ConfigMaps

Let’s Assume that

We are building containerized image to share with other teammates or open source community. So, it very common to have a custom application configuration, we need to have on each organization or an each environment.

How do we **make** **containerize app Portable?**

We can use the **ConfigMaps** to make the containerize app portable. Before knowing this concept, we need to have knowledge on **PODS, kubectl**

# Objectives

## Concept

Configuring Containerized Application

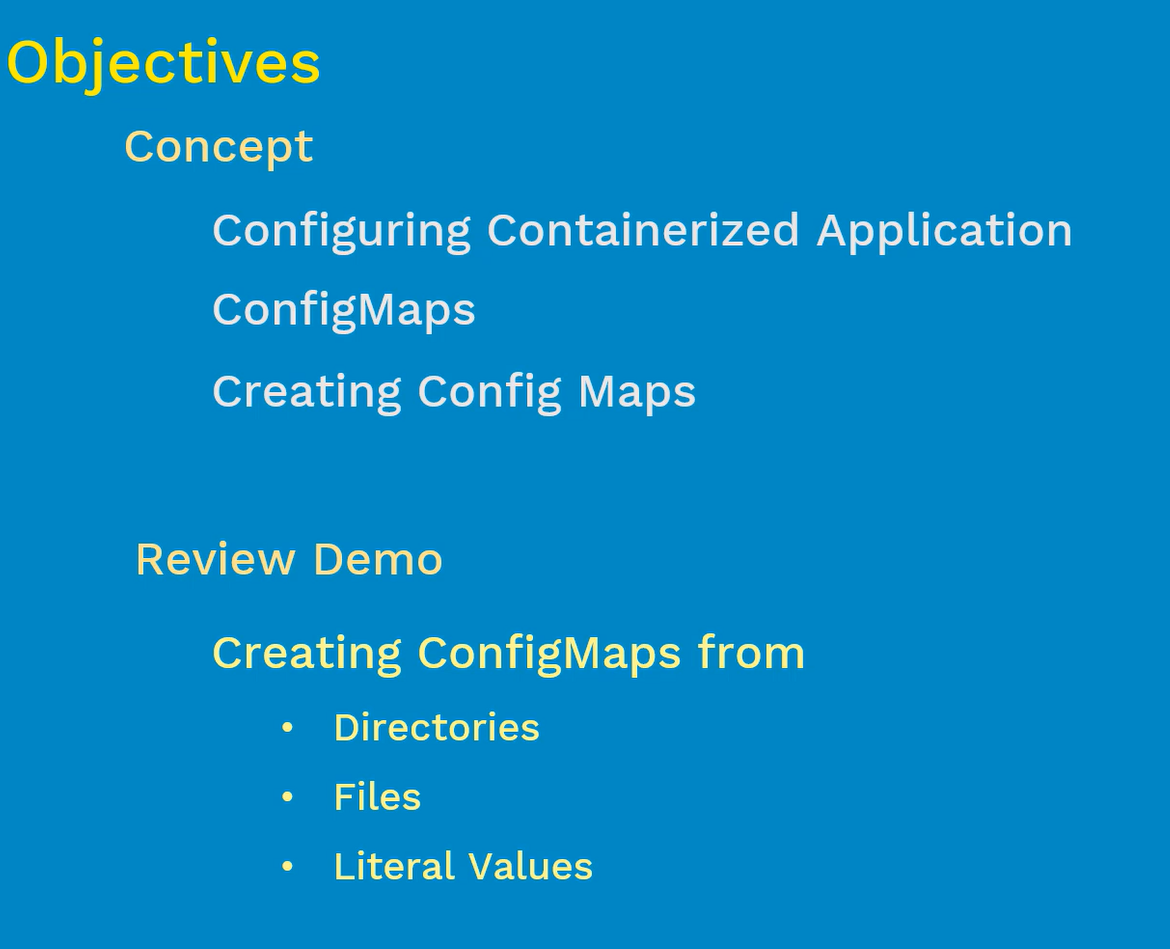
Overview of ConfigMaps

Creating ConfigMaps

## Review Demo 🡪 (deploy on live kubernetes cluster and in advance)

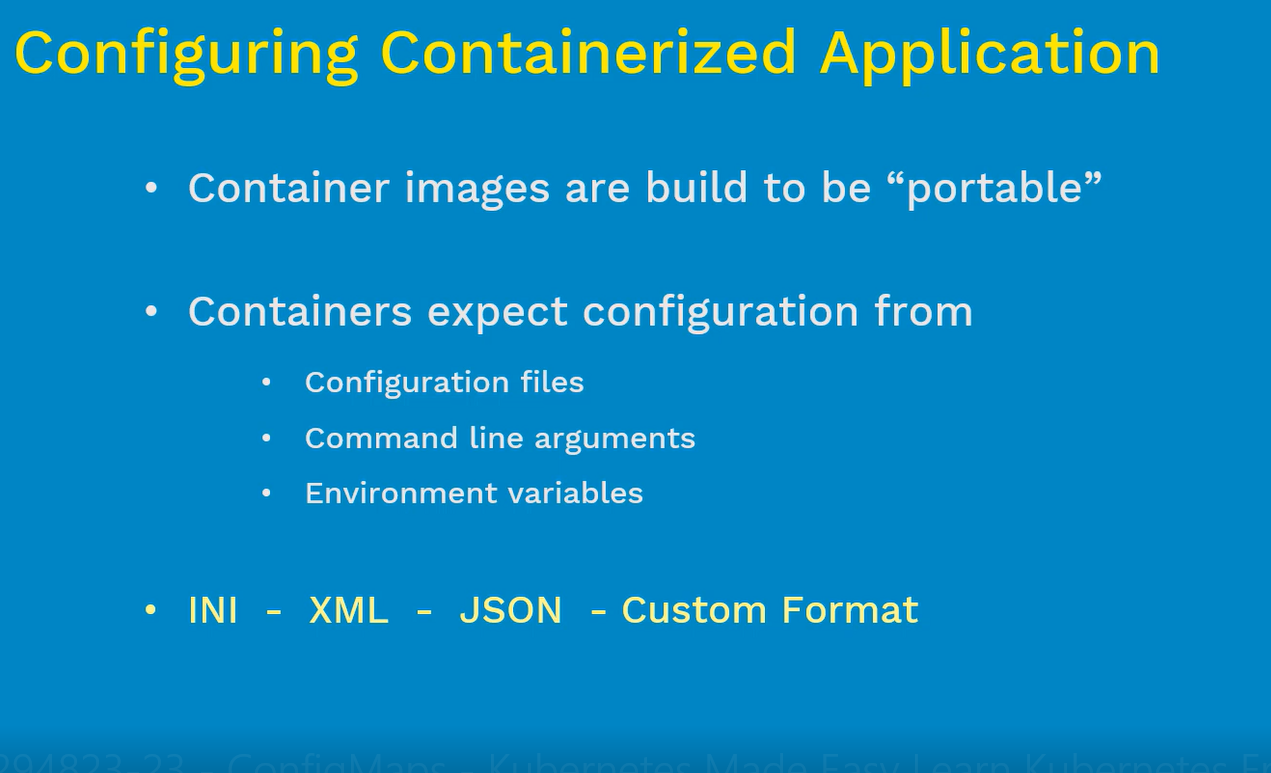
Creating ConfigMaps from

* Directories
* Files
* Literal Values



