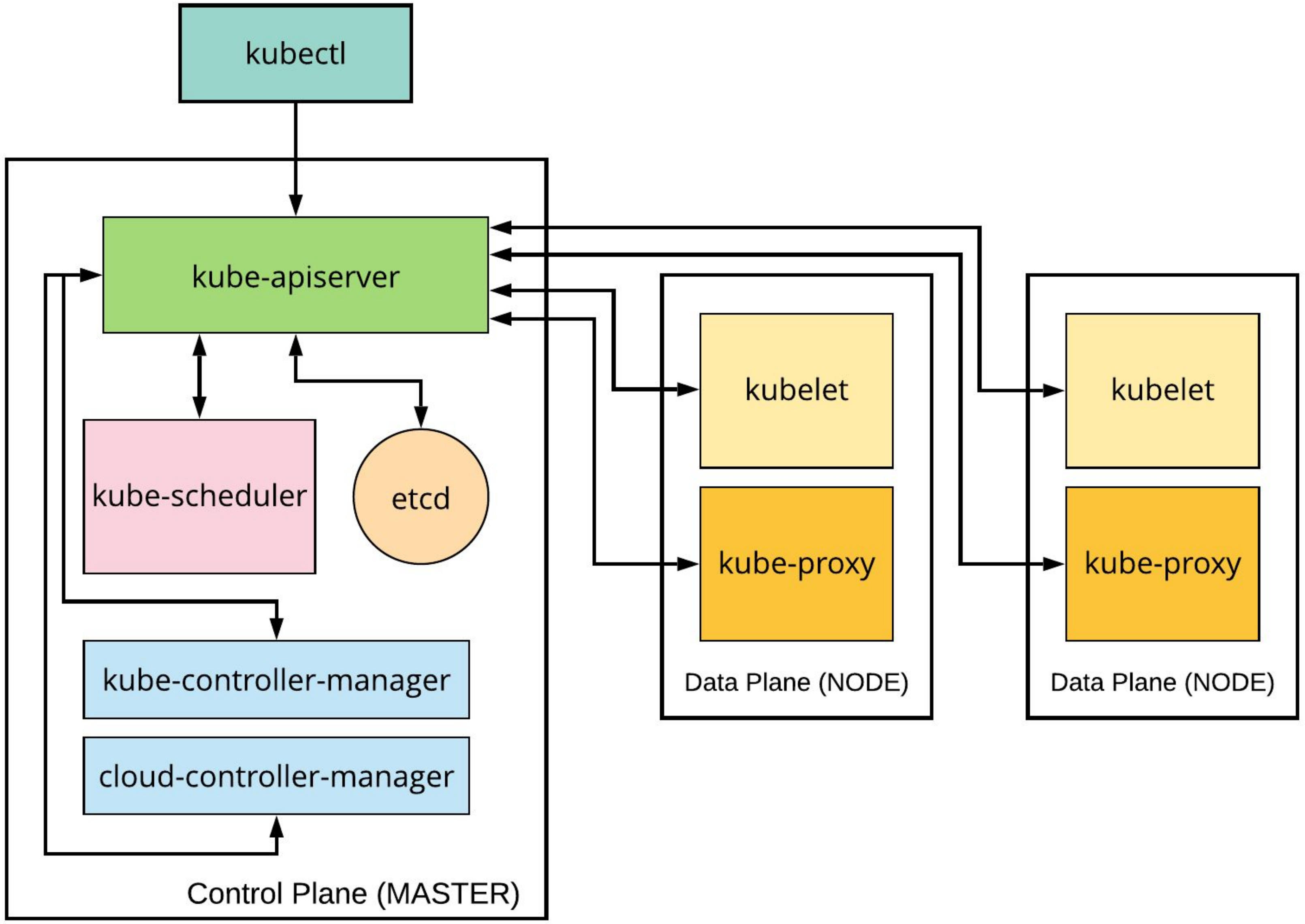
What we'll discuss today

Kubernetes API

- API Structure
- Optimistic Concurrency
- Versioning: Alpha, Beta, and Stable
- GroupVersionKind and GroupVersionResource
- Metadata, Spec, and Status
- Subcommands, API Actions, and HTTP Methods
- Deletions and Garbage Collection
- API Command Line Interaction

An Operator takes advantage of what Kubernetes does best

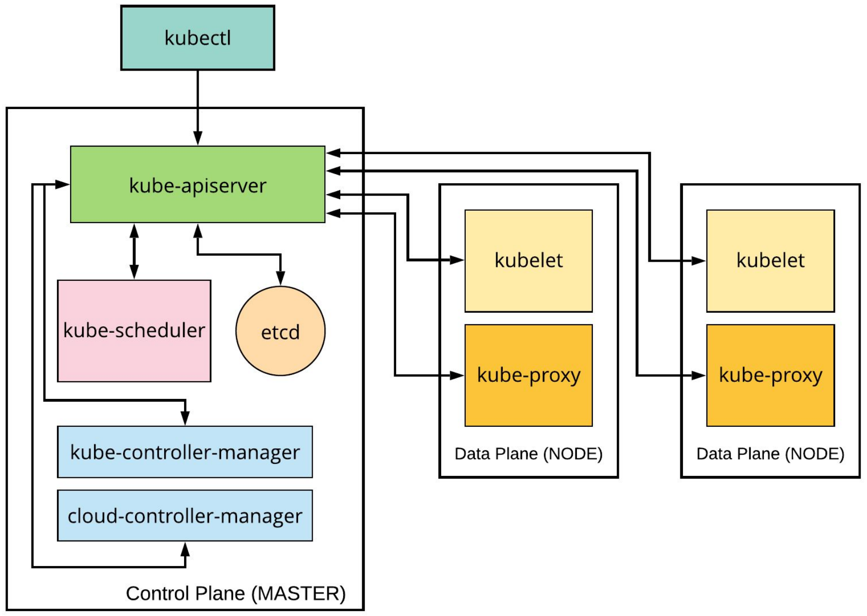
Kubernetes API



Source:
https://medium.com/@karthikeyan_krishnaswamy/overview-of-kubernetes-34d8e0e59b26

