# Splitting the Pod and Establishing Communication through Services

In this lesson, we will split up the Pods, create a separate DB pod and a Service to communicate with it.

#### WE'LL COVER THE FOLLOWING ^

- Looking into the Definition
- Creating the ReplicaSet
- Creating the Service

### Looking into the Definition #

Let's take a look at a ReplicaSet definition for a Pod with only the database.

```
cat svc/go-demo-2-db-rs.yml
```

The **output** is as follows.

```
apiVersion: apps/v1
                                                                                          kind: ReplicaSet
metadata:
 name: go-demo-2-db
spec:
  selector:
    matchLabels:
      type: db
     service: go-demo-2
  template:
    metadata:
      labels:
        type: db
        service: go-demo-2
        vendor: MongoLabs
    spec:
      containers:
      - name: db
        image: mongo:3.3
        ports:
        - containerPort: 28017
```

We'll comment only on the things that changed.

Since this ReplicaSet defines only the database, we reduced the number of replicas to 1. Truth be told, MongoDB should be scaled as well, but that's out of the scope of this chapter. For now, we'll pretend that one replica of a database is enough.

Since selector labels need to be unique, we changed them slightly. The service is still go-demo-2, but the type was changed to db.

The rest of the definition is the same except that the **containers** now contain only **mongo**. We'll define the API in a separate ReplicaSet.

#### Creating the ReplicaSet #

Let's create the ReplicaSet before we move to the Service that will reference its Pod.

```
kubectl create \
-f svc/go-demo-2-db-rs.yml
```

One object was created, three are left to go.

## Creating the Service #

The next one is the Service for the Pod we just created through the ReplicaSet.

```
cat svc/go-demo-2-db-svc.yml
```

The **output** is as follows.

```
apiVersion: v1
kind: Service
metadata:
   name: go-demo-2-db
spec:
   ports:
   - port: 27017
   selector:
     type: db
     service: go-demo-2
```

This Service definition does not contain anything new.

- There is no type, so it'll default to ClusterIP.
- Since there is no reason for anyone outside the cluster to communicate with the database, there's no need to expose it using the NodePort type.
- We also skipped specifying the **NodePort**, since only internal communication within the cluster is allowed.
- The same is true for the <a href="protocol">protocol</a>. TCP is all we need, and it happens to be the default one.
- Finally, the selector labels are the same as the labels that define the Pod.

Let's create the Service.

```
kubectl create \
   -f svc/go-demo-2-db-svc.yml
```

We are finished with the database. The ReplicaSet will make sure that the Pod is (almost) always up-and-running and the Service will allow other Pods to communicate with it through a fixed DNS.