### Configuring Containerized Application

Generally when we build any container image, we need to make sure image is portable and share to others using docker hub or private registry. When we need, we will download the images from the respective registry



Customer using custom configuration on environments and how can we make sure the configuration to make portable.

Application containers can able to read and understand by using above formats

**How does the configurations is handled inside kubernetes?**

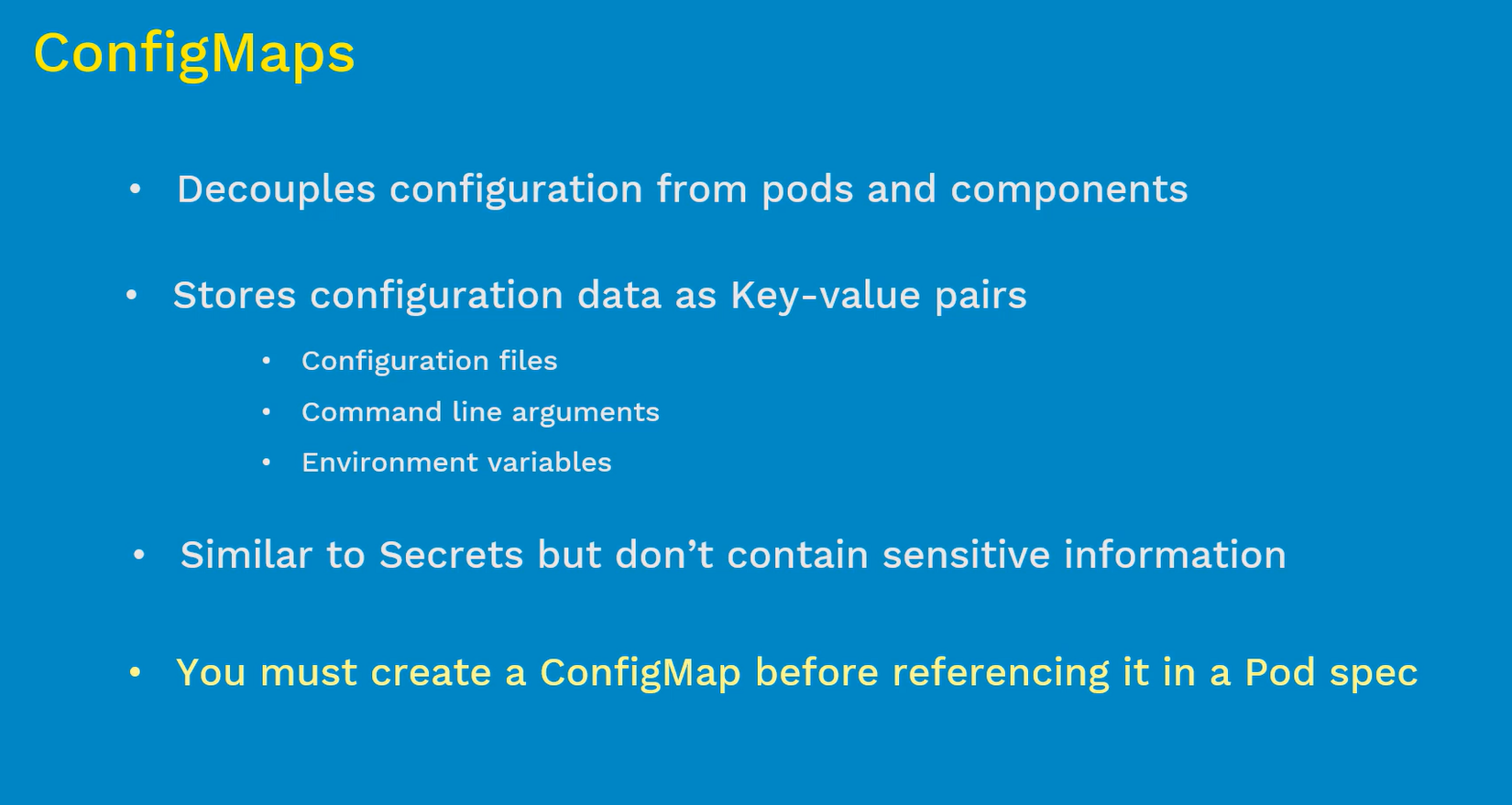
**Using ConfigMaps**

**ConfigMaps:** It is a kubernetes object that allows separately the configurations from pods and components

* It keeps the container portable and makes the configuration easier to change and manage. Prevent the hard-coding configuration data into pod spec
* It stores the configuration data into **Key-Value** pairs

Eg: If we passing any configuration in file, name of file is **Key** and content of the file is **Value**