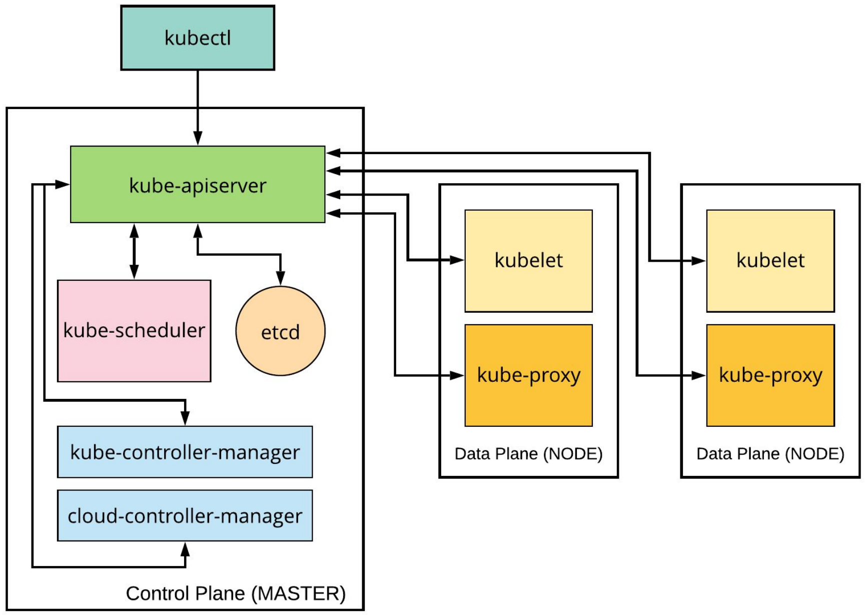
Control Plane (Master)

- kube-apiserver
  - the only component that all other master and worker components directly communicate with.
  - validates and configures data for the api objects which include pods, services, deployments, and others.
- kube-scheduler
  - responsible for managing the scheduling of pods.
- kube-controller-manager
  - embeds the core control loops shipped with Kubernetes.
  - performs cluster-level functions like keeping track of workers and handling node failures.
- cloud-controller-manager
  - embeds the cloud specific control loops shipped with Kubernetes.
- etcd
  - distributed data store that persistently stores the cluster configuration.



Data Plane (Worker)

- kube-proxy
  - network proxy that runs on each node.
  - can do simple TCP, UDP, and SCTP stream forwarding or round robin TCP, UDP, and SCTP forwarding across a set of backends.
  - reflects services as defined in the Kubernetes API on each node.
- kubelet
  - primary agent running on the node and registers the node with the API server.
  - ensures containers described in PodSpecs are running.
  - able to receive container manifests via File, HTTP endpoint, HTTP server
- container runtime
  - cri-o
  - rkt
  - containerd



Command Line Interaction

oc proxy --port=8080

curl http://127.0.0.1:8080/apis/batch/v1

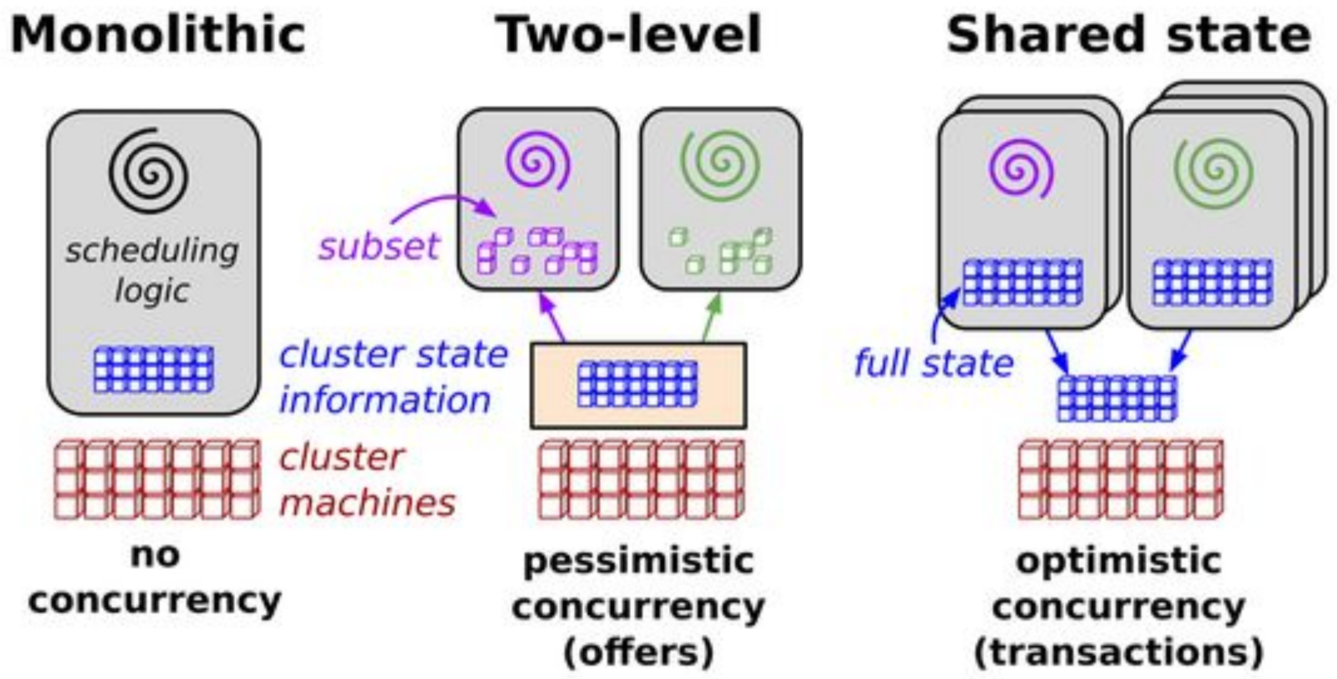
oc get --raw /apis/batch/v1

oc api-resources

oc api-versions

* kubectl and oc are essentially the same (oc utilizes the packages of kubectl)

