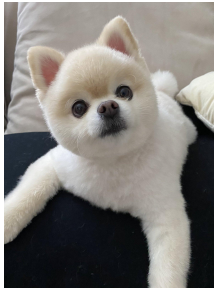
KubeCon | CloudNativeCon

Europe 2020

Virtual

Dynamic Configuration with `ComponentConfig` and the Control Loop

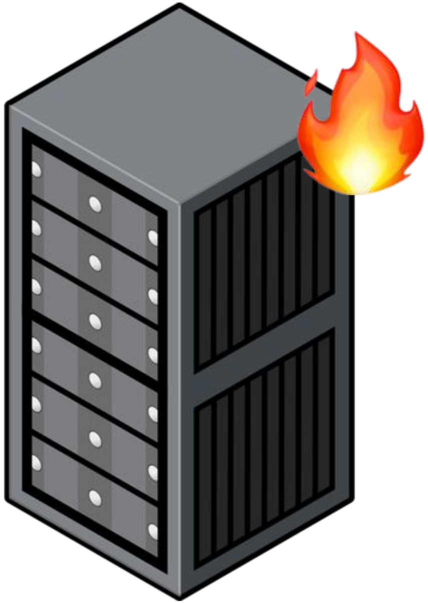


Leigh Capili	Chris Hein	(kevin)
@capileigh	@christopherhein	thedogkevin
stealthybox	christopherhein	

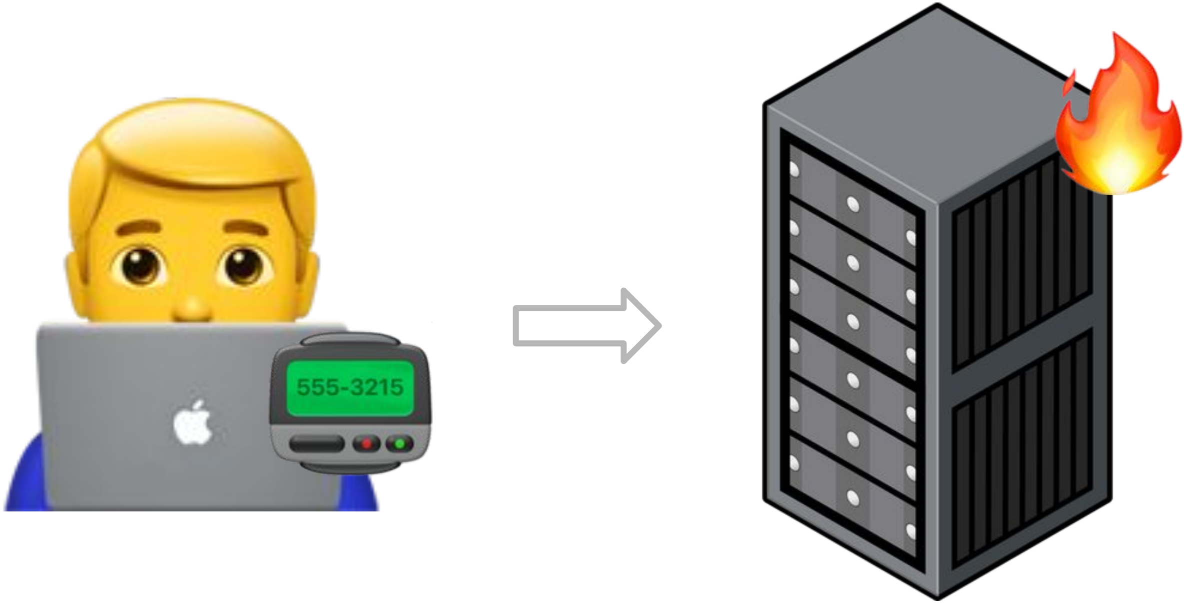
Let’s talk about config...

