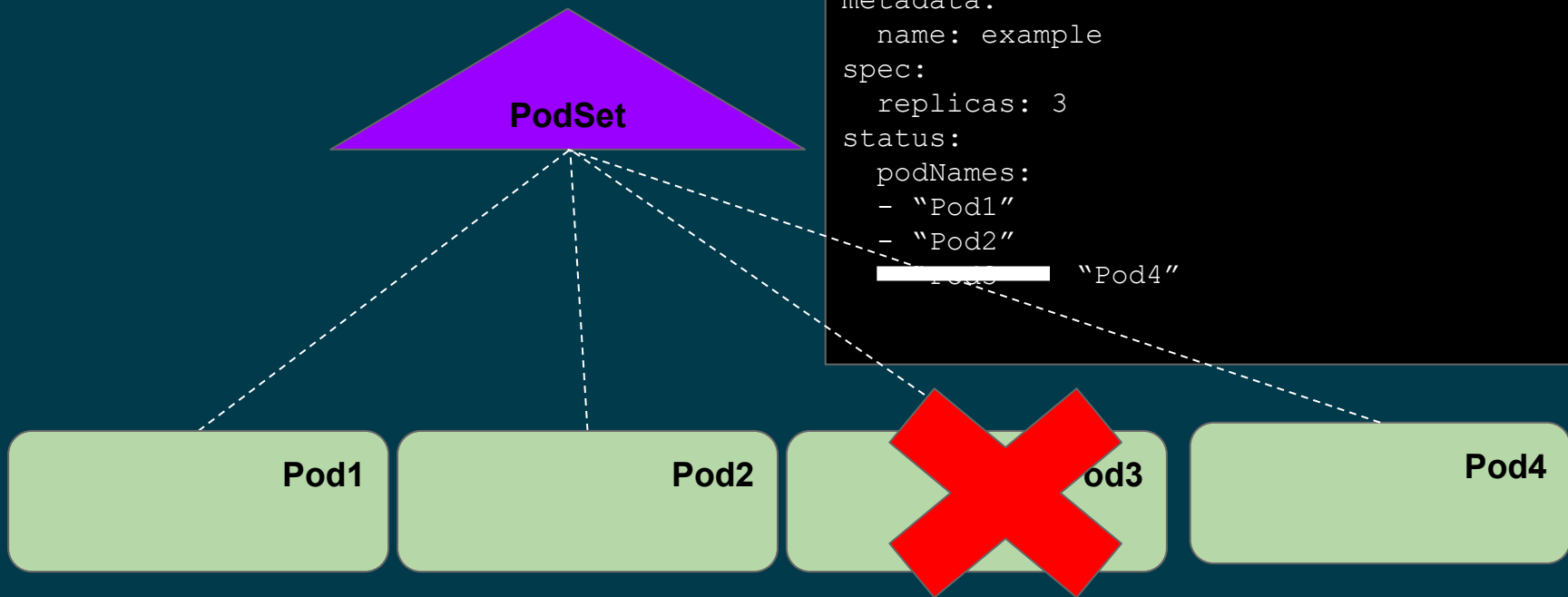


Pod Set Operator Challenge

Let's Create a Simple Operator.

