Introduction

This lesson provides a short recap of what we have discussed so far and also the limitations of Metrics Server.

WE'LL COVER THE FOLLOWING

Insufficient facts always invite danger.

- Spock

Short recap

So far, we explored how to leverage some of Kubernetes's core features. We used `HorizontalPodAutoscaler` and `Cluster Autoscaler`. While the former relies on **Metrics Server**, the latter is not based on metrics, but on Scheduler's inability to place Pods within the existing cluster capacity. Even though **Metrics Server** does provide some basic metrics, we are in desperate need of more.

Limitations of `Metrics Server`

We have to be able to monitor our cluster and `Metrics Server` is just not enough. It contains a limited amount of metrics, it keeps them for a very short period, and it does not allow us to execute anything but the simplest queries. I can't say that we are blind if we rely only on `Metrics Server`, but that we are severely impaired. Without increasing the number of metrics we're collecting, as well as their retention, we get only a glimpse into what's going on in our Kubernetes clusters.

Being able to fetch and store metrics cannot be the goal by itself. We also need

to be able to query them in search of a cause of an issue. For that, we need metrics to be "rich" with information, and we need a powerful query language.

Creating a self-adaptive system

Finally, being able to find the cause of a problem is not worth much without being notified that there is an issue in the first place. That means we need a system that will allow us to define alerts that, when certain thresholds are reached, will send us notifications or, when appropriate, send them to other parts of the system that can automatically execute steps that will remedy issues. If we accomplish that, we'll be a step closer to having not only a self-healing (Kubernetes already does that) but also a self-adaptive system that will react to changed conditions. We might go even further and try to predict that "bad things" will happen in the future and be proactive in resolving them before they even arise.

All in all, we need a tool, or a set of tools, that will allow us to fetch and store "rich" metrics, that will allow us to query them, and that will notify us when an issue happens or, even better, when a problem is about to occur.

We might not be able to build a self-adapting system in this chapter, but we can try to create a foundation. But, first things first, we need a cluster that will allow us to "play" with some new tools and concepts.

In the next lesson, we will again first create a cluster.