WG Component Standard

Leigh Capili
Mike Taufen

Today's Talk

- Some history of how we got started.
- A little bit about our working group.
  - Projects, contributors, and mentorship efforts.
  - How you can get involved!

History first

Kubernetes-style APIs

If you work with Kubernetes, you're probably pretty familiar with these yaml things:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
 labels: ...
spec:
 replicas: 3
 selector:
   matchLabels: ...
 template:
   metadata:
     labels: ...
   spec:
     containers: ...
```

Kubernetes-style APIs


```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
 labels: ...
spec:
 replicas: 3
 selector:
   matchLabels: ...
 template:
   metadata:
     labels: ...
   spec:
     containers: ...
```

Maybe all that config annoys you…

But these yamls have some nice properties.

Kubernetes-style APIs


```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
 labels: ...
spec:
 replicas: 3
 selector:
   matchLabels: ...
 template:
   metadata:
     labels: ...
   spec:
     containers: ...
```

One important property is that they each conform to a *versioned schema*.

Kubernetes calls this a *GroupVersionKind*, or *GVK* for short.

Kubernetes-style APIs


```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
 labels: ...
spec:
 replicas: 3
 selector:
   matchLabels: ...
 template:
   metadata:
     labels: ...
   spec:
     containers: ...
```

The *API group* (apps) has a *version* (v1).

This versioned group contains several *Kinds* (e.g. Deployment).

What does yaml + versions get us?


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
 labels: ...
spec:
 replicas: 3
 selector:
   matchLabels: ...
 template:
   metadata:
     labels: ...
   spec:
     containers: ...
```

- Version can express stability guarantees for configuration APIs.
- Config written against one version works as long as that version is available.
- Structure makes it easy to read, write, and parse.
- Common tooling (kubectl, Kustomize, etc).


KubeCon
CloudNativeCon
North America 2019
Great, then what's the problem?

Command line flags

If you use Unix-style computer systems, you're probably familiar with the command line:

```
$ do-something --foo 1 --bar 2,3,4,5
```

Command line flags

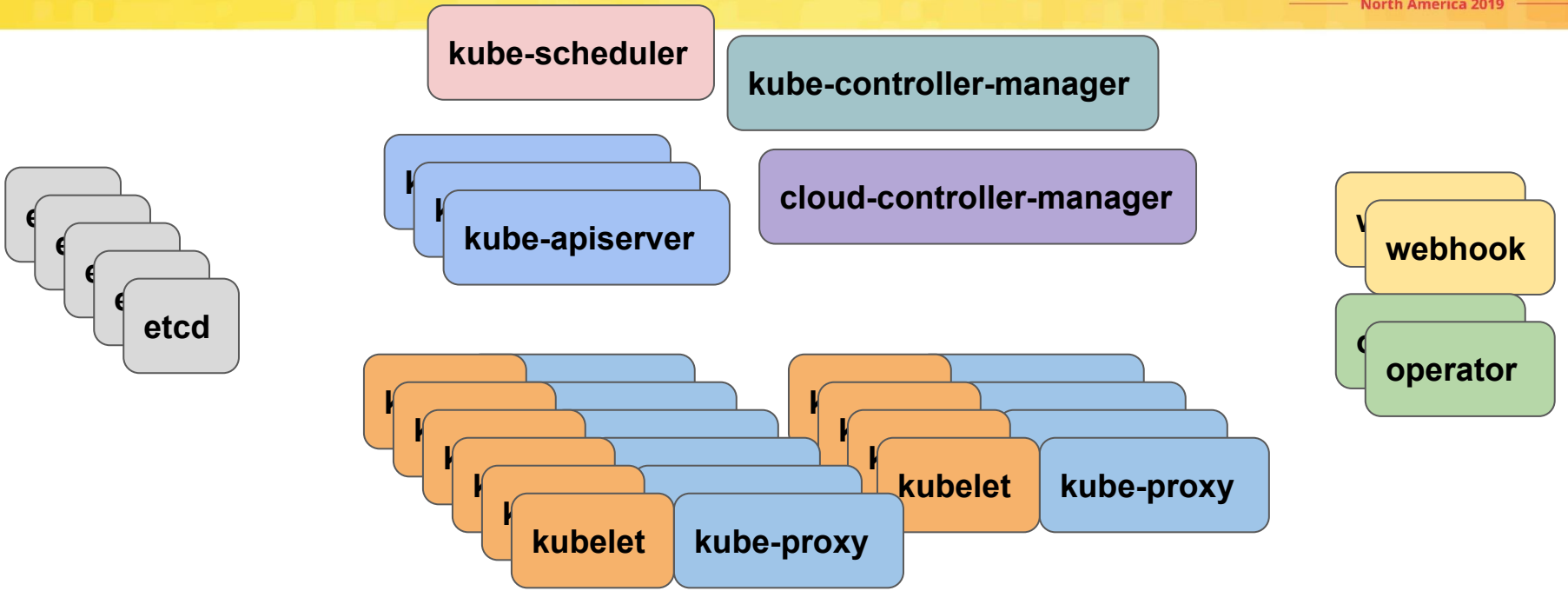
Commands can take *flags* that describe configuration.

```
$ do-something --foo 1 --bar 2,3,4,5
```

The values are arbitrary strings parsed by the program.

