Proposal: Custom Metrics **Watch API**

Motivation:

The custom-metrics API (and the external-metrics API, the following document will use custom-metrics API to mean both of these APIs) is a great way of modelling arbitrary flows of metrics, backed by whatever system seems fitting to do the job.

The API however only exposes endpoints to explicitly query metrics at a given point in time. For many use-cases (including autoscaling) this can be fine. However, there are use-cases, which require the system to act timely based on a change in metrics. One such example is scale-from-zero.

There are systems like Knative and soon the HPA itself, which allow Pods to be scaled down to 0 if there is no work to be done. However, once traffic appears, these systems need to generate Pods to fulfill that work. This should happen as quick as possible.

Prior Art:

Kubernetes' own API:
https://kubernetes.io/docs/reference/using-api/api-concepts/#efficient-detection-of-changes

Considerations:

"Incompatible" backends

Based on the list of current implementations, the common backends (Prometheus, Azure, Stackdriver, Datadog) don't usually provide an API to watch for metric changes like this which makes them a lot less suitable for the scale-from-zero scenario described above.

However, the Watch API could still be implemented for those backends as well, reducing the amount of data sent to the autoscalers if metrics do not change often and/or if changes are not reflected as often in those backend systems. Many metrics in these systems have a fixed sample interval. The Watch API could therefore be implemented as a poll in that frequency to reduce load on the backend and only nudge the autoscaler if necessary which improves efficiency of the system and its backends overall. The implementation of said polling would also be local to the specific custom-metrics API provider and thus the implementor (e.g. Stackdriver) has all the context necessary to set this polling interval to whatever value is a good fit for the specific backend.

Alternative backends or plain apiserver implementations

Using the custom-metrics apiserver one can implement the custom-metrics API in whichever way fits best towards the metrics that it's supposed to provide. Taking the scale-from-zero use-case mentioned above, Knative currently implements a push-based metrics system for time critical metrics. A websocket connection pushes the relevant metrics as soon as they are available and the autoscaler itself is nudged to immediately recompute its aggregations and scale accordingly, resulting in an almost instant scale-to-N once traffic arrives.

Using the custom-metrics apiserver, all of this push-based handling can be used as a "backend" for the custom-metrics API. The Watch API could then be used to efficiently implement the scenario without ever having to leave the custom-metrics API surface and thus keeping compatibility between different autoscaler implementations. Likewise, the HPA could in the future make use of Knative's metric system in a just as efficient way as our custom system with websocket connections.

Conclusion

The Watch API as laid out is an extension to the custom-metrics API. Where applicable it enables use-cases that require rapid reactions to changes in collected metrics. If backends do not support it, the addition can still be implemented using backend polling mechanisms.