Concurrency


Monolithic
Two-level
Shared state
scheduling logic
subset
cluster state information
cluster machines
full state
no concurrency
pessimistic concurrency (offers)
optimistic concurrency (transactions)

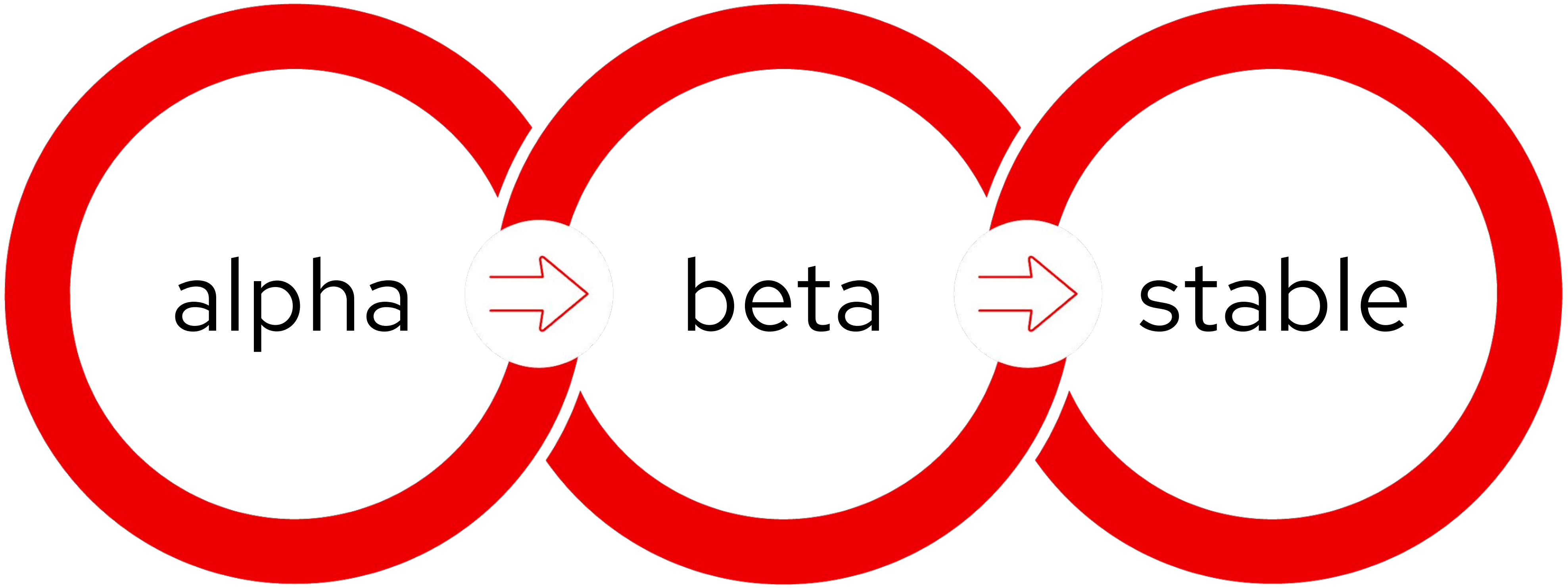
Pessimistic Concurrency

- conflicts will happen often
- uses locks

Optimistic Concurrency

- we believe conflicts will not happen
- if they happen react in some way

API Versioning



alpha

The version names contain alpha (e.g. v1alpha1).

May be buggy. Enabling the feature may expose bugs. Disabled by default.

Support for feature may be dropped at any time **without notice**.

The API may change in incompatible ways in a later software release **without notice**.

Recommended for use only in short-lived testing clusters, due to the increased risk of bugs and lack of long-term support.

https://kubernetes.io/docs/concepts/overview/kubernetes-api/#api-versioning

beta

The version names contain beta (e.g. v2beta3).

Code is well tested. Enabling the feature is considered safe. **Enabled by default**.

Support for the overall feature will not be dropped, though details may change.

The schema and/or semantics of objects may change in incompatible ways in a subsequent beta or stable release.

Recommended for only **non-business-critical uses**.

https://kubernetes.io/docs/concepts/overview/kubernetes-api/#api-versioning

stable

The version name is vX where X is an integer.

Stable versions of features will appear in released software for many subsequent versions.

https://kubernetes.io/docs/concepts/overview/kubernetes-api/#api-versioning

GroupVersionKind or GVK

```
apiVersion: batch/v1
kind: Job
```

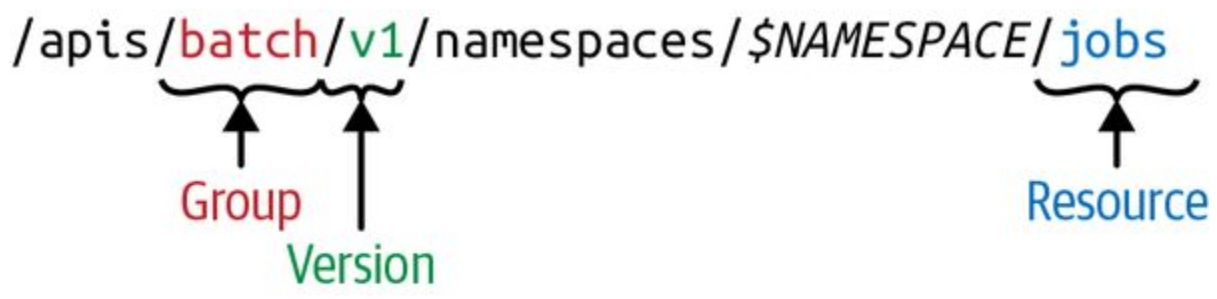
The entity **Group** is similar to package in a language. It disambiguates different APIs that may happen to have identically named Kinds. Groups often contain a domain name, such as redhat.com.

