Getting Started with ReplicaSets

This lesson will introduce ReplicaSets to us and we will create a cluster for getting started with ReplicaSets.

WE'LL COVER THE FOLLOWING

Understanding ReplicaSets

> Most applications should be scalable and all must be fault tolerant. Pods do not provide those features, **ReplicaSets** do.

We learned that Pods are the smallest unit in Kubernetes. We also learned that Pods are **not fault tolerant**. If a Pod is destroyed, Kubernetes will do nothing to remedy the problem. That is if Pods are created without **Controllers**.

The first Controller we'll explore is called *ReplicaSet*. Its primary, and pretty much only function, is to ensure that a specified number of replicas of a Pod matches the actual state (almost) all the time. That means that ReplicaSets make Pods scalable.

We can think of ReplicaSets as a *self-healing* mechanism. As long as elementary conditions are met (e.g., enough memory and CPU), Pods associated with a ReplicaSet are guaranteed to run. They provide fault-tolerance and high availability.

If you're familiar with Replication Controllers, it is worth mentioning that ReplicaSet is the next-generation *ReplicationController*. The only significant difference is that ReplicaSet has extended support for selectors. Everything else is the same. ReplicationController is considered deprecated, so we'll focus only on ReplicaSet.

> ReplicaSet's primary function is to ensure that the specified number of replicas of a service are (almost) always running.

Let's explore ReplicaSet through examples and see how it works and what it does.

The first step is to create a Kubernetes cluster.

> All the commands from this chapter are available in the 04-rs.sh Gist

Creating A Cluster

We'll continue using Minikube to simulate a cluster locally.

```
minikube start --vm-driver=virtualbox
kubectl config current-context
```

We created a single-node cluster and configured `kubectl` to use it.

Before we explore the first ReplicaSet example, we'll enter into the local copy of the vfarcic/k8s-spec repository and pull the latest version.

```
cd k8s-specs
git pull
```

Now that the cluster is running and the repository with the specs is up-to-date, we can create our first ReplicaSet.

In the next lesson, we will create our first ReplicaSet.