Which is fine and convenient for tools and small programs.

What about Kubernetes?



Even though things inside the cluster use K8s-style configs,

the cluster itself is still using command line flags.

Why does this matter?

If you've ever configured a Kubernetes cluster from scratch, you may be familiar with something like this:

```
kubelet --v=2 --cloud-provider=gce --experimental-check-node-
capabilities-before-mount=true --allow-privileged=true --expe
rimental-mounter-path=/home/kubernetes/containerized_mounter/mo
unter --cert-dir=/var/lib/kubelet/pki/ --cni-bin-dir=/home/ku
bernetes/bin --kubeconfig=/var/lib/kubelet/kubeconfig --exper
imental-kernel-memcg-notification=true --max-pods=110 --netwo
rk-plugin=kubenet --node-labels=beta.kubernetes.io/fluentd-ds-
ready=true,cloud.google.com/gke-nodepool=default-pool,cloud.goo
gle.com/gke-os-distribution=cos --volume-plugin-dir=/home/kube
rnetes/flexvolume --bootstrap-kubeconfig=/var/lib/kubelet/boot
strap-kubeconfig --node-status-max-images=25 --registry-qps=1
0 --registry-burst=20 --pod-sysctls='net.core.somaxconn=1024,
net.ipv4.conf.all.accept_redirects=0,net.ipv4.conf.all.forwardi
ng=1,net.ipv4.conf.all.route_localnet=1,net.ipv4.conf.default.f
orwarding=1,net.ipv4.ip_forward=1,net.ipv4.tcp_fin_timeout=60,n
et.ipv4.tcp_keepalive_intvl=75,net.ipv4.tcp_keepalive_probes=9,
net.ipv4.tcp_keepalive_time=7200,net.ipv4.tcp_max_syn_backlog=1
28,net.ipv4.tcp_max_tw_buckets=16384,net.ipv4.tcp_syn_retries=6
,net.ipv4.tcp_tw_reuse=0,net.netfilter.nf_conntrack_generic_tim
eout=600,net.netfilter.nf_conntrack_tcp_timeout_close_wait=3600
,net.netfilter.nf_conntrack_tcp_timeout_established=86400' --a
nonymous-auth=false --authentication-token-webhook=true --cli
ent-ca-file=/etc/srv/kubernetes/pki/ca-certificates.crt --auth
orization-mode=webhook --cgroup-root=/ --cluster-dns=10.27.24
0.10 --cluster-domain=cluster.local --enable-debugging-handle
rs=true --eviction-hard="memory.available<100Mi,nodefs.availab
le<10%,nodefs.inodesFree<5%" --feature-gates=DynamicKubeletCon
fig=false,ExperimentalCriticalPodAnnotation=true,NodeLease=true
,RotateKubeletServerCertificate=false,TaintBasedEvictions=false --kub
```

Problems with flags

- Flags are a public API, but breaking changes are not communicated by the overall K8s version.
  - Flag breakages are *allowed* across K8s minor versions as long as warnings were logged for enough releases.
- Tools don't understand the custom structures (component-specific string parsers) built into command lines. *Only* the component binary knows how to read them.
- Flags embed structured data in strings, and components invent one-off parsers to process their flags. This invites bugs. Many of these structures (lists, maps) *could* be expressed in basic yaml.

```
kubelet --v=2 --cloud-provider=gce --experimental-check-node-
capabilities-before-mount=true --allow-privileged=true --expe
rimental-mounter-path=/home/kubernetes/containerized_mounter/mo
unter --cert-dir=/var/lib/kubelet/pki/ --cni-bin-dir=/home/ku
bernetes/bin --kubeconfig=/var/lib/kubelet/kubeconfig --exper
imental-kernel-memcg-notification=true --max-pods=110 --netwo
rk-plugin=kubenet --node-labels=beta.kubernetes.io/fluentd-ds-
ready=true,cloud.google.com/gke-nodepool=default-pool,cloud.goo
gle.com/gke-os-distribution=cos --volume-plugin-dir=/home/kube
rnetes/flexvolume --bootstrap-kubeconfig=/var/lib/kubelet/boot
strap-kubeconfig --node-status-max-images=25 --registry-qps=1
0 --registry-burst=20 --pod-sysctls='net.core.somaxconn=1024,
net.ipv4.conf.all.accept_redirects=0,net.ipv4.conf.all.forwardi
ng=1,net.ipv4.conf.all.route_localnet=1,net.ipv4.conf.default.f
orwarding=1,net.ipv4.ip_forward=1,net.ipv4.tcp_fin_timeout=60,n
et.ipv4.tcp_keepalive_intvl=75,net.ipv4.tcp_keepalive_probes=9,
net.ipv4.tcp_keepalive_time=7200,net.ipv4.tcp_max_syn_backlog=1
28,net.ipv4.tcp_max_tw_buckets=16384,net.ipv4.tcp_syn_retries=6
,net.ipv4.tcp_tw_reuse=0,net.netfilter.nf_conntrack_generic_tim
eout=600,net.netfilter.nf_conntrack_tcp_timeout_close_wait=3600
,net.netfilter.nf_conntrack_tcp_timeout_established=86400' --a
nonymous-auth=false --authentication-token-webhook=true --cli
ent-ca-file=/etc/srv/kubernetes/pki/ca-certificates.crt --auth
orization-mode=webhook --cgroup-root=/ --cluster-dns=10.27.24
0.10 --cluster-domain=cluster.local --enable-debugging-handle
rs=true --eviction-hard="memory.available<100Mi,nodefs.availab
le<10%,nodefs.inodesFree<5%" --feature-gates=DynamicKubeletCon
fig=false,ExperimentalCriticalPodAnnotation=true,NodeLease=true
,RotateKubeletServerCertificate=false,TaintBasedEvictions=false
```