The entity **Version** defines the stability of the API and backward compatibility guarantees - such as v1beta1 or v1.

The entity **Kind** is the name of the API - such as Deployment or Service.

GroupVersionResource or GVR

Red Hat



The resource **Group** is similar to package in a language. It disambiguates different APIs that may happen to have identically named Kinds.

The resource **Version** defines the stability of the API and backward compatibility guarantees - such as v1beta1 or v1.

The **Resource** is the name of the path

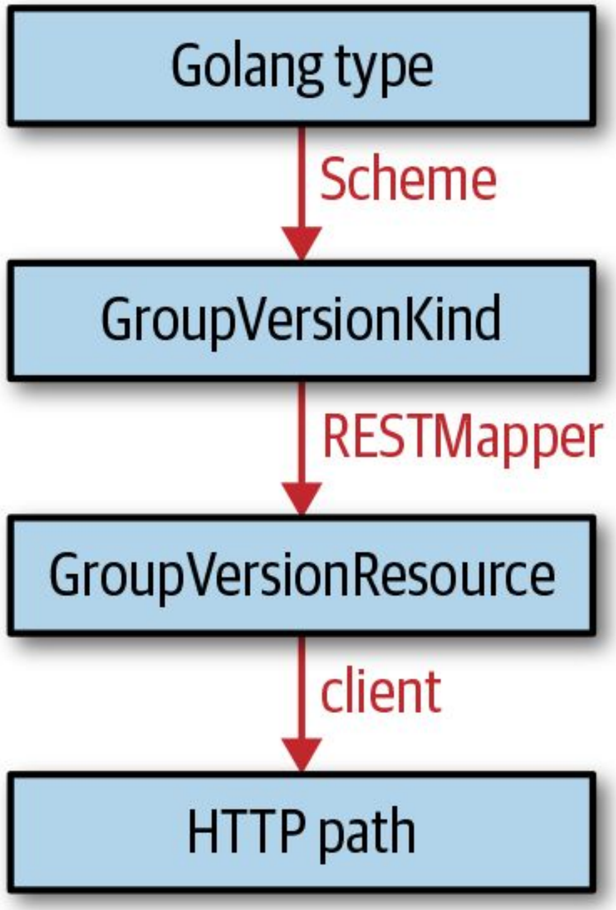
GVK (TypeMeta)

Metadata (ObjectMeta)

