Exercise 2. Kubernetes client-go

Create API extensions client

```
import "k8s.io/client-go/rest
import "k8s.io/apiextensions-apiserver/pkg/client/clientset/clientset"

restConfig = &rest.Config{
        Host: "https://$(MINIKUBE_IP):8443", // $(minikube ip)
        TLSClientConfig: rest.TLSClientConfig{
                CertFile: "/home/user/.minikube/apiserver.crt",
                KeyFile:  "/home/user/.minikube/apiserver.key",
                CAFile:   "/home/user/.minikube/ca.crt",
        },
}

k8sExtClient := clientset.NewForConfig(restConfig)
```

Create CRD object

```
import "k8s.io/apiextensions-apiserver/pkg/apis/apiextensions/v1beta1"

crd := &v1beta1.CustomResourceDefinition{
        TypeMeta: ...,
        ObjectMeta: ...,
        Spec: ...,
}

k8sExtClient.
        ApiextensionsV1beta1().
        CustomResourceDefinitions().
        Create(crd)
```

Wait for custom resource

```
endpoint := "/apis/GROUP/VERSION/PLURAL"
err := k8sExtClient.
        Apiextensions().
        RESTClient().
        Get().
        AbsPath(endpoint).
        Do().
        Error()

if err == nil {
        // retry...
}
```

Create custom runtime objects

```
import "github.com/giantswarm/operator-workshop/customobject"
import "k8s.io/apimachinery/pkg/apis/meta/v1"

type PostgreSQLConfig struct {
    v1.TypeMeta   `json:",inline"`
    v1.ObjectMeta `json:"metadata,omitempty"`

    customobject.PostgreSQLConfig `json:",inline"`
}

type PostgreSQLConfigList struct {
    v1.TypeMeta `json:",inline"`
    v1.ListMeta `json:"metadata,omitempty"`

    Items []*PostgreSQLConfig `json:"items"`
}
```

Create scheme

```
import "k8s.io/apimachinery/pkg/apis/meta/v1"
import "k8s.io/apimachinery/pkg/runtime/schema"
import "k8s.io/apimachinery/pkg/runtime"

groupVersion := schema.GroupVersion{Group: group, Version: version}

scheme := runtime.NewScheme()
scheme.AddKnownTypes(
        groupVersion,
        &PostgreSQLConfig{},
        &PostgreSQLConfigList{},
)

v1.AddToGroupVersion(scheme, groupVersion)
```

Creating custom REST client

```
import "k8s.io/client-go/rest
import "k8s.io/apimachinery/pkg/runtime"
import "k8s.io/apimachinery/pkg/runtime/serializer"

restConfig = &rest.Config{
        ...,
        APIPath:     "/apis",
        ContentConfig: rest.ContentConfig{
            GroupVersion: &groupVersion,
            ContentType:  runtime.ContentTypeJSON,
            NegotiatedSerializer: serializer.DirectCodecFactory{
                    CodecFactory: serializer.NewCodecFactory(scheme),
            },
        },
}

k8sCustRestClient := rest.RESTClientFor(restConfig)
```

Create event handlers

```
import "k8s.io/client-go/tools/cache"

handler := cache.ResourceEventHandlerFuncs{
        AddFunc: ...,
        UpdateFunc: ...,
        DeleteFunc: ...,
}
```

Create informer

```
import "k8s.io/client-go/tools/cache"

handler := cache.ResourceEventHandlerFuncs{AddFunc: ..., UpdateFunc: ..
listWatch := cache.NewListWatchFromClient(
        k8sCustomRestClient, // custom Rest client
        "PLURAL",
        "",                  // namespace
        fields.Everything(), // get all fields of the custom objects
)
_, informer := cache.NewInformer(
        listWatch,
        &PostgreSQLConfig{}, // custom object instance for decoding
        time.Minute,         // resync period
        handler,
)
informer.Run(make(chan struct{}))
```

In-cluster mode

```
restConfig, err = rest.InClusterConfig()

if err != nil { ... }
```

- Run deployment

```
$ kubectl apply -f manifest/operator.yaml
```