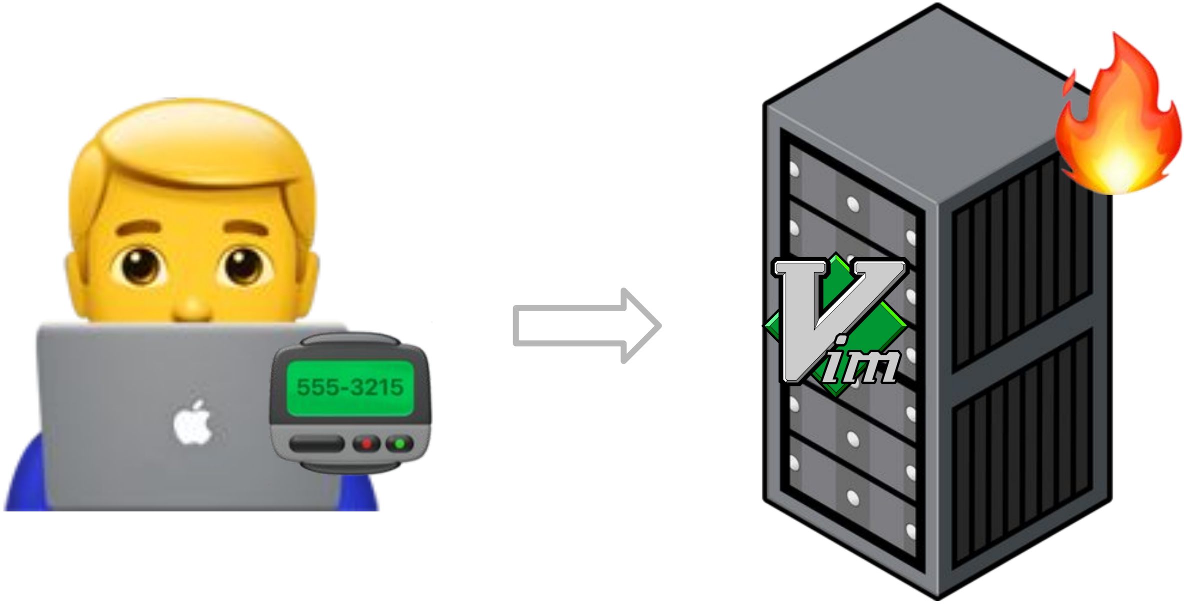
@capileigh @christopherhein

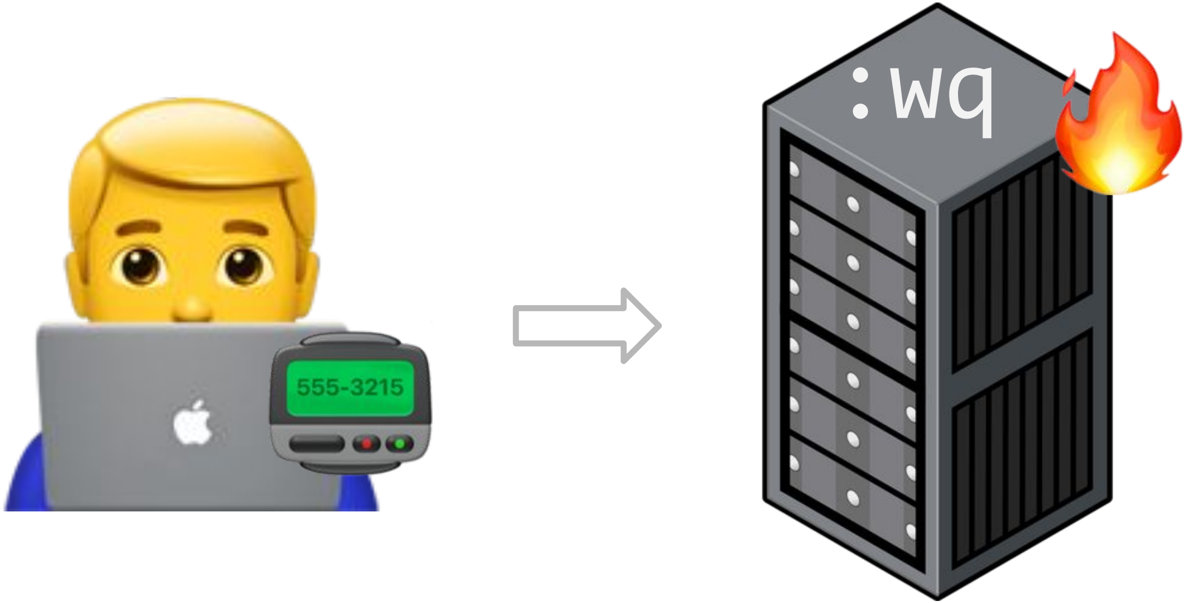
555-3215

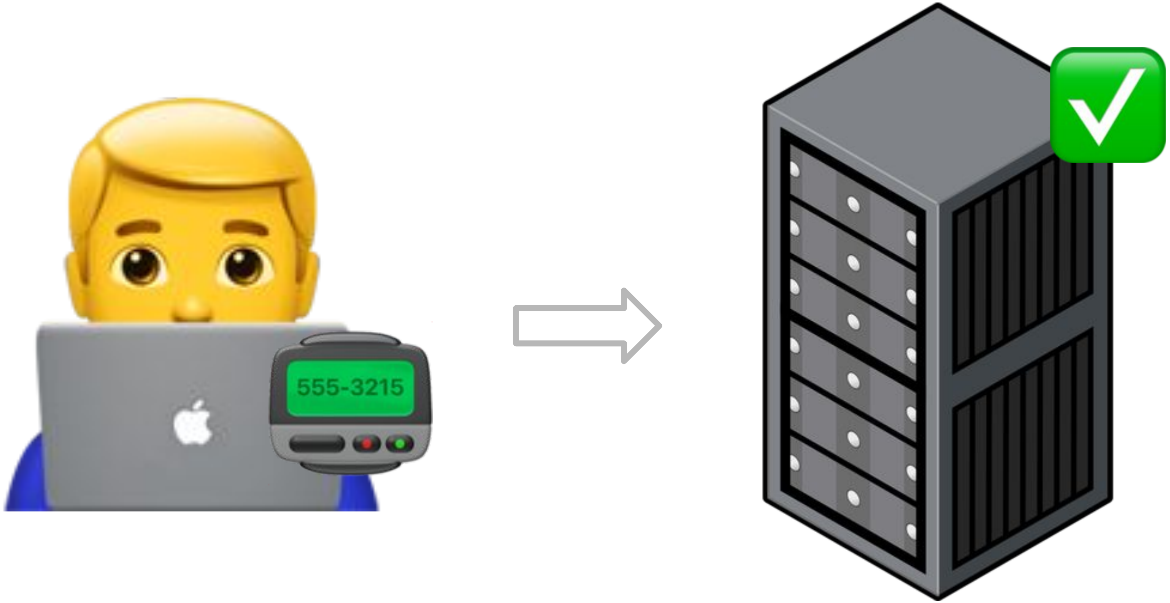


@capileigh @christopherhein

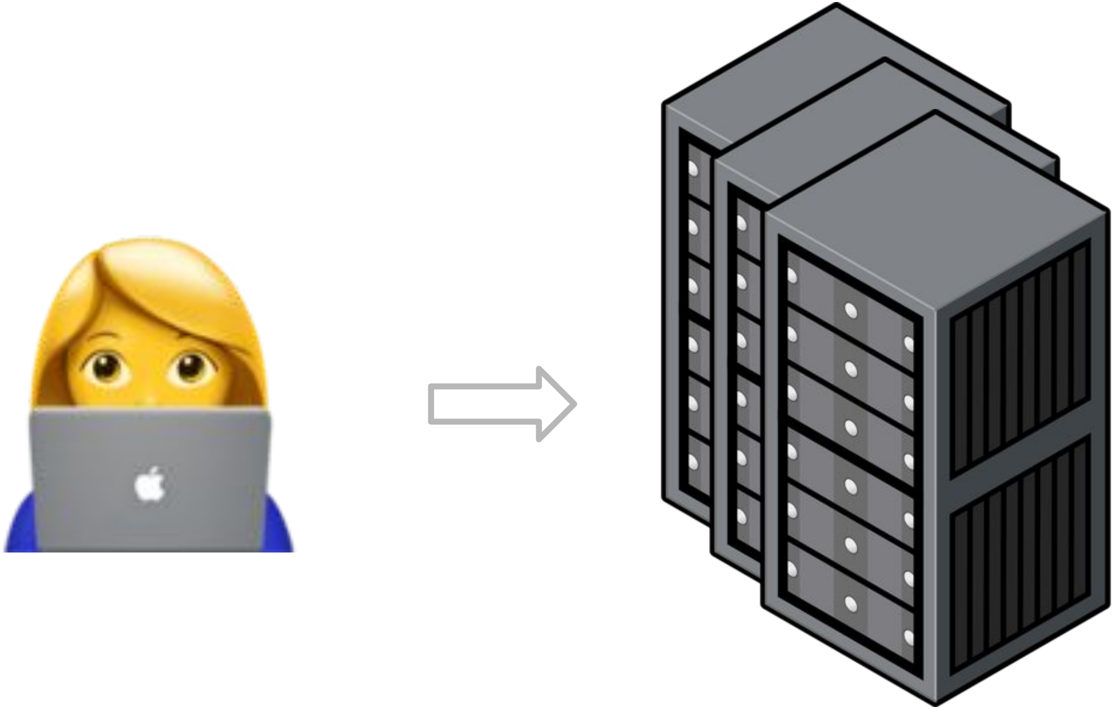

555-3215


555-3215
Vim

