**Why ConfigMaps and Secrets?**

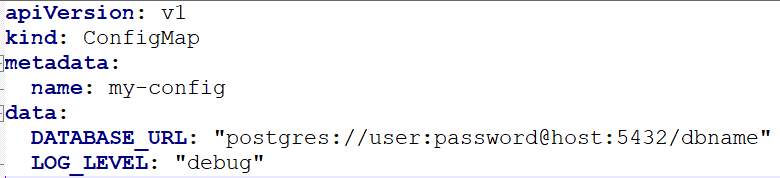
* **ConfigMap** → Store **non-sensitive** configuration data (example: database URLs, feature flags, environment variables).
* **Secret** → Store **sensitive** data (example: passwords, API keys, tokens) in **Base64 encoded** form.

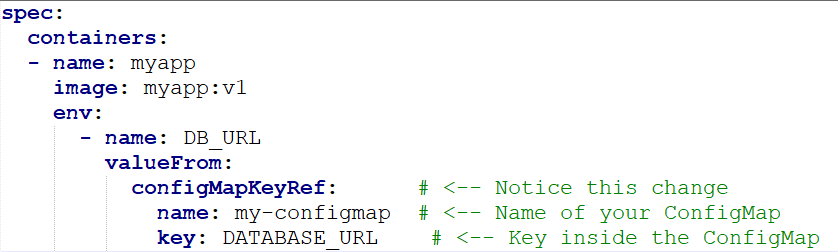
**Idea:**

Don't hardcode configurations inside images. Make them **external** and **injectable** into Pods when running.

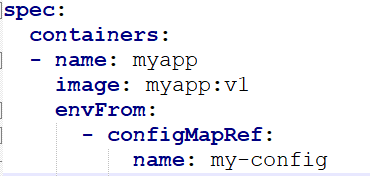
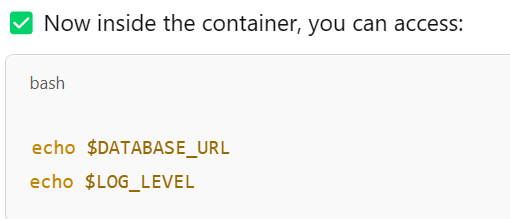
**1. ConfigMap**

* Key-value pairs of **configuration data**.
* Example: app settings, environment variables, URLs, etc.



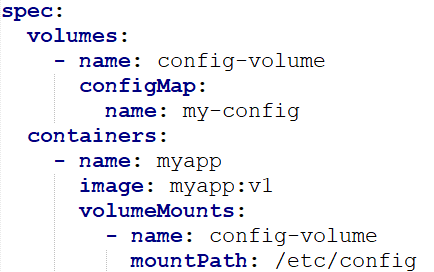
**Loading each variable from ConfigMap**

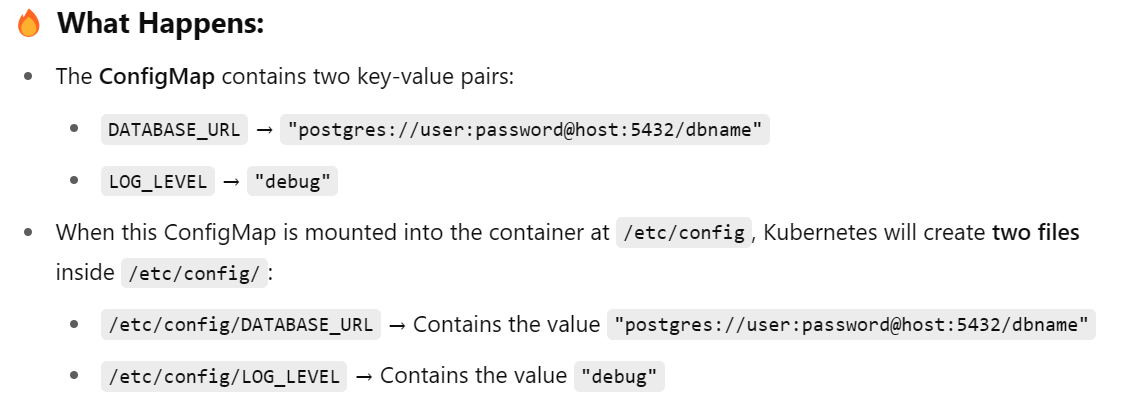
**Loading Entire ConfigMap at once:**

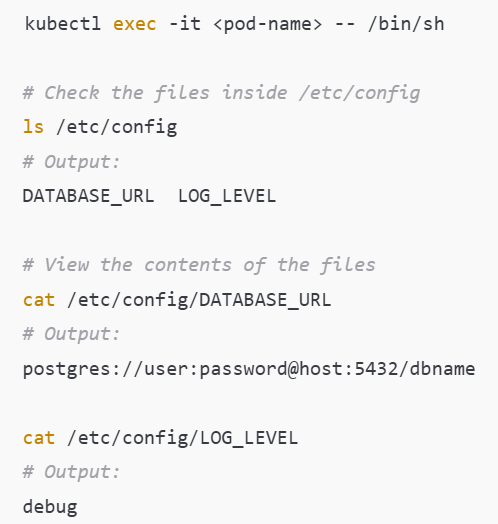
** **

**But if we use the configMaps as env variables, if you update the data in configMap that does not reflect immediately in container unless it restarts. So you will be facing a downtime.**

