What’s New in Kubernetes 1.13

CLOUD NATIVE
COMPUTING FOUNDATION

Presenters

Kendrick Coleman

1.13 Enhancements Lead

Saad Ali

SIG-Storage

Kaitlyn Barnard

1.13 Communications

Tim St. Clair

SIG-Cluster Lifecycle

Agenda

1.13 Enhancements Overview

Storage updates

Kubeadm

Q&A

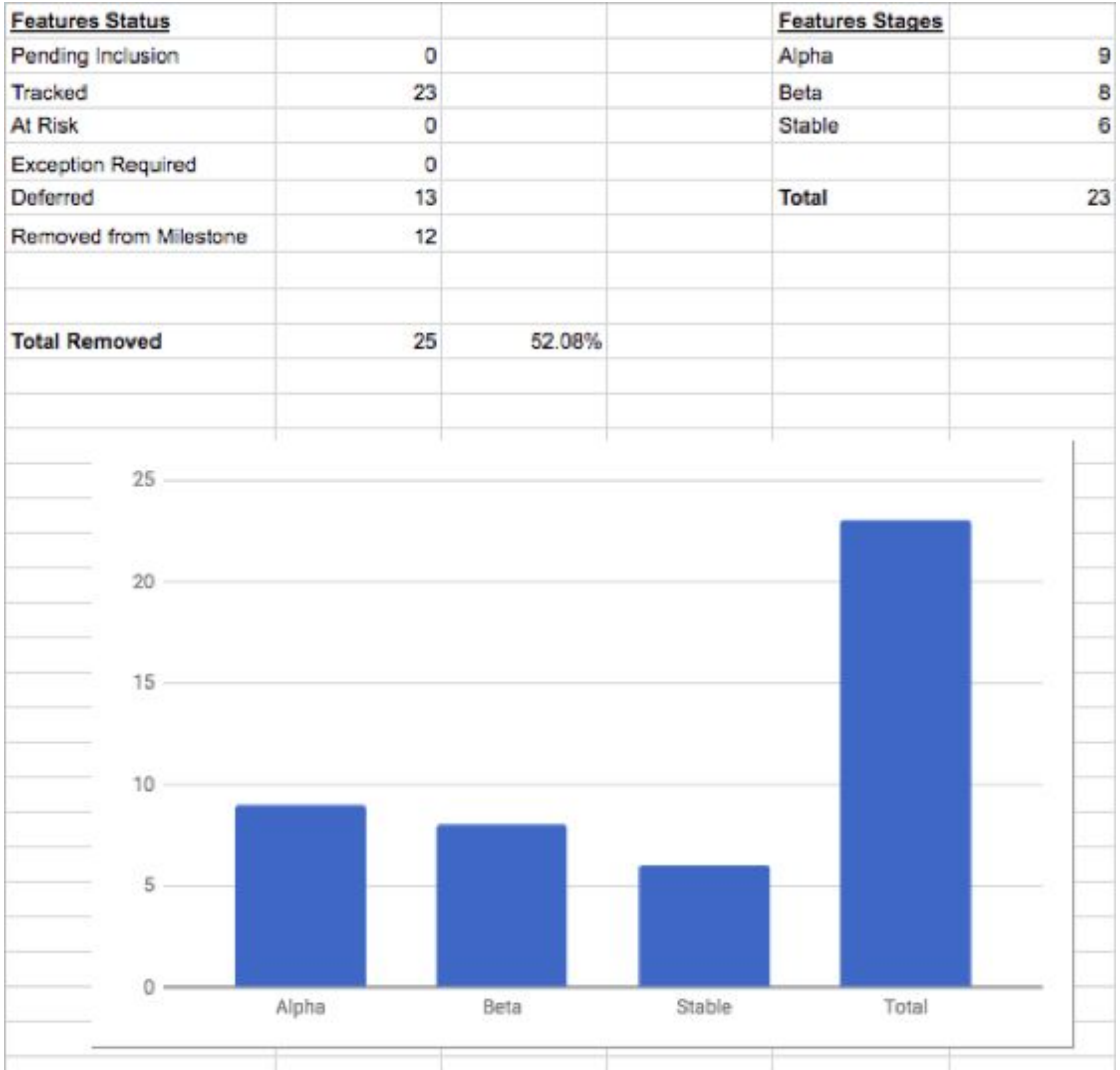
1.13 Enhancements

CLOUD NATIVE
COMPUTING FOUNDATION

Major Themes

- Focus on stability
  - Shortened release cycle from 12-13 weeks to 10
    - KubeCons and US Holidays
  - Remove lofty goals
  - Slipped features from 1.12 into 1.13
  - Required extensive communication by leads to counterpart SIGs
- De-vendoring and Extensibility
  - More on removing vendor code
  - Added more interfaces

Overview



Features Status				**Features Stages**	
Pending Inclusion	0			Alpha	9
Tracked	23			Beta	8
At Risk	0			Stable	6
Exception Required	0				
Deferred	13			**Total**	23
Removed from Milestone	12				
Total Removed	25	52.08%			

Highlights

Kubeadm

Graduated to Stable / SIG Cluster Lifecycle

Kubeadm is a tool built to provide kubeadm init and kubeadm join as best-practice “fast paths” for creating Kubernetes clusters.

kubeadm performs the actions necessary to get a minimum viable cluster up and running. By design, it cares only about bootstrapping, not about provisioning machines. Kubeadm also supports other cluster lifecycle functions, such as upgrades, downgrade, and managing bootstrap tokens. Likewise, installing various nice-to-have addons, like the Kubernetes Dashboard, monitoring solutions, and cloud-specific addons, is not in scope.