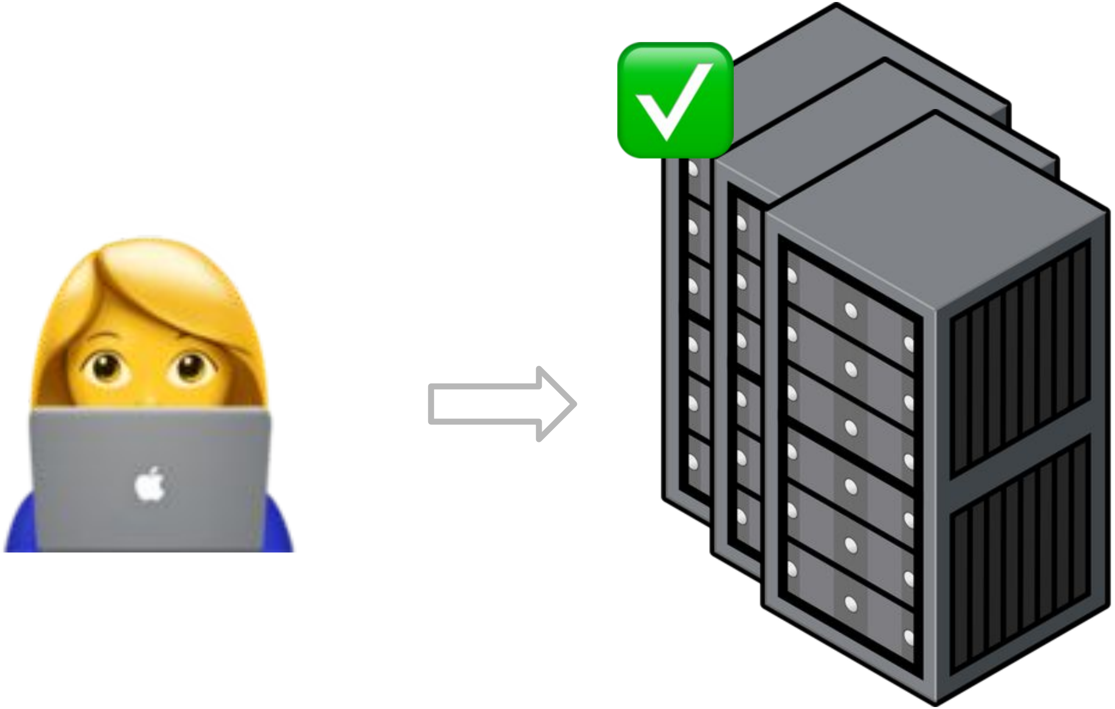

555-3215
:wq

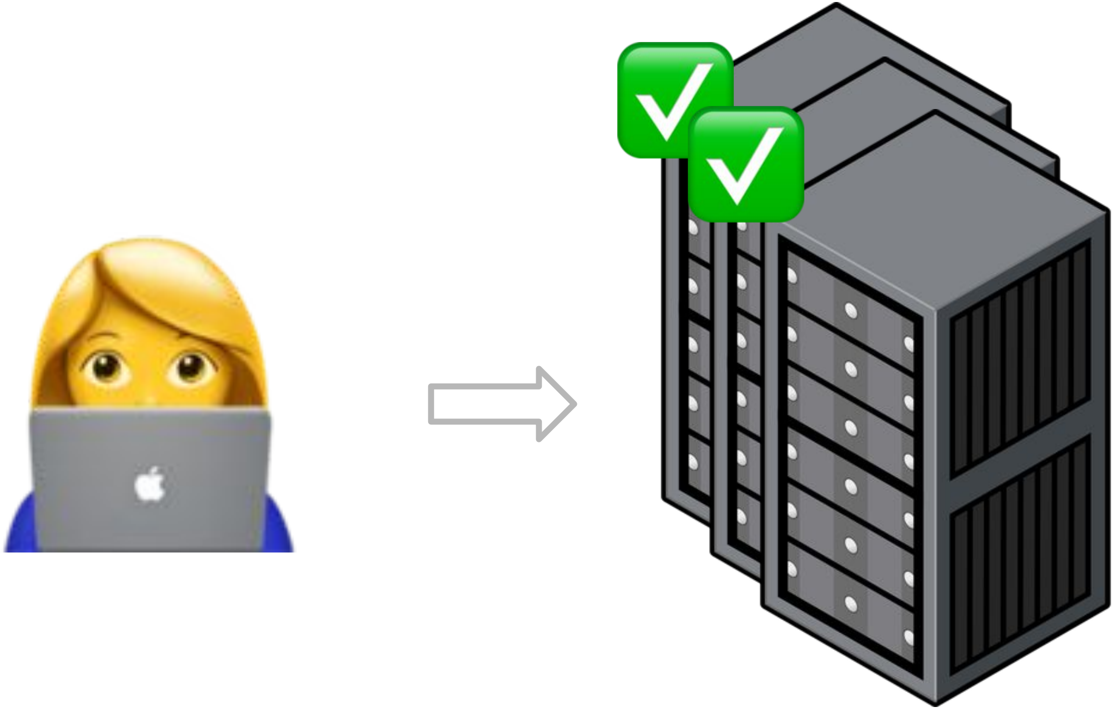

555-3215

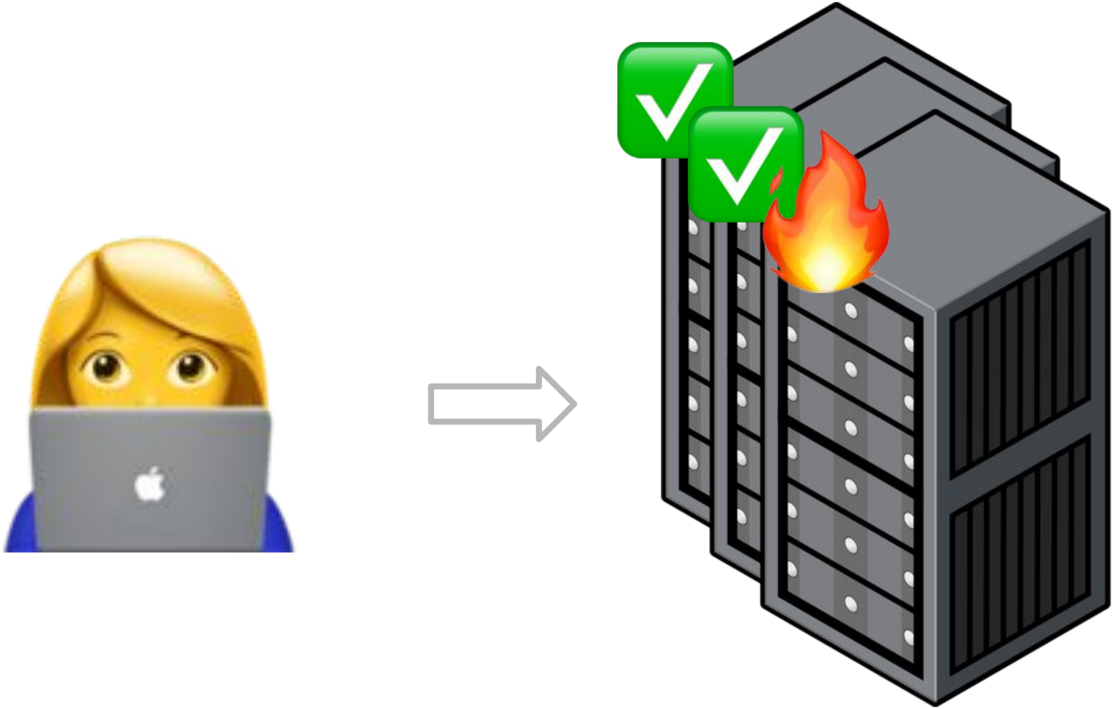
24 hours later…



@capileigh @christopherhein



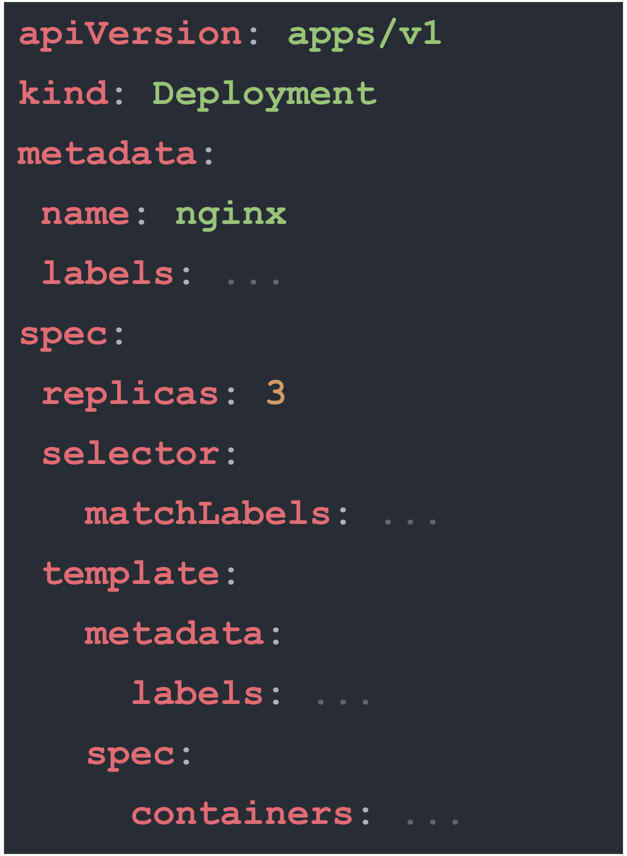




What can we do?