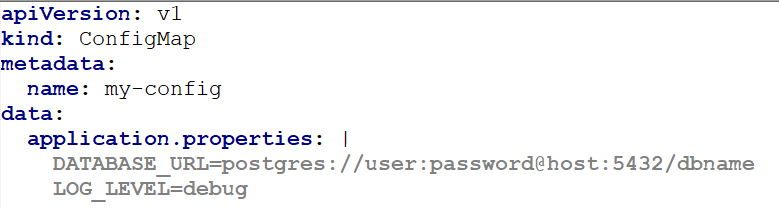
**It’s recommended to use configMaps as volumes(Files) to reflect the data immediately.**

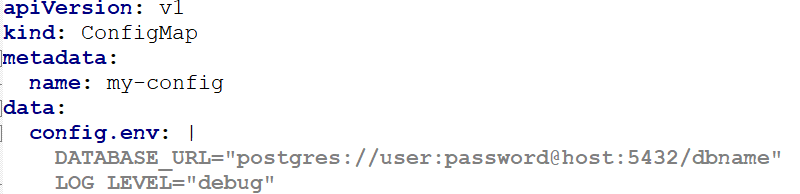


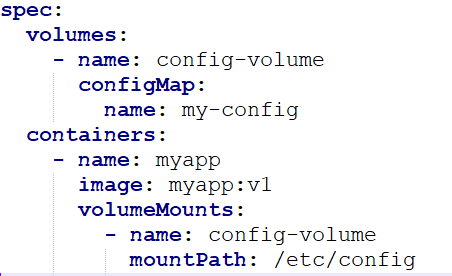
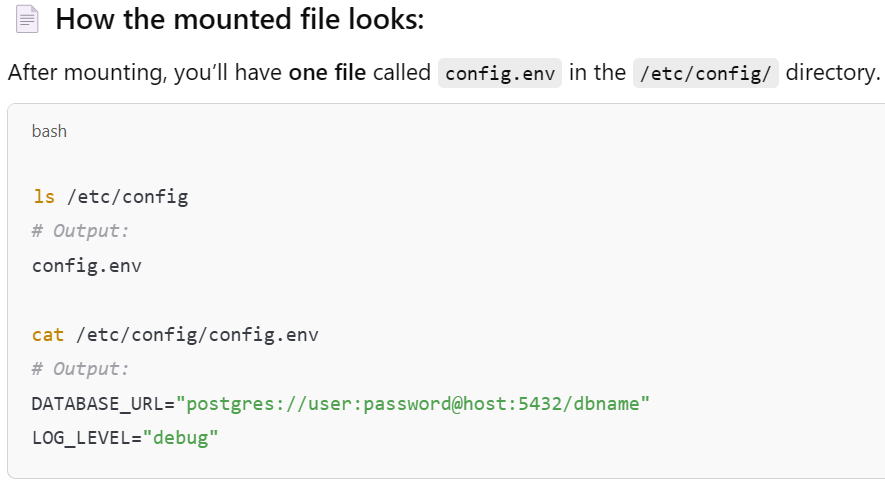




Here, you are getting multiple files each file with each key name, what if you need all the keys present in a single file. You will be passing the data in a file and that file will be present in that voume.

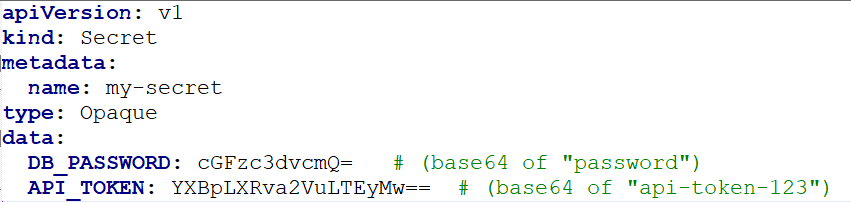


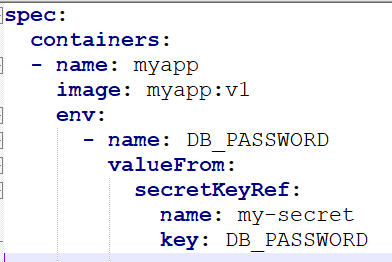
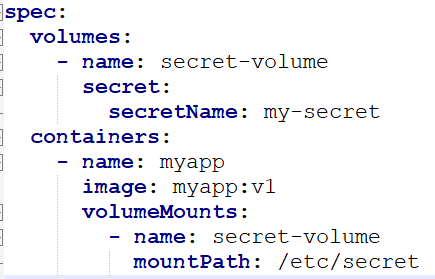
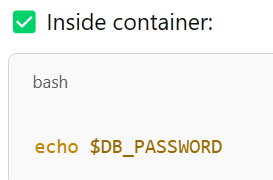


**2. Secrets**

* Store **sensitive data**.
* Stored in **Base64** encoded format.
* Kubernetes can encrypt them at rest (optional).
* Can be accessed securely by Pods.



✅ Files like /etc/secret/DB\_PASSWORD will exist inside container.

You can create a file to store secrets data and mount that as volume inside a container.