Spec

Status

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-first-replica-set
  namespace: myproject
spec:
  selector:
    matchLabels:
     app: nginx
  replicas: 5
  template:
    metadata:
      labels:
        app: nginx
      ...
status:
  availableReplicas: 1
  fullyLabeledReplicas: 1
  ...
```



Kubernetes API Actions and HTTP Methods

Verb (API)	HTTP Method
get	GET
list	GET
watch	GET
create	POST
update	PUT
patch	PATCH
delete	DELETE
deletecollection	DELETE

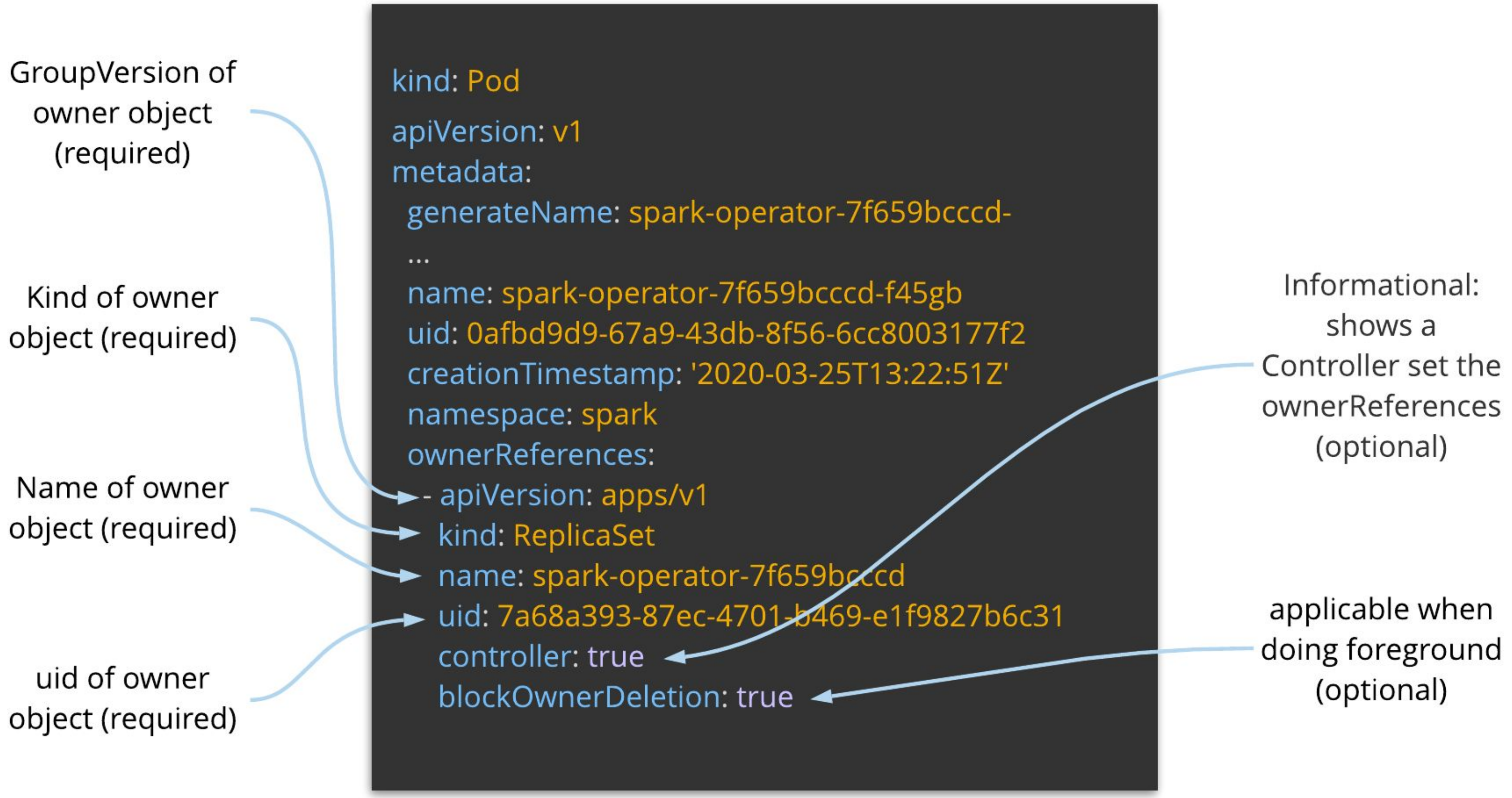
Kubernetes Subcommand and HTTP Methods

Subcommand	Object Does Not Exist	Object Exists
apply	POST	PATCH | DELETE
create	POST	error!
replace	error!	PUT
delete	error!	DELETE
patch	error!	PATCH

Garbage Collection assists in deleting objects that have an **owner** that no longer exists.

ownerReferences

Parent/child or owner/dependents Relationship



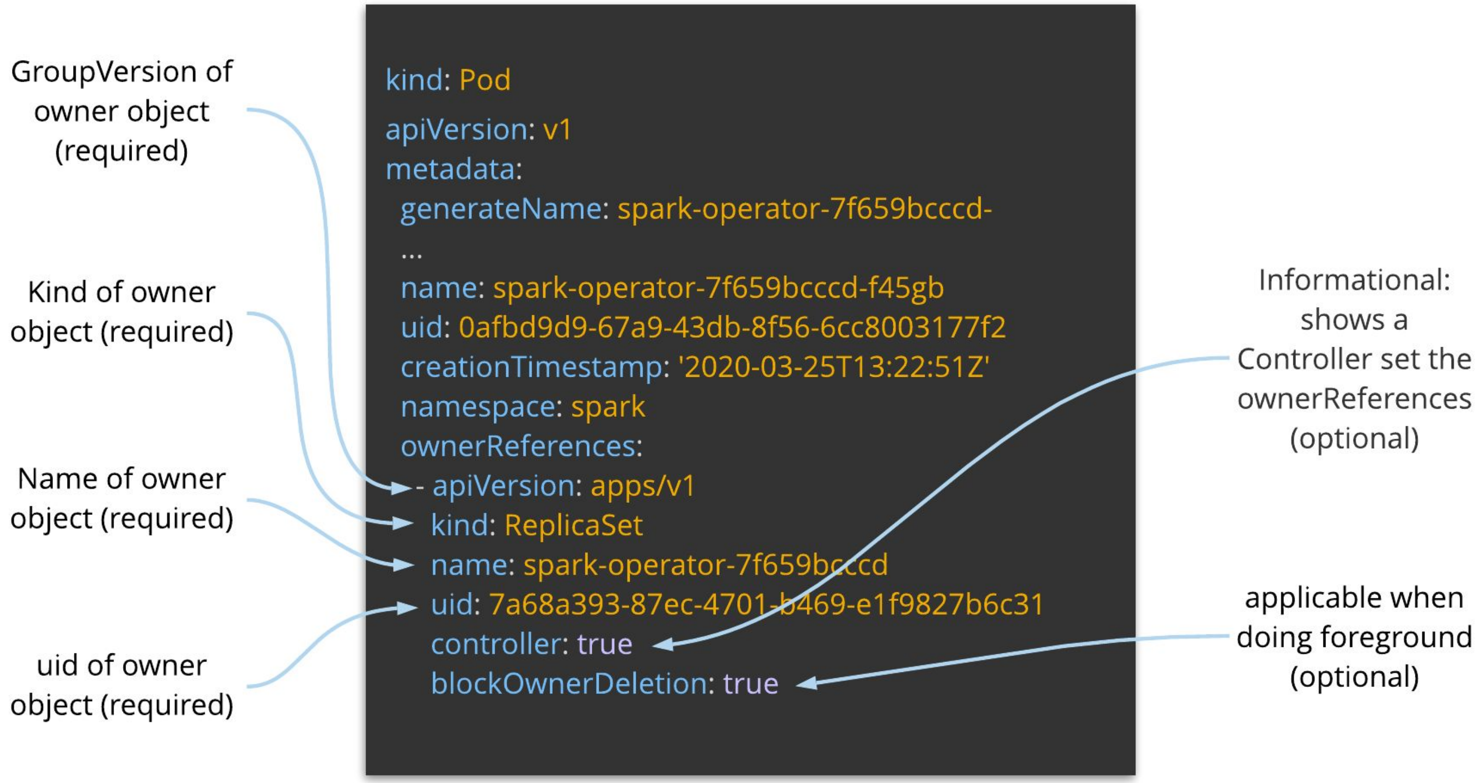
```
kind: Pod
apiVersion: v1
metadata:
 generateName: spark-operator-7f659bcccd-
 ...
 name: spark-operator-7f659bcccd-f45gb
 uid: 0afbd9d9-67a9-43db-8f56-6cc8003177f2
 creationTimestamp: '2020-03-25T13:22:51Z'
 namespace: spark
 ownerReferences:
  - apiVersion: apps/v1
    kind: ReplicaSet
    name: spark-operator-7f659bcccd
    uid: 7a68a393-87ec-4701-b469-e1f9827b6c31
    controller: true
    blockOwnerDeletion: true
```

GroupVersion of owner object (required)
Kind of owner object (required)
Name of owner object (required)
uid of owner object (required)
Informational: shows a Controller set the ownerReferences (optional)
applicable when doing foreground (optional)

Sources:
https://kubernetes.io/docs/concepts/workloads/controllers/garbage-collection/
https://kubernetes.io/docs/tasks/access-kubernetes-api/custom-resources/custom-resource-definitions/

ownerReferences

Parent/child or owner/dependents Relationship