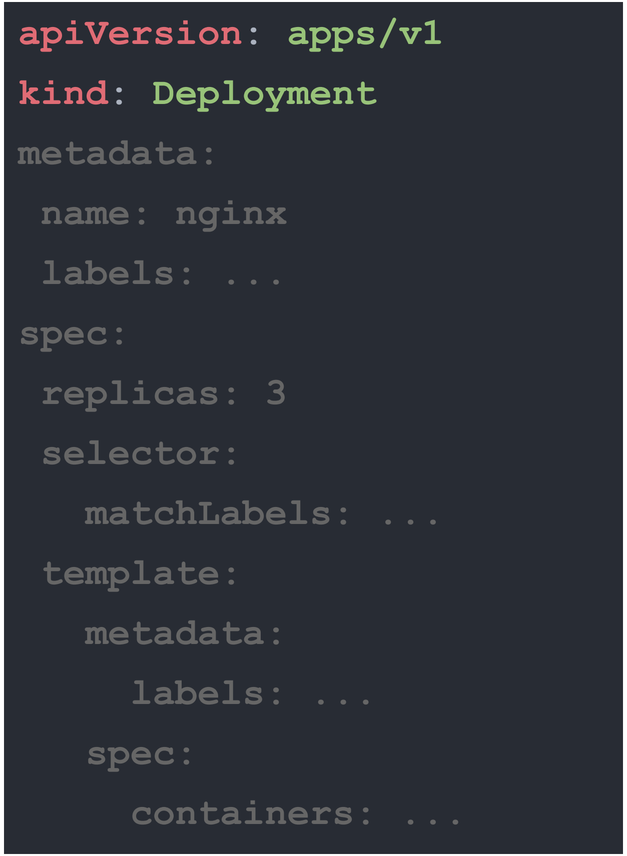
Kubernetes-style APIs

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



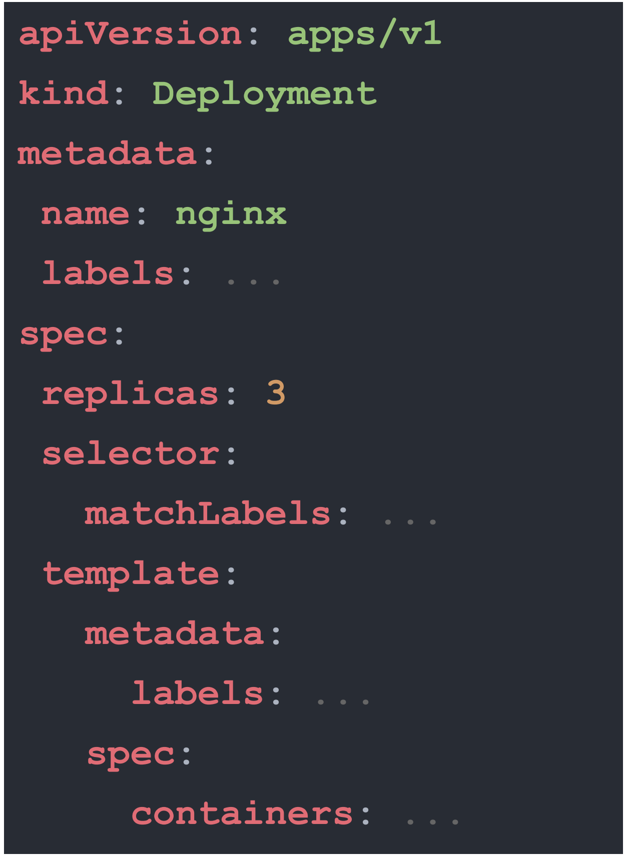
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
 labels: ...
spec:
 replicas: 3
 selector:
   matchLabels: ...
 template:
   metadata:
     labels: ...
   spec:
     containers: ...
```

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

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

@capileigh @christopherhein

Kubernetes-style APIs

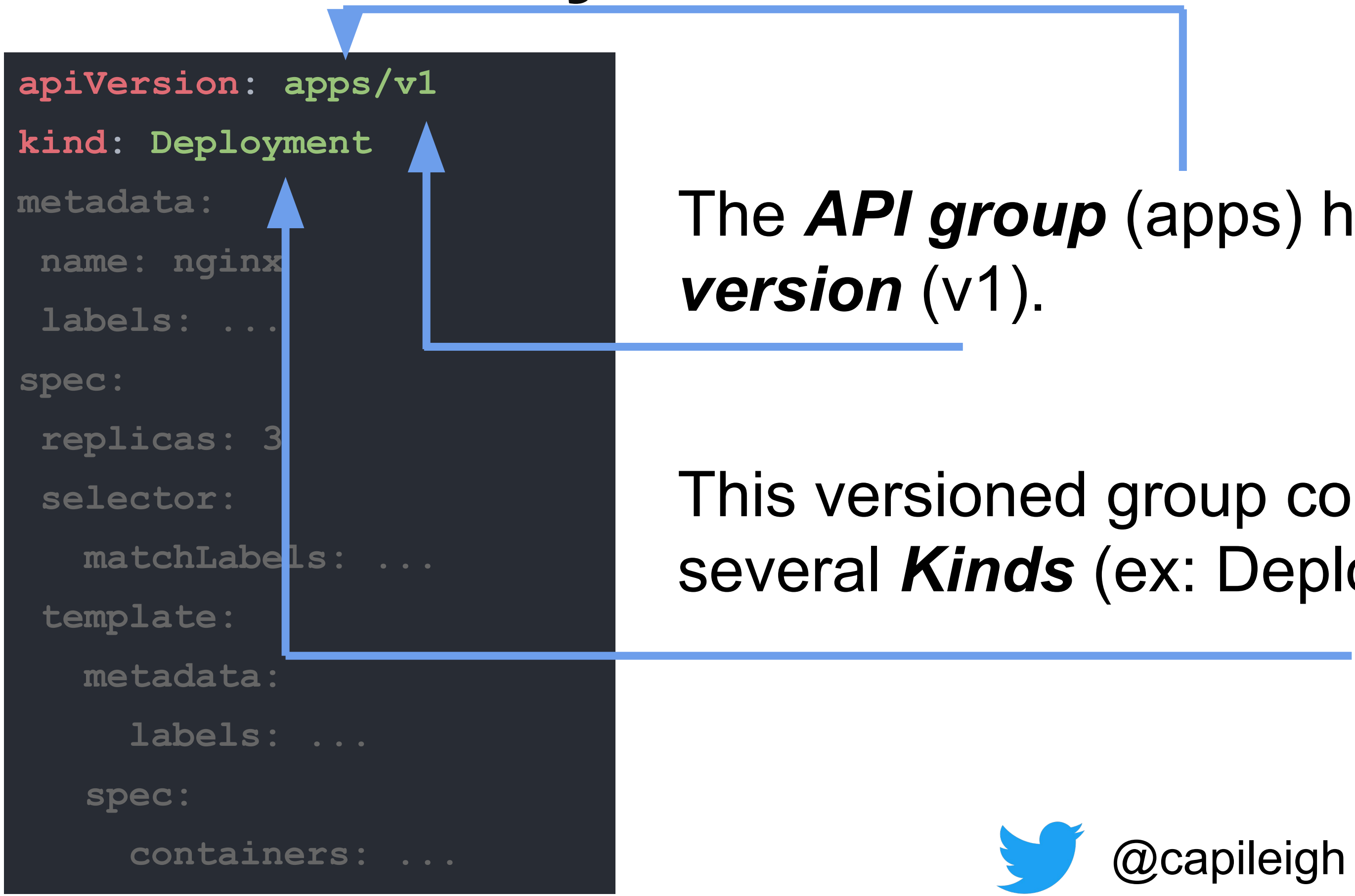


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
 labels: ...
spec:
 replicas: 3
 selector:
   matchLabels: ...
 template:
   metadata:
     labels: ...
   spec:
     containers: ...
```

Version fields can be annoying when there are already so many fields...

... but these YAML objects 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: ...
```

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

This versioned group contains several ***Kinds*** (ex: 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).

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

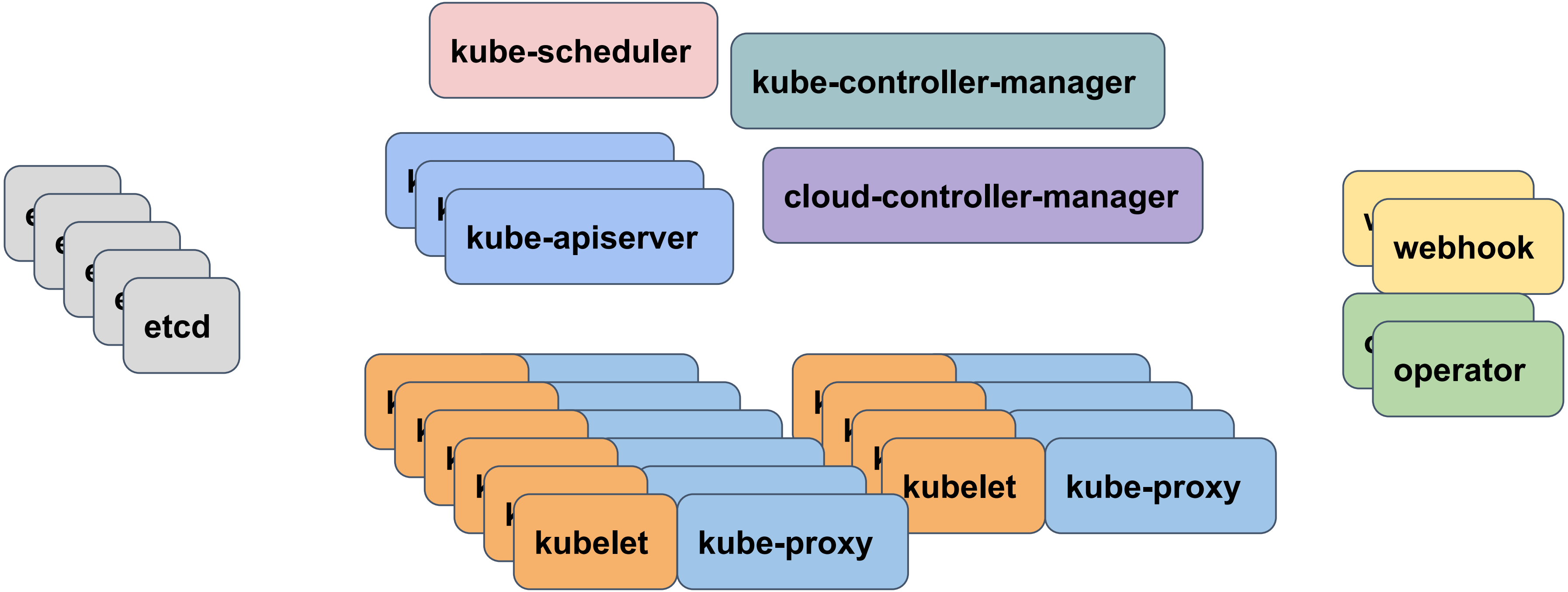
Commands can take *flags* that describe configuration.

```
$ 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

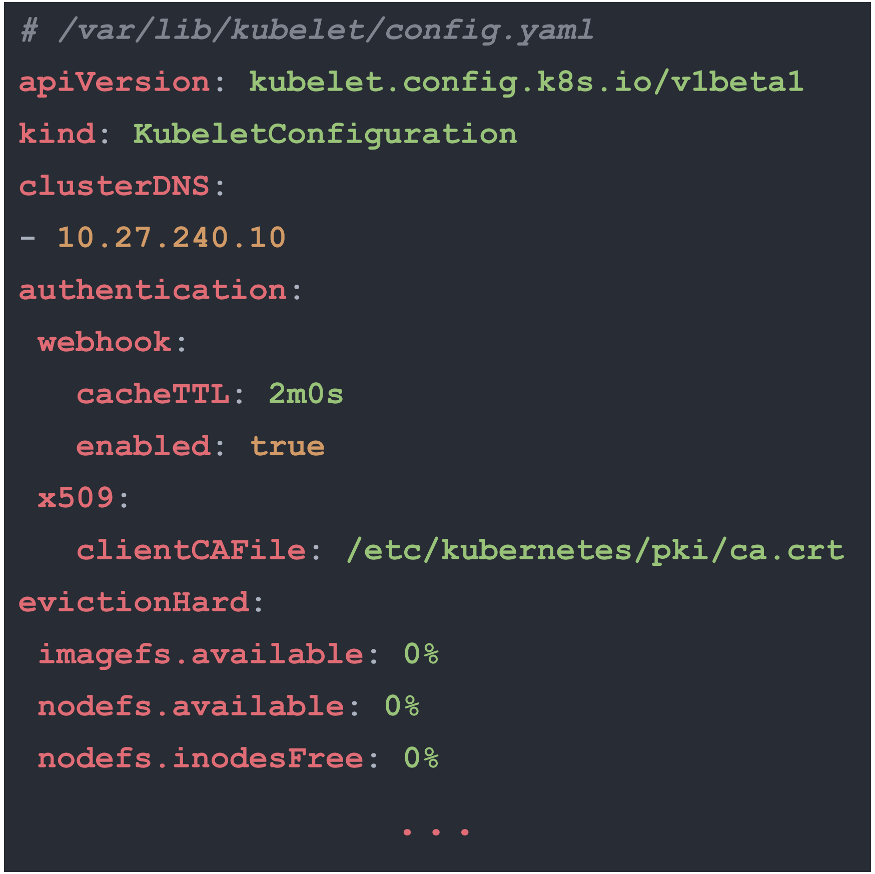
- Flags are a public API, but breaking changes are not communicated by the overall K8s version.
    - Flag breakages are *allowed* across K8s minor versions as long as warnings were logged for enough releases.
- Tools don't understand the custom structures (component-specific string parsers) built into 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!

