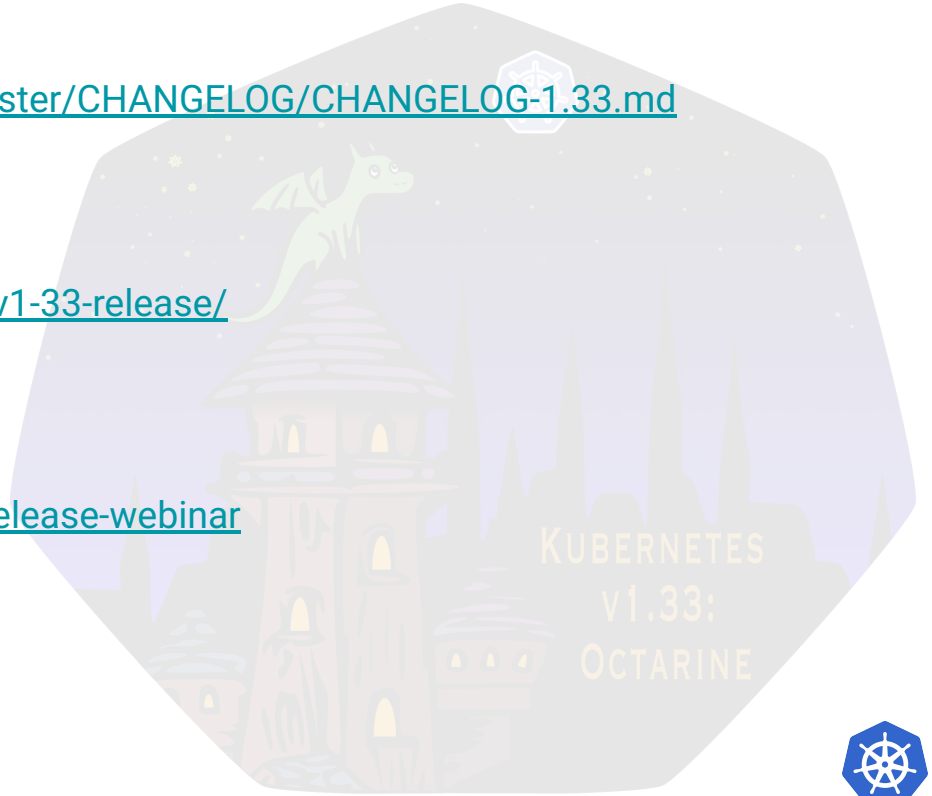
- This release a new team structure adopted, combining Release Notes and Docs subteams into a unified subteam of Docs. Thanks to the meticulous effort in organizing the relevant information and resources from the new Docs team, both Release Notes and Docs tracking have seen a smooth and successful transition!
- There were sixteen feature blogs published for v1.33 release cycle.
- During the v1.33 release cycle, Kubernetes received contributions from as many as 121 different companies and 570 individuals.

Recap

Release Notes
https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.33.md

Release Announcement Blog
https://kubernetes.io/blog/2025/04/23/kubernetes-v1-33-release/

Slide deck
https://github.com/npolshakova/kubernetes-v1-33-release-webinar



XXXX

Some description

Status: Beta | Alpha

- Alpha: v1.23
- Beta: v1.31

KEP

KEP-2644: Always Honor PersistentVolume Reclaim Policy (features.k8s.io/2644)

Join the v1.34 Release Team!

Kubernetes v1.34 Release Team Shadow Application

Thank you for applying to be a shadow on the Kubernetes v1.34 Release Team!

This questionnaire is to learn a little bit more about you, your journey with Kubernetes, your understanding of the Release Team processes and procedures, as well as to determine if you're a good fit for the current Release Team.

Please note: you are expected to be inexperienced in some areas, the shadow program is an apprenticeship.

Participation in the Release Team can include somewhat open-ended expectations. We never know what blocking/critical issue may come up at what point during the release cycle. The Release Team Lead will endeavour to make sure any such inconveniences do not unfairly hit individuals or specific time zones. To learn more please read the doc on shadows here: https://git.k8s.io/sig-release/release-team/shadows.md.

While the release calendar has not yet been finalized, the release is tentatively planned to start on ***Monday, May 19, 2025.*** The tentative release date for Kubernetes v1.34 is ***Wednesday, August 27, 2025***.
Please keep these dates in mind when you consider your availability and the time commitments.
Please note: These dates are currently approximate and subject to change!

Before proceeding further, you must browse and read the relevant Role Handbooks: https://git.k8s.io/sig-release/release-team/role-handbooks

https://forms.gle/hH85eUDU1fDUf3uz9

Thanks!