```
kind: Pod
apiVersion: v1
metadata:
  generateName: spark-operator-7f659bcccd-
  ...
  name: spark-operator-7f659bcccd-f45gb
  uid: 0afbd9d9-67a9-43db-8f56-6cc8003177f2
  creationTimestamp: '2020-03-25T13:22:51Z'
  namespace: spark
  ownerReferences:
  - apiVersion: apps/v1
    kind: ReplicaSet
    name: spark-operator-7f659bcccd
    uid: 7a68a393-87ec-4701-b469-e1f9827b6c31
    controller: true
    blockOwnerDeletion: true
```

GroupVersion of owner object (required)
Kind of owner object (required)
Name of owner object (required)
uid of owner object (required)
Informational: shows a Controller set the ownerReferences (optional)
applicable when doing foreground (optional)

- one can decide the parent and child relationship for a pair or more number of objects
- multiple owners of a dependent is possible
- when a dependent has multiple owners **all** owners have to be “out-of-scope” for the dependent to be marked for garbage collection

Sources:
https://kubernetes.io/docs/concepts/workloads/controllers/garbage-collection/
https://kubernetes.io/docs/tasks/access-kubernetes-api/custom-resources/custom-resource-definitions/

Background, Foreground, and Finalizers

BACKGROUND

- Kubernetes deletes the **owner** object immediately and the garbage collector then deletes the dependents in the background

FOREGROUND

- object is still visible via the REST API
- the root object first enters a “deletion in progress” state
- once the garbage collector has deleted all “blocking” dependents it deletes the **owner** object

FINALIZERS

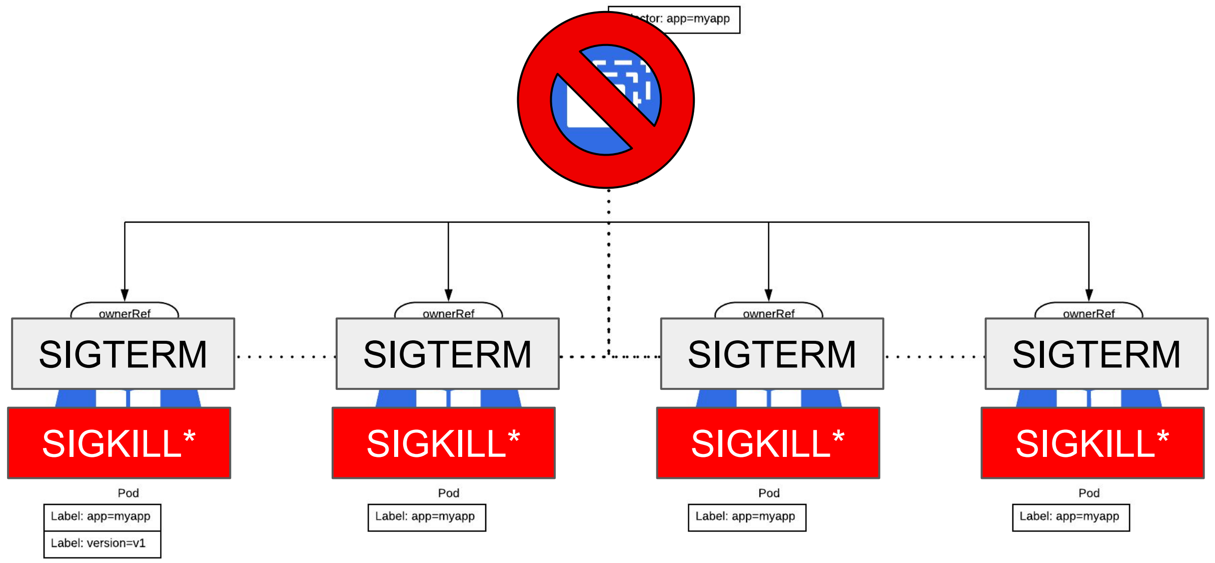
- allow controllers to implement asynchronous pre-delete hooks
- arbitrary string values, that when present ensure that a hard delete of a resource is not possible while they exist
- Kubernetes only finally deletes the object if the list of finalizers is empty, meaning all finalizers have been executed

Sources:
https://kubernetes.io/docs/concepts/workloads/controllers/garbage-collection/
https://kubernetes.io/docs/tasks/access-kubernetes-api/custom-resources/custom-resource-definitions/

Background Deletion

oc delete -f spark-rs.yaml