Solution: ComponentConfig

Use Kubernetes-style config files for configuring the cluster too!

- Humans like them.
  - Readable and writable.
  - Clear stability policy.
- Tools like them.
  - Common format with wide support.
  - Avoids nonstandard structures that prevent interop.
- *Versioned schemas help everyone.*

```
# /var/lib/kubelet/config.yaml
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
clusterDNS:
- 10.27.240.10
authentication:
 webhook:
   cacheTTL: 2m0s
   enabled: true
 x509:
   clientCAFile: /etc/kubernetes/pki/ca.crt
evictionHard:
 imagefs.available: 0%
 nodefs.available: 0%
 nodefs.inodesFree: 0%
                    ...
```

Progress so far:

- Kubelet has a v1beta1 ComponentConfig.
- Several components (kube-proxy, kube-scheduler) have v1alpha1 ComponentConfig APIs.
- Kubeadm is driven by ComponentConfig and generates configs.
- Large, in-progress migration across multiple releases. Many flags still need to become available in configs, and there are still some design issues to solve.
- Prior to the WG progress was sometimes slow, as individuals' time ebbed and flowed, but now:
- Many new contributors in **#wg-component-standard** are helping out, and we're starting to make progress again. **Looking forward to 2020!**

More about the WG!

Elevator Pitch

Come to WG Component Standard when you want Kubernetes components to do something the same way.

Mission

Develop a standard foundation (philosophy and libraries) for core Kubernetes components to build on top of.

Areas Include:

- Configuration (flags, ComponentConfig APIs, ...)
- Status Endpoints (healthz, configz, ...)
- Integration Points (delegated authn/z, ...)
- Logging / Metrics

Details in KEP 0032: kubernetes/enhancements:keps/sig-cluster-lifecycle/wgs/0032-create-a-k8s-io-component-repo.md

Current Projects

- Continuing the flag to ComponentConfig migrations for:
  - kubelet
  - kube-proxy
  - kube-scheduler
  - controller-managers
- Strict decoders that reject invalid field names
- Improved testing for ComponentConfig
- Cleanup of existing ComponentConfig backwards-compatibility layers
- Standardizing, breaking out common component server endpoints like /healthz, /metrics, etc.
- Structured logging
- **And more!**

Mentorship group

We started offering direct mentorship to new contributors in September 2019.

Currently **15 new contributors working on key projects in our WG's mentorship group**.

@savitharaghunathan
@McCoyAle
@mayankshah1607
@palnabarun
@lalatendum
@bharaththiruveedula
@RainbowMango
@alejandrox1
@praveensastry
@obitech
@phenixblue
@tahsinrahman
@pjbgf
@Abhik1998
@conwaychriscosmo

Recent PRs

@stealthybox, @obitech, @phenixblue: Enabling strict decoders across core components that support ComponentConfig.

- Merged **core implementation:** #76805
- Merged **kube-scheduler**: #83030, #84129
- Merged **kubelet**: #83204
- Merged **kube-proxy**: #82927, #84143

@tahsinrahman: Increase test coverage for ComponentConfig APIs.

- Open **core implementation:** #84688

@mtaufen, @alejandrox1: LegacyFlag prototypes (possible cleanup of ComponentConfig compatibility layers).

- Merged **core implementation:** kubernetes-sigs/legacyflag #1
- Open **kube-proxy:** #79916

How you can get involved

Weekly meeting: Tuesdays 8:30am-9:00am PT
Weekly office hours: Tuesdays 10:00am-11:00am PT

Mailing list:
kubernetes-wg-component-standard@googlegroups.com
Join for meeting invites!

GitHub:
kubernetes/community/tree/master/wg-component-standard

`wg/component-standard`

Slack:
Chairs: @mtaufen, @stealthybox, @sttts
#wg-component-standard #wg-component-standard-mentorship


KubeCon
CloudNativeCon
North America 2019
Thank you!


KubeCon
CloudNativeCon
North America 2019

Extra/alt slides

Version Conversions