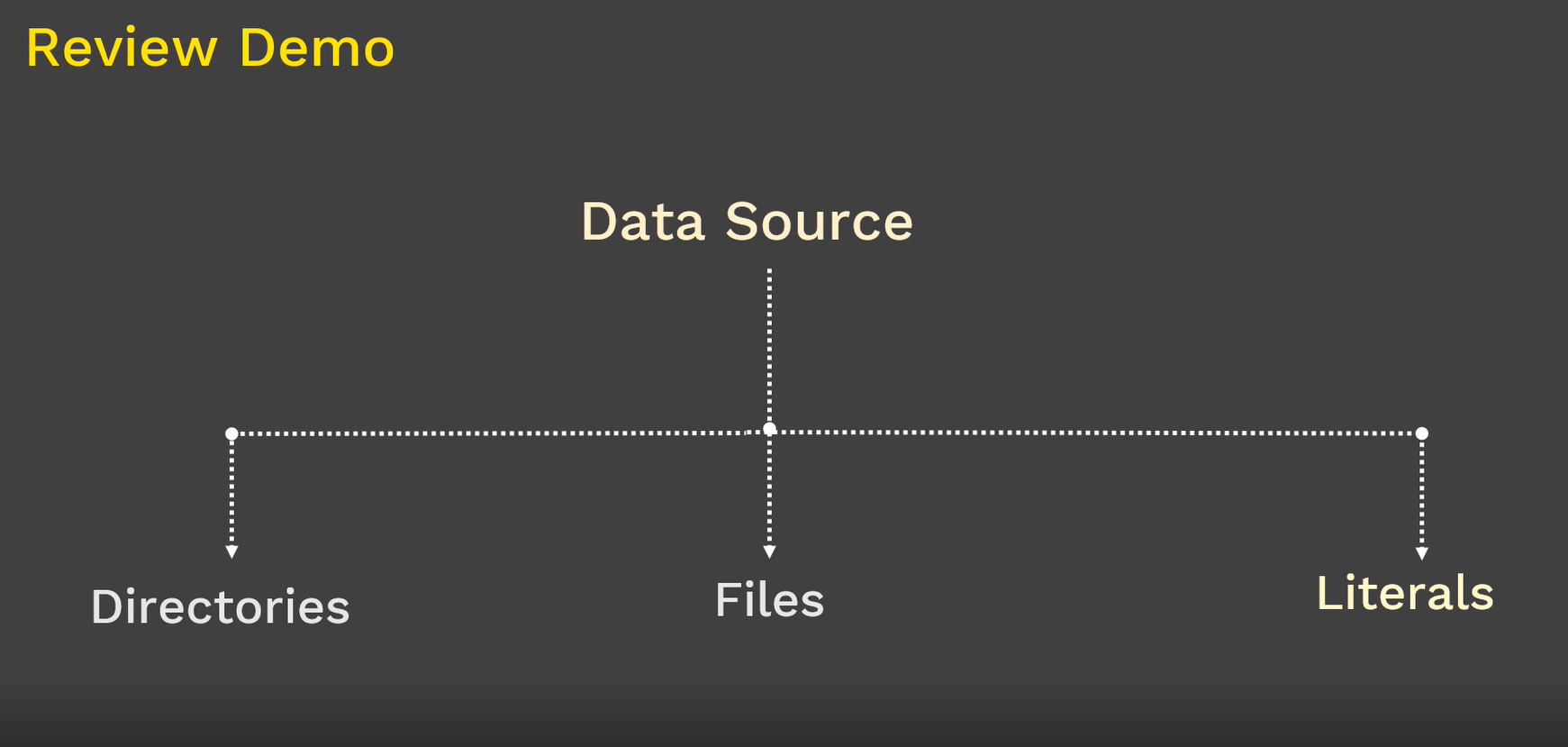
* We can pass the data in **configuration files, command line arguments, Environment Variables**
* With the help of ConfigMaps, we can manage configuration in containers
* If we have any sensitive data passing on the configuration files, we can use another type of object called as “**secrets”**
* We can still run your sensitive information using ConfigMaps, but it will high risk for company
* We must create the ConfigMap before referencing it in a pod spec – {Incase, if we reference a configmap inside the pod spec, if that doesn’t exist then pod won’t start. Also if we reference keys inside a pod that doesn’t exist in configmap then it will prevent pod from starting. Before you reference the configmap and keys inside the pod spec, please do make sure they are created and exist }



