
KubeCon
CloudNativeCon
Europe 2025
LONDON BUS

KubeCon | CloudNativeCon
Europe 2025

Making CRDs Delightful: Beyond the Pitfalls

Evan Anderson
Stacklok

Who I Am

Knative Founder / ToC Member

Kubernetes user for 7 years

Lead on VMware Tanzu

Contributor to 10+ CNCF projects

... and a lot of other open source

Why Does UX Matter?

no, really!

does it matter?

`git` has terrible UX. Linux UX is mixed at best.
Why should we care about Kubernetes UX?

Why Does UX Matter?

Kubernetes UX is DevOps UX

Easier UX reduces barriers to participation

Difficult UX increases the chance of errors under stress

Accidental complexity vs intrinsic complexity

KubeCon | CloudNativeCon
Europe 2025

CRD Basics

☑️ Do this first

`status`: for Humans *and* Machines

`status` is where controllers tell the world about an object.

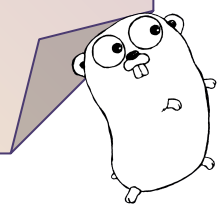
Humans:

- Descriptions
- Summaries
- Sentences
- Resources

Machines:

- Numbers
- Enums
- URLs
- Resources

```yaml
apiVersion: foo.bar/v1
kind: MyThing
metadata:
  name: …
  namespace: …
spec:
  # What the user wants
  replicas: 4
status:
  # How the thing actually is
  replicas: 2
```



status: for Humans *and* Machines

KubeCon CloudNativeCon Europe 2025

```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: …
  namespace: …
spec:
  template:
    spec:
      containers:
      - image: …
status:
  url: https://mysvc.clustername/
  latestReadyRevisionName: …
  traffic:
  - revisionName: msvc-00001
    percent: 100
```

Knative

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: …
  namespace: …
spec:
  sources: …
status:
  status: Synced
  resources:
  - kind: …
    name: …
    namespace: …
  history:
  - deployedAt: …
    # etc
  - …
```

ArgoCD

```yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: …
  namespace: …
spec:
  dnsNames: [ … ]
  secretName: …
status:
  conditions:
  - lastTransitionTime: …
    message: Certificate is up to
date and has not expired
    observedGeneration: 2
    reason: Ready
    status: "True"
    type: Ready
  notAfter: …
  notBefore: …
  renewalTime: …
```

Cert-Manager

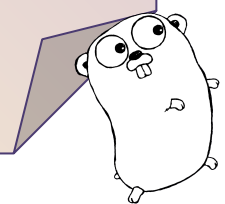
Status-free objects: Policies and Classes

Sometimes a custom resource doesn't actually change anything.

Two common use cases:

- Define a pattern or configuration for other resources.
- Define a policy which is applied to many resources.

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
---
apiVersion: gateway.networking…
kind: GatewayClass
---
apiVersion: rbac.authorization…
kind: RoleBinding
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
```



Events: Better Than Controller Logs

Events are objects that describe occurrences related to a resource.

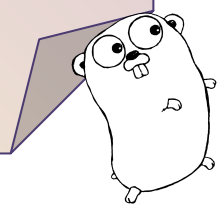
Visible to non-admin users!

State change ⇒ explicitly create event

Aged out by Kubernetes

```
$ kubectl describe ksvc hello
Events:
Type    Reason    Age   From         Message
----    ------    ----  ----         -------
Normal  Created   7m9s  controller   Created Configuration "hello"
Normal  Created   7m9s  controller   Created Route "hello"
```

```
- apiVersion: v1
  kind: Event
  metadata:
    creationTimestamp:
"2025-04-01T08:29:37Z"
    name: ping-pong.1832731dccbb23e3
    namespace: default
    resourceVersion: "76068"
    uid: 36bac17c-...-f72cc2fdb3d8
  involvedObject:
    apiVersion: serving.knative.dev/v1
    kind: Service
    name: hello
    namespace: default
    resourceVersion: "76066"
    uid: ff1c52e4-...-7aa80b48522a
  count: 1
  eventTime: null
  firstTimestamp: "2025-04-01T08:29:37Z"
  lastTimestamp: "2025-04-02T08:29:37Z"
  message: Created Configuration "hello"
  reason: Created
  reportingComponent: service-controller
  reportingInstance: ""
  source:
    component: service-controller
  type: Normal
```



Day-1 RBAC for everyone

Kubernetes ships with handy built-in cluster roles you can apply to namespaces or the cluster:

`view`, `edit`, `admin`

You can play nicely with these roles by shipping ClusterRoles annotated with:

`rbac.authorization.k8s.io/aggregate-to-$role: true`

You can also define your own for plugins to use.

```yaml
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac…/aggregate-to-admin: "true"
    rbac…/aggregate-to-edit: "true"
    rbac…/aggregate-to-view: "true"
  name: cert-manager-view
rules:
- apiGroups:
  - cert-manager.io
  resources:
  - certificates
  - certificaterequests
  - issuers
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - acme.cert-manager.io
  resources:
  - challenges
  - orders
  verbs:
  - get
  - list
  - watch
```

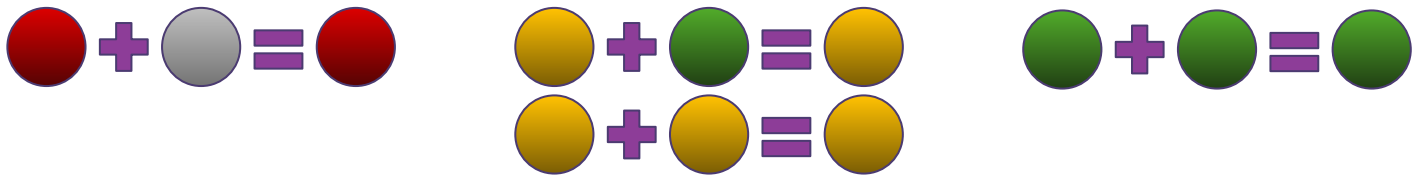
One Rule for Condition Superpowers

KubeCon CloudNativeCon Europe 2025

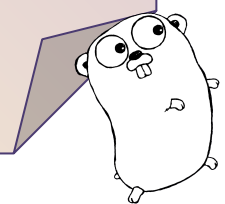
`status.condition` is a convention for providing a summary of resource status.

- Have a well-known top-level type (`Ready` or `Succeeded` are common)
- Keep all types the same “polarity” for reduced confusion.
  `true` ⇒ desired works well.

Automated summary of `Ready`:



```
status:
  conditions:
  - type: Ready  # Or “Succeeded”
    status: false
    reason: CamelCaseEnum
    message: # A sentence
    observedGeneration: 3
    lastTransitionTime: …
  - type: FooWorked
    status: true
    observedGeneration: 3
  - type: BarFetched
    status: true
    observedGeneration: 3
  - type: SyncedBaz
    status: unknown
    message: Baz sync in progress
    observedGeneration: 2
  - type: Allocated
    status: false
    message: Quota exceeded
    observedGeneration: 3
```



KubeCon | CloudNativeCon
Europe 2025

Beyond the CRD

☑️ Command Lines

How to avoid needing to build a CLI

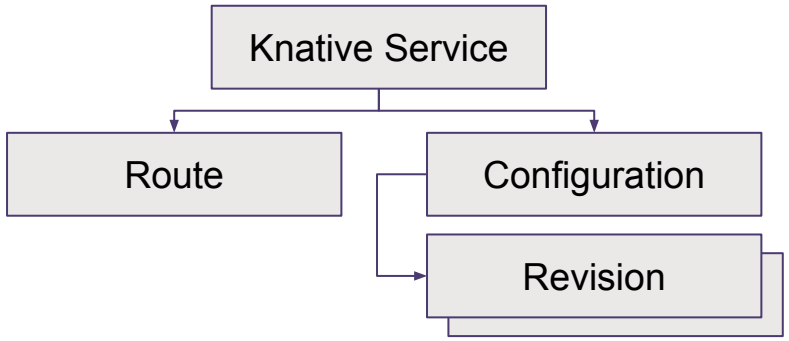
We love `kubectl`! With a little work, you can get:

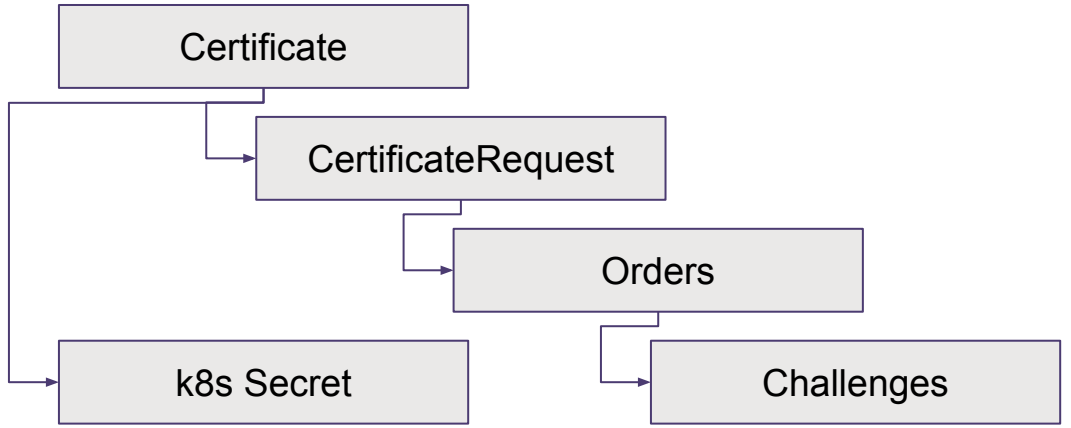
- Status reporting
  `spec.versions[*].additionalPrinterColumns`
- Short names (e.g. `app` or `cert`)
  `spec.names.shortNames`
- `kubectl get all` lists your resources
  `spec.names.categories`

How to avoid needing to build a CLI

Have a lot of coordinating resources?

Think about adding adding *even more* resources that roll up common constellations.





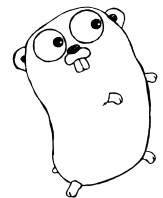

```
$ kubectl get ksvc
NAME    URL                                        LATEST         AGE   CONDITIONS   READY
hello   http://hello.default.majordemo.app         hello-00001    13s   3 OK / 3     True

$ kubectl get certs -o wide
NAME          READY   SECRET        ISSUER        STATUS                                  AGE
minder-cert   True    minder-cert   letsencrypt   Certificate is up to date and has not expired   28d
```

When to Build a CLI Anyway

You might want a CLI for:

- Orchestrating multiple operations
  This is basically a client-operated workflow.
- Highlighting or reporting status
  A CLI is a “baby GUI”. If you’re thinking you need a CLI, think about whether a GUI will also help.



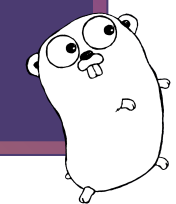
CLI examples


```
$ kn service create hello --image ghcr.io/knative/helloworld-go:latest --env TARGET=World
Creating service 'hello' in namespace 'default':

  0.049s The Route is still working to reflect the latest desired specification.
  0.108s Configuration "hello" is waiting for a Revision to become ready.
  0.145s ...
  6.662s ...
  6.730s Ingress has not yet been reconciled.
  6.772s Waiting for load balancer to be ready
  6.967s Ready to serve.

Service 'hello' created to latest revision 'hello-00001' is available at URL:
http://hello.default.127.0.0.1.sslip.io
```

```
$ flux bootstrap github --owner=user --repository=infra --branch=main --path=./clusters/my-cluster
► connecting to github.com
✔ repository created
✔ repository cloned
✚ generating manifests
✔ components manifests pushed
► installing components in flux-system namespace
...
✔ install completed
► generating sync manifests
✔ sync manifests pushed
► applying sync manifests
◎ waiting for cluster sync
✔ bootstrap finished
```



KubeCon | CloudNativeCon
Europe 2025

Advanced CRD-Foo

☑️ Count your bullets

Borrowing: Embedding Known Types

KubeCon CloudNativeCon Europe 2025

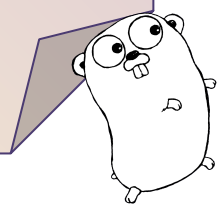
Familiarity: knowledge transfer from existing patterns

Duck Typing: manipulate several different resources using common field locations

Pass-through: don't repeat the wheel – people will eventually want each combination of options

```
<ObjectReference>:
  apiVersion:
  kind:
  namespace:
  name:
  resourceVersion:
  uid:
  fieldPath:
---
<LabelSelector>
---

---
kind: Deployment
spec:
  template: <PodTemplateSpec>
---
kind: DaemonSet
spec:
  template: <PodTemplateSpec>
---
apiVersion: serving.knative.dev/v1
kind: Service
spec:
  template: <PodTemplateSpec>
```



The Beauty of Zero

By default, Kubernetes will omit zero-valued fields. If you can, use `0` as a "reasonable default" value.

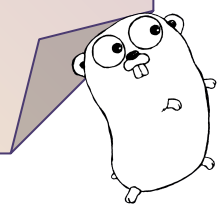
This has the benefit of making your resources shorter to read, *and* defaulting to reasonable settings.

