Single vs. Multi-Container Pods

In this lesson, we will briefly discuss a single-container and a multi-container Pod.

WE'LL COVER THE FOLLOWING

Why Single-container Pods?

We should not think of Pods as resources that should do anything beyond a definition of the smallest unit in our cluster. A Pod is a collection of containers that share the same resources. Not much more. Everything else should be accomplished with higher-level constructs. We'll explore how to scale Pods without changing their definition in one of the next chapters.

Let's go back to our original multi-container Pod that defined `api` and `db` containers. That was a terrible design choice since it tightly couples one with the other. As a result, when we explore how to scale Pods (not containers), both would need to match. If, for example, we scale the Pod to three, we'd have three APIs and three DBs. Instead, we should have defined two Pods, one for each container (`db` and `api`). That would give us enough flexibility to treat each independently from the other.

There are quite a few other reasons not to put multiple containers in the same Pod. For now, just be patient. Most of the scenarios where you might think that multi-container Pod is a good solution will probably be solved through other resources.

📝 A **Pod** is a collection of containers. However, that does not mean that multi-container Pods are common. They are rare. Most Pods you'll create will be single container units.

Multi-container Pods

Does that mean that multi-container Pods are useless? They're not. There are scenarios when having multiple containers in a Pod is a good idea. However, they are very specific and, in most cases, are based on one container that acts as the main service and the rest serving as side-cars.

A frequent use case is multi-container Pods used for:

- Continuous integration **(CI)**
- Continious Delivery **(CD)**
- Continuous Deployment processes **(CDP)**

We'll explore them later. For now, we'll focus on single-container Pods.

Let's remove the Pod before we move onto container health.

```
kubectl delete -f pod/go-demo-2.yml
```

That's it for now. In the next lesson, we will discuss monitoring the health of a Pod.