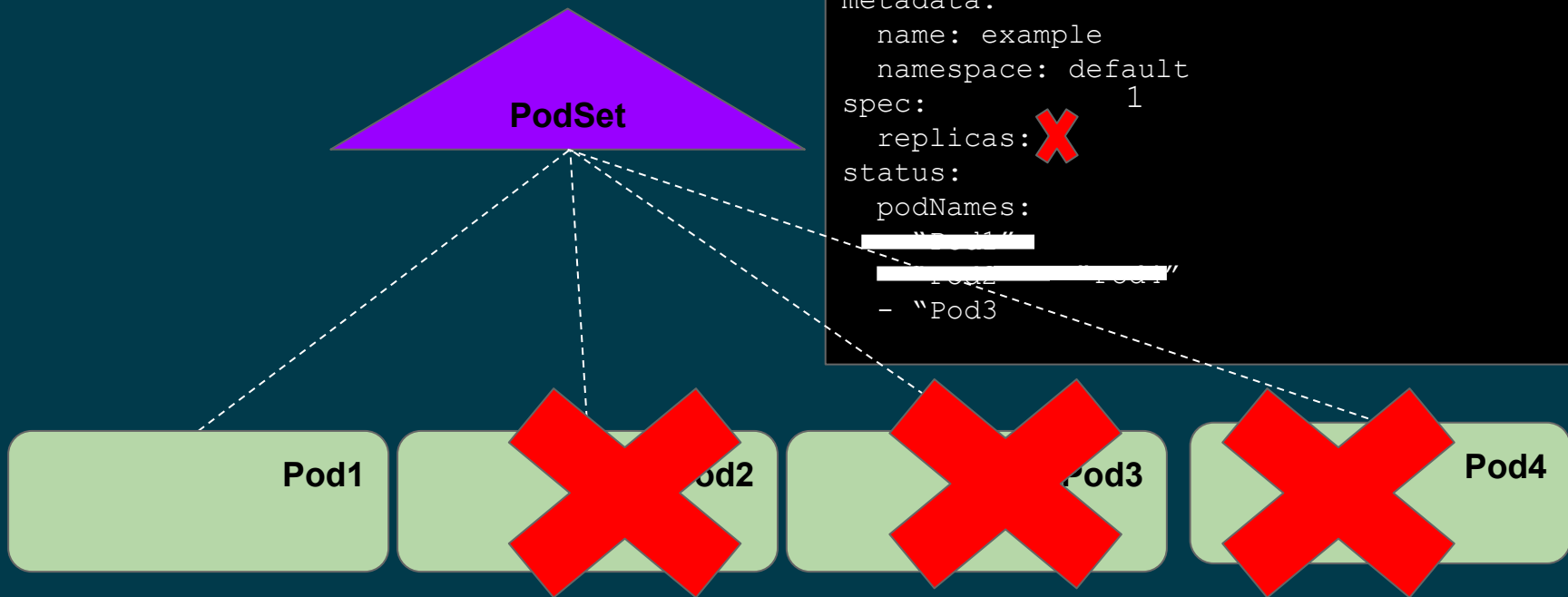
It's Called a "Pod Set".

A Simple Controller that Manages Pods.



```
apiVersion: podset.redhat.com/v1alpha1
kind: PodSet
metadata:
  name: example
spec:
  replicas: 3
status:
  podNames:
    - "Pod1"
    - "Pod2"
    - "Pod3"
    - "Pod4"
```

A Pod Set Allows You to Scale Up/Down.



```
apiVersion: podset.redhat.com/v1alpha1
kind: PodSet
metadata:
  name: example
  namespace: default
spec:
  replicas: 1
status:
  podNames:
    - "Pod1"
    - "Pod2"
    - "Pod3"
```

Pod Set Requirements

- Spec: replicas
- Status: podNames

Always Run

operator-sdk generate k8s
after modifying *_types.go.

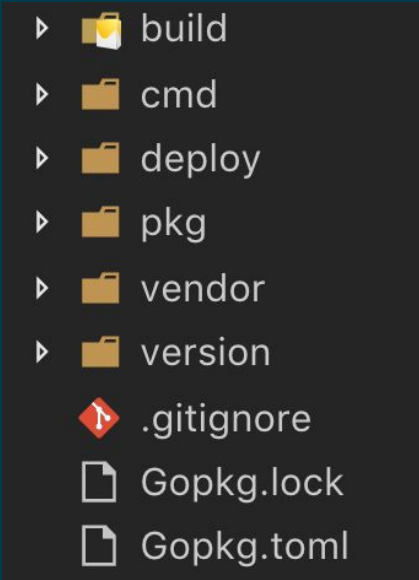
Hints Before You Begin










- **Identify Primary and Secondary Resources.**
 - These will determine what you Watch.
- **Use OwnerRefs.**
 - `controllerutil.SetControllerReference` to set OwnerRefs on all pods!
- **Use Labels** to manage the relationship between Primary and Secondary Resources.
 - Use `labels.SelectorFromSet` to assist in converting an existing map of labels to selectors.

Pod Set Solution

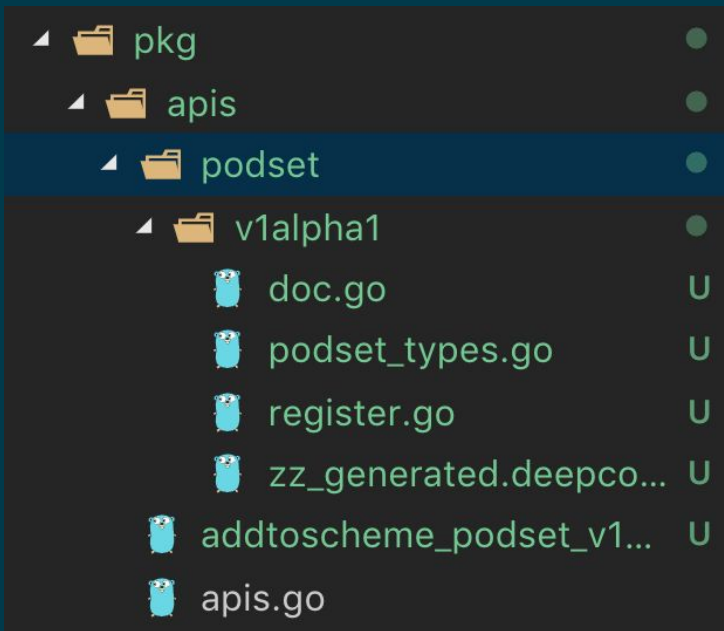
Easy Stuff First.

