# Getting Started with ConfigMaps

In this lesson, we will explore a bit about the ConfigMap Volume type.

WE'LL COVER THE FOLLOWING



## The Need of the Hour #

> ConfigMaps allow us to keep configurations separate from application images. Such separation is useful when other alternatives are not a good fit.

Almost every application can be fine-tuned through configuration. Traditional software deployment methods fostered the use of configuration files. However, we are not discussing traditional, but advanced, distributed, and immutable deployments through Kubernetes schedulers.

Usage of fundamentally new technology often requires new processes and different architecture, if we are to leverage its potential to its maximum. On the other hand, we cannot just throw away everything we have and start new.

We'll have to try to balance new principles and legacy needs.

If we were to start developing a new application today, it would be, among other things, distributed, scalable, stateless, and fault tolerant. Those are some of today's needs. While we might question how many of us know how to design an application with those quality attributes in mind, hardly anyone would argue against having any of them. What is often forgotten is the configuration. Which mechanism should your new application use to

configure itself? *How about environment variables?*

> Environment variables fit well into distributed systems. They are easy to define, and they are portable. They are the ideal choice for configuration mechanism of new applications.

However, in some cases, the configuration might be too complex for environment variables. In such situations, we might need to fall back to files (hopefully YAML). When those cases are combined with legacy applications which are almost exclusively using file-based configuration, it is evident that we cannot rely only on environment variables.

When a configuration is based on files, the best approach we can take is to bake the configuration into a Docker image. That way, we are going down the fully-immutable road. Still, that might not be possible when our application needs different configuration options for various clusters (e.g., testing and production).

We don't want to convert this into a discussion that ends with "you do NOT need a different configuration for different environments". Rather just assume that you might have an excellent reason for something like that. In such a case, baking config files into images will not do the trick. That's where ConfigMaps comes into play.

## The ConfigMap #

ConfigMap allows us to "inject" configuration into containers. The source of the configs can be files, directories, or literal values. The destination can be files or environment variables.

> **ConfigMap** takes a configuration from a source and mounts it into running containers as a *volume*.

That's all the theory you'll get up-front. Instead of a lengthy explanation, we'll run some examples, and comment on the features we experience. We'll be learning by doing, instead of learning by memorizing theory.

Let's prepare the cluster and see ConfigMaps in action.

> ℹ All the commands from this chapter are available in the 09-config-map.sh Gist.

## Creating A Cluster #

It's still the same process as before, so let's get over with it silently.

```
cd k8s-specs

git pull

minikube start --vm-driver=virtualbox

minikube addons enable ingress

kubectl config current-context
```

Now we can try out the first variation of a ConfigMap.

---

In the next lesson, we will learn to inject configurations from files into containers.