Clean Up Your Room!

What Does It Mean to Delete Something in K8s

Aaron Alpar

kasten

Introduction ...

- Deleting can be difficult …
- What properties on a resource govern deletion?
    - finalizers and owner references
- Demonstrate how it works.
- This is a short presentation … the goal is to inform, and inspire …

    … I recommend you experiment on a test cluster

Kubernetes KRUD

- Create: `create`, ...
- Read: `get`, `describe`, `logs`, ...
- Update: `label`, `edit`, `label`, `annotate`, `patch`, …
- Delete: `**delete**`, …

- `delete` sometimes removes more than what you want

  … sometimes removes less of what you want.

Kubernetes KRUD

- Create: `**create**`, ...
- Read: `**get**`, `describe`, `logs`, ...
- Update: `label`, `edit`, `label`, `annotate`, `**patch**`, …
- Delete: `**delete**`

- All examples use configmaps
- All examples use a unix shell

Basic delete ...

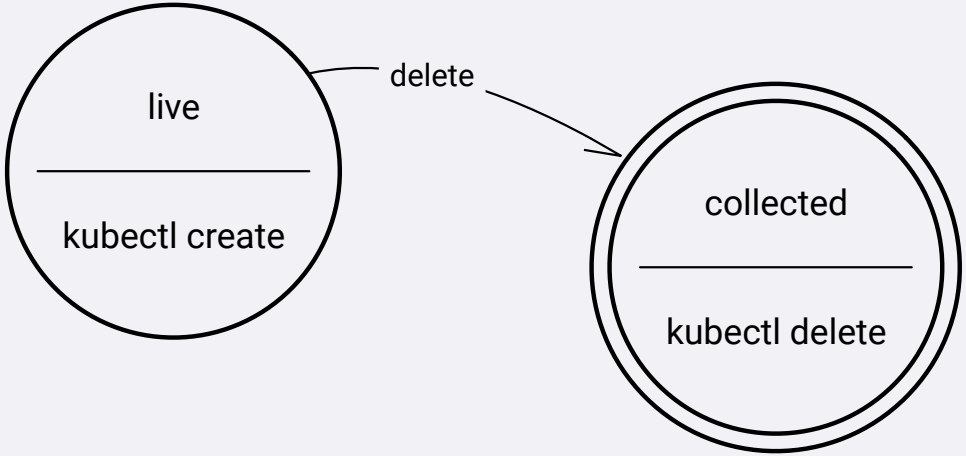
```
$ kubectl create configmap mymap
configmap/mymap created

$ kubectl get configmap/mymap
NAME    DATA   AGE
mymap   0      12s

$ kubectl delete configmap/mymap
configmap "mymap" deleted

$ kubectl get configmap/mymap
Error from server (NotFound): configmaps "mymap" not found
```

Basic delete ...



Finalizers

- Keys on resources that signal pre-delete operations
  - Controllers are responsible for running finalizers
- ”dead” finalizers can prevent deletion
- Knowledge of a finalizers application is helpful

Finalizers

```
$ cat <<EOF | kubectl create -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: mymap
  finalizers:
  - kubernetes
EOF
```

Finalizers

```
$ cat <<EOF | kubectl create -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: mymap
  finalizers:
  - kubernetes
EOF
```

Finalizers

```
$ cat <<EOF | kubectl create -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: mymap
  finalizers:
  - kubernetes
EOF
```

Finalizers

```
$ kubectl get configmap/mymap -o yaml
apiVersion: v1
kind: ConfigMap
metadata:
  creationTimestamp: "2020-10-22T21:30:18Z"
  finalizers:
  - kubernetes
  name: mymap
  namespace: default
  resourceVersion: "311456"
  selfLink: /api/v1/namespaces/default/configmaps/mymap
  uid: 93a37fed-23e3-45e8-b6ee-b2521db81638
```

Finalizers

```
$ kubectl get configmap/mymap -o yaml
apiVersion: v1
kind: ConfigMap
metadata:
  creationTimestamp: "2020-10-22T21:30:18Z"
  finalizers:
  - kubernetes
  name: mymap
  namespace: default
  resourceVersion: "311456"
  selfLink: /api/v1/namespaces/default/configmaps/mymap
  uid: 93a37fed-23e3-45e8-b6ee-b2521db81638
```

Finalizers

```
$ kubectl delete configmap/mymap &
configmap "mymap" deleted

$ jobs
[1]+  Running kubectl delete configmap/mymap
```