operator-sdk new podset-operator



- ▶  build
- ▶  cmd
- ▶  deploy
- ▶  pkg
- ▶  vendor
- ▶  version
-  .gitignore
-  Gopkg.lock
-  Gopkg.toml

```
operator-sdk add api \
--api-version=app.example.com/v1alpha1 \ --kind=PodSet
```




Modify Spec/Status in podset_types.go

```
10 // PodSetSpec defines the desired state of PodSet
11 type PodSetSpec struct {
12     // INSERT ADDITIONAL SPEC FIELDS - desired state of cluster
13     // Important: Run "operator-sdk generate k8s" to regenerate code after modifying this file
14     Replicas int32 `json:"replicas"`
15 }
16
17 // PodSetStatus defines the observed state of PodSet
18 type PodSetStatus struct {
19     // INSERT ADDITIONAL STATUS FIELD - define observed state of cluster
20     // Important: Run "operator-sdk generate k8s" to regenerate code after modifying this file
21     PodNames []string `json:"podNames"`
22 }
```

operator-sdk generate k8s

```
operator-sdk add controller \  
--api-version=app.example.com/v1alpha1 --kind=PodSet
```



A file explorer view showing the directory structure of a project. The root is 'pkg', which contains 'apis', 'controller', and 'podset' subdirectories. The 'podset' directory contains three Go files: 'podset_controller.go', 'add_podset.go', and 'controller.go'. Each file has a small blue robot icon next to it. To the right of each file name is a green circle, and to the right of each directory name is a green circle. The 'podset' directory is expanded, showing its contents.

- pkg
 - apis
 - controller
 - podset
 - podset_controller.go
 - add_podset.go
 - controller.go

Identify our Primary and Secondary
Resources to Watch.

The Core Component of the Controller is the Watch function.

1

```
Watch func(src source.Source, eventhandler  
handler.EventHandler, predicates ...predicate.Predicate)  
error
```

2

3

The Core Component of the Controller is the Watch function.

1

```
Watch func(src source.Source, eventhandler  
handler.EventHandler, predicates ...predicate.Predicate)  
error
```

2

3

1

Source of events ("sigs.k8s.io/controller-runtime/pkg/source")
One Kind: Your Custom GVK, Pods, Deployments, Routes, etc)
One Channel: Trigger reconcile based on something outside the cluster.

