Elasticity and Resilience

Table of Contents

Your Microservices should be **highly available** and resilient to failure. Ideally each Microservice should also be *elastic* so that you can easily scale up or down the number of containers used for each Microservice. Some Microservices may only require one container; others may require many.

Many times when considering elasticity, you have to consider whether your app is stateless or stateful. Stateless apps should be trivial to scale up and down, however stateful apps require more care. For example, a stateful data store would need to shard and replicate its state across the members in the cluster and know how to rebalance itself during scaling events.

Fabric8 solves these elasticity and resilience problems by using Kubernetes Replica Sets (which used to be called Replication Controllers). Just like most configurations for Kubernetes, a Replica Set is a way to reconcile a desired state: you tell Kubernetes what state the system should be and Kubernetes figures out how to make it so. A Replica Set controls the number of `replicas` or exact copies of the app that should be running at any time.

A *Replia Set* defines a template for running on or more pods which then can be scaled either by an operator or automatically by Kubernetes based on some system high watermarks.

The Replica Set uses a *selector* to keep watching the available pods matching the selectors labels. If there are not enough pods running it will spin up more; or if there are too many pods running it will terminate the extra pods.

Operator-based scaling

To scale your Replica Set you just need to specify how many `replicas` you wish by default in your Replica Set YAML file. The default value of 1 should ensure that there is always a pod running. If a pod terminates (or the host running the pod terminates) then Kubernetes will automatically spin up another pod for you.

Autoscaling

To autoscale you need to annotate your Replica Set with the metadata required, such as CPU limits or custom metrics so that Kubernetes knows when to scale up or down the number of pods. Then you can create a HorizontalPodAutoscaler to let Kubernetes know certain pods/ReplicaSets should participate in autoscaling.