

ConfigMaps

Many applications require configuration via some combination of config files, command line arguments, and environment variables. These configuration artifacts should be decoupled from image content in order to keep containerized applications portable. The ConfigMap API resource provides mechanisms to inject containers with configuration data while keeping containers agnostic of Kubernetes. ConfigMap can be used to store fine-grained information like individual properties or coarse-grained information like entire config files or JSON blobs.

## Overview of ConfigMap

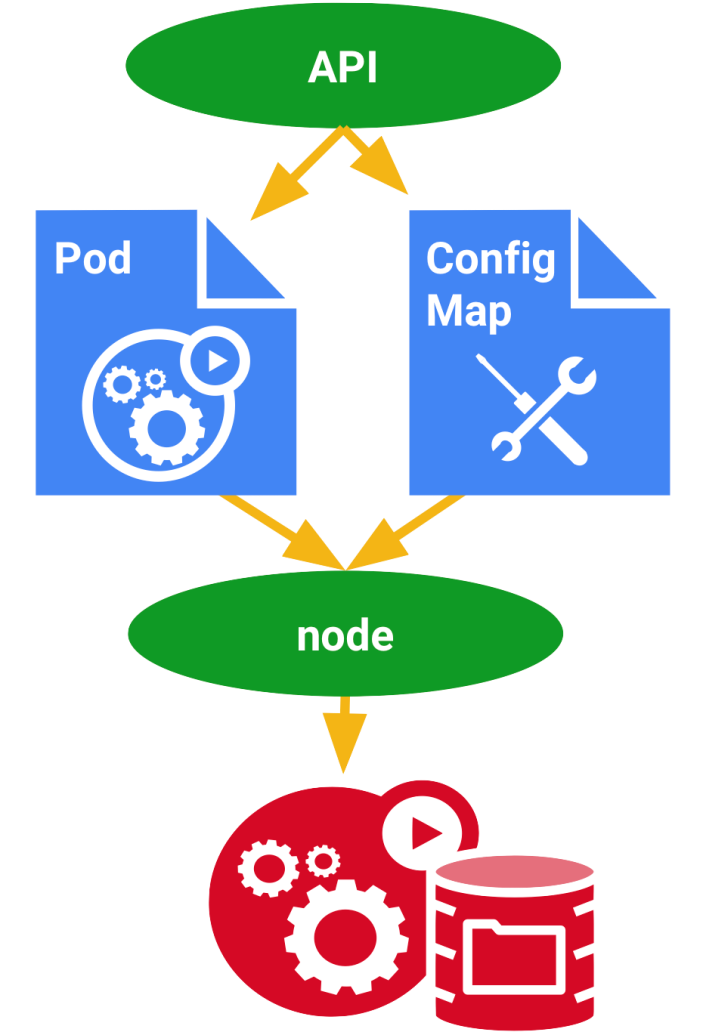
The ConfigMap API resource holds key-value pairs of configuration data that can be consumed in pods or used to store configuration data for **system components such as controllers**.

ConfigMap is similar to Secrets, but designed to more conveniently support working with strings that do not contain sensitive information.

Note: ConfigMaps are not intended to act as a replacement for a properties file. ConfigMaps are intended to act as a reference to multiple properties files. You can think of them as way to represent something similar to the /etc directory, and the files within, on a Linux computer. One example of this model is creating Kubernetes Volumes from ConfigMaps, where each data item in the ConfigMap becomes a new file.

ConfigMaps allow you to decouple configuration artifacts from image content to keep containerized applications portable. This page provides a series of usage examples demonstrating how to create ConfigMaps and configure Pods using data stored in ConfigMaps.

ConfigMaps are useful for storing and sharing non-sensitive, unencrypted configuration information



Syntax

**Create a ConfigMap**

Use the kubectl create configmap command to create configmaps from directories, files, or literal values:

kubectl create configmap <map-name> <data-source>

where <map-name> is the name you want to assign to the ConfigMap and <data-source> is the directory, file, or literal value to draw the data from.

The data source corresponds to a key-value pair in the ConfigMap, where

* key = the file name or the key you provided on the command line, and
* value = the file contents or the literal value you provided on the command line.

**Using ConfigMaps**

To use a ConfigMap with your workloads, you can specify an environment variable that reference the ConfigMap's values, or mount a volume containing the ConfigMap.