Instead, we expect higher-level and more tailored tooling to be built on top of kubeadm, and ideally, using kubeadm as the basis of all deployments will make it easier to create conformant clusters.

https://github.com/kubernetes/enhancements/issues/11

Switch default DNS plugin to CoreDNS

Graduated to Stable / SIG Network

CoreDNS is another CNCF project

- 3000+ GitHub Stars
- 114+ contributors

A single executable and single process.

Written in Go

Supports more use cases

https://github.com/kubernetes/enhancements/issues/566

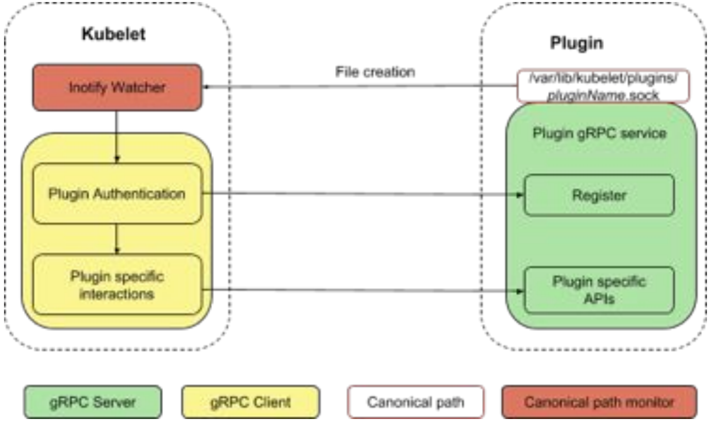
Kubelet Device Plugin Registration

Graduated to Stable / SIG Node

A common Kubelet plugin discovery model that can be used by different types of node-level plugins, such as device plugins, CSI, and CNI, to establish communication channels with Kubelet.

- Currently, a plugin registers with Kubelet through grpc
- New implementation is a Kubelet watches new plugins under a canonical path through inotify.
  - Kubelet will have a new module, PluginWatcher, which will probe a canonical path recursively
  - On detecting a socket creation, Watcher will try to get plugin identity details using a gRPC client on the discovered socket and the RPCs of a newly introduced Identity service
  - Plugins must implement Identity service RPCs for initial communication with Watcher.

https://github.com/kubernetes/enhancements/issues/595



Out-of-tree CSI Volume Plugins

Graduated to Stable / SIG Storage

Allows out-of-tree plugins to be created for volume purposes. Kubernetes volume plugins are currently all "in-tree" meaning that their source code is included in the main Kubernetes repo. All volume plugins are compiled and ship along with kubernetes binaries. This allows plug-ins to:

