Operator Task

PostgreSQL Operator

Functionality

- Create PostgreSQLConfig CRD
- Create database
- Update database owner
- Delete database

PostgreSQL Server

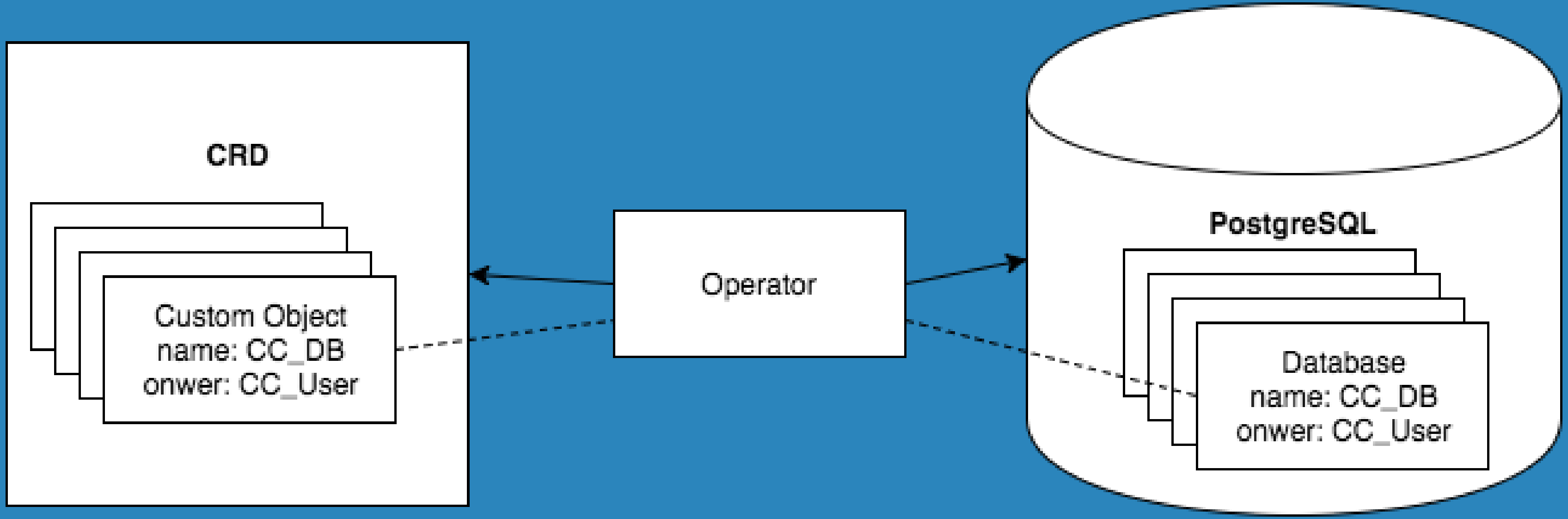
- Kubernetes resources
  - Secret with admin password
  - Deployment with single PostgreSQL pod
  - Service with a node port
- See giantswarm/operator-workshop

postgresqlops

- github.com/giantswarm/operator-workshop/postgresqlops
- Go package provides database access methods

customobject.Resource

- customobject Go package applies database changes
- Resource.EnsureCreated
- Resource.EnsureDeleted


CRD
Custom Object
name: CC_DB
onwer: CC_User
Operator
PostgreSQL
Database
name: CC_DB
onwer: CC_User

Exercise 1. Kubernetes REST API

Create minikube VM

```
$ minikube start --kubernetes-version 'v1.8.0'
```

Create PostgreSQL database

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

Connecting to REST API (Minikube)

- `minikube ip`
- APIServer is on port 8443
- TLS certificates

```
~/.minikube/apiserver.crt
~/.minikube/apiserver.key
~/.minikube/ca.crt
```

Load certificates

```
import "crypto/tls"
import "crypto/x509"

crt, err := tls.LoadX509KeyPair(
        "/home/user/.minikube/apiserver.crt",
        "/home/user/.minikube/apiserver.key",
)
if err != nil { ... }

caCert, err := ioutil.ReadFile(".minikube/ca.crt")
if err != nil { ... }

certPool := x509.NewCertPool()
certPool.AppendCertsFromPEM(caCert)
```

Create HTTP client

```go
import "crypto/tls"
import "net/http"

tlsConfig := &tls.Config{
        Certificates: []tls.Certificate{crt},
        RootCAs:      certPool,
}
tlsConfig.BuildNameToCertificate()

k8sClient := &http.Client{
        Transport: &http.Transport{TLSClientConfig: tlsConfig},
}
```

Create CRD

```
import "strings"

crdJson := `{
        "apiVersion": ...,
        "kind": ...,
        "metadata": ...,
        "spec": ...
}`

url := "https://$(MINIKUBE_IP):8443" + // $(minikube ip)
        "/apis/apiextensions.k8s.io/v1beta1/customresourcedefinitions"
res, err := k8sClient.Post(
        url,
        "application/json",
        strings.NewReader(crdJson),
)
```

Wait for custom resource

```
url := "https://$(MINIKUBE_IP):8443" + // $(minikube ip)
        "/apis/GROUP/VERSION/PLURAL"

res, err := k8sClient.Get(url)
if err != nil {
        // repeat
}
if res.StatusCode != http.StatusOK {
        // repeat
}
```

List object

```
import "github.com/giantswarm/operator-workshop/customobject"

type PostgreSQLConfigList struct {
	Items []*customobject.PostgreSQLConfig `json:"items"`
}
```

Reconciliation loop

```
for {
        url := "https://$(MINIKUBE_IP):8443" + // $(minikube ip)
                "/apis/GROUP/VERSION/PLURAL"

        res, err := k8sClient.Get(url)
        if err != nil { ... }
        defer res.Body.Close()

        var objs customobject.PostgreSQLConfigList
        err = json.NewDecoder(res.Body).Decode(&objs)
        if err != nil { ... }

        // reconcile objs ...

        time.Sleep(interval)
}
```

Reconciliation util

```
import "github.com/giantswarm/operator-workshop/customobject"
import "github.com/giantswarm/operator-workshop/postgresqlops"

ops, err := postgresqlops.New(config)
if err != nil { ... }

resource := customobject.NewResource(ops)

// resource.EnsureCreated(obj)
// resource.EnsureDeleted(obj)
```