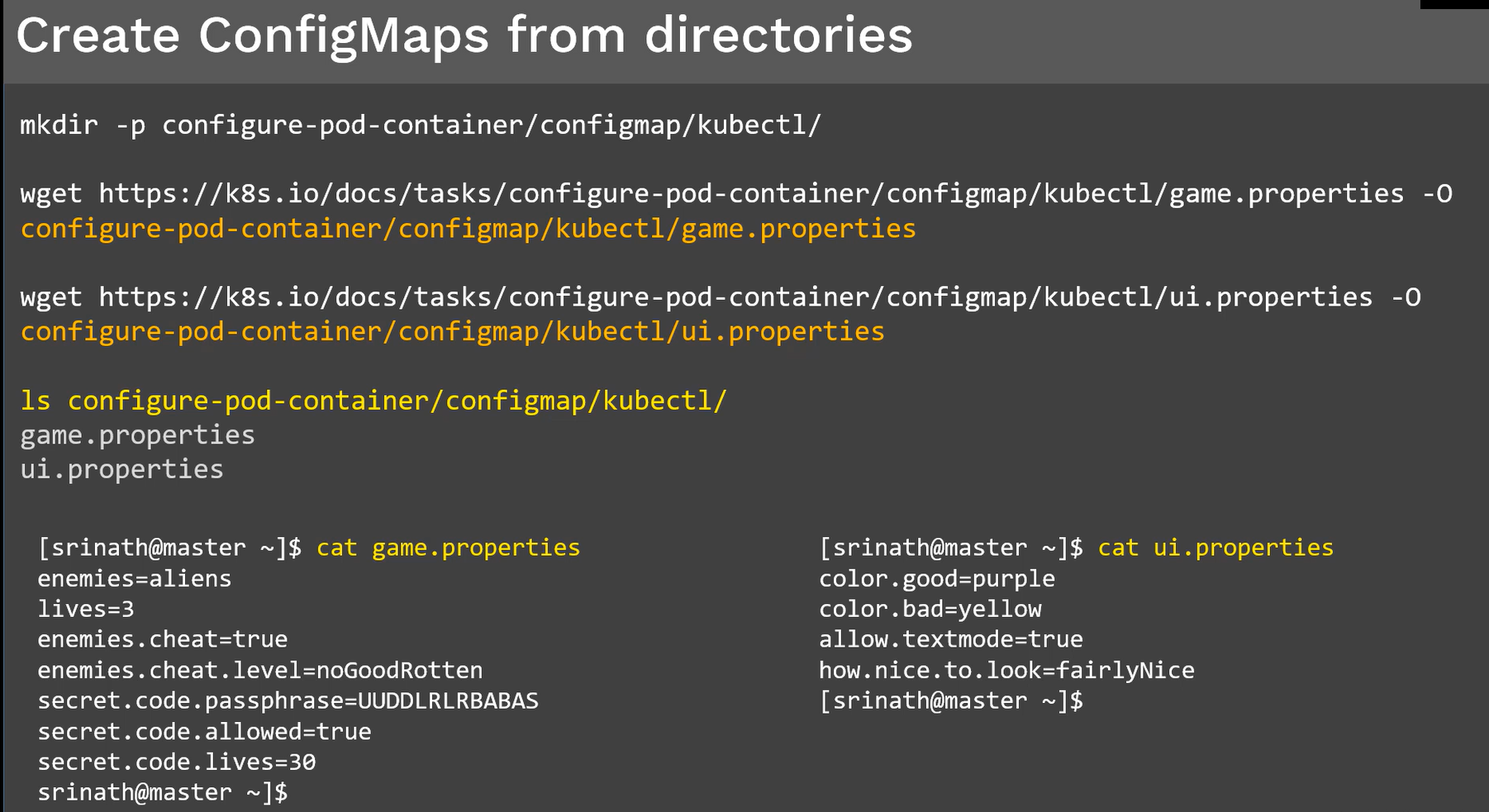
**Syntax:**

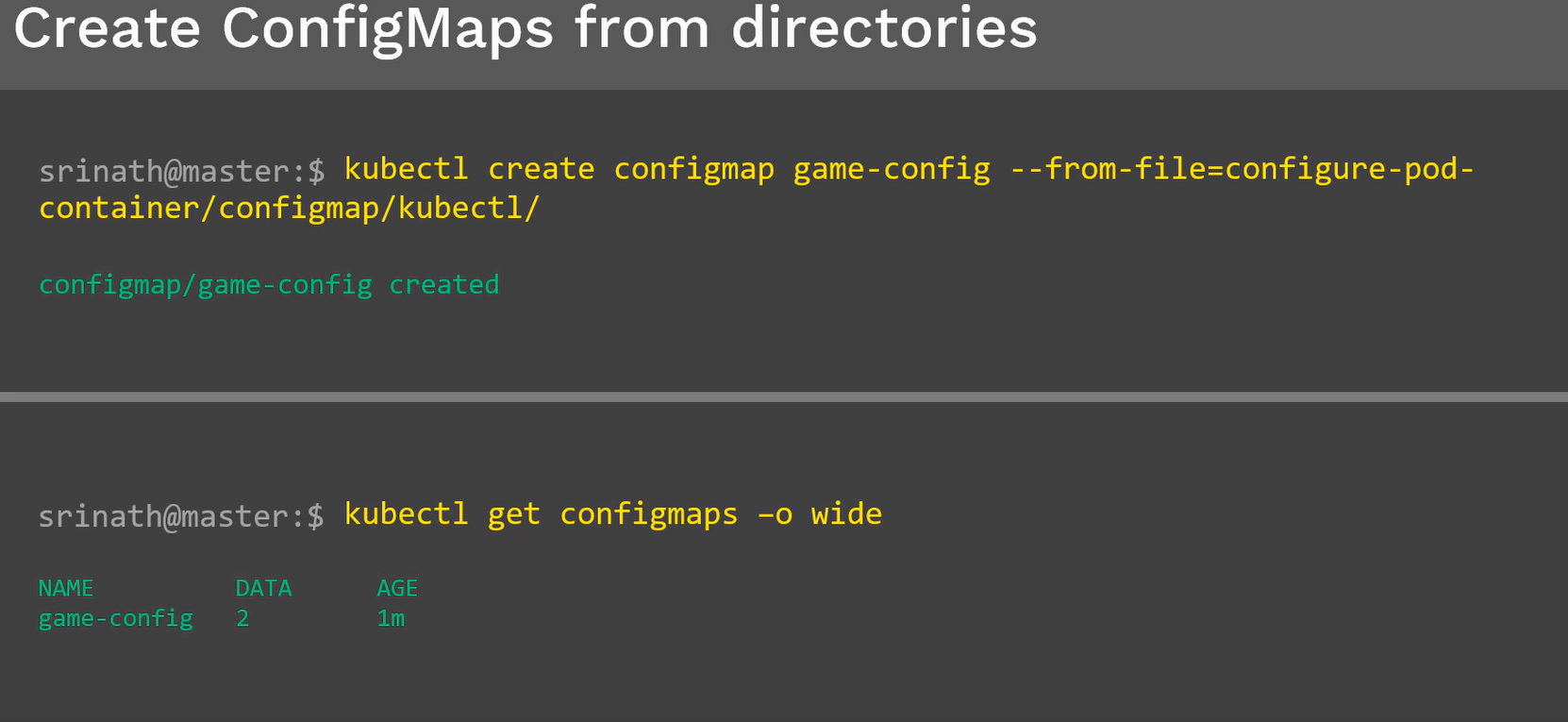
Kubectl create config-map <map-name> <data-source>