(You probably know reasonable defaults as well the average user.)

```
concurrency: 0 → choose for me
timeout: 0     → choose for me
tag: ""        → choose for me
matches: []    → match any


in Go:

                        Units!
type Foo struct {
  timeoutSeconds int32   // Yes
  maxIdleTime    *int32  // No
}
```

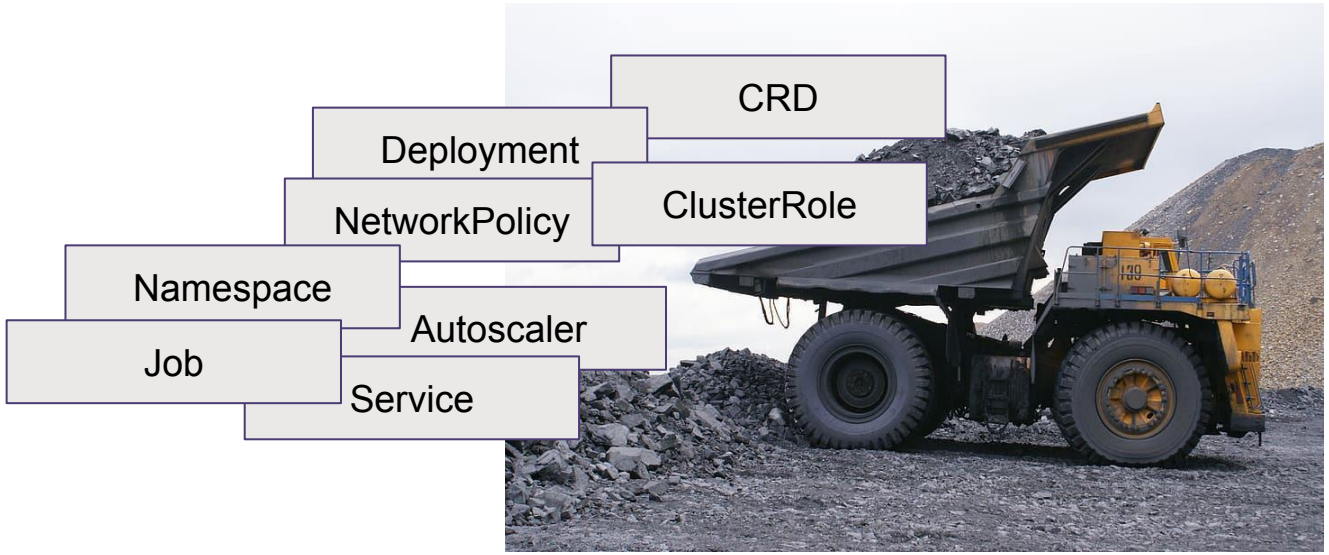


Supporting GitOps Gracefully

When your resources are run by GitOps, they tend to all be dumped in the cluster at once:

- Non-parent-child resources initialize in arbitrary order
- Any workflows need explicit sequencing
- Items need to be re-creatable

⇒ ClusterRoles and CRDs are two dependency-ordering examples of breaking GitOps



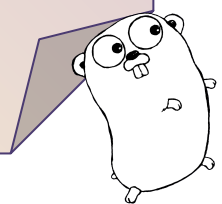
Operating Someone Else's CRD

KubeCon CloudNativeCon Europe 2025

Problem: how to extend resources that are defined by another project?

- `labels`: short values, indexed
- `annotatons`: longer values with data
- `selectors`: re-use existing labels
- `ownerReferences`: manage external resources

```
apiVersion:
gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: shared
  namespace: envoy-gateway-system
  annotations:
    cert-manager.io/cluster-issuer: foo
```



CRD Authoring Checklist

- ☑️ Do you need a top-level object?
- ☑️ `status` should let me skip your docs
- ☑️ Define `status.conditions`
- ☑️ Explicitly create Events on state changes
- ☑️ Link to sub-resources that are created
- ☑️ Define aggregated ClusterRoles
- ☑️ Define `kubectl` metadata:
  - ☑️ shortnames and categories
  - ☑️ status columns
- ☑️ Reuse! Can you embed existing types?
- ☑️ Make empty / zero meaningful
- ☑️ Enable up-front validation with OpenAPI, CEL, or webhooks

Thank you!

KubeCon CloudNativeCon Europe 2025

https://www.linkedin.com/in/evankanderson

KubeCon | CloudNativeCon

Europe 2025