Intended Audience and Course Requirements

This lesson focuses on the intended audience and course requirements for this course.

WE'LL COVER THE FOLLOWING

Course overview

In this course, we assume that you are familiar with Kubernetes and that there is no need to explain how Kube API works, nor the difference between master and worker nodes, and especially not resources and constructs like Pods, Ingress, Deployments, StatefulSets, ServiceAccounts, and so on. Therefore we'll master subjects we often dive into after we learn the basics and after we automate all the processes. We'll explore **monitoring**, **alerting**, **logging**, **auto-scaling**, and other subjects aimed at making our cluster **resilient**, **self-sufficient**, and **self-adaptive**.

Course requirements

Apart from assumptions based on knowledge, there are some technical requirements as well. If you are a **Windows user**, please run all the examples from **Git Bash**. It will allow you to run the same commands as macOS and Linux users do through their terminals. Git Bash is set up during Git installation. If you don't have it already, please run the Git setup.

Since we'll use a Kubernetes cluster, we'll need kubectl. Most of the applications we'll run inside the cluster will be installed using Helm, so please make sure that you have the client installed as well. Finally, install jq as well. It's a tool that helps us format and filter JSON output.

It's a tool that helps us format and filter JSON output.

> The commands in this book assume that you're using **Helm v3.+**. If you're running an *older version* of **Helm**, please upgrade it.

Kubernetes flavors

Finally, we'll need a Kubernetes cluster. All the examples are tested using **Docker for Desktop**, **minikube**, **Google Kubernetes Engine (GKE)**, **Amazon Elastic Container Service for Kubernetes (EKS)**, and **Azure Kubernetes Service (AKS)**. I will provide requirements (e.g., number of nodes, CPU, memory, Ingress, etc.) for each of those Kubernetes flavors.

You're free to apply the lessons to any of the tested Kubernetes platforms, or you might choose to use a different one. There is no good reason why the examples from this course shouldn't work in every Kubernetes flavor. You might need to tweak them here and there, but I'm confident that won't be a problem.

Choosing the right Kubernetes flavor

Before you select a Kubernetes flavor, you should know that not all the features will be available everywhere. In the case of local clusters based on **Docker for Desktop** or **minikube**, scaling nodes will not be possible since both are single-node clusters. Other clusters might not be able to use more specific features. I'll use this opportunity to compare different platforms and give you additional insights you might want to use if you're evaluating which Kubernetes distribution to use and where to host it. Or, you can choose to run some chapters with a local cluster and switch to a multi-node cluster only for the parts that do not work in local. That way you'll save a few bucks by having a cluster in Cloud for very short periods.

If you're unsure which Kubernetes flavor to select, choose **GKE**. It is currently the most advanced and feature-rich managed Kubernetes on the market. On the other hand, if you're already used to EKS or AKS, they are, more or less, OK as well. Most, if not all of the things featured in this book will work. Finally, you might prefer to run a cluster locally, or you're using a different (probably on-prem) Kubernetes platform. In that case, you'll learn what you’re missing and which things you’ll need to build on top of “standard offerings” to accomplish the same result.

In the next chapter, we will see how to autoscale Deployments and StatefulSets.