```
curl -x DELETE
http://localhost:8080/apis/apps/v1/namespaces/spark/replicasets/spark-operator-7f659bcccd \
-d '{"kind":"DeleteOptions","apiVersion":"v1","propagationPolicy":"Background"}' \
-H "Content-Type: application/json"
```



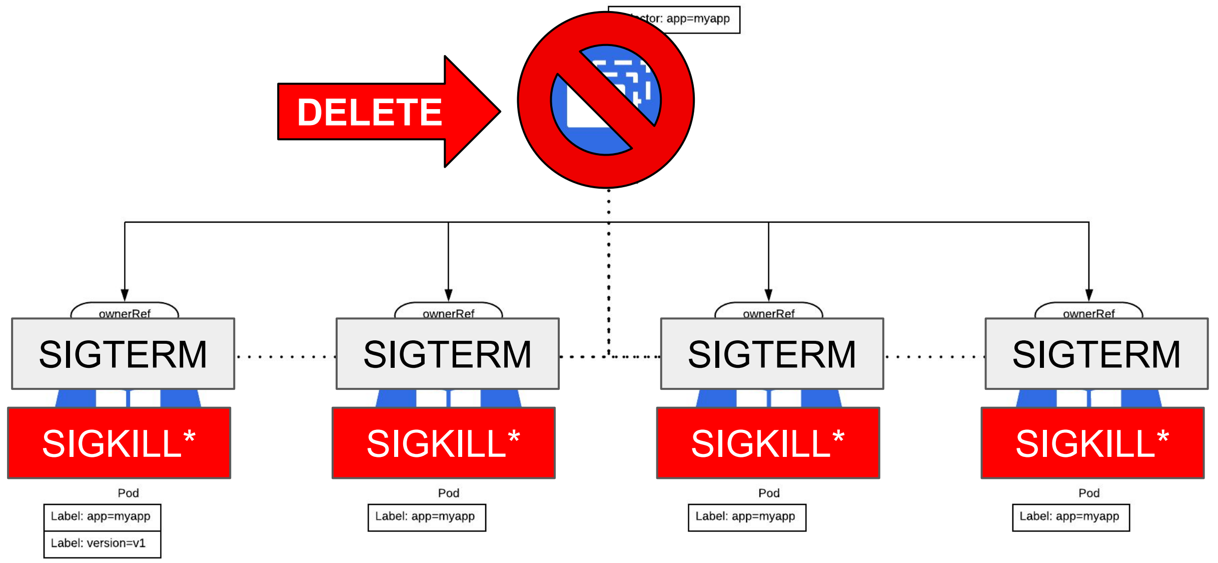
```
ownerReferences:
  - apiVersion: apps/v1
    kind: ReplicaSet
    name: spark-operator-7f659bcccd
    uid: 7a68a393-87ec-4701-b469-e1f9827b6c31
    controller: true
    blockOwnerDeletion: true
```

*wait 30s after SIGTERM then SIGKILL

Foreground Deletion

oc delete -f spark-rs.yaml

```
curl -x DELETE
http://localhost:8080/apis/apps/v1/namespaces/spark/replicasets/spark-operator-7f659bcccd \
-d ‘{“kind”:”DeleteOptions”,”apiVersion”:”v1”,”propagationPolicy”:”Foreground”}’ \
-H “Content-Type: application/json”
```



```
ownerReferences:
  - apiVersion: apps/v1
    kind: ReplicaSet
    name: spark-operator-7f659bcccd
    uid: 7a68a393-87ec-4701-b469-e1f9827b6c31
    controller: true
    blockOwnerDeletion: true
```

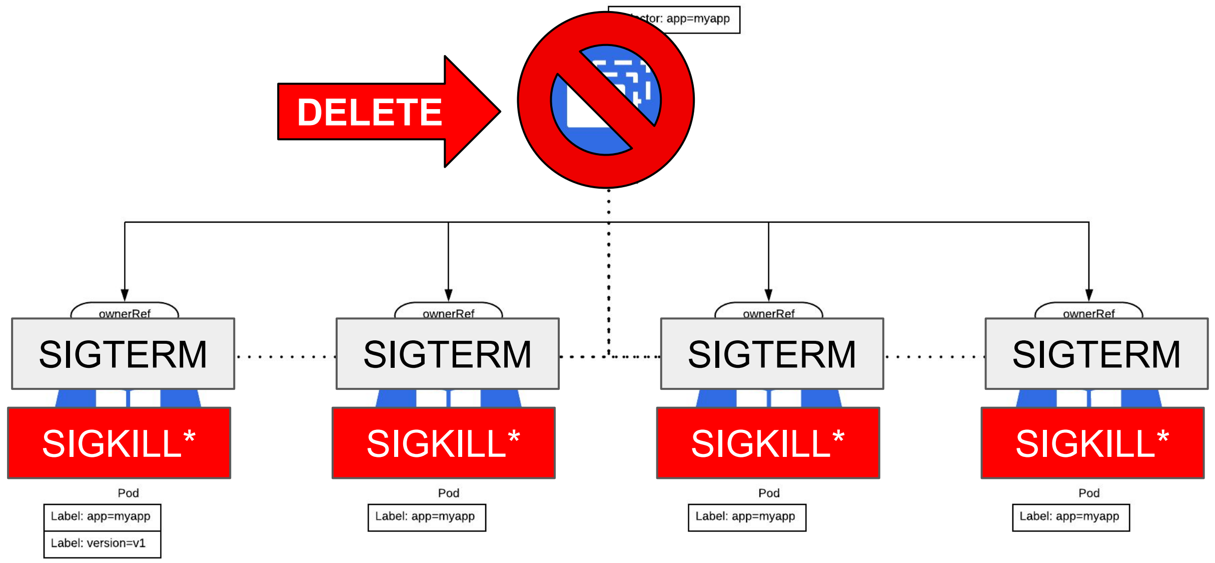
*wait 30s after SIGTERM then SIGKILL

Foreground Deletion

```
metadata:
  deletionTimestamp: 2019-10-20T01:16:04Z
  Finalizers: “foregroundDeletion”
```

oc delete -f spark-rs.yaml

```
curl -x DELETE
http://localhost:8080/apis/apps/v1/namespaces/spark/replicasets/spark-operator-7f659bcccd \
-d ‘{“kind”:”DeleteOptions”,”apiVersion”:”v1”,”propagationPolicy”:”Foreground”}’ \
-H “Content-Type: application/json”
```