- be developed out-of-tree.
- Not require building volume plugins (or their dependencies) into Kubernetes binaries
- Not requiring direct machine access to deploy new volume plugins (drivers)

CONTAINER STORAGE INTERFACE

https://github.com/kubernetes/enhancements/issues/178

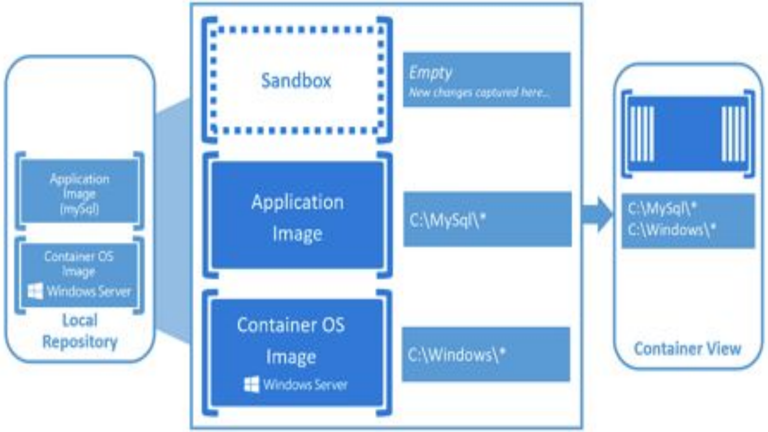
Windows Container Support

Didn’t Happen

Where’s Windows??

SIG Architecture and SIG Windows met to discuss the current status and determined it was not ready to graduate to GA/Stable

SIG Architecture has requested a formal KEP to be filed. This is to make sure there is an expectation of what will and won’t be supported. This is also to make sure there is community consensus moving forward.



https://github.com/kubernetes/enhancements/issues/116

API-Machinery

CLOUD NATIVE
COMPUTING FOUNDATION

APIserver "dry-run"

Graduated to Beta

Dry-run is a new feature implemented in the api-server

The goal is to send requests to modifying endpoints, and see if the request would have succeeded and/or what would have happened without having it actually happen. The response body for the request should be as close as possible to a non dry-run response.

Dry-run is triggered by setting the “dryRun” query parameter on modifying verbs: POST, PUT, PATCH and DELETE.

https://github.com/kubernetes/enhancements/issues/576

Webhook Conversion for Custom Resource Definitions

Alpha

CRD supports multiple version but no conversion between them (something called nopConverter which only change the apiVersion of the CR). With this proposal, it introduced a conversion mechanism for CRDs based on an external webhook. Detail API changes, use cases and upgrade/downgrade scenarios are discussed.

https://github.com/kubernetes/enhancements/issues/598

Drop support for etcd2

Stable

Deprecation announced in 1.10

Support for etcd3 was added in 1.5 and made stable in 1.6

https://github.com/kubernetes/enhancements/issues/622

Auth

Dynamic Audit Configuration

Alpha

Kubernetes provides auditing via the API server to generate a record of chronological events to document activated performed. It answers the following questions:

- what happened?
- when did it happen?
- who initiated it?
- on what did it happen?
- where was it observed?
- from where was it initiated?
- to where was it going?

Dynamic Audit Control provide a means of configuring the advanced auditing features post cluster provisioning.

https://github.com/kubernetes/enhancements/issues/600

CLI

Kubectl diff

Beta

Users can run a kubectl command to view the difference between a locally declared object configuration and the current state of a live object

Kubectl would support a diff command referencing the object configuration (OC) sources referenced above (LOCAL, LIVE, LAST, MERGED) using a similar convention to how *git-diff* references the tip of a branch as HEAD.

