CHAPTER 10

Singleton Service

The *Singleton Service* pattern ensures only one instance of an application is active at a time and yet is highly available. This pattern can be implemented from within the application, or delegated fully to Kubernetes.

Problem

One of the main capabilities provided by Kubernetes is the ability to easily and transparently scale applications. Pods can scale imperatively with a single command such as `kubectl scale`, or declaratively through a controller definition such as ReplicaSet, or even dynamically based on the application load as we describe in Chapter 24, *Elastic Scale*. By running multiple instances of the same service (not a Kubernetes Service, but a component of a distributed application represented by a Pod), the system usually increases throughput and availability. The availability increases because if one instance of a service becomes unhealthy, the request dispatcher forwards future requests to other healthy instances. In Kubernetes, multiple instances are the replicas of a Pod, and the Service resource is responsible for the request dispatching.

However, in some cases only one instance of a service is allowed to run at a time. For example, if there is a periodically executed task in a service and multiple instances of the same service, every instance will trigger the task at the scheduled intervals, leading to duplicates rather than having only one task fired as expected. Another example is a service that performs polling on specific resources (a filesystem or database) and we want to ensure only a single instance and maybe even a single thread performs the polling and processing. A third case occurs when we have to consume messages from a messages broker in order with a single-threaded consumer that is also a singleton service.