Finalizers

```
$ kubectl get configmap/mymap -o yaml
apiVersion: v1
kind: ConfigMap
metadata:
  creationTimestamp: "2020-10-22T21:30:18Z"
  deletionGracePeriodSeconds: 0
  deletionTimestamp: "2020-10-22T21:30:34Z"
  finalizers:
  - kubernetes
  name: mymap
  namespace: default
  resourceVersion: "311456"
  selfLink: /api/v1/namespaces/default/configmaps/mymap
  uid: 93a37fed-23e3-45e8-b6ee-b2521db81638
```

Finalizers

```
$ kubectl get configmap/mymap -o yaml
apiVersion: v1
kind: ConfigMap
metadata:
  creationTimestamp: "2020-10-22T21:30:18Z"
  deletionGracePeriodSeconds: 0
  deletionTimestamp: "2020-10-22T21:30:34Z"
  finalizers:
  - kubernetes
  name: mymap
  namespace: default
  resourceVersion: "311456"
  selfLink: /api/v1/namespaces/default/configmaps/mymap
  uid: 93a37fed-23e3-45e8-b6ee-b2521db81638
```

Finalizers

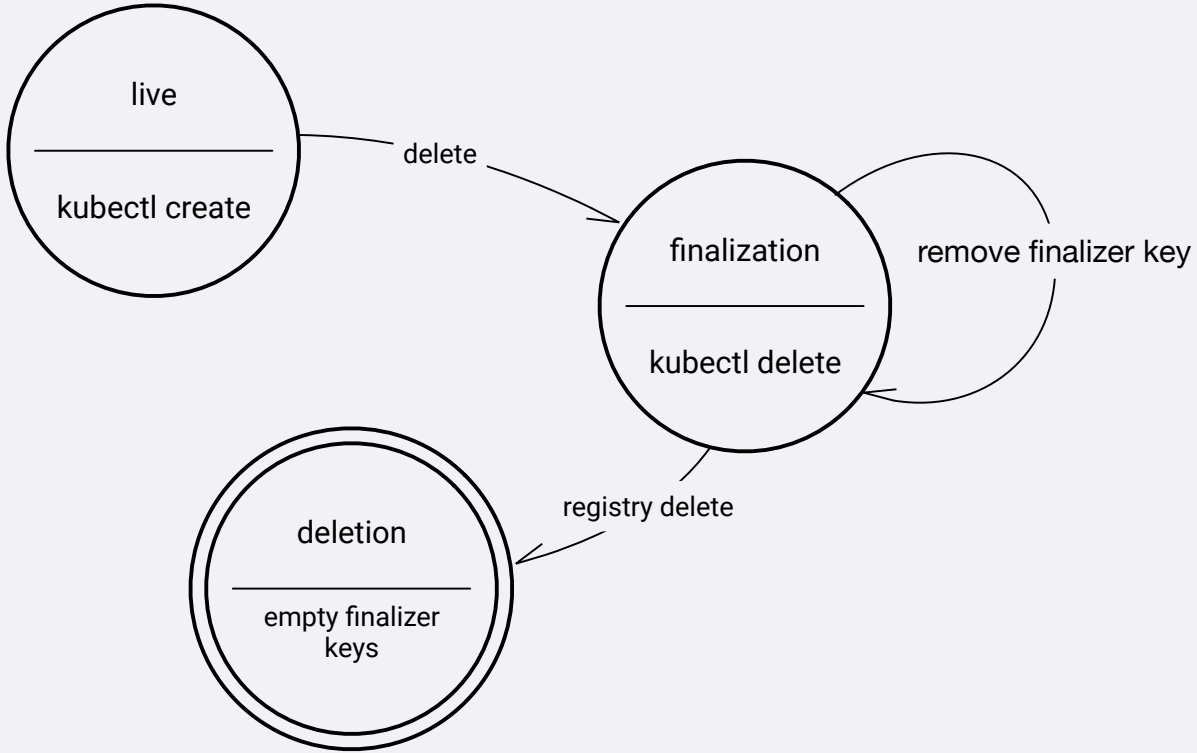
```
$ kubectl get configmap/mymap -o yaml
apiVersion: v1
kind: ConfigMap
metadata:
  creationTimestamp: "2020-10-22T21:30:18Z"
  deletionGracePeriodSeconds: 0
  deletionTimestamp: "2020-10-22T21:30:34Z"
  finalizers:
  - kubernetes
  name: mymap
  namespace: default
  resourceVersion: "311456"
  selfLink: /api/v1/namespaces/default/configmaps/mymap
  uid: 93a37fed-23e3-45e8-b6ee-b2521db81638
```

Finalizers

```
$ kubectl patch configmap/mymap \
    --type json \
    --patch='[ { "op": "remove", "path": "/metadata/finalizers" } ]'
configmap/mymap patched
[1]+  Done  kubectl delete configmap/mymap

$ kubectl get configmap/mymap -o yaml
Error from server (NotFound): configmaps "mymap" not found
```