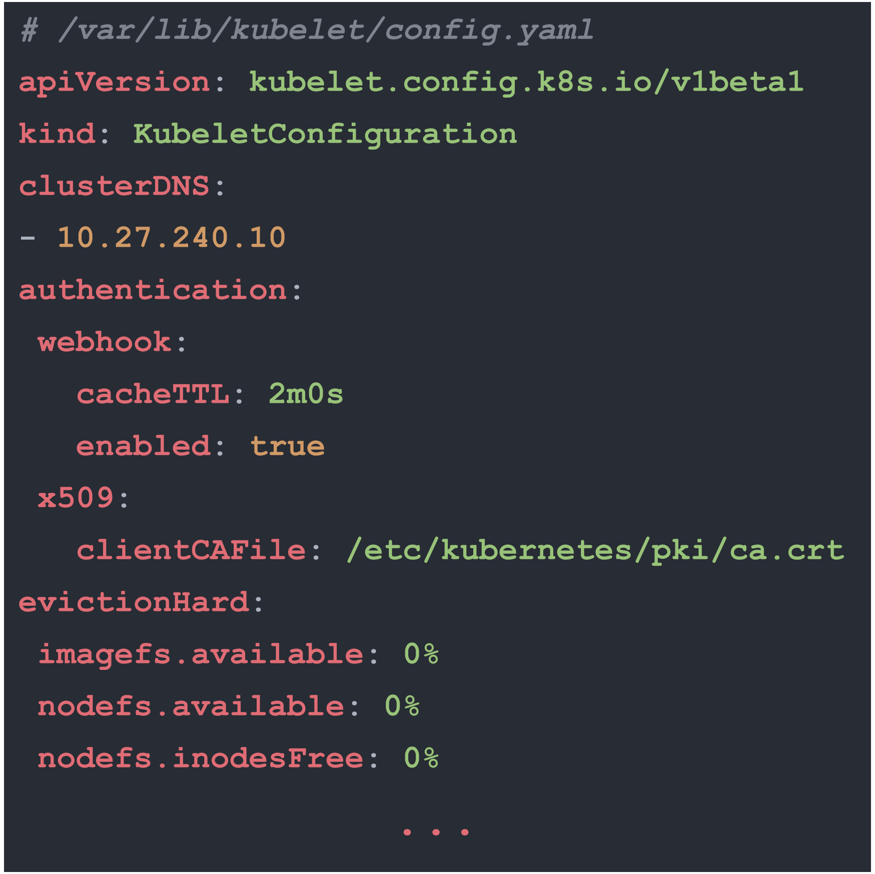
- Humans like them.
  - Readable and writable.
  - Clear stability policy.
- Tools like them.
  - Common format with wide support.
  - Avoids nonstandard structures that prevent interop.
- *Versioned schemas help everyone.*



```yaml
# /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%
                    ...
```

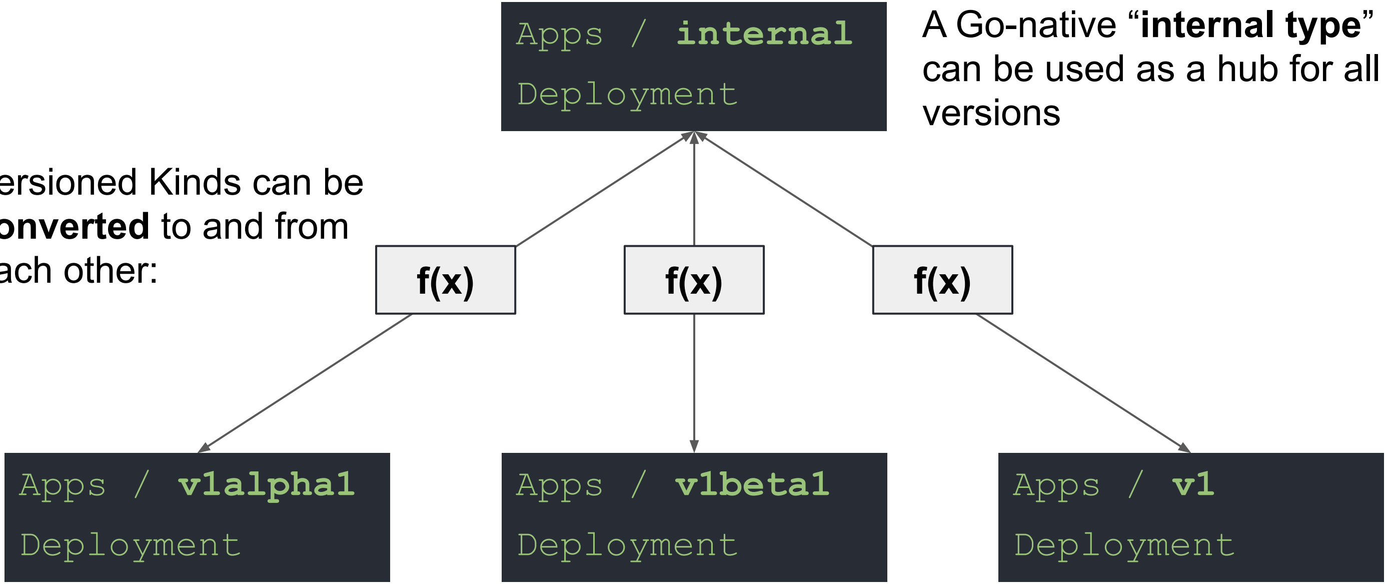
DynamicKubeletConfiguration

- Special feature built for the Kubelet
- Kubelets bootstrap their **KubeletConfiguration** ComponentConfig from the filesystem
- After connecting to the API Server, Kubelet loads a new **KubeletConfiguration** from a **ConfigMap**
- The API Server does not own the kubelet.config.k8s.io API Group. The kubelet manages it.
- Kubelet can reload the ConfigMap



```yaml
# /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%
                    . . .
```

Conversions defined in Go



A Go-native "**internal type**" can be used as a hub for all versions

Versioned Kinds can be **converted** to and from each other:

@capileigh @christopherhein

CustomResourceDefinitions

CustomResourceDefinitions

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: repositories.github.go.hein.dev
spec:
  group: github.go.hein.dev
  names:
    kind: Repository
    listKind: RepositoryList
    plural: repositories
    singular: repository
  versions:
  - name: v1alpha1
```

@capileigh @christopherhein

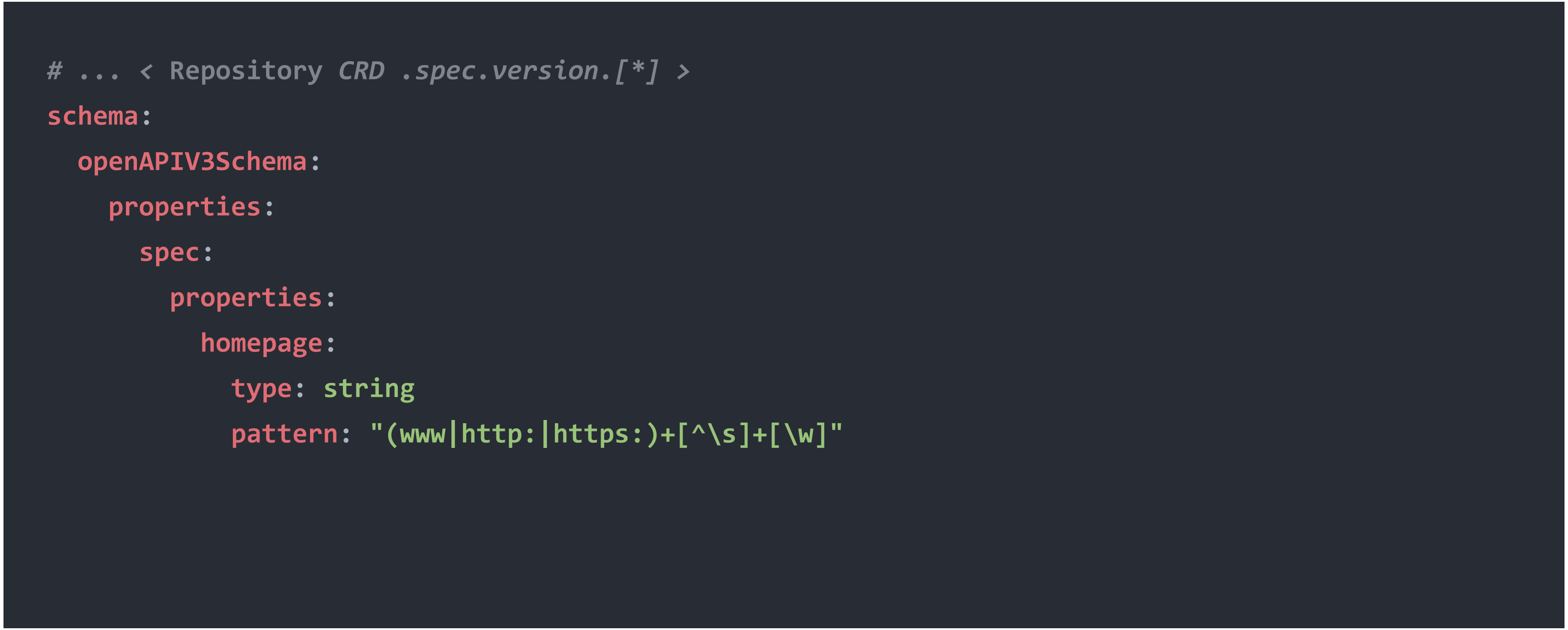
kubectl can create API’s with CRD’s

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: repositories.github.go.hein.dev
spec:
  group: github.go.hein.dev
  names:
    kind: Repository
    listKind: RepositoryList
    plural: repositories
    singular: repository
  versions:
  - name: v1alpha1
```