2

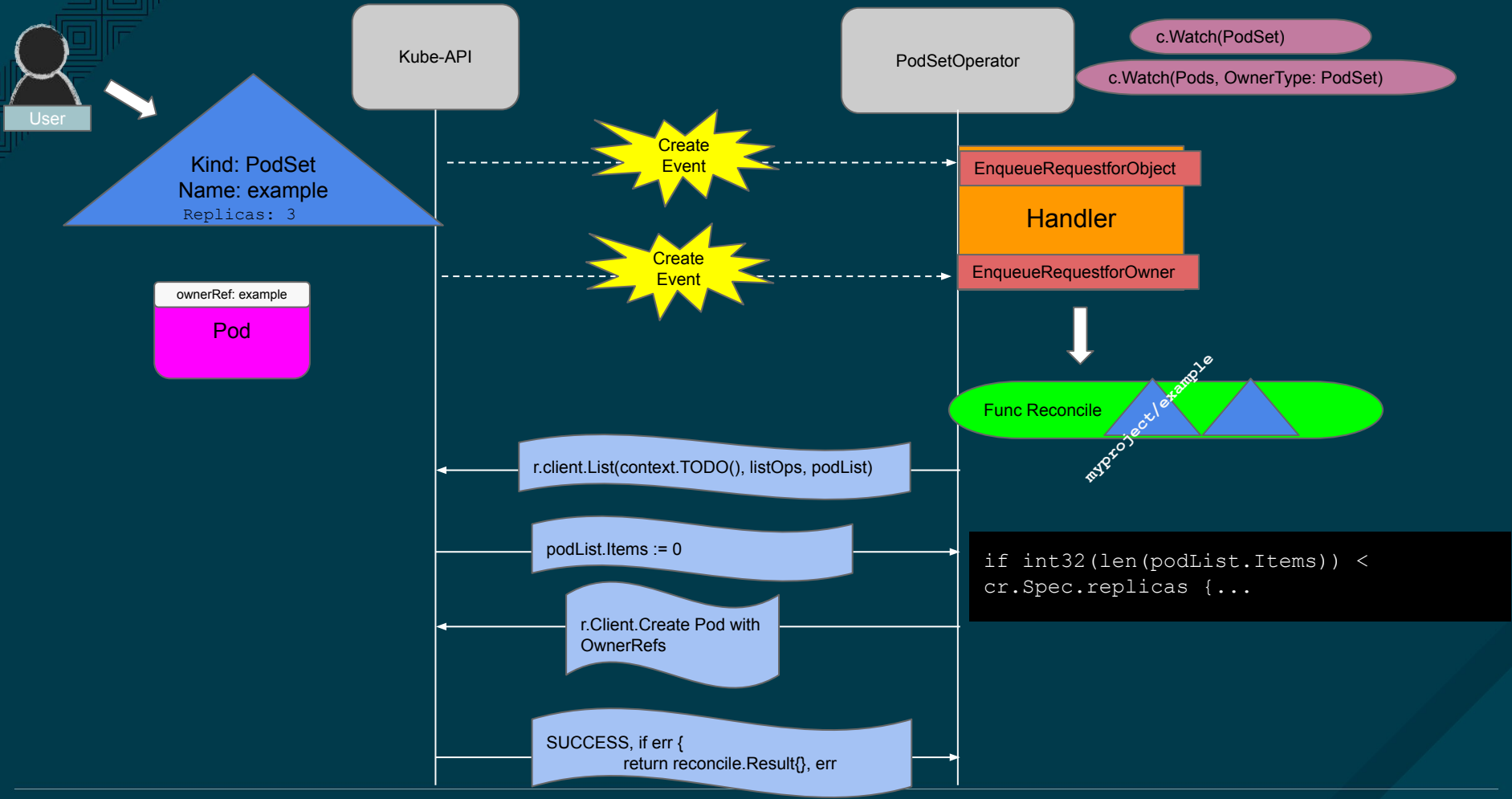
EventHandler ("sigs.k8s.io/controller-runtime/pkg/handler")
3 Types: EnqueueRequestForObject, EnqueueRequestForOwner, EnqueueRequestsFromMapFunc

3

Predicate: ("sigs.k8s.io/controller-runtime/pkg/predicate")
Filter events before enqueueing keys: i.e. Only enqueue Delete Events.


```
38 // add adds a new Controller to mgr with r as the reconcile.Reconciler
39 func add(mgr manager.Manager, r reconcile.Reconciler) error {
40     // Create a new controller
41     c, err := controller.New("podset-controller", mgr, controller.Options{Reconciler: r})
42     if err != nil {
43         return err
44     }
45
46     // Watch for changes to primary resource PodSet
47     err = c.Watch(&source.Kind{Type: &podsetv1alpha1.PodSet{}}, &handler.EnqueueRequestForObject{})
48     if err != nil {
49         return err
50     }
51
52     // TODO(user): Modify this to be the types you create that are owned by the primary resource
53     // Watch for changes to secondary resource Pods and requeue the owner PodSet
54     err = c.Watch(&source.Kind{Type: &corev1.Pod{}}, &handler.EnqueueRequestForOwner{
55         IsController: true,
56         OwnerType:     &podsetv1alpha1.PodSet{}},
57     })
58     if err != nil {
59         return err
60     }
61
62     return nil
63 }
```

Let's Visualize the Watch.



Create a Function for the
Creation of the Pod.

func newPodForCR

```
128 // newPodForCR returns a busybox pod with the same name/namespace as the cr
129 func newPodForCR(cr *podsetv1alpha1.PodSet) *corev1.Pod {
130     labels := map[string]string{
131         "app": cr.Name,
132     }
133     return &corev1.Pod{
134         // ObjectMeta
135         Name:      cr.Name + "-pod",
136         Namespace: cr.Namespace,
137         Labels:    labels,
138     },
139     Spec: corev1.PodSpec{
140         Containers: []corev1.Container{
141             {
142                 Name:      "busybox",
143                 Image:     "busybox",
144                 Command: []string{"sleep", "3600"},
145             },
146         },
147     },
148 }
149 }
```