Finalizers


live
kubectl create
delete
finalization
kubectl delete
remove finalizer key
registry delete
deletion
empty finalizer keys

Finalizers

- Deletion of an object with a finalizer will add a `deletionTimestamp` making it a read-only object
  - With the exception of removing finalizers
- Finalizers are just keys
- Controllers can add and remove finalizers
- kubectl can add and remove finalizers
- Some common finalizers:
  - kubernetes.io/pv-protection kubernetes
    kubernetes.io/pvc-protection foregroundDeletion

Owner References

- Child resource objects can reference parents
- Trees of resources may be deleted
- Finalizer rules are followed
- `ownerReference` field
- ReplicaSets are owner of Pods

Owner References

```
$ cat <<EOF | kubectl create -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: mymap-parent
EOF
```

Owner References

```
$ CM_UID=$(kubectl get configmap mymap-parent -o jsonpath="{.metadata.uid}")
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: mymap-child
  ownerReferences:
  - apiVersion: v1
    kind: ConfigMap
    name: mymap-parent
    uid: $CM_UID
EOF
```

Owner References

```
$ CM_UID=$(kubectl get configmap mymap-parent -o jsonpath="{.metadata.uid}")
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: mymap-child
  ownerReferences:
  - apiVersion: v1
    kind: ConfigMap
    name: mymap-parent
    uid: $CM_UID
EOF
```

Owner References - child deletion

```
$ kubectl get configmap
NAME            DATA    AGE
mymap-child     0       12m4s
mymap-parent    0       12m4s

$ kubectl delete configmap/mymap-child
configmap "mymap-child" deleted

$ kubectl get configmap
NAME            DATA    AGE
mymap-parent    0       12m10s
```

Owner References - parent deletion

```
$ kubectl get configmap
NAME            DATA    AGE
mymap-child     0       10m2s
mymap-parent    0       10m2s

$ kubectl delete configmap/mymap-parent
configmap "mymap-parent" deleted

$ kubectl get configmap
No resources found in default namespace.
```

Owner References - cascade

```
$ kubectl get configmap
NAME            DATA    AGE
mymap-child     0       13m8s
mymap-parent    0       13m8s

$ kubectl delete --cascade=false configmap/mymap-parent
configmap "mymap-parent" deleted

$ kubectl get configmap
NAME           DATA    AGE
mymap-child    0       13m21s
```

Owner References - cascade

- `metadata.ownerReferences` tells Kubernetes which objects own the referrer (child)
- The "`--cascade=false`" option for "`kubectl delete`" can be used to delete and object and orphan its children.

Owner References - propagationPolicy

```
$ kubectl proxy --port=8080 &
Starting to serve on 127.0.0.1:8080
```

Owner References - propagationPolicy

```
$ curl -X DELETE \
localhost:8080/api/v1/namespaces/default/configmaps/mymap-parent \
  -d '{
"kind":"DeleteOptions",
"apiVersion":"v1",
"propagationPolicy":"Background"
}' \
  -H "Content-Type: application/json"
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {},
  "status": "Success",
  "details": { ... }
```

Owner References - propagationPolicy

```
$ curl -X DELETE \
localhost:8080/api/v1/namespaces/default/configmaps/mymap-parent \
  -d '{
"kind":"DeleteOptions",
"apiVersion":"v1",
"propagationPolicy":"Background"
}' \
  -H "Content-Type: application/json"
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {},
  "status": "Success",
  "details": { ... }
```

Owner References - propagationPolicy

- Foreground: children are deleted before parent (post-order)
- Background: parent is deleted before children (pre-order)
- orphan: owner references are ignored
    - Deleting a parent will leave children intact

Forcing finalization

```
$ cat <<EOF | curl -X PUT \
  localhost:8080/api/v1/namespaces/test/finalize \
  -H "Content-Type: application/json" \
  --data-binary @-
{
  "kind": "Namespace",
  "apiVersion": "v1",
  "metadata": {
    "name": "test"
  },
  "spec": {
    "finalizers": null,
  }
}
EOF
```

Summary

- Finalizers will prevent deletion of resources
- Generally there are reasons for a finalizers presence. Investigate first
- `ownerReferences` allow trees of resources to be removed
- `propogationPolicy` can change which order resources are deleted

References …

https://kubernetes.io/docs/concepts/workloads/controllers/garbage-collection

https://book.kubebuilder.io/reference/using-finalizers.html

https://kubernetes.io/docs/concepts/storage/persistent-volumes/#storage-object-in-use-protection

https://thenewstack.io/deletion-garbage-collection-kubernetes-objects