Introduction

This lesson provides an Introduction to this course.

WE'LL COVER THE FOLLOWING

Computers make excellent and efficient servants, but I have no wish to serve under them.

- Spock

`HorizontalPodAutoscaler (HPA)` phases

The adoption of `HorizontalPodAutoscaler (HPA)` usually goes through three phases.

First phase

The first phase is **discovery**. The first time we find out what it does, we usually end up utterly amazed. “Look at this. It scales our applications automatically. I don’t need to worry about the number of replicas anymore.”

Second phase

The second phase is the **usage**. Once we start using `HPA`, we quickly realize that scaling applications based on memory and CPU is not enough. Some apps do increase their memory and CPU usage with the increase in load, while many others don’t. Or, to be more precise, not proportionally. `HPA`, for some applications, works well. For many others, it does not work at all, or it is not

enough. Sooner or later, we'll need to extend `HPA` thresholds beyond those based on memory and CPU. This phase is characterized by *disappointment*. "It seemed like a good idea, but we cannot use it with most of our applications. We need to fall back to alerts based on metrics and manual changes to the number of replicas."

Third phase

The third phase is **re-discovery**. Once we go through the `HPA` v2 documentation (still in beta at the time of this writing) we can see that it allows us to extend it to almost any type of metrics and expressions. We can hook `HPAs` to `Prometheus`, or nearly any other tool, though adapters. Once we master that, there is almost no limit to the conditions we can set as triggers of automatic scaling of our applications. The only restriction is our ability to transform data into Kubernetes' custom metrics.

Our next goal is to extend `HorizontalPodAutoscaler` definitions to include conditions based on data stored in `Prometheus`. But first, let's create a cluster in the next lesson.