December 2021

METHOD TO PROVIDE ACTIVE/STANDBY LOGIC FOR KUBERNETES SINGLE-INSTANCE SERVICES

Tim Kuik

Matthew Chou

METHOD TO PROVIDE ACTIVE/STANDBY LOGIC FOR KUBERNETES SINGLE-INSTANCE SERVICES

AUTHORS:
Tim Kuik
Matthew Chou

ABSTRACT

Services that were originally designed to run as a single-instance are unable to be run within a high availability model, which is expected in a clustered environment. Presented herein are techniques through which single-instance services can be converted into multi-instance services without any changes to their binary service logic. This provides an immediate benefit in providing a higher level of availability for any Kubernetes single-instance service.

DETAILED DESCRIPTION

During operation in a network environment, if a node upon which a single-instance service is running fails, the service would be unavailable until the node outage is detected and the service is brought up to a functioning state on a different node in the cluster. In some instance, it has been observed on some cloud platforms that recovery from a (virtual) node outage can take up to 20-30 minutes. Thus, it would be advantageous to mimic the functionality and benefits of high availability services in order to minimize downtime, without touching the original service at all.

In many network service implementations, there are often a number of services that are not yet multi-instance capable. With Kubernetes, it has been discovered that it can take minutes to detect node outages, and potentially several minutes on some cloud platforms. One solution is to make all services multi-instance, however, this takes time and, in some rare cases, perhaps some services will remain single-instance. In order to improve the availability for this category of services, this proposal provides a method that allows single-instance services to run in an active/standby mode without any change to the service logic itself.

The solution proposed herein may or may not leverage the use of a sidecar to determine a leader for a service. The service instance that is elected will be the active one. All other services will be run in a standby mode. Thus, the method essentially provides a wrapper for the service that monitors for leadership changes, which is run on a per-service basis. The leader instance is activated while the other instances remain as a hot standby. When an active service's role changes from active to standby, it is guaranteed that the deployable unit, such as the POD containing the service, is restarted, so that the service is brought to an acquiesced state. Figure 1A illustrates example details associated with the method of this proposal, which is discussed in further detail below, and Figure 1B illustrates an exemplary environment in which the method can be utilized involving two PODs that are alive but only one is ready, such that a first POD (POD1) is active and a second POD (POD2) is in a standby mode.

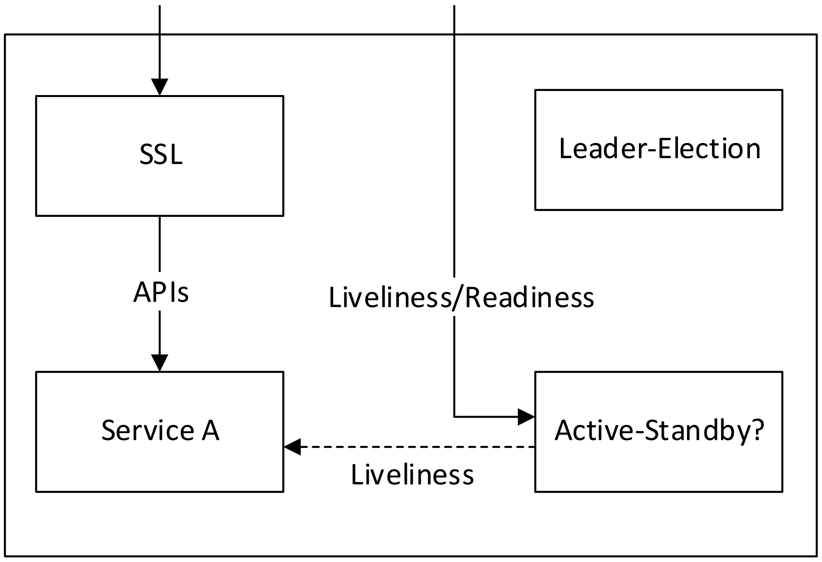

Figure 1A: Example Method Flow

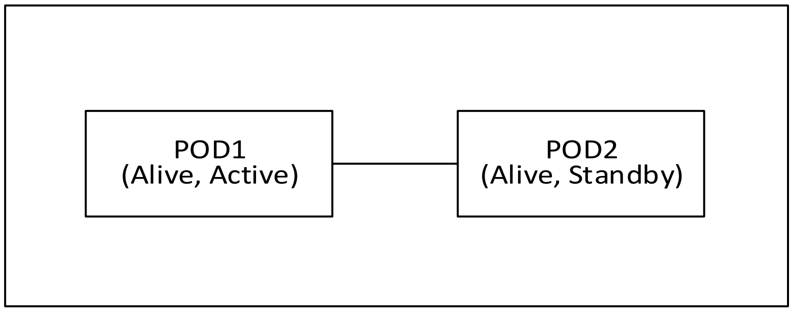

Figure 1B: Exemplary Environment

Logic associated with the method of this proposal may or may not involve the use of a leader sidecar. For instances involving a leader sidecar, the sidecar can determine which instance is the active instance, with the leader being the instance that would be selected to be active. In either case, a new wrapper script is injected, as needed, to perform the discussed logic.

The active/standby script can be injected into the service container, which can be performed at the time of imaging, but it is also be possible to do this dynamically, for example, with a layer rebuild, leveraging a shared PersistentVolumeClaim (PVC), or other similar mechanism. This script performs the following functions:

- The Kubernetes POD is monitored to see if it is the leader. When it is the leader, the service is started. If one were to do this dynamically the old command would be passed as a new environment (ENV) variable in the specification file but for the method proposed herein, it is baked into the layer during build time.
- The readiness probe for the POD is altered to allow for the wrapper to handle the probe. When the POD is the leader, it returns the readiness result from the service, which is configurable via the service YAML (Yet Another Markup Language) specification. When the POD is not the leader, the readiness probe returns that it is not ready so that no other service will send traffic to the POD.
- The liveness probe for the POD is also altered to allow for the wrapper to handle the probe. When the POD is the leader, it returns the liveness result from the service, which is configurable via the service YAML specification. When the POD is not the leader, the liveness probe returns that it is alive.
- To handle the logic when a leader is demoted for some reason, as is generally applicable in an extended connectivity case, the wrapper also detects this case and will modify the liveness probe response to return a failure condition so that the POD is restarted.

One of the key elements of the method described herein is that there are required YAML ENV variable declarations that are added to configure how to propagate the probes for the active service Accordingly, with these changes, the services that are single-instance are able to become multi-instance without a single modification within the service binary.

Thus, the method of this proposal provides an immediate benefit in providing a higher level of availability for any Kubernetes single-instance service.