Applying this CRD creates REST routes in the API Server

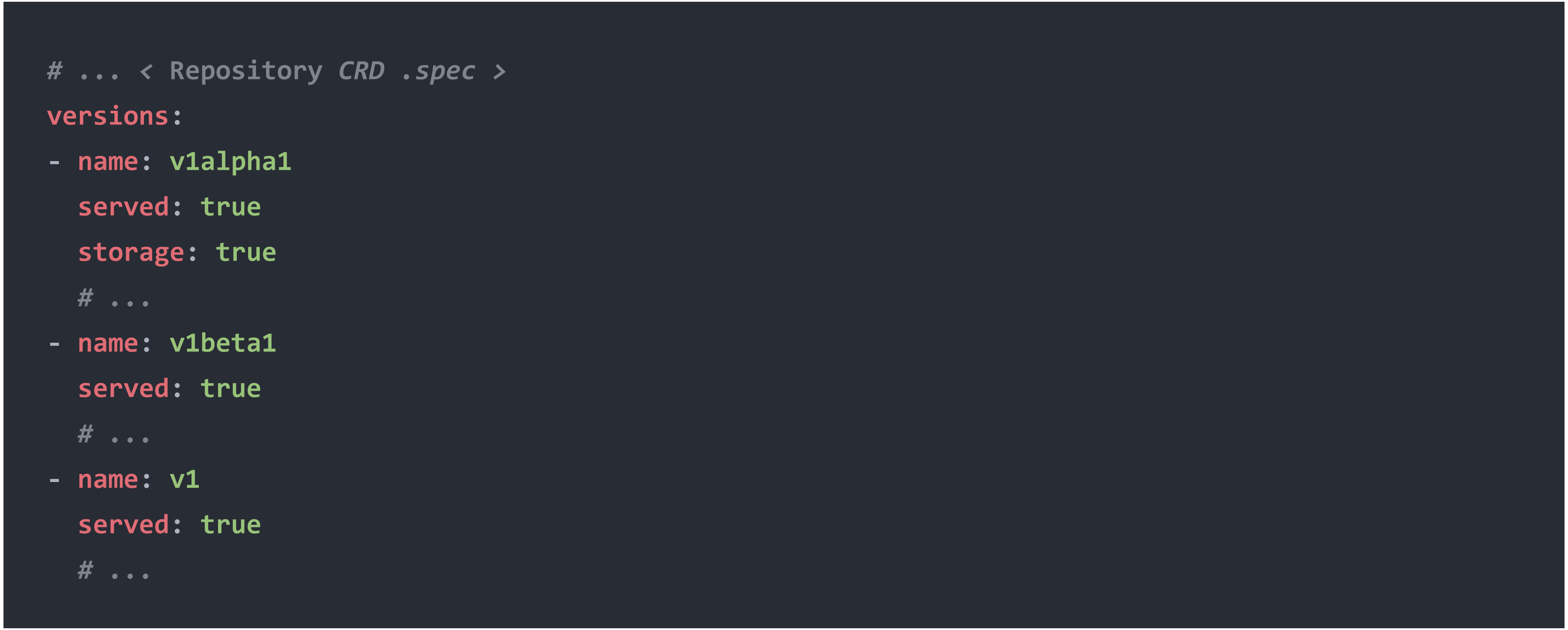
```
$ kubectl get -o yaml \
    repositories repository-sample
apiVersion: github.go.hein.dev/v1alpha1
kind: Repository
metadata:
  name: repository-sample
  selfLink: /apis/github.go.hein.dev/v1alpha1/repositories/repository-sample
data:
  organization: acme
  description: Sample Repo
```

Validation is part of the CRD spec.



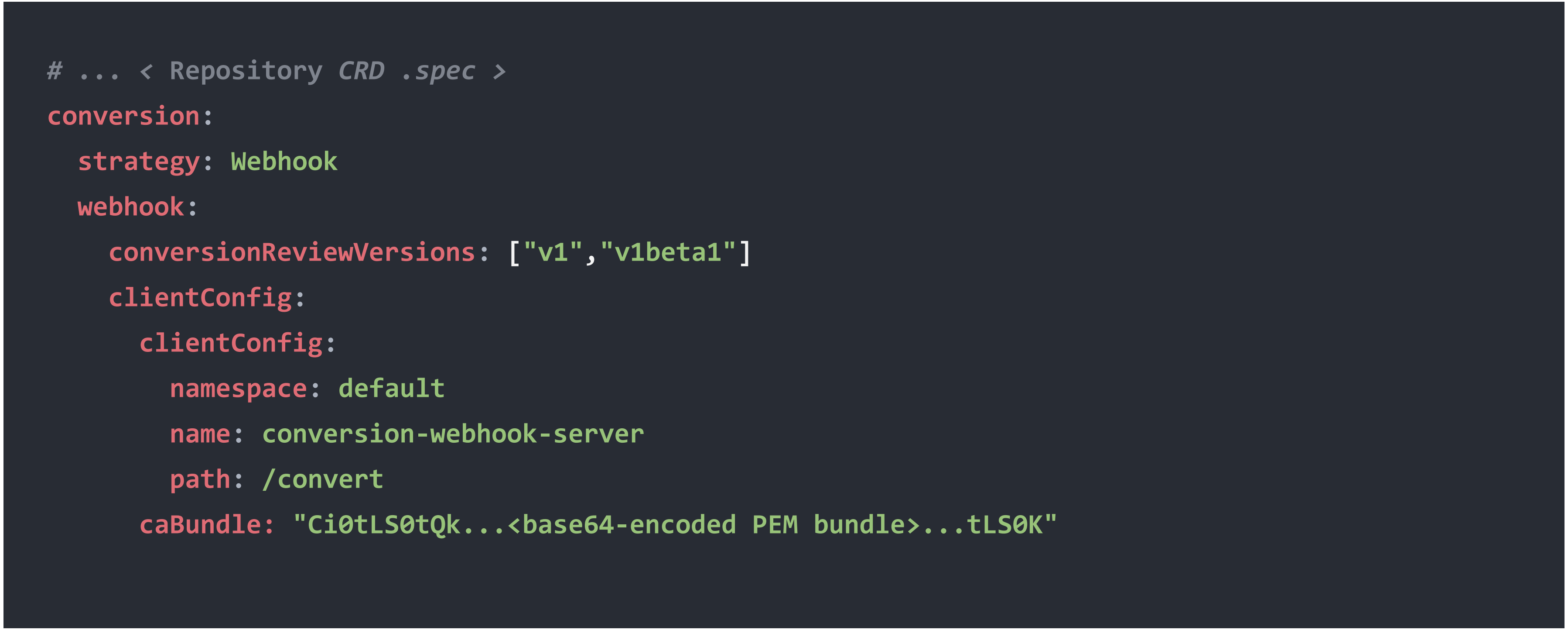
```yaml
# ... < Repository CRD .spec.version.[*] >
schema:
  openAPIV3Schema:
    properties:
      spec:
        properties:
          homepage:
            type: string
            pattern: "(www|http:|https:)+[^\s]+[\w]"
```

CRD’s can specify many Versions of the same Kind



```yaml
# ... < Repository CRD .spec >
versions:
- name: v1alpha1
  served: true
  storage: true
  # ...
- name: v1beta1
  served: true
  # ...
- name: v1
  served: true
  # ...
```