Implementation

- `kubectl diff -f something.yaml -f somethingelse.yaml LOCAL MERGED`

https://github.com/kubernetes/enhancements/issues/491

Plugin mechanism for kubectl

Beta

Provide users with a way to extend the functionality of kubectl, beyond what is offered by its core commands

Avoid any kind of installation process (no additional config, users drop an executable in their PATH, for example, and they are then able to use that plugin with kubectl). No additional configuration is needed, only the plugin executable. A plugin's filename determines the plugin's intention, such as which path in the command tree it applies to: `/usr/bin/kubectl-educate-dolphins` would, for example be invoked under the command `kubectl educate dolphins --flag1 --flag2`. It is up to a plugin to parse any arguments and flags given to it. A plugin decides when an argument is a subcommand, as well as any limitations or constraints that its flags should have.

https://github.com/kubernetes/enhancements/issues/579

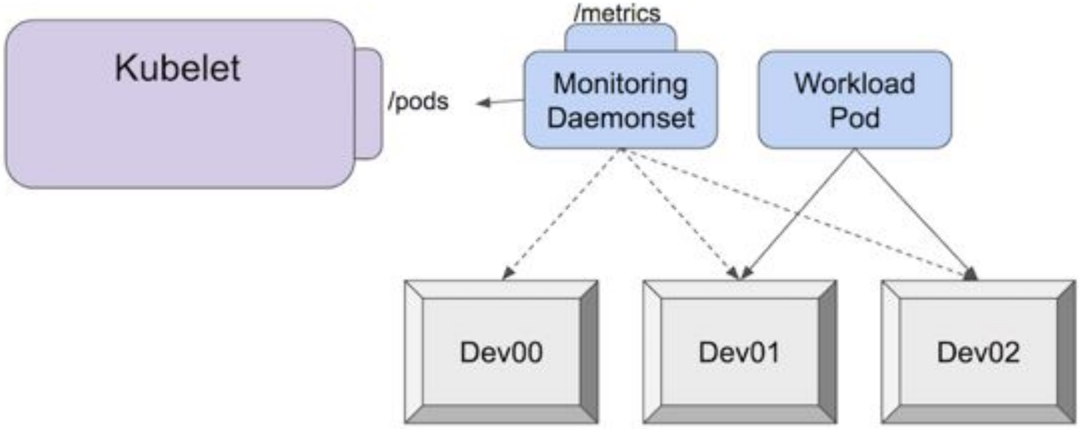
Node

CLOUD NATIVE
COMPUTING FOUNDATION

Support 3rd party device monitoring plugins

Alpha

Device Monitoring requires external agents to be able to determine the set of devices in-use by containers and attach pod and container metadata for these devicesDynamic Audit Control provide a means of configuring the advanced auditing features post cluster provisioning.



https://github.com/kubern

Move frequent Kubelet heartbeats to Lease API

Alpha

Kubelet creates and periodically renews a Lease on the node; node lifecycle controller treats this lease as a health signal.

Introducing a new Lease built-in API in the newly created API group coordination.k8s.io. To make it easily reusable for other purposes it will be namespaced.

https://github.com/kubernetes/enhancements/issues/606

Scheduling

CLOUD NATIVE
COMPUTING FOUNDATION

Taint Based Eviction

Graduated to Beta

This is an addon to tainting nodes which is similar to anti-affinity in VM-speak.

TaintBasedEvictions feature is promoted to beta and enabled by default, hence the taints are automatically added by the NodeController (or kubelet) and the normal logic for evicting pods from nodes based on the Ready NodeCondition is disabled.

- `node.kubernetes.io/not-ready`: Node is not ready. This corresponds to the NodeCondition Ready being "False".
- `node.kubernetes.io/unreachable`: Node is unreachable from the node controller. This corresponds to the NodeCondition Ready being "Unknown".
- `node.kubernetes.io/out-of-disk`: Node becomes out of disk.
- `node.kubernetes.io/memory-pressure`: Node has memory pressure.
- `node.kubernetes.io/disk-pressure`: Node has disk pressure.
- `node.kubernetes.io/network-unavailable`: Node's network is unavailable.
- `node.kubernetes.io/unschedulable`: Node is unschedulable.
- `node.cloudprovider.kubernetes.io/uninitialized`: When the kubelet is started with "external" cloud provider, this taint is set on a node to mark it as unusable. After a controller from the cloud-controller-manager initializes this node, the kubelet removes this taint.

https://github.com/kubernetes/enhancements/issues/166

Scheduler checks feasibility and scores a subset of all cluster nodes

Beta

Today, kube-scheduler checks feasibility of all of the nodes in a cluster for every pod in every scheduling attempt. All of those feasible pods are then scored. Performance data shows that 90th percentile of running predicate and priority functions takes about 30ms per pod and 99th percentile is as high as 60ms/pod.

