What's Next?

This lesson summarises what we have learned so far and what we are going to learn next.

WE'LL COVER THE FOLLOWING

Summary

Kubernetes relies heavily on available resources spread throughout the cluster. Still, it cannot do magic. We need to help it out by defining resources we expect our containers will consume.

Even though Heapster is not the best solution for collecting metrics, it is already available in our Minikube cluster, and we used it to learn how much resources our applications use and, through that information, we refined our resource definitions.

Even though Metrics Server is not the best solution for collecting metrics, it is already available in our Minikube cluster, and we used it to learn how much resources our applications use and, through that information, we refined our resource definitions.

Without metrics, our definitions are pure guesses. When we guess, Kubernetes needs to guess as well. A stable system is a predictable system based on facts, not someone's imagination. Metrics Server helped us transform our assumptions into measurable facts which we fed into Kubernetes which, in turn, used them in its scheduling algorithms.

Exploration of resource definitions led us to the Quality Of Service (QoS). Even though Kubernetes decides which QoS will be used, knowing the rules used in the decision process is essential if we are to prioritize applications and their availability.

All that leads us to the culmination of the strategies that make our clusters secure, stable, and robust. Dividing a cluster into Namespaces and employing RBAC is not enough.

RBAC prevents unauthorized users from accessing the cluster and provides permissions to those we trust. However, RBAC does not prevent users from accidentally (or intentionally) putting the cluster in danger through too many deployments, too big applications, or inaccurate sizing. Only by combining RBAC with resource defaults, limitations, and quotas can we hope for a fault tolerant and robust cluster capable of reliably hosting our applications.

Destroying Everything

We are about to delete the Minikube cluster for the last time.

```
minikube delete
```

We learned almost all the essential Kubernetes objects and principles. The time has come to move to a “real” cluster. In the next chapter, we explore creating a production-ready cluster.

In the next lesson, we will go through a comparison of Kubernetes resource management with the Docker Swarm equivalent.