```
ownerReferences:
  - apiVersion: apps/v1
    kind: ReplicaSet
    name: spark-operator-7f659bcccd
    uid: 7a68a393-87ec-4701-b469-e1f9827b6c31
    controller: true
    blockOwnerDeletion: true
```

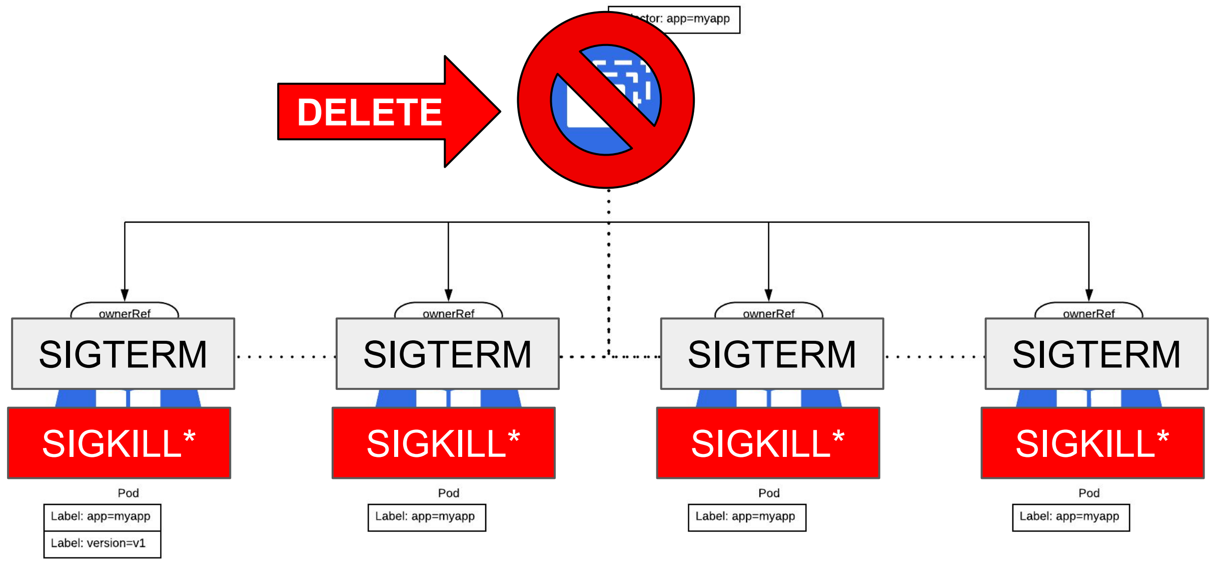
*wait 30s after SIGTERM then SIGKILL

Foreground Deletion

```
metadata:
  deletionTimestamp: 2019-10-20T01:16:04Z
  Finalizers: “foregroundDeletion”
```

oc delete -f spark-rs.yaml

```
curl -x DELETE
http://localhost:8080/apis/apps/v1/namespaces/spark/replicasets/spark-operator-7f659bcccd \
-d ‘{“kind”:”DeleteOptions”,”apiVersion”:”v1”,”propagationPolicy”:”Foreground”}’ \
-H “Content-Type: application/json”
```



```
ownerReferences:
  - apiVersion: apps/v1
    kind: ReplicaSet
    name: spark-operator-7f659bcccd
    uid: 7a68a393-87ec-4701-b469-e1f9827b6c31
    controller: true
    blockOwnerDeletion: true
```

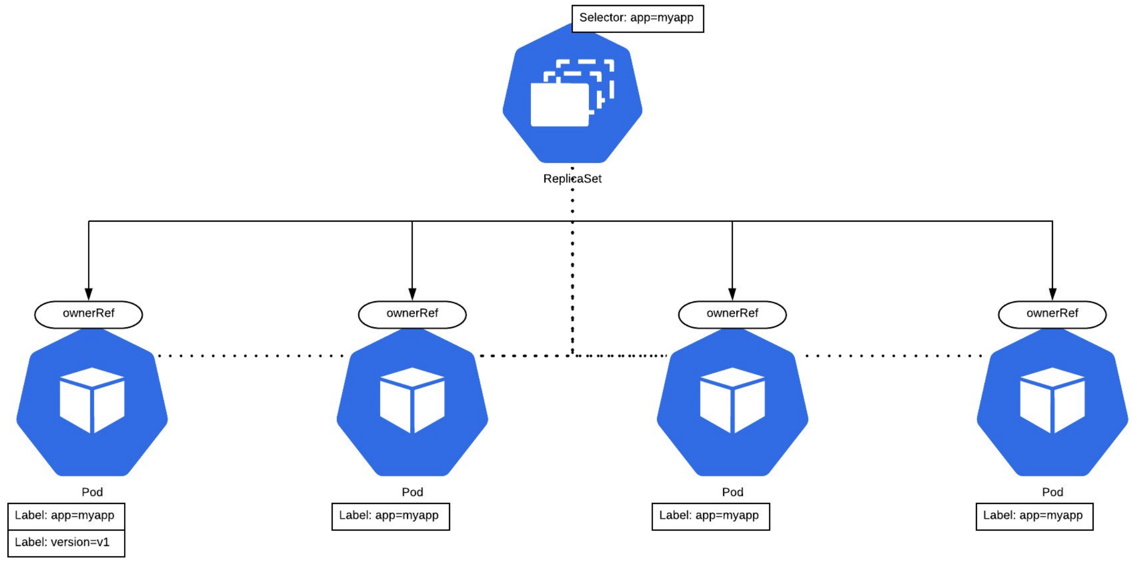
*wait 30s after SIGTERM then SIGKILL

Finalizers

```
metadata:
  deletionTimestamp: 2019-10-20T01:16:04Z
  Finalizers: “foregroundDeletion”
```

oc delete -f spark-rs.yaml

```
curl -x DELETE
http://127.0.0.1:8080/apis/apps/v1/namespaces/spark/replicasets/spark-operator-7f659bcccd \
-d ‘{“kind”:”DeleteOptions”,”apiVersion”:”v1”,”propagationPolicy”:”Background”}’ \
-H “Content-Type: application/json”
```



```
ownerReferences:
  - apiVersion: apps/v1
    kind: ReplicaSet
    name: spark-operator-7f659bcccd
    uid: 7a68a393-87ec-4701-b469-e1f9827b6c31
    controller: true
    blockOwnerDeletion: true
```

```
metadata:
  deletionTimestamp: 2019-10-20T01:16:04Z
  Finalizers: “foregroundDeletion”
```