This feature in the scheduler finds a certain number of feasible nodes in a cluster, and passes those nodes for scoring. This will improve the scheduler's performance.

Today, kube-scheduler checks all the feasible nodes and picks the highest score in the whole cluster. With this feature, the chosen node may not be the best node in the whole cluster. The chosen node will be the highest score among those nodes found feasible.

https://github.com/kubernetes/enhancements/issues/593

Storage

CLOUD NATIVE
COMPUTING FOUNDATION

Topology Aware Volume Scheduling

Graduated to Stable

Makes the scheduler aware of a Pod's volume's topology constraints, such as zone or node making the PersistentVolumeClaim (PVC) binding aware of scheduling decisions.

- Allow a Pod to request one or more topology-constrained Persistent Volumes (PV) that are compatible with the Pod's other scheduling constraints, such as resource requirements and affinity/anti-affinity policies.
- Support arbitrary PV topology constraints (i.e. node, rack, zone, foo, bar).
- Support topology constraints for statically created PVs and dynamically provisioned PVs.
- No scheduling latency performance regression for Pods that do not use topology-constrained PVs.

https://github.com/kubernetes/enhancements/issues/490

Raw block device using persistent volume source

Beta

By extending the API for volumes to specifically request a raw block device, we provide an explicit method for volume consumption, whereas previously any request for storage was always fulfilled with a formatted file system, even when the underlying storage was block. In addition, the ability to use a raw block device without a filesystem will allow Kubernetes better support of high performance applications that can utilize raw block devices directly for their storage. Block volumes are critical to applications like databases (MongoDB, Cassandra) that require consistent I/O performance and low latency. For mission critical applications, like SAP, block storage is a requirement.

https://github.com/kubernetes/enhancements/issues/351

Add resize support for FlexVolume

Alpha

Add resize call support for FlexVolume to support volume resizing like LVM expansion

When user uses FlexVolume and corresponding volume driver to connect to the backend storage system, they can expand the PV size by updating PVC in Kubernetes.

https://github.com/kubernetes/enhancements/issues/304

Kubeadm - GA

Command Line UX

Graduated to Stable

- **Stable command-line UX** — The kubeadm CLI conforms to #5a GA rule of the Kubernetes Deprecation Policy, which states that a command or flag that exists in a GA version must be kept for at least 12 months after deprecation.
  - ***init/join/upgrade/config/reset/token/version***
- **Upgrades between minor versions** — The kubeadm upgrade command is now fully GA. It handles control plane upgrades for you, which includes upgrades to etcd, the API Server, the Controller Manager, and the Scheduler. You can seamlessly upgrade your cluster between minor or patch versions (e.g. v1.12.2 -> v1.13.1 or v1.13.1 -> v1.13.3).

Command Line UX - Cont.

Graduated to Stable

- **The “toolbox” interface of kubeadm** — Also known as **phases**. If you don’t want to perform all kubeadm init tasks, you can instead apply more fine-grained actions using the kubeadm init phase command (for example generating certificates or control plane Static Pod manifests).
    - Currently this only applies to `*kubeadm init*`
    - In 2019 - `kubeadm join phases`
- **etcd setup** — etcd is now set up in a way that is secure by default, with TLS communication everywhere, and allows for expanding to a highly available cluster when needed.
- **HA Ready** - (feature still currently experimental)

Configuration File Schema

Beta

- **Configuration file schema** — With the new **v1beta1** API version, you can now tune almost every part of the cluster declaratively and thus build a "GitOps" flow around kubeadm-built clusters. In future versions, we plan to graduate the API to version **v1** with minimal changes.
  - Examples and references are now in standard Godoc format
  - Config is split into parts
    - InitConfiguration
    - ClusterConfiguration - stored on cluster in a configmap
    - JoinConfiguration

Questions?

CLOUD NATIVE
COMPUTING FOUNDATION

Thank You

CLOUD NATIVE
COMPUTING FOUNDATION