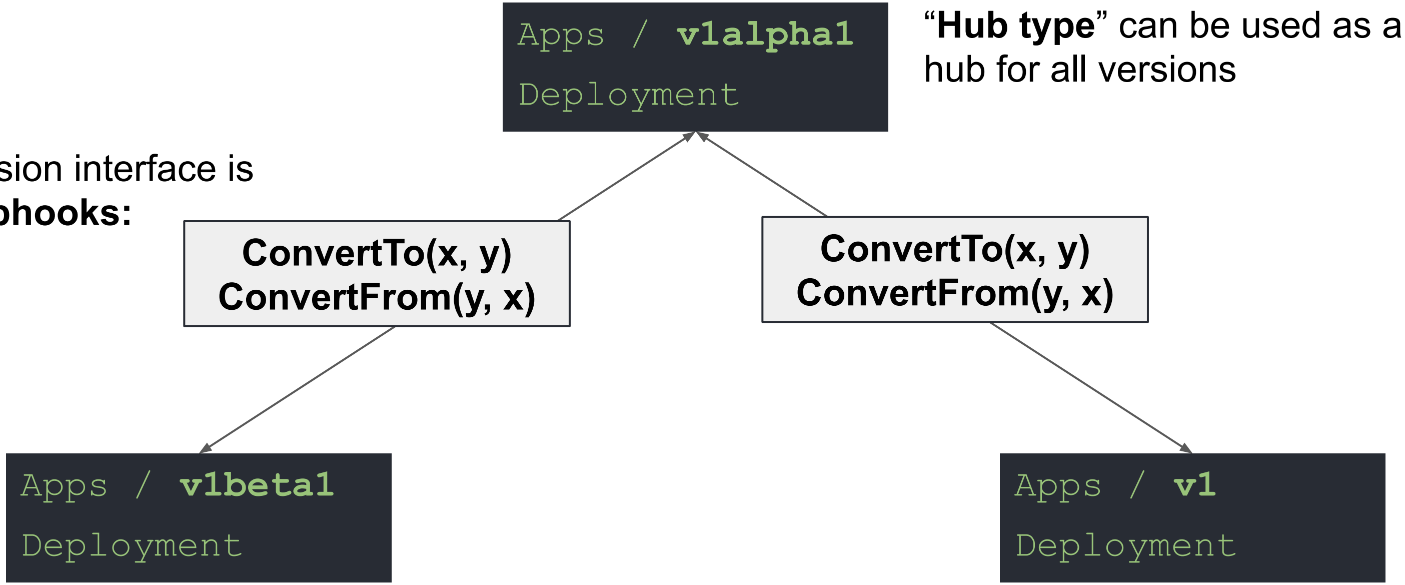
CRD Conversions are done w/ webhooks in the spec (external URL or Service)



```
# ... < Repository CRD .spec >
conversion:
  strategy: Webhook
  webhook:
    conversionReviewVersions: ["v1","v1beta1"]
    clientConfig:
      clientConfig:
        namespace: default
        name: conversion-webhook-server
        path: /convert
      caBundle: "Ci0tLS0tQk...<base64-encoded PEM bundle>...tLS0K"
```

@capileigh @christopherhein

CRD Version Conversions

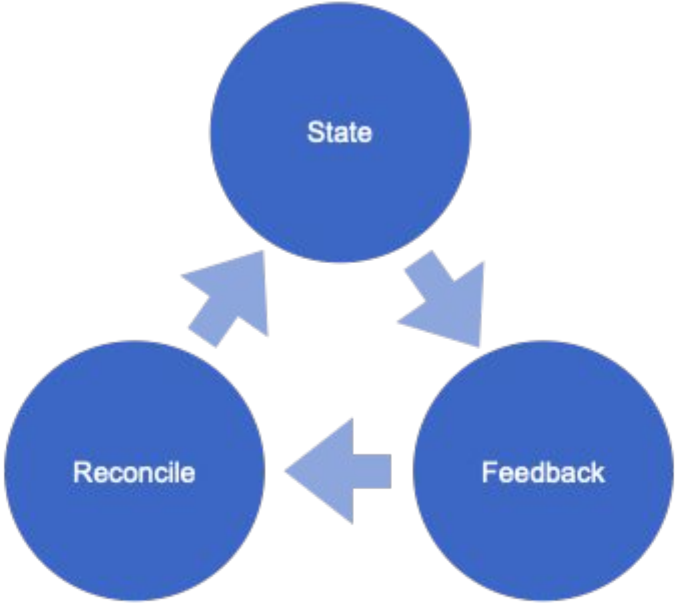


"**Hub type**" can be used as a hub for all versions

Conversion interface is via **Webhooks:**

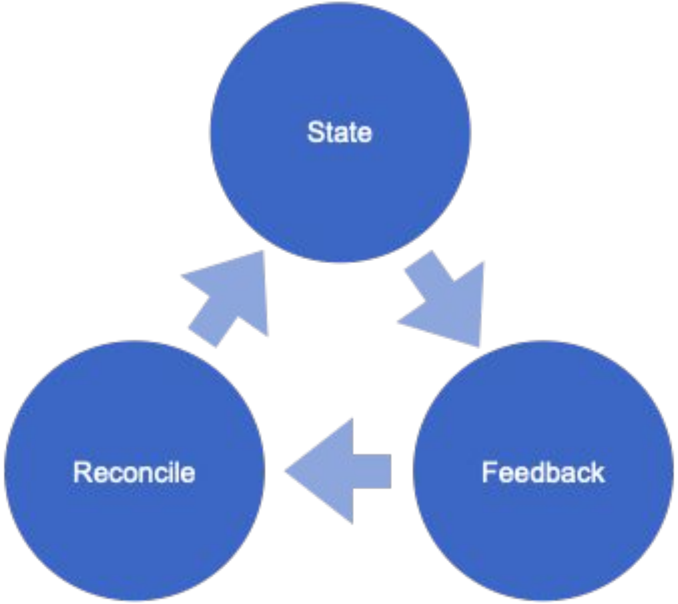
@capileigh @christopherhein

Controllers and their Value



Controllers and their Value

- Level-triggered systems that reliably converge
- K8s API watches allow for rapid reconciliation
- with k8s API extensions / CRD’s, allows us to model declarative control of error-prone operations

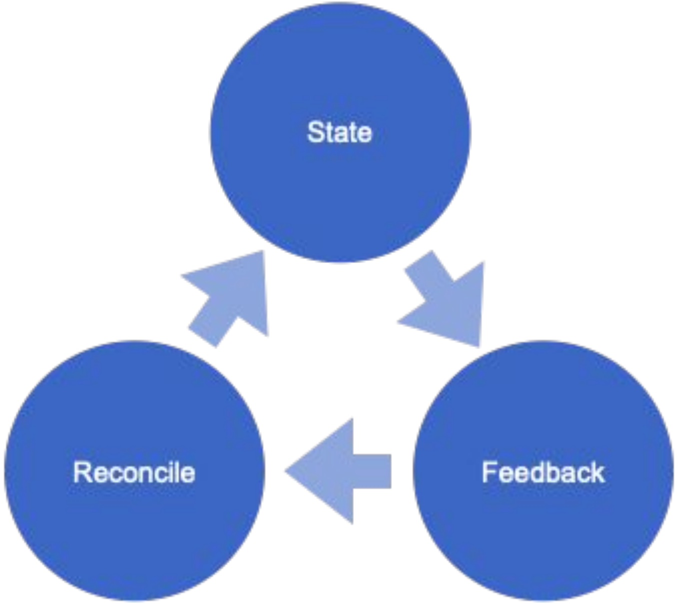


@capileigh @christopherhein

Operators

- Controllers that use Custom Resources to operate more complex systems, enforcing policies, and converging them to their desired states

- Excels at converting imperative systems into controllable, declarative ones.



Operator Overhead

- Complex solution
- Need to be deployed, often inside the same cluster
- Has config that should be managed

Command line flags

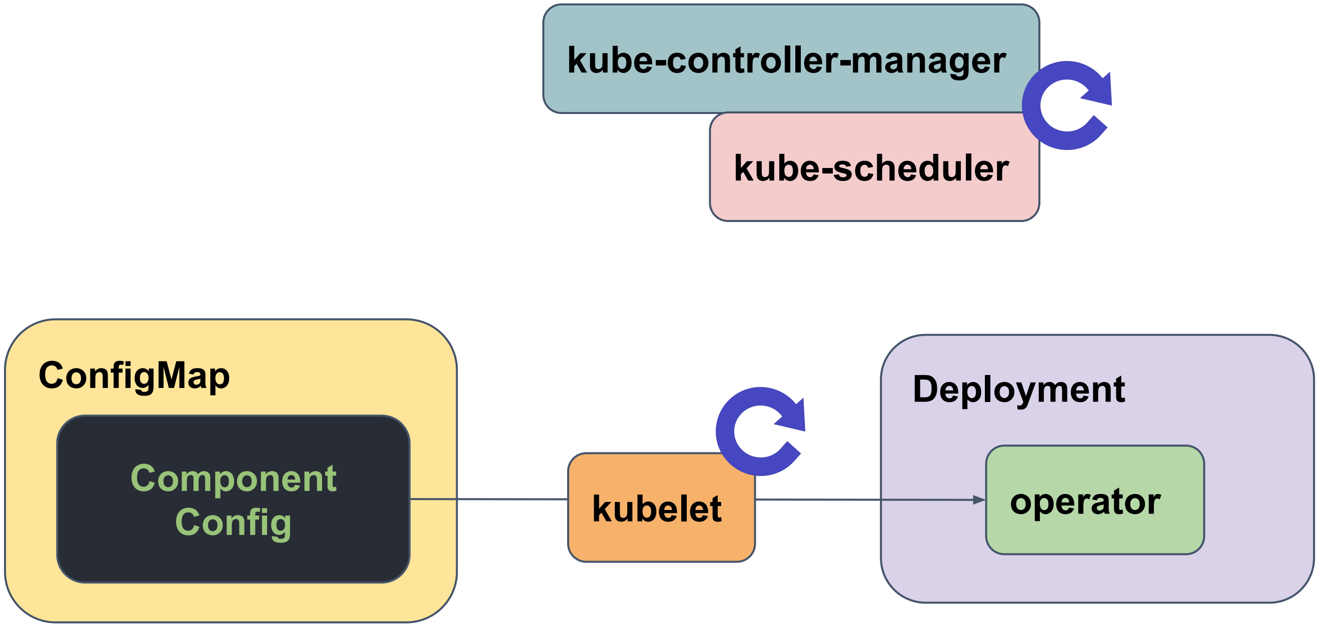
Commands can take *flags* that describe configuration.