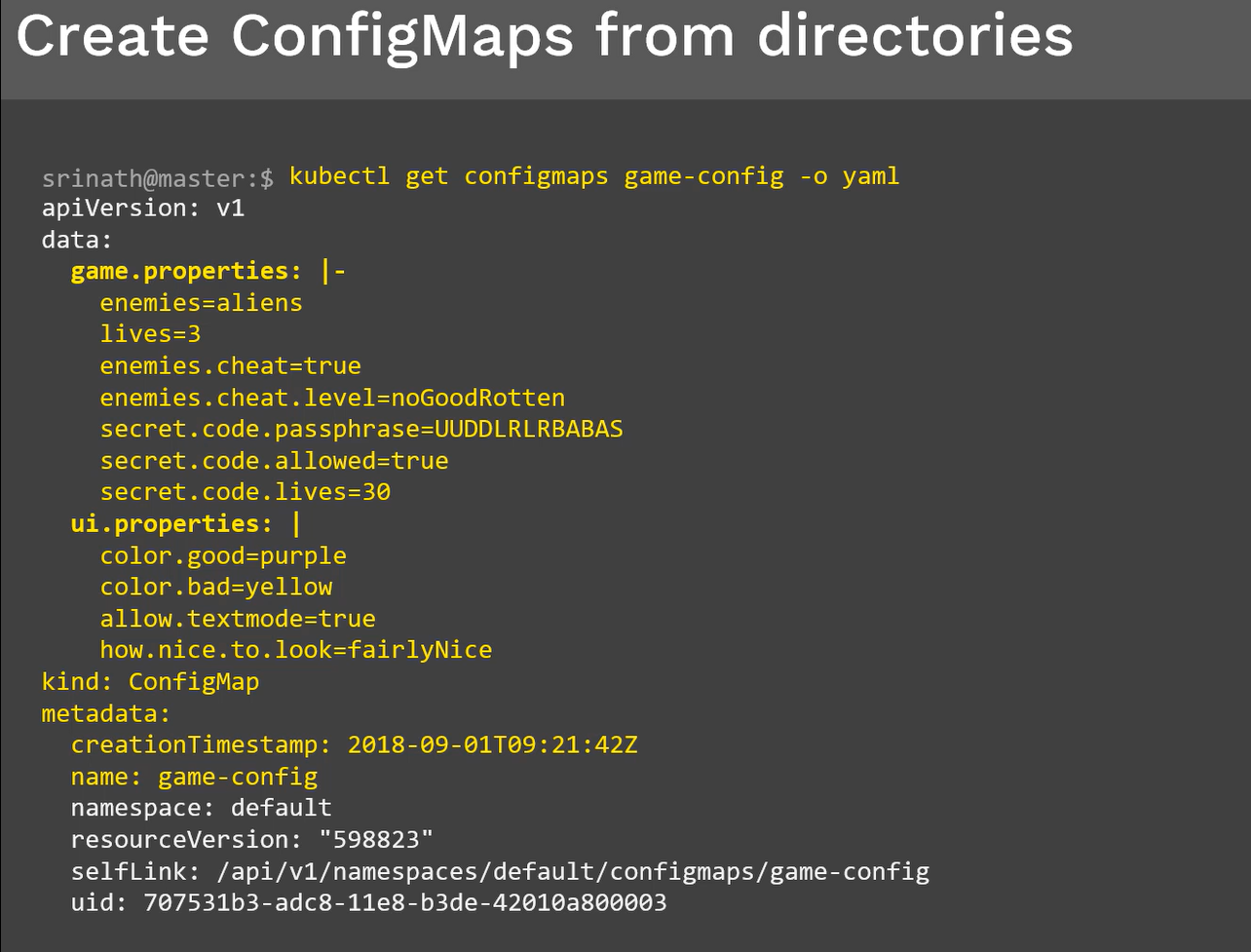




### ConfigMaps using directory:





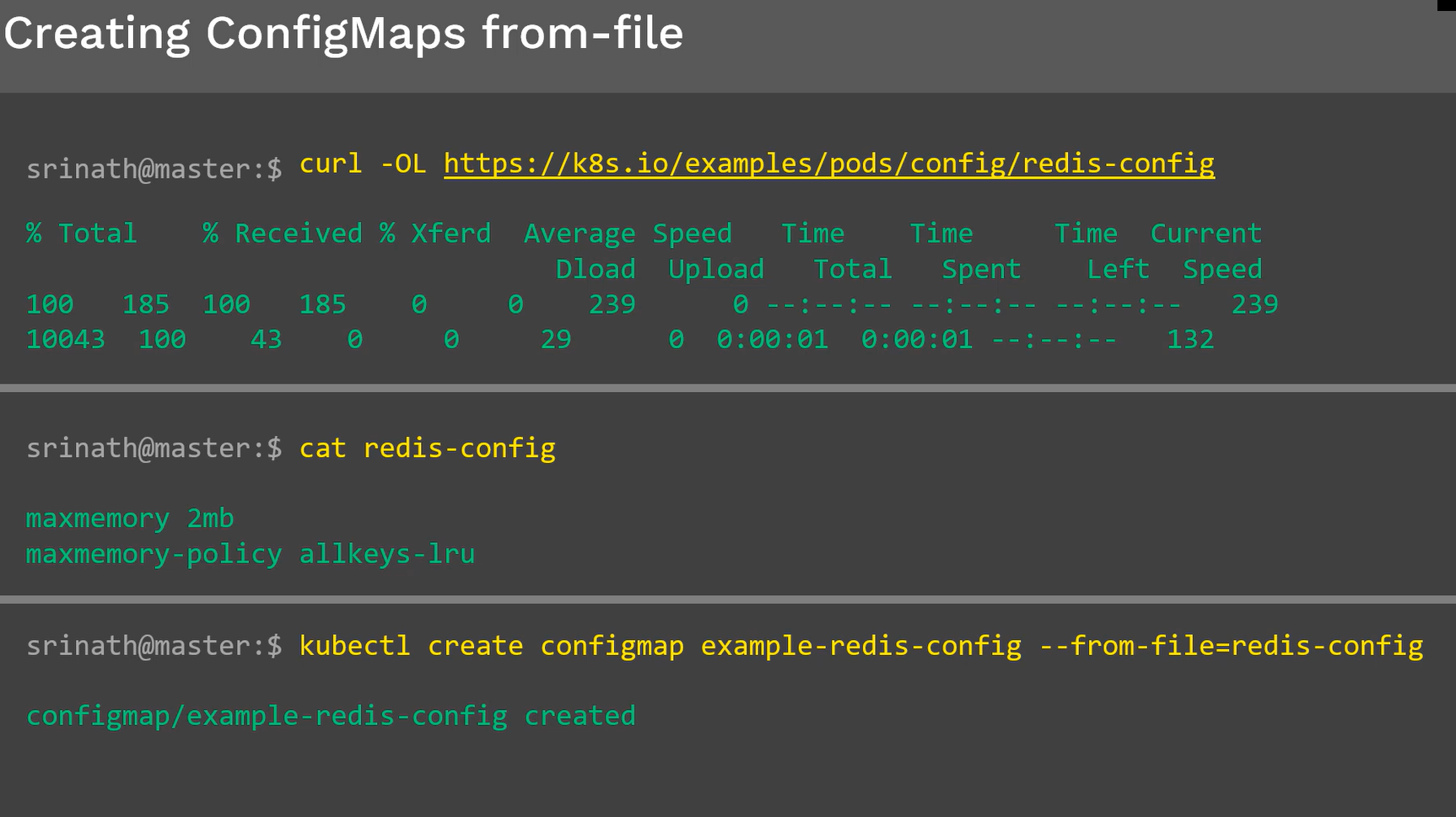


### ConfigMaps using file:

We will create configmap using single file. First we download the sample file from online, then we create configmap using that file after then we create pod spec using that configmap and finally we verify the configuration whether correctly applied or not.



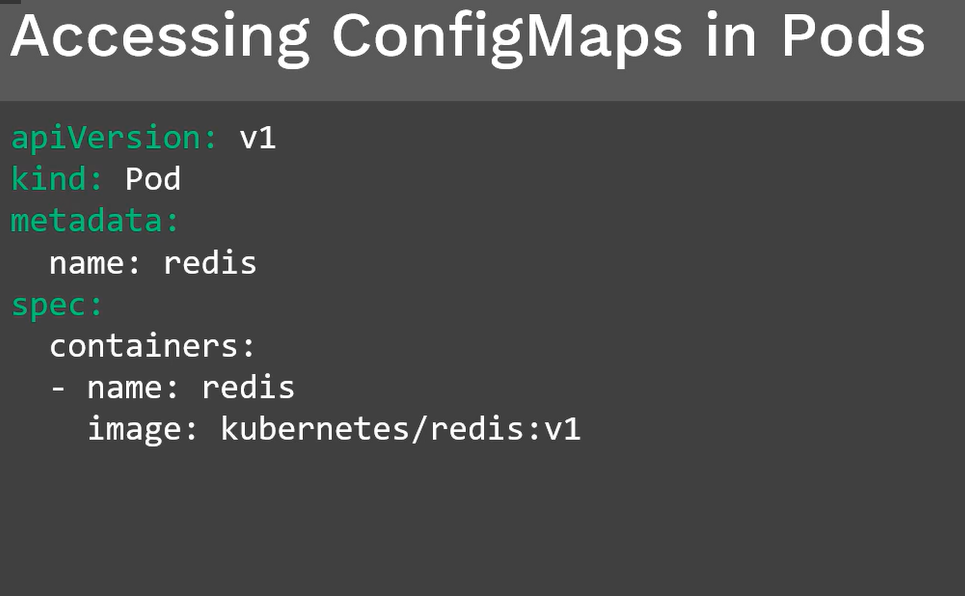
**Ex: configure redis using configmap**



Kubectl get or describe to display configmaps

