Kubernetes and Golang
CloudNative Week

Wojciech Barczynski
(wbarczynski.pro@gmail.com)

Contents

1 Prerequiments

1.1 Audience

We design the workshop with the following assumptions about the audience:

- Can read and write Golang code, and understand its basics concepts.
- Have been working at lease 1 month with Kubernetes
- Feel good with Command Line Interface.

1.2 Your workstation

- Linux or OSX recommended.
- Basic:
  - Golang,
  - a configured IDE or editor.
  - Git.
- Kubernetes.
  - minikube,
  - kubectl.
- Docker:
  - Installed.
  - Account on hub.docker.com or quay.io.

2 Check our dev environment

Let's start a connection:

```
$ kubectl config use-context minikube
$ kubectl get po --all-namespaces
```

You should see:

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS AGE
kube-system   coredns-5c98db65d4-kmz2d 1/1     Running   2        3d20h
kube-system   coredns-5c98db65d4-kwq6g 1/1     Running   2        3d20h
```

3 client-go for kubernetes

1. Create your project

```
$ mkdir my-kube
$ cd my-kube
$ go mod init my-kube
```

2. Le's get the library (https://github.com/kubernetes/client-go):

```
$ go get k8s.io/client-go@kubernetes-1.15.3
```

3. Write a program (based on example.go):

```
package main

import (
        "flag"
        "fmt"
        "os"
        "path/filepath"
        "time"

        "k8s.io/apimachinery/pkg/api/errors"
        metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
        "k8s.io/client-go/kubernetes"
        "k8s.io/client-go/tools/clientcmd"
)

func main() {
        var kubeconfig *string
        home := os.Getenv("HOME")
        if home == "" {
                panic("HOME not found")
        }

        kubeconfig = flag.String("kubeconfig",
                filepath.Join(home, ".kube", "config"),
                "(optional) absolute path to the kubeconfig file")

        flag.Parse()
```

```
        // use the current context in kubeconfig
        config, err := clientcmd.BuildConfigFromFlags("", *kubeconfig)
        if err != nil {
                panic(err.Error())
        }

        // create the clientset
        clientset, err := kubernetes.NewForConfig(config)
        if err != nil {
                panic(err.Error())
        }
        for {
                check(clientset, "kube-system", "etcd-minikube")
        }
}

func check(clientset *kubernetes.Clientset, namespace string, pod string) {
        pods, err := clientset.CoreV1().Pods("").List(metav1.ListOptions{})
        if err != nil {
                panic(err.Error())
        }
        fmt.Printf("There are %d pods in the cluster\n", len(pods.Items))

        _, err = clientset.CoreV1().Pods(namespace).Get(pod, metav1.GetOptions{})
        if errors.IsNotFound(err) {
                fmt.Printf("Pod %s in namespace %s not found\n", pod, namespace)
        } else if statusError, isStatus := err.(*errors.StatusError); isStatus {
                fmt.Printf("Error getting pod %s in namespace %s: %v\n",
                        pod, namespace, statusError.ErrStatus.Message)
        } else if err != nil {
                panic(err.Error())
        } else {
                fmt.Printf("Found pod %s in namespace %s\n", pod, namespace)
        }

        time.Sleep(10 * time.Second)
}
```

Please change the program to display `Endpoints` and, later, to get all

`Deployments`.

4. Please create in a golang program a service and ingress based on this example.

5. Let's see how to use fake-client for testing your program. Follow the instructor.

4 operator-SDK

1. Install operator SDK following instructions from https://github.com/operator-framework/operator-sdk. You can also download the binary from https://github.com/operator-framework/operator-sdk/releases.

2. Add the operator-sdk to your PATH.

3. Create a sample program, following quick start: https://github.com/operator-framework/operator-sdk. You might need to create an account on hub.docker.com or quay.io.
Notice: you might need to make your docker image public.

4. Let's extend change the command to:

```
Spec: corev1.PodSpec{
    Containers: []corev1.Container{
        {
            Name:    "busybox2",
            Image:   "busybox",
            Command: []string{"/bin/sh","-c", "echo 'hello world' && sleep 3600"},
        },
    },
}
```

5. We would like now our operator to ensure the command is as specified.
Otherwise it should recreate the BusyBox. Let's add `appservice_types.go`:

```
type AppServiceSpec struct {
        BoxCommand []string `json:"command,omitempty"`
}
```

and regenerate:

```
$ operator-sdk generate k8
$ operator-sdk generate openapi
```

Change CRD:

```
$ cat deploy/crds/app_v1alpha1_appservice_cr.yaml
```

Let's spcify it:

```
spec:
  command: [ "/bin/sh","-c", "echo 'hello world x2' && sleep 1000"]
  # Add fields here
  size: 3
```

6. Check the CR:

```
$ kubectl get appservice
```

5 kubebuilder

Kubebuilder (https://github.com/kubernetes-sigs/kubebuilder) is a framework for building Kubernetes APIs.

6 Observability

Follow the instructor.

- Monitoring - prometheus
- Logging - framework overview

See: rest service instruction, github.com/wojciech12/talk_observability_logging, and github.com/wojciech12/talk_monitoring_with_prometheus.

7 Golang app on kubernetes

Which signals to handle, how to communicate with kubernetes?
See github.com/wojciech12/talk_zero_downtime_deployment_with_kubernetes

8 References

- https://github.com/golang/go/wiki/CodeReviewComments
- https://golang.com/doc/effective_go.html
- http://devs.cloudimmunity.com/gotchas-and-common-mistakes-in-go-golang