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

The values are arbitrary strings parsed by the program.

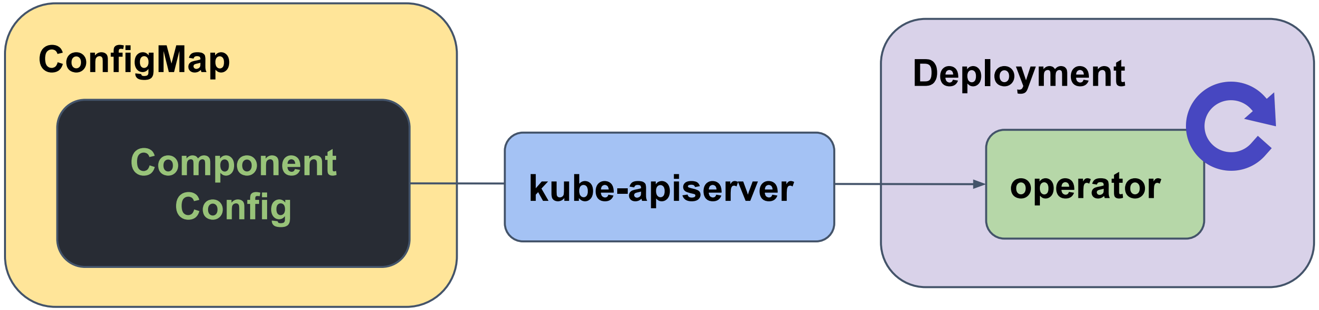
Which is fine and convenient for tools and small programs.

Flags Solution: ComponentConfig

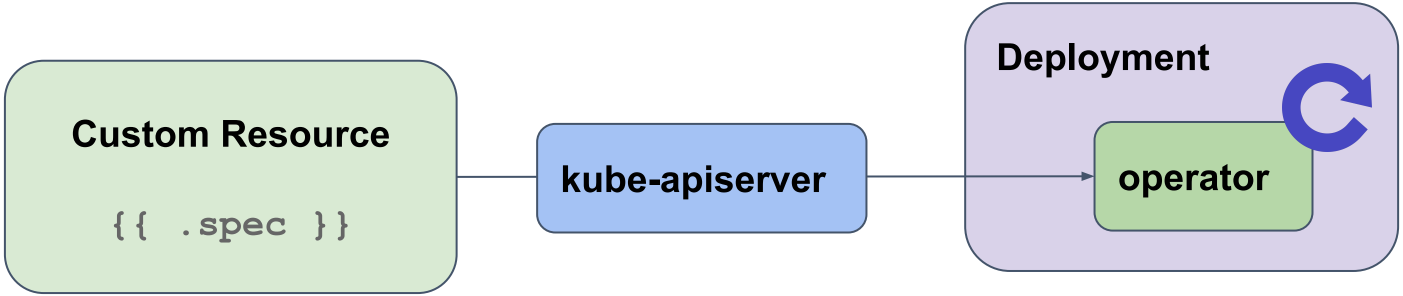


@capileigh @christopherhein

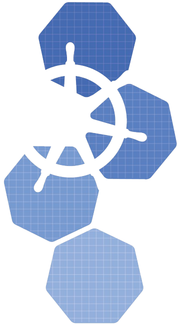
Dynamic ComponentConfig



Dynamic Config w/ Custom Resources



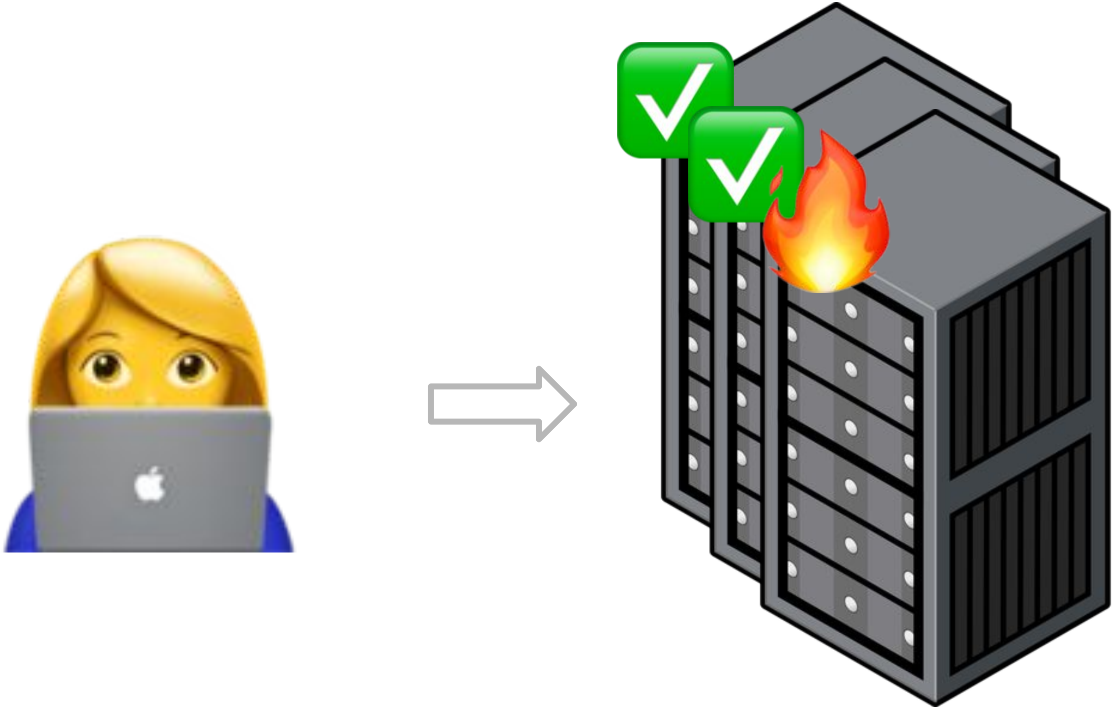
Examples



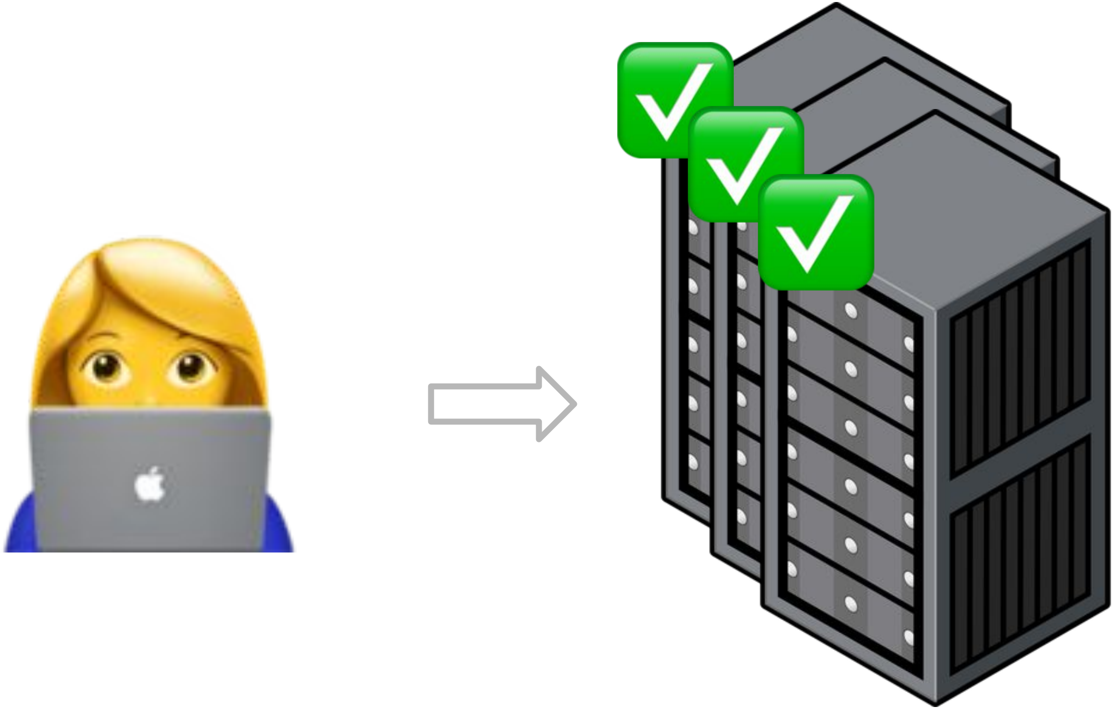
https://bit.ly/2X5v8T2

https://bit.ly/3g9CJaH

https://bit.ly/3gITIfL

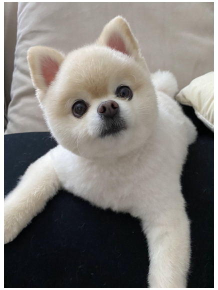


@capileigh @christopherhein



Leigh Capili

Chris Hein



(kevin)

@capileigh	@christopherhein		thedogkevin
stealthybox	christopherhein		

KubeCon | CloudNativeCon

Europe 2020

Virtual

KEEP CLOUD NATIVE CONNECTED