Creating a Cluster

This lesson focuses on creating a cluster and the necessary requirements and gists for this chapter.

WE'LL COVER THE FOLLOWING

Pulling the code

The vfarcic/k8s-specs repository will continue to serve as our source of Kubernetes definitions. We'll make sure that it is up-to-date by pulling the latest version.

> All the commands from this chapter are available in the 05-hpa-custom-metrics.sh Gist.

```
cd k8s-specs

git pull
```

Gists and specifications

Choose the flavor you want and run the commands from its `.sh` file to create the cluster and the required specifications needed in this chapter. The requirements are the same as those we had in the previous chapter. The only exception is **EKS**. We'll continue using the same Gists as before for all other Kubernetes flavors.

> **NOTE:** In the end, you will see a command to `DELETE` the cluster too. Don't execute that command. Use the `DELETE` command only when you

> need to delete the cluster, preferably at the end of the chapter.

> ### A note to EKS users
>
> 🔍 Even though three t2.small nodes we used so far have more than enough memory and CPU, they might not be able to host all the Pods we'll create. EKS (by default) uses AWS networking. A t2.small instance can have a maximum of three network interfaces, with four IPv4 addresses per each. That means that we can have up to twelve IPv4 addresses on each t2.small node. Given that each Pod needs to have its own address, that means that we can have a maximum of twelve Pods per node. In this chapter, we might need more than thirty-six Pods across the cluster. Instead of creating a cluster with more than three nodes, we'll add Cluster Autoscaler (CA) to the mix, and let the cluster expand if needed. We already explored CA in one of the previous chapters and the setup instructions are now added to the Gist eks-hpa-custom.sh.

Please use one of the Gists that follow to create a new cluster. If you already have a cluster you'd like to use for the exercises, please use the Gists to validate that it fulfills all the requirements.

GKE

- gke-instrument.sh: **GKE** with 3 n1-standard-1 worker nodes, **nginx Ingress** and **Prometheus** Chart, and environment variables **LB_IP**, **PROM_ADDR**, and **AM_ADDR**

EKS

- eks-hpa-custom.sh: **EKS** with 3 t2.small worker nodes, **nginx Ingress**, **Metrics Server**, **Prometheus** Chart, environment variables **LB_IP**, **PROM_ADDR**, and **AM_ADDR**, and **Cluster Autoscaler**

AKS

- aks-instrument.sh: **AKS** with 3 Standard_B2s worker nodes, **nginx Ingress** and **Prometheus** Chart, and environment variables **LB_IP**, **PROM_ADDR**, and **AM_ADDR**

Docker for Desktop

- docker-instrument.sh: **Docker for Desktop** with **2 CPUs, 3 GB RAM**, **nginx Ingress**, **Metrics Server**, **Prometheus** Chart, and environment variables **LB_IP**, **PROM_ADDR**, and **AM_ADDR**

Minikube

- minikube-instrument.sh: **minikube** with **2 CPUs, 3 GB RAM**, **ingress**, **storage-provisioner**, **default-storageclass**, and **metrics-server** addons enabled, **Prometheus** Chart, and environment variables **LB_IP**, **PROM_ADDR**, and **AM_ADDR**

Now we're ready to extend our usage of `HorizontalPodAutoscaler (HPA)`. But, before we do that, let's briefly explore (again) how `HPA` works out of the box in the next lesson.