Next we will discuss the use of the above configmap inside the pod spec

General pod spec file:



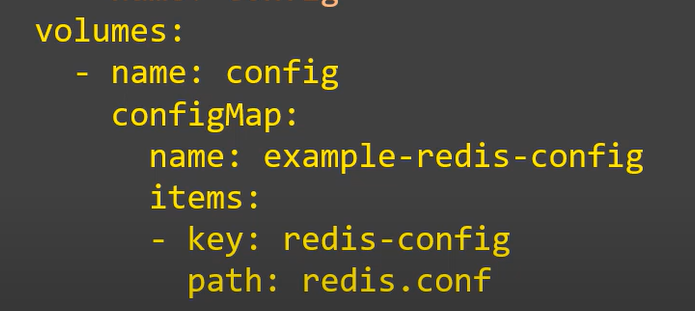
How and where do we use configmap inside the pod spec?

There are two different ways.

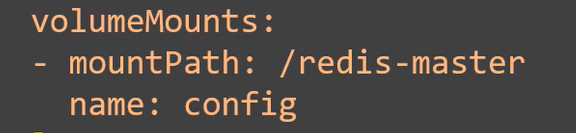
1. By volumes
2. By Environment Variables

#### By Volumes:

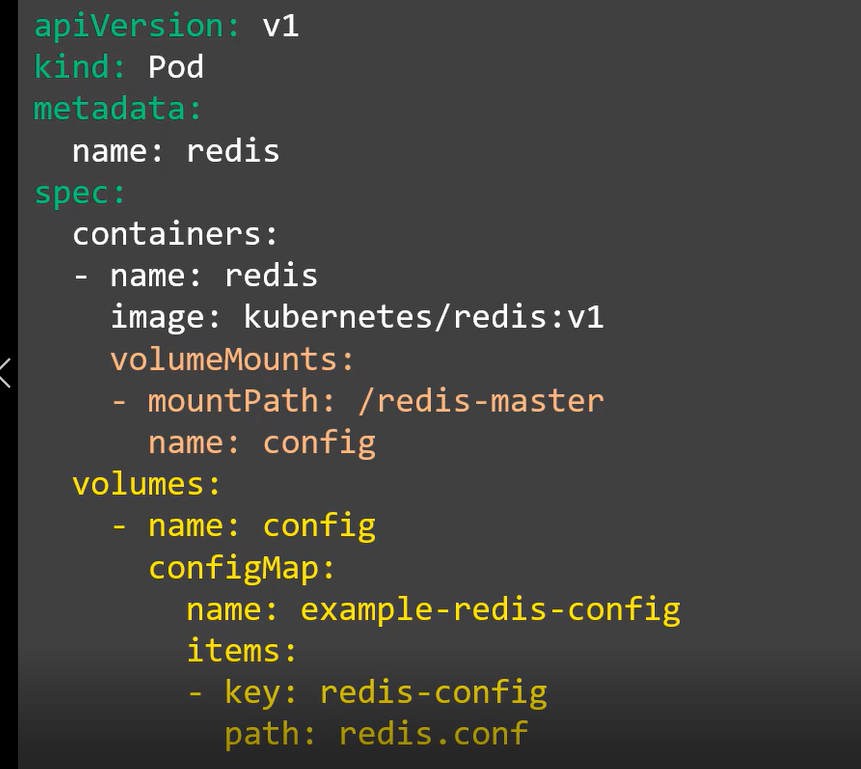
* We will mount the configmap by using volumes
* Volume section name, type of volume(configmap)
* Under Configmap, name of the configmap, path of the configmap