GenerateName

```
reconciler.Errorf(err, "failed to delete pod %s", podName, podName);
return reconcile.Result{}, err
}
return reconcile.Result{
    type ObjectMeta struct {
        Name      string `json:"name,omitEmpty" protobuf:"bytes,1,opt,name=name"`
        GenerateName string `json:"generateName,omitEmpty" protobuf:"bytes,2,opt,name=generateName"`
        Namespace string `json:"namespace,omitEmpty" protobuf:"bytes,3,opt,name=namespace"`
        SelfLink   string `json:"selfLink,omitEmpty" protobuf:"bytes,4,opt,name=selfLink"`
        UID        types.UID `json:"uid,omitEmpty" protobuf:"bytes,5,opt,name=uid,casttype=k8s.io/kubernetes/pkg/types.UID"`
        ResourceVersion string `json:"resourceVersion,omitEmpty" protobuf:"bytes,6,opt,name=resourceVersion"`
        Generation int64 `json:"generation,omitEmpty" protobuf:"varint,7,opt,name=generation"`
    }
}
PodForCR returns a builder for creating a new PodForCR(cr *appsv1.Pod, labels := map[string]string{"app": cr.Name, "version": "v0.1"}, return &corev1.Pod{
    ObjectMeta: metav1.ObjectMeta{
        Name: cr.Name + "-pod",
        Namespace: cr.Namespace,
        Labels: labels,
    },
},
```

example-podset-podc6xlj

[illegible][illegible]

NAME	READY	STATUS	RESTARTS	AGE
example-appservice-podxvnk9	0/1	Pending	0	0s
example-appservice-podxvnk9	0/1	Pending	0	0s
example-appservice-podxvnk9	0/1	ContainerCreating	0	0s
example-appservice-podtznfz	0/1	Pending	0	0s
example-appservice-podtznfz	0/1	Pending	0	0s
example-appservice-podtznfz	0/1	ContainerCreating	0	0s
example-appservice-podrwlnw	0/1	Pending	0	0s
example-appservice-podrwlnw	0/1	Pending	0	0s
example-appservice-podrwlnw	0/1	ContainerCreating	0	0s
example-appservice-podwl2z6	0/1	Pending	0	0s
example-appservice-podwl2z6	0/1	Pending	0	0s
example-appservice-podwl2z6	0/1	ContainerCreating	0	0s
example-appservice-podhr87v	0/1	Pending	0	0s
example-appservice-podhr87v	0/1	Pending	0	0s
example-appservice-podhhbd2	0/1	Pending	0	0s
example-appservice-podhhbd2	0/1	Pending	0	0s
example-appservice-podffjlq	0/1	Pending	0	0s
example-appservice-podffjlq	0/1	Pending	0	0s
example-appservice-pod647j7	0/1	Pending	0	0s
example-appservice-pod647j7	0/1	Pending	0	0s
example-appservice-podhr87v	0/1	ContainerCreating	0	0s
example-appservice-pod6dv9d	0/1	Pending	0	0s
example-appservice-pod6dv9d	0/1	Pending	0	0s
example-appservice-podxkz5q	0/1	Pending	0	0s
example-appservice-podxkz5q	0/1	Pending	0	0s
example-appservice-podx8zt5	0/1	Pending	0	0s
example-appservice-podx8zt5	0/1	Pending	0	0s
example-appservice-podzhlj4	0/1	Pending	0	0s
example-appservice-podzhlj4	0/1	Pending	0	0s
example-appservice-pod8axbw	0/1	Pending	0	0s

Listing Pods with List Options

```
103 // List all pods owned by this PodSet instance
104 podList := &corev1.PodList{}
105 lbs := map[string]string{
106     "app":    podSet.Name,
107     "version": "v0.1",
108 }
109 labelSelector := labels.SelectorFromSet(lbs)
110 listOps := &client.ListOptions{Namespace: podSet.Namespace, LabelSelector: labelSelector}
111 if err = r.client.List(context.TODO(), listOps, podList); err != nil {
112     return reconcile.Result{}, err
113 }
```


Be Careful When Listing Pods...

Some May Be in Terminating Status..

```
example-appservice-pod 0/1 Terminating 0 1m
```

Only Count Pods that are “Available”

```
115 // Count the pods that are pending or running as available
116 var available []corev1.Pod
117 for _, pod := range podList.Items {
118     if pod.ObjectMeta.DeletionTimestamp != nil {
119         continue
120     }
121     if pod.Status.Phase == corev1.PodRunning || pod.Status.Phase == corev1.PodPending {
122         available = append(available, pod)
123     }
124 }
125 numAvailable := int32(len(available))
126 availableNames := []string{}
127 for _, pod := range available {
128     availableNames = append(availableNames, pod.ObjectMeta.Name)
129 }
```