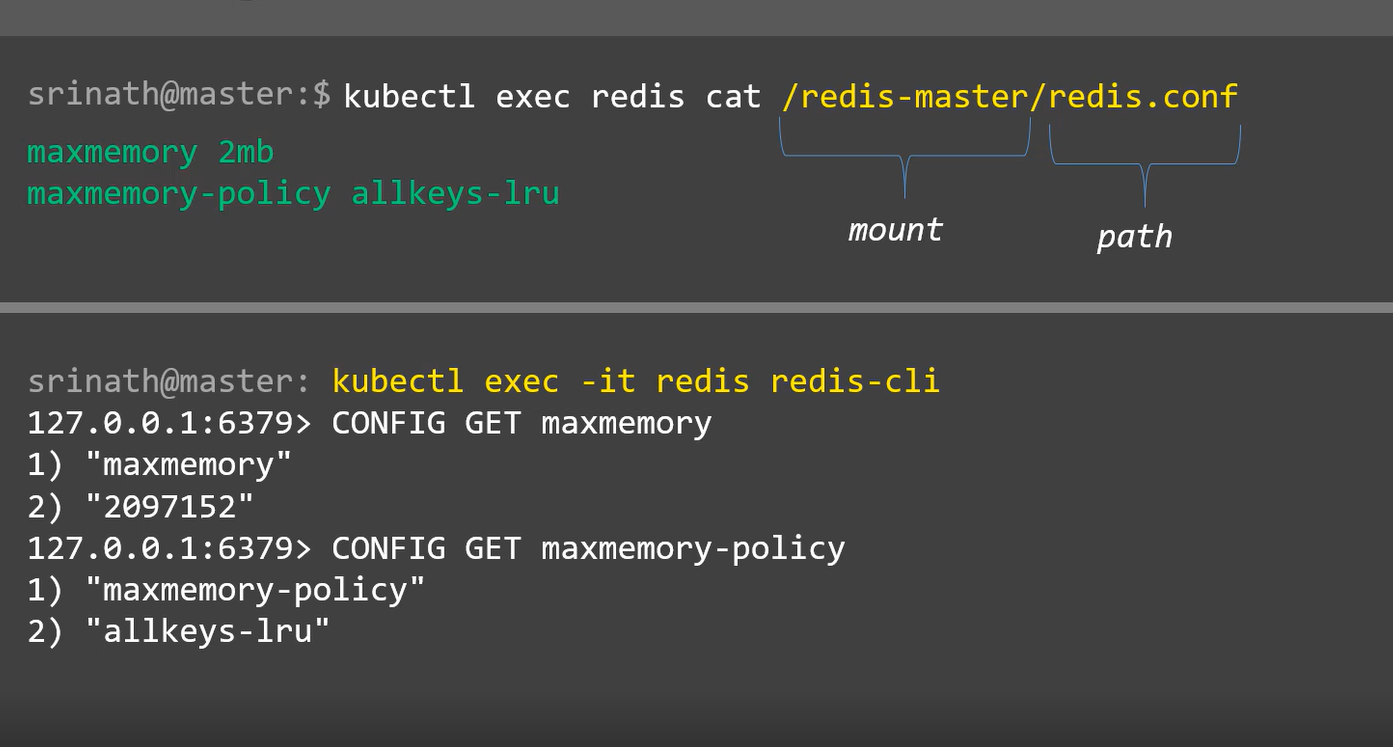
* One more section, mount the configmap using **volumeMounts**



Ex: example of configmap using volumes



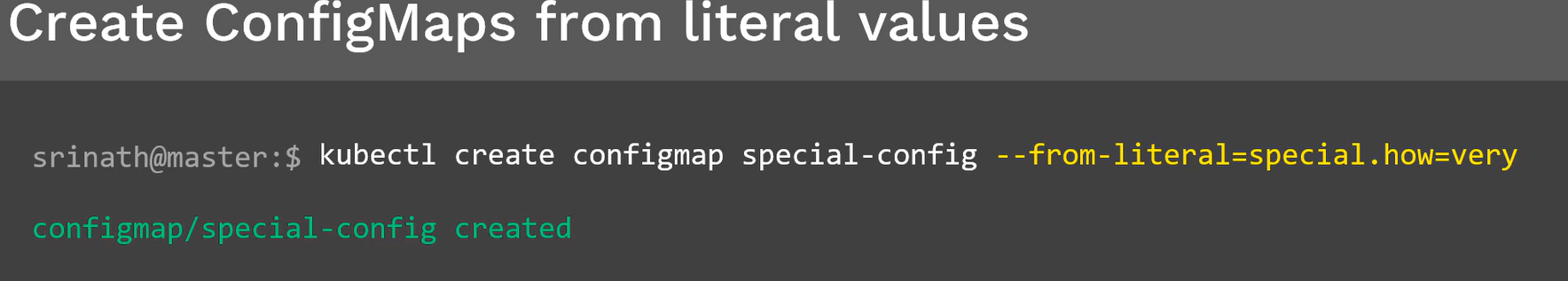
Testing:



### Configmap Using Literals:

Creating configmap taking inputs from the literals values straight away from the command line. Once we have configmap in place then we will create pod spec and finally we will verify the data

Ex:

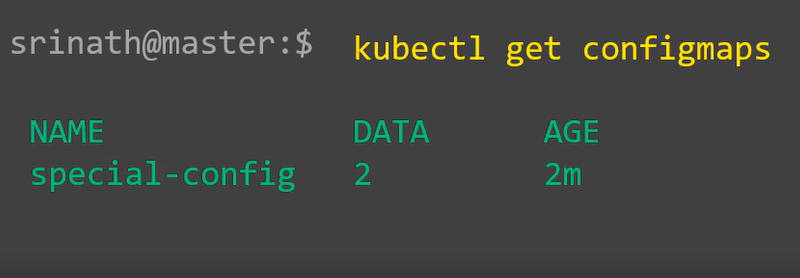


We are creating configmap name as “**special-config**” using literal values **--from-literal**, key is **special.how** and value is **very**

When we use the reference of the configmap in pod spec, we will use **key** not the **value**

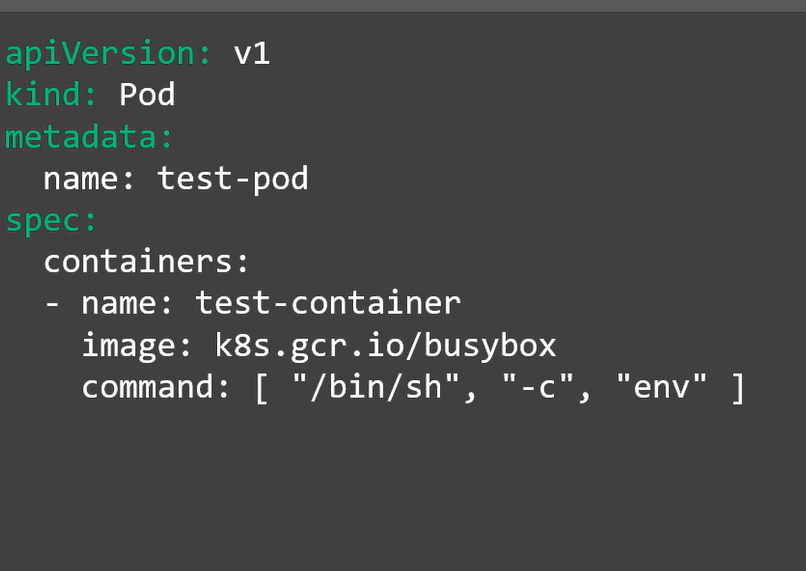
As we discussed earlier, there are two different ways to use configmaps inside pod spec 1. Volumes

2. Environment variables



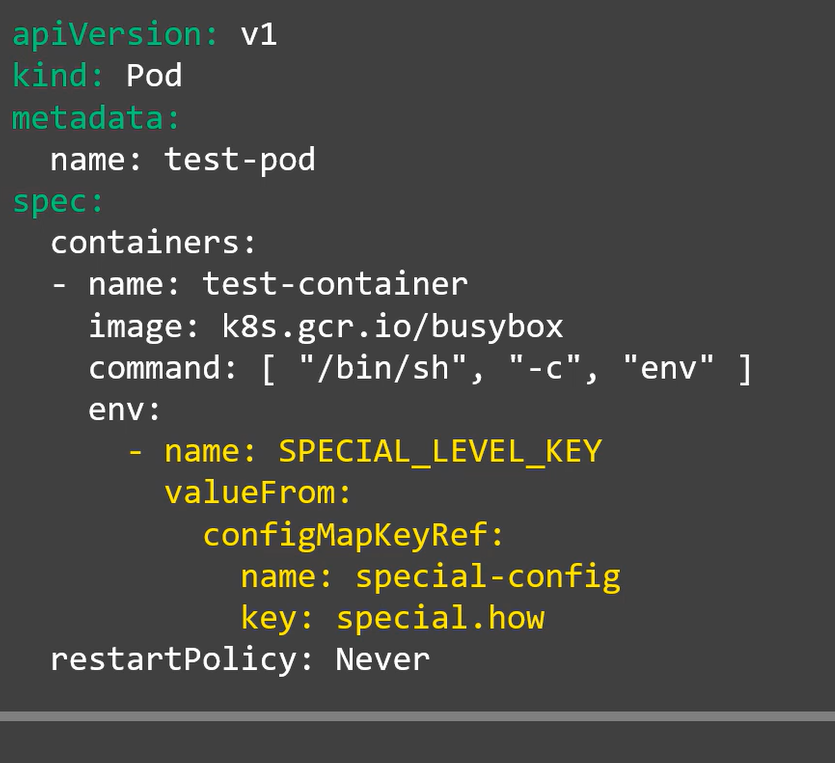
Eg:

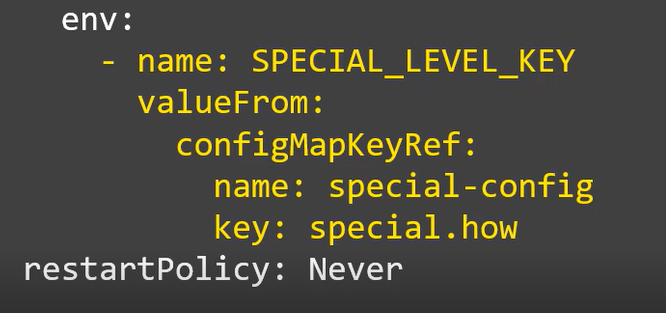
Standard Pod spec in busybox container



There is command to print the env variables once it started

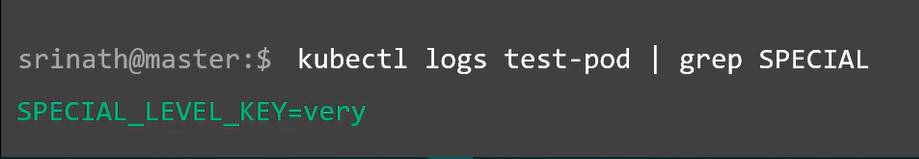
How we can use configmap inside pod using env variables?





* First, we define **name**  of the env variable
* Second, where do we get the value of the “SPECIAL\_LEVEL\_KEY” --🡪 from configmap
* Third, We define the name of the config that we define earlier then followed by key is **special.how**
* **Use kubectl create command to create pod**

Testing:



For Demo:

Please check this file